Institut National Polytechnique de Lorraine

Département de formation doctorale en informatique — École doctorale IAE + M

Qualité de Service et Ordonnancement dans les Systèmes de Communication Temps Reel

THÈSE

présentée et soutenue publiquement le 13 octobre 2000

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(**Spécialité Informatique**)

par

Miguel Angel LEON CHAVEZ

Composition du jury

Président :	Ken CHEN	Prof. à l'Université de Paris Nord
Rapporteurs :	Marc BOURCERIE	Prof. à l'Université d'Angers
	José A. FONSECA	Prof. à l'Université d'Aveiro Portugal
	Laurent ROMARY	CR. CNRS HDR
E	RINQUET	MdC. HDR IRCyN Nantes
	erre THOMESSE	Prof. à l'INPL

Laboratoire Lorrain de Recherche en Informatique et ses Applications – UMR 7503

Institut National Polytechnique de Lorraine

Département de formation doctorale en informatique — École doctorale IAE + M

Qualité de Service et Ordonnancement dans les Systèmes de Communication Temps Reel

THÈSE

présentée et soutenue publiquement le 13 octobre 2000

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(**Spécialité Informatique**)

par

Miguel Angel LEON CHAVEZ

Composition du jury

Président :	Ken CHEN	Prof. à l'Université de Paris Nord
Rapporteurs :	Marc BOURCERIE	Prof. à l'Université d'Angers
	José A. FONSECA	Prof. à l'Université d'Aveiro Portugal
	Laurent ROMARY	CR. CNRS HDR
Examinateurs :	Yvon TRINQUET	MdC. HDR IRCyN Nantes
	Jean-Pierre THOMESSE	Prof. à l'INPL

Laboratoire Lorrain de Recherche en Informatique et ses Applications – UMR 7503

Remerciements

Je remercie Monsieur Jean-Pierre THOMESSE pour son accueil dans l'équipe Temps Réel et InterOpérabilité (TRIO) du LORIA, pour avoir encadré mon travail de recherche et pour son soutien, son encouragement et son amitié.

Je remercie également Madame Françoise SIMONOT, responsable de l'équipe TRIO, pour son accueil et pour avoir toujours veillé à ce que nous puissions travailler dans les meilleures conditions.

J'exprime toute ma reconnaissance au "Consejo Nacional de Ciencia y Tecnología" (CONACyT), à l'Université Autonome du Puebla (UAP) et au PROMEP du Mexique, également à la Société Française d'Exportation des Ressources Éducatives (SFERE) et l'ENSEM pour leur confiance et leur appui.

Je remercie

- Monsieur Marc BOURCERIE, Professeur à l'Université d'Angers,
- Monsieur José A. FONSECA, Professeur à l'Université d'Aveiro Portugal,
- Monsieur Laurent ROMARY, Chargé de Recherche CNRS,
- Monsieur Ken CHEN, Professeur à l'Université de Paris Nord,
- Monsieur Yvon TRINQUET, Maître de Conférence à l'École Centrale de Nantes,

de m'avoir fait l'honneur de faire partie du jury de cette thèse

Merci à Luis et Madalina pour leur amitié, à Nicolas Navet pour les trois années de cohabitation dans le bureau. Je remercie mes autres collègues et amis, Gladys et Christophe, Emmanuel, Laurent, Raphael, Paolo I et II, Titi, Olivier, Fabrice, Marcel, Michel, Bruno, Jörn, Mme. Dalbourg et Josette.

Je tiens à exprimer mes vifs remerciements à toutes et tous mes amis : Rosa María, Dení, Paty, Rosana, Flor, Angélica, Flora, Fabiola, David, Dario, Rafael, Herminio, Gilberto, Oscar, Carlos, Cristiani, Jaime Diaz et Izas, Jaime Cid et Galia, Pepe et Elsa, Enrique, Jesus, Albores, Cristina et Frederic.

A mis padres, hermanos y sobrinos

À Laura

Table des matières

Chapitre 1

Introduction Générale

Ce chapitre présente l'introduction générale ainsi que les sujets principaux de notre recherche qui seront la base du document. En même temps, il présente les motivations, les objectifs, l'organisation du document et finalement un résume par chapitre.

1.1 Les systèmes temps réel distribués

De manière générale, les *systèmes temps réels* sont caractérisés par la production de leurs résultats dans des intervalles de temps spécifiés, par exemple une définition typique est la suivante [Stankovic, 1988] : un système temps réel est un système dont l'exactitude dépend non seulement de la correction logique de ses résultats mais également du temps ou du moment où les résultats sont produits.

Nous nous sommes intéressés aux systèmes utilisés notamment dans les milieux industriels dont l'objectif principal est de contrôler et de surveiller un processus de production et la qualité des produits. Dans ce but, le système doit réagir dans un temps borné à chaque occurrence d'un événement qui pourrait changer l'état du processus mettant en danger la qualité des produits ou le fonctionnement correct du matériel de production (ou pire encore des vies humaines). C'est pourquoi ces systèmes sont appelés temps réel ou parfois systèmes réactifs.

Généralement, ces systèmes sont décomposés en sous-systèmes tels que l'objet contrôlé (le processus de production, les machines, les capteurs, les actionneurs, etc.), l'application temps réel (l'ensemble de programmes applicatifs organisés sous forme de tâches ou processus) et l'opérateur [Kopetz, 1991]. Nous appellerons l'objet contrôlé et l'opérateur les utilisateurs de l'application temps réel.

Dans les milieux industriels typiquement l'objet contrôlé est distribué, c'est-à-dire, les produits (articles, objets, biens, etc. proposés sur le marché par les entreprises) sont traités dans différentes étapes de production qui sont physiquement distribuées. Le seul moyen de communication est donc l'échange de messages entre les tâches de l'application temps réel, et aussi entre les utilisateurs de l'application. Dans ce cas là, on dit que l'application est distribuée, l'appellation distribuée [Andrews, 1991] provient du fait que les tâches s'exécutent de façon concurrente ou parallèle sur des processeurs distribués sans mémoire partagée et interconnectés par un ou plusieurs réseaux de communication.

En ce qui concerne la communication, le besoin majeur de l'utilisateur est le transfert de messages dans des intervalles de temps bornés. Néanmoins, ce n'est pas le seul, il y a d'autres besoins temporels selon les utilisateurs qui sont présentés au chapitre deux. Généralement, les besoins de l'utilisateur sont décrits en termes fonctionnels, temporels et de sûreté de fonctionnement. Nous ne nous sommes intéressés qu'aux besoins temporels.

L'analyse identifie le besoin d'échanges périodiques et parfois d'échanges sporadiques entre les utilisateurs et l'application. Le besoin d'échanges périodiques, que nous appellerons périodicité, provient du fait que l'application possède la connaissance de l'état courant de l'objet contrôlé par le biais de données qui sont périodiquement échantillonnées, selon la théorie d'échantillonnage. Le besoin d'échanges sporadiques provient des données aléatoires produites pour notifier l'occurrence d'alarmes ou d'événements imprévisibles.

La gigue est un phénomène temporel associé au besoin de périodicité, elle définit la variation de la période de la transmission des données, laquelle peut être variable ou non et entre une borne minimale et maximale. Le contrôle de la gigue peut être nécessaire.

Par ailleurs, l'analyse fait apparaître le besoin de cohérence temporelle et spatiale. La cohérence temporelle appliquée à la communication indique si les données ont été transmises et/ou reçues dans l'intervalle de temps spécifié. Elle peut être appliquée à d'autres opérations comme les productions de données, des traitements, etc. La cohérence spatiale signale si plusieurs copies de la valeur d'une donnée sont égales dans l'intervalle de temps spécifié.

1.1.1 Besoin, contrainte, caractéristique et propriété temporels

Dans la littérature spécialisée, la plupart des auteurs utilisent l'expression *contrainte temporelle* afin de faire référence à une restriction imposée sur une caractéristique temporelle de l'application ou sur l'un de ses composants. Les contraintes temporelles peuvent être exprimées et spécifiées de plusieurs façons, par exemple : langages de tests [Dasarathy, 1985], logique temporelle [Manna and Pnueli, 1992], automates temporisés [Alur and Dill, 1994], etc.

D'ailleurs, une *caractéristique temporelle* est définie comme un attribut qui s'exprime en temps ou en fonction du temps, c.-à-d. qui est quantifié à l'aide du temps, cet attribut porte sur le comportement de l'application ou l'un de ses composants. Une *propriété temporelle* est donc une assertion sur le respect d'une contrainte associée à une caractéristique temporelle. Une propriété est dite vraie si la caractéristique respecte la contrainte [Vega, 1996].

Par exemple, dans [Arvind *et al.*, 1991] on dit que l'environnement impose des contraintes temporelles sur tous les éléments, composants et activités de l'application. L'échange d'information se réalise grâce aux services fournis par un système de communication qui est généralement un réseau local. Les contraintes sont alors projetées sur le système de communication et elles sont traduites en échéances associées à chaque message.

Ainsi, dans [Malcolm and Zhao, 1995] on dit que la principale caractéristique qui permet de distinguer les systèmes de communication temps réel des autres types de systèmes de communication réside dans la nécessité de respecter les contraintes temporelles des messages, notamment leurs échéances. Un message doit respecter ses contraintes, dans le cas contraire il est considéré comme perdu.

Par contre nous utiliserons, tout au long du document, le terme *besoin temporel* pour faire référence aux caractéristiques temporelles des services de communication souhaitables par l'utilisateur de l'application temps réel. Ainsi, l'utilisateur devra demander non seulement le service mais il devra aussi spécifier les paramètres des caractéristiques temporelles

concernées. Dans un premier temps nous exprimerons les paramètres comme des bornes, minimales et maximales, et dans le quatrième chapitre nous montrerons qu'ils peuvent être exprimés de façon statistique ou déterministe à l'aide d'un concept plus générale tel que la Qualité de Service (QoS).

Dans notre approche, les systèmes de communication dit temps réel doivent fournir les services nécessaires pour satisfaire aux besoins de l'utilisateur. Ces services auront un ensemble de qualités associées qui seront perçues par l'utilisateur.

On peut remarquer que la différence entre les termes besoin et contrainte se trouve dans le point de vue : on utilise le terme besoin par rapport à l'utilisateur, par contre le terme « contrainte » est utilisé par rapport au système de communication.

1.1.2 Conception des applications temps réel

Le développement des applications temps réel se caractérise par un cycle de vie auquel est associée la prise en compte des besoins temporels tout au long du processus de développement. Plus la prise en compte du temps est cruciale dans l'application, plus sa gestion doit être stricte. Ainsi, dans [Adrion *et al.*, 1982] les applications temps réel sont caractérisées par un processus de vérification à chaque phase et entre chaque phase de son cycle de vie dans le but de pouvoir démontrer que le développement de l'application est complet, consistant et non ambigu par rapport aux besoins établis à la phase précédente («building the system right» [Boehm, 1976]). Ce processus de vérification permet habituellement de vérifier ou de garantir une certaine justesse de l'application finale par rapport aux besoins de l'utilisateur («building the right system» [Boehm, 1976]).

Les besoins temporels doivent être présents dès l'analyse des besoins de l'utilisateur, lors de l'élaboration du cahier de charges. C'est d'abord dans cette phase que les besoins temporels associés aux services et aux fonctions de l'application doivent être pris en compte dans le but de spécifier de manière générale l'application temps réel et ce d'un point de vue [Kopetz, 1997] : fonctionnelle et temporelle (prévisibilité [Stankovic, 1988; Stankovic and Ramamritham, 1990]) et sûreté de fonctionnement [Laprie, 1992] (fiabilité, disponibilité, sécurité, maintenance).

La spécification fonctionnelle décrit la structure et le comportement externe des activités de l'application. Cette spécification est issue d'une analyse fonctionnelle dont le but est de définir l'ensemble des services et des traitements que l'application doit fournir et accomplir [Bayart et Simonot-Lion, 1995].

La spécification temporelle devra être associée à la spécification fonctionnelle et devra prendre en compte d'une part les besoins temporels de l'utilisateur et d'autre part les services que l'application doit fournir dans le but de décrire les caractéristiques temporelles des services que l'utilisateur pourra modifier par le biais de paramètres. Ainsi, l'utilisateur demandera non seulement un service mais il devra aussi spécifier entre autres les paramètres des caractéristiques temporelles concernées.

La prévisibilité [Stankovic, 1988] a été définie comme la propriété la plus importante d'un système temps réel, c.-à-d., son comportement fonctionnel et temporel doit être aussi déterministe que possible pour satisfaire la spécification du système.

Cependant, le déterminisme est basé sur la capacité de prédire quelque chose dans le futur à partir de la connaissance du passé et du présent. Même si ce dilemme a été analysé par plusieurs philosophes et chercheurs, c'est un sujet de discussion dans les systèmes informatiques et plus précisément dans les systèmes temps réel. On trouve très souvent l'expression « protocole déterministe pour la communication temps réel », mais qu'est que cela veut dire exactement ? Le transfert d'un message dans un intervalle borné ou l'émission d'un message à l'instant précis ? D'ailleurs, sous quelles suppositions de charge, de débit, d'erreur, etc. dit-on que le protocole est déterministe ? En générale, le déterminisme n'existe pas à moins de faire plusieurs hypothèses sur la sûreté de fonctionnement [Thomesse and Leon, 1999].

La *sûreté de fonctionnement* [Laprie *et al.*, 1989] d'un système est la propriété qui permet de placer une confiance justifiée dans le service qu'il délivre. La *fiabilité* mesure la délivrance continue d'un service approprié ou le temps jusqu'à une défaillance. La *disponibilité* mesure la délivrance d'un service approprié par rapport à l'alternance service approprié - service inapproprié. La *sécurité* mesure le temps jusqu'à défaillance catastrophique ou maligne. La *maintenance* mesure le temps pour réparer le système après l'occurrence d'une défaillance bénigne.

C'est pourquoi, nous partageons l'idée de spécifier l'application temps réel dans les termes précédents. La spécification peut se composer d'un ensemble de spécifications informelles, semi-formelles et formelles comme il a été proposé dans [Bayart and Simonot, 1995].

D'ailleurs, un aspect qui doit être pris en compte dans la phase de spécification de l'application est sa performance. Plusieurs mesures ont été définies afin de la mesurer, par exemple [ISO 13236] : la capacité de traitement (la quantité de traitement en mesure d'être exécutée dans une période de temps), le débit du système (la quantité de traitement exécutée dans une période de temps), la charge d'opération (le rapport entre la capacité utilisée et la capacité disponible), etc.

La spécification de performances du système de communication est d'une importance capitale dans toutes les applications. Etant donné un ensemble de dispositifs, avec certaines caractéristiques de trafic, un besoin essentiel est que le système de communication ait la capacité appropriée pour supporter la charge attendue [Stallings, 1984]. Sans cette spécification, on pourrait attendre la même performance pour n'importe quelle taille des messages et n'importe quelle distance [Stallings, 1984 ; Fine and Tobagi, 1984].

L'évaluation de performances des réseaux locaux est faite typiquement par le biais de mesures tels que le débit d'information, l'utilisation du canal et différentes sortes de délais [Stallings, 1984 ; Fine and Tobagi, 1984 ; Abeysundara and Kamal, 1991]. Le débit d'information est défini par le nombre total de bits d'information transmis par unité de temps. L'utilisation du canal est définie par la fraction de temps employé pour la transmission de bits d'information comparé au temps total employé pour la transmission de bits d'information et de bits d'en-tête et de contrôle. Un délai peut être mesuré de différentes façons, selon les instants de temps pris

en compte dans la mesure. Une mesure de délai est le délai de transfert des messages (nous utiliserons le terme message pour faire référence à la trame, au paquet, au N-PDU, etc.) lequel est l'intervalle de temps nécessaire pour transmettre le message à travers le système de communication. Ce délai peut être mesuré en termes de valeur moyenne, maximale or le pire des cas [Joseph and Pandya, 1986]. Nous discuterons plus en détail ce délai dans le chapitre 3.

Dans les systèmes de communication dit temps réel, un message est généralement considéré comme perdu s'il est reçu hors de l'intervalle de temps spécifié même s'il est reçu correctement [Kurose *et al.*, 1984]. Ainsi, la performance est mesurée en termes de probabilité de perte, du taux de perte maximal, ou du nombre maximal de messages qui n'ont pas respecté leur échéance.

1.2 Les systèmes de communication

Les réseaux de communication ont été conçus originellement pour permettre l'échange d'information entre les utilisateurs. Ils ont été structurés par l'Organisation Internationale de Normalisation (ISO) dans une architecture à sept couches, appelée Modèle de Référence pour l'Interconnexion des Systèmes Ouverts (OSI) [ISO 7498]. Ce modèle décrit simplement les fonctions concernées dans la communication entre systèmes, et il définit les services qui sont nécessaires pour accomplir ces fonctions.

Le service à chaque couche est fourni par le biais d'un protocole, c.-à-d., la procédure de communication entre deux entités protocolaires en utilisant les services qui sont offerts par la couche inférieure. Ainsi, chaque couche est un « client » de la couche inférieure, et chaque couche est un « serveur » de la couche supérieure.

Le concept de service contribue à organiser les phases du processus de développement des protocoles de l'OSI. Pratiquement, n'importe quel modèle de développement du logiciel peut être utilisé dans ce but, par exemple, les phases suivantes ont été proposées dans [Perhson, 1990] et sont montrées dans la Fig. 1.1 :

- Analyse des besoins de l'utilisateur.
- Spécification des services de communication qui satisfont les besoins de l'utilisateur.
- Spécification des protocoles qui fournissent les services de communication spécifiés.
- Mise en œuvre des entités protocolaires conformément à la spécification du protocole.

L'analyse des besoins de l'utilisateur permet d'établir les services que le système de communication doit fournir et les contraintes sous lesquelles il fonctionnera. Après avoir réalisé l'analyse, les services et les protocoles peuvent être spécifiés. La spécification est le processus de décrire un système et ses propriétés souhaitables. La mise en œuvre d'une entité protocolaire consiste simplement en un logiciel [Voelcker, 1986], très souvent certaines entités protocolaires sont embarquées dans des circuits spéciaux inclus dans des cartes ou des équipements de communication.

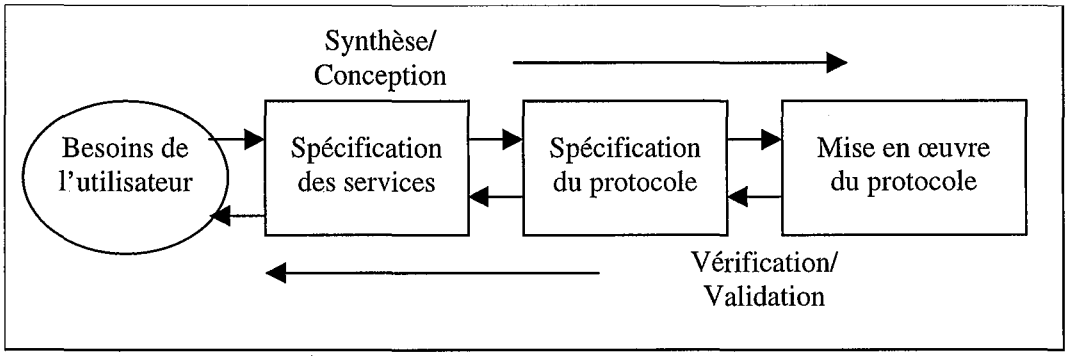

Fig. 1.1. Conception des protocoles de communication.

Dans la figure 1.1 les activités de conception, parfois appelées synthèse, sont représentées par les flèches orientées de gauche à droite. Les flèches de la droite vers la gauche représentent les activités vérifiant que la spécification est complète, consistante et non ambiguë, c.-à-d. le processus de validation.

1.2.1 Les réseaux de terrain

Les réseaux de terrain sont des réseaux locaux spéciaux qui sont utilisés fondamentalement pour connecter des équipements de terrain et des dispositifs de contrôle (capteurs et actionneurs) par le biais d'un canal partagé sur une topologie telle que le bus et la boucle.

Les réseaux de terrain sont vus comme une architecture à trois ou quatre couches, ne prenant en compte que la couche physique, la couche de liaison de données (divisée en sous-couches de Contrôle d'Accès au Médium (MAC) et de Contrôle de la Liaison Logique (LLC)) et la couche application. Néanmoins, certaines fonctions des couches manquantes sont parfois présentes dans des réseaux de terrain mais elles sont intégrées dans la couche application et liaison de données, par exemple le support pour l'interopérabilité, le contrôle de bout en bout, la fragmentation et le réassemblage des messages, le routage sur plusieurs réseaux de terrain interconnectés [Thomesse, 1998].

La couche MAC offre le service d'accès à un canal de transmission partagé, les protocoles MAC déterminent donc le délai de transfert expérimenté par les messages. Le protocole MAC, lorsque cela est nécessaire, doit déterminer le message à transmettre; il se comporte donc de façon implicite comme un ordonnanceur de messages.

Etant donné l'importance des protocoles MAC, beaucoup d'efforts ont été faits dans le but de les étudier, les classifier et les standardiser. Parmi les classifications, on trouve les suivantes qui sont basées sur des critères comme les méthodes d'accès [Kurose *et al.*, 1984], les méthodes d'allocation du canal [Tanenbaum, 1988], la performance [Abeysundara and Kamal, 1991], les méthodes d'accès au canal selon les conventions qui déterminent le site qui a le droit de l'utiliser [Mammeri and Thomesse, 1991], les modèles de coopération et de gestion du temps [Song and Thomesse, 1994], l'évolution des techniques d'accès au médium [van As, 1994], les stratégies de garantie et de meilleur effort utilisées pour garantir le respect des contraintes de temps [Arvind *et al.*, 1991 ; Malcolm and Zhao, 1995].

Parmi les standards on trouve les suivants : l'IEEE par son comité 802 a défini un modèle structuré en trois couches et un ensemble de standards régissant leurs fonctionnements. Le comité a standardisé entre autres : CSMA/CD [IEEE 802.3], le Bus à jeton [IEEE 802.4] et la Boucle à jeton [IEEE 802.5]. L'EN-50170 a standardisé les réseaux de terrain P-Net, Profibus et WorldFIP [EN 50170-1-2-3]. L'IEC 61158 a standardisé les réseaux de terrain TS 61158, ControlNet, Profibus, P-Net, Foundation Fieldbus, SwiftNet, WorldFIP et Interbus [IEC 61158].

Nous proposons dans le troisième chapitre du document, une analyse des principaux protocoles MAC et des réseaux de terrain à partir des besoins temporels de l'utilisateur identifiés au deuxième chapitre. Le but de l'analyse est d'identifier les méthodes d'accès et les algorithmes d'ordonnancement qui composent les services satisfaisant ces besoins temporels.

L'analyse fait voir que la plupart des services sont assurés par une configuration statique ou hors-ligne de l'ordonnancement des messages, c'est-à-dire, au moment de la configuration du réseau. Cependant, certains utilisateurs ont besoin de services dynamiques, c'est-à-dire qui sont spécifiés pendant l'exécution de l'application.

1.3 Motivations et objectifs

Les réseaux de terrain sont également connus sous le nom de systèmes de communication à architecture réduite et leur utilisation est de plus en plus répondue, par exemple : l'industrie de l'automobile, l'industrie textile, la fabrication des semi-conducteurs, la fabrication des appareils électroménagers, l'industrie alimentaire, l'industrie de la chimie, etc. En un mot toute industrie relevant aussi bien des processus continus que manufacturiers. Ils ont aussi pénétré le monde des machines, des voitures, des avions, de la gestion de bâtiments, etc.

L'analyse des besoins temporels de l'utilisateur, proposée dans le deuxième chapitre, montre qu'en plus du besoin de transférer les messages dans un délai borné, les utilisateurs des applications ont besoin d'échanger des données qui peuvent être caractérisées par la fréquence de production. Les données *périodiques* sont produites à intervalles réguliers. Les données *sporadiques* sont produites aléatoirement.

L'analyse des protocoles MAC et des réseaux de terrain, proposée dans le troisième chapitre, révélera que certains protocoles ont été conçus pour offrir des services qui garantissent la transmission des données à un instant précis ou dans un temps borné. Cependant, ces services sont assurés par une configuration statique ou hors-ligne de l'ordonnancement des messages, c'est-à-dire, au moment de la configuration du réseau.

Notre thèse propose une solution dynamique à la place des solutions statiques. Cette solution s'inspire de l'architecture de Qualité de Service définie dans les grands réseaux et du modèle de Services Intégrés de l'Internet qui est généralement utilisée au niveau de la couche réseau.

Nous proposons donc un protocole de réservation dynamique de ressources qui peut être implanté sur n'importe quel protocole MAC de type 'droit de parole' ou centralisé pour n'importe quelle topologie. Ce protocole accepte par ailleurs différents algorithmes d'ordonnancement. Il a été spécifié en termes de systèmes états-transitions et a été validé en utilisant l'outil ObjectGEODE.

1.4 Organisation du document

Le document est organisé de la manière suivante : le chapitre deux présente une analyse des besoins temporels des applications temps réel distribuées, l'analyse est illustrée par des applications industrielles [Almeida *et al.*, 1999b]. L'analyse permet d'identifier les besoins temporels et de les classifier.

Le chapitre trois analyse les systèmes de communication à architecture réduite et leurs protocoles MAC en mettant en lumière les méthodes d'accès et les algorithmes d'ordonnancement qui composent des services satisfaisant les besoins temporels de l'application [Leon and Thomesse, 2000].

Plusieurs architectures de QoS ont été définies pour les systèmes de communication, nous analysons dans le quatrième chapitre les architectures proposées par l'ISO [ISO 13236], par la communauté de l'Internet [Braden *et al.*, 1994 ; Blake *et al.*, 1998 ; Bernet *et al.*, 1999 ; Ghanwani *et al.*, 1999 ; Yavatkar *et al.*, 2000 ; Seaman *et al.*, 1999] et par l'IEC [IEC 61158].

Le cinquième chapitre présente une architecture de QoS pour le modèle client-serveur ainsi que des extensions pour les modèles multiclient-serveur et client-multiserveur. Cette architecture de QoS prend en compte le modèle des Services Intégrés de l'Internet. Nous proposons également un Protocole à RéSerVation des Ressources pour le modèle client-serveur (CS-RSVP). Le protocole est spécifié par le biais du Langage de Spécification et Description (SDL) et validé par simulation avec l'outil ObjectGEODE (Annexes A, B et C). Nous avons choisi le modèle client serveur parce qu'il modélise les interactions entre les couches adjacentes et homologues du Modèle de Référence OSI.

Finalement, nous présentons les conclusions et les perspectives de notre recherche.

Avant de commencer le deuxième chapitre, nous présentons les résumés des chapitres 2, 3, 4 et 5 rédigés en français.

Résumé du Chapitre 2

Analyse des besoins temporels des utilisateurs

De façon générale, l'analyse de besoins permet d'avoir un bon aperçu et une compréhension profonde de tous les aspects d'un domaine donné. Les besoins de l'utilisateur peuvent être groupés dans les besoins fonctionnels, temporels et de sûreté de fonctionnement (cf. § 1.1.2). Spécifiquement, nous proposons dans le Chapitre 2 une analyse des besoins temporels de l'utilisateur dans le but d'établir les caractéristiques temporelles des services que le système de communication doit fournir dans les systèmes temps réel.

Les besoins de l'utilisateur qui sont pris en compte par l'analyse sont ceux d'un système temps réel spécial utilisé dans les milieux industriels où les utilisateurs sont les personnes responsables du contrôle, de la maintenance, de l'ordonnancement de production, de la gestion technique, etc. Mais les « utilisateurs » peuvent également être des capteurs, des actionneurs, des contrôleurs des machines de production, un système de transport, etc.

En ce qui concerne la communication, tous les utilisateurs ont des besoins temporels différents qui peuvent être caractérisés par les propriétés temporelles des données échangées entre eux, c'est-à-dire, les données peuvent être caractérisées par la criticité, la fréquence de transmission, la gigue, la cohérence temporelle et la cohérence spatiale. Les données peuvent être également caractérisées par les propriétés spatiales, c.-à-d. la taille des données par transaction.

Le chapitre est structuré de la façon suivante : la section 2.1 présente l'introduction, la section 2.2 présente l'analyse de besoins temporels, la section 2.3 présente l'analyse de besoin spatial.

2.1 Introduction

Les applications industrielles sont de plus en plus distribuées et intégrées, non seulement dans l'usine elle-même mais entre les sites de production, de gestion, du développement des produits et de leur distribution. Ces applications sont donc composées d'une grande quantité de logiciels qui doivent coopérer, et qui doivent être coordonnés dans le but de fournir la qualité de service demandé par les différents utilisateurs. Plusieurs réseaux de communication servent de support à cette coopération et sont devenus l'infrastructure principale des applications industrielles.

Les systèmes de communication doivent donc fournir différents services selon les différentes nécessités et un point important de comparaison entre eux est leur capacité de satisfaire les besoins de l'utilisateur, essentiellement les besoins temporels. Pour considérer ces aspects, ce chapitre les analysera et dans le chapitre suivant les principales solutions seront étudiées en tenant compte des réseaux de communication temps réel qui sont connus sous le nom de réseaux de terrain.

2.2 Besoins temporels de l'utilisateur

Cette section présente l'analyse des besoins de l'utilisateur en termes des propriétés temporelles des données échangées entre les utilisateurs, c.-à-d. l'importance critique, la fréquence de transmission, la gigue, la cohérence temporelle et la cohérence spatiale.

2.2.1 Délai de transfert des messages

Typiquement, les applications temps réel interagissent avec les utilisateurs (l'objet contrôlé et l'opérateur) par le biais d'un ensemble de données. La dynamique de l'objet contrôlé produit des données qui peuvent être caractérisées de diverses façons, comme par exemple la *durée de vie* (voir figure 2.1), c.-à-d. l'intervalle de temps pendant lequel la valeur de la donnée est valable.

Lorsque la durée de vie des données est bornée et petite, l'instant auquel l'application produit ses actions doit se trouver dans l'intervalle de temps durant lequel les données sont encore valables dans le but de garantir effectivement que l'application agit sur l'état courant de l'objet contrôlé. Celui-ci signifie que les temps de réponse de toutes les activités de l'application (production, transmission, consommation, traitement, etc.) doivent être aussi bornés.

Ce type de données est appelé *critique* et les contraintes sur les données peuvent être encore caractérisées comme *légères* ou *relatives* et *dures* ou *strictes*. Dans le premier cas, l'objet contrôlé peut tolérer quelques délais, cependant, dans le deuxième cas, les actions de l'application doivent se trouver dans les fenêtres de vie respectives des données puisque dans le cas contraire, les actions seront appliquées à des données qui ne sont plus valables et qui peut mettre en danger des vies humaines ou endommager sérieusement l'environnement ou l'équipement.

En plus de données critiques, l'objet contrôlé produit des données dont la durée de vie n'est pas bornée dans le temps. Ces données sont appelées *non-critiques* et sont habituellement traitées par les tâches de l'application s'exécutant à l'arrière plan.

En bref, par rapport à ces sortes de données, les besoins de l'utilisateur peuvent être récapitulés de la manière suivante :

- Il est essentiel de faire la distinction entre le trafic critique et non-critique. Le système de communication doit transporter l'un et l'autre [ISO 184/SC5/WG2].

- Le système de communication doit garantir que tous les transferts de données critiques seront achevés dans intervalles de temps bornés.

- Le système de communication doit faire de son mieux pour transférer les données critiques-relatives.

- Les transferts de données non-critiques ne doivent pas mettre en danger la livraison de données critiques.

2.2.2 Périodicité

Habituellement les applications temps réel ont la connaissance de l'état courant de l'objet contrôlé par le biais des données qui sont produites et transmises par des capteurs ou des dispositifs de plus haut niveau de façon périodique et aléatoire. De la même manière, l'objet contrôlé est sous contrôle de l'application par le biais de données de contrôle transmises vers les actionneurs ou les dispositifs de contrôle. Dans les deux cas, lorsque la transmission des données est faite à intervalles de temps réguliers, les données sont appelées *périodiques*, dans le cas contraire elles sont appelées *apériodiques*. Ces données peuvent être critiques ou non-critiques selon la discussion précédente.

Dans certains cas, il est possible de délimiter le temps de séparation minimale entre deux transmissions apériodiques consécutives de la même source. Cette sorte de données apériodiques est appelée *sporadique*.

Ainsi, en plus des besoins précédents, les besoins de l'utilisateur par rapport à la fréquence de transmission des données sont les suivants :

- Il est important de faire la distinction entre le trafic périodique et apériodique. Le système de communication doit transporter l'un et l'autre de la source vers la (ou les) destination(s).

- Il doit être possible faire la différence entre l'urgence relative des données, notamment entre les données critiques et non-critiques.

On peut remarquer que la plupart des messages de l'application contiennent des données périodiques qui doivent être transmises avant le début de leur période suivante. Ainsi, le besoin de périodicité signifie que le système de communication doit fournir le service pour transmettre ces données avant le début de leur période suivante.

On peut aussi remarquer que la principale différence entre les messages périodiques et apériodiques c'est l'instant auquel leurs caractéristiques temporelles sont connues par le système de communication. Les caractéristiques temporelles des messages périodiques peuvent être connues à l'avance, au moment de la configuration du réseau. Par contre, celles des messages apériodiques ne sont connues qu'au moment où les messages arrivent au système de communication, pendant l'exécution de l'application.

La discussion précédente sur les données selon leur criticité et leur fréquence de transmission est résumée par la figure 2.2. La région ombrée représente les données critiques, parfois appelées temps réel, lesquelles peuvent avoir deux besoins différents : d'une part, un délai borné de transfert (associé à leur criticité) et d'autre part, la périodicité (associée à la régularité de leur transmission). Cependant, les données temps réel et non-temps réel peuvent être de nature périodique, apériodique ou sporadique.

Données	Critique		Non-critique
	Stricte	Relative	
Périodique			
Apériodique	Temps Réel		Non-temps réel
Sporadique			

Fig. 2.2. Caractéristiques temporelles des données dans les applications temps réel.

2.2.3 Gigue

Ce besoin est associé à la périodicité. La gigue mesure la variation de la période d'une opération ou d'un événement, de la transmission périodique des données, de la production, de la consommation, laquelle peut être variable ou non et entre une borne inférieure ou supérieure.

2.2.4 Cohérence temporelle

La cohérence temporelle met en relation l'occurrence de deux ou plusieurs événements qui doivent se produire dans un intervalle de temps. Elle peut être utilisée pour spécifier des actions simultanées telles que la production de valeurs, la transmission de données, etc. [Thomesse, 1999].

Dans de nombreuses applications, les sites ont besoin de données qui sont produites et transmises par différents sites sources. La cohérence temporelle indique donc si les messages ont été émis et/ou reçus dans un intervalle de temps spécifié [ISO 184/SC5/WG2 ; Saba *et al.*, 1993].

2.2.5 Cohérence spatiale

La cohérence spatiale signale si toutes les copies d'une variable sont égales à un instant donné ou dans un intervalle de temps spécifié [Saba *et al.*, 1993; Thomesse, 1999].

Dans de nombreuses applications, les sites stockent les valeurs des variables dans une liste ou dans une structure de données quelconque. Lorsque les sites échangent les valeurs, les copies de la liste peuvent être différentes, notamment à cause de différentes fréquences de production, de transmission et de consommation, mais aussi à cause des délais, de messages perdus et de retransmissions.

Ainsi, le besoin de cohérence spatiale impose au système de communication l'existence d'un mécanisme dont le but est de garantir que les diverses copies de la liste seront toutes égales à un instant donné ou dans un intervalle de temps spécifié. Ce mécanisme devra informer les sites producteurs et consommateurs de l'état de cohérence de la liste ; parallèlement le mécanisme devra prendre les mesures nécessaires pour mettre à jour les valeurs de la liste [ISO 184/SC5/WG2].

2.3 Taille des données par transaction

Par rapport à la taille des données échangées par transaction, quelques utilisateurs ont besoin d'échanger des bits de données, d'autres d'échanger des mots de données, quelques-uns d'autres d'échanger des flux de données de longueur variable.

Les bits de données sont associés habituellement au statut de dispositifs très simples, limités à l'état allumé/éteint. Dans ce cas, les applications ne nécessitent que le transfert de quelques bits d'E/S par site. Cependant, d'autres dispositifs, comme les capteurs ou les actionneurs, sont plus élaborés et ils utilisent plus d'information telles que le statut, le diagnostic et le contrôle. Dans ce cas, le système de communication ne nécessite que de transférer un ou deux octets de données d'E/S.

En montant dans la hiérarchie de complexité des dispositifs, il existe des dispositifs de haut niveau tels que les réseaux à logique programmée, les machines de contrôle, les robots et autres équipements. Le fonctionnement de ces dispositifs a besoin d'un flux de plusieurs octets de données associés au statut, à la configuration, au diagnostic, au calibrage et au contrôle. Dans ce cas, le système de communication ne nécessite que de transférer plusieurs octets de données d'E/S. Au plus haut niveau, il existe le besoin d'échanger des fichiers, d'accéder aux bases de données, de recueillir des données associées à tout le système, de permettre des accès à distance au système, etc. Ces opérations ont besoin d'échanger de plus longs flux de données par transaction.

Il est intéressant de remarquer qu'au fur et à mesure que la taille de données par transaction augmente entre les différents utilisateurs, les besoins temporels deviennent plus détendus [Almeida *et al.*, 1999b]. En fait, les transactions sur des bits ou des octets de données sont associées aux capteurs et actionneurs lesquels sont impliqués dans les boucles internes de contrôle d'un sous-système de l'application industrielle, en conséquence les utilisateurs demandent des transmissions fréquentes avec une gigue minimale et un délai de transfert petit et borné. Les flux de plusieurs octets de données sont associés aux dispositifs de plus haut niveau et ils sont impliqués dans les boucles externes de contrôle, ainsi les dispositifs demandent des services de communication moins stricts en termes de contraintes de temps. Au niveau le plus haut des applications industrielles, les utilisateurs ont besoin de longs flux de données associés à la surveillance et le contrôle du système entier, le contrôle par retour de l'information est réalisé avec une intervention humaine, en conséquence les besoins temporels sont plus détendus.

À n'importe quel niveau de communication, une analyse temporelle est toujours nécessaire en associant des attributs tels que les temps d'exécution, les temps de transfert, les échéances et fréquences des tâches et des messages de l'application en utilisant une méthode commune pour les spécifier (logique temporelle [Manna and Pnueli, 1992], automate temporisé [Alur and Dill, 1994], ...). La seule différence est l'unité de temps utilisée pour exprimer les besoins temporels.

Par rapport à la taille de données, les besoins de l'utilisateur peuvent être résumés ainsi :

- Il est nécessaire de faire la distinction entre les tailles des données. Le système de communication doit fournir le service pour transmettre les données de la source vers la destination quel qu'il soit la taille des messages.

2.4 Conclusion

Nous avons présenté une analyse des besoins des utilisateurs afin d'identifier les caractéristiques des services que les systèmes de communication doivent fournir dans les systèmes temps réel. L'analyse s'est concentrée sur les besoins ayant rapport au temps et à l'espace et ne prend pas en compte d'autres besoins parfois importants des utilisateurs tels que la performance ni la sûreté de fonctionnement. Cependant, l'analyse permet d'organiser la discussion des systèmes de communication du prochain chapitre.

Cette analyse a montré que le principal besoin des utilisateurs est le délai borné du transfert des messages. Cependant, il y a d'autres besoins temporels selon les utilisateurs, par exemple dans les applications industrielles, ces besoins sont la périodicité, la gigue, les cohérences temporelles et la cohérence spatiale.

Le besoin de périodicité provient du fait que la plupart des données échangées entre les utilisateurs sont périodiquement échantillonnées, ainsi le système de communication doit fournir le service de transmission pour ce type de messages. La transmission des données doit être également périodique, la gigue mesure son éventuelle variabilité.

La cohérence temporelle de transmission ou de réception indique si les messages ont été transmis et/ou reçus dans un intervalle de temps donné. La cohérence spatiale indique si toutes les copies d'une variable sont identiques à un instant donné ou dans une fenêtre de temps donnée.

Résumé du Chapitre 3

Les Systèmes de Communication dans les Systèmes Temps Réels

Les Réseaux de Terrain et les Protocoles MAC

Ce chapitre présente une analyse des principaux protocoles de Contrôle d'Accès au Médium qui sont utilisés par les réseaux de terrain. L'analyse prend en compte deux points de vue, d'une part, les besoins temporels de l'utilisateur définis dans le Chapitre 2 tels que le délai de transfert des messages, la périodicité, la gigue, les cohérences temporelles et spatiale, d'autre part, l'analyse prend en compte les caractéristiques des protocoles MAC identifiant les méthodes d'accès et les algorithmes d'ordonnancement qui satisfont les besoins précédents.

Le chapitre est organisé de la manière suivante : la section 3.1 présente l'introduction, la section 3.2 présente l'analyse des protocoles MAC selon les besoins temporels de l'utilisateur, la section 3.3 présente les principales caractéristiques des réseaux de terrain et des protocoles MAC.

3.1 Introduction

Même si tous les choix en matière de protocoles de communication peuvent influer sur la capacité à fournir des services de communication ayant rapport au temps, le protocole MAC possède la plus haute importance dans les réseaux locaux parce qu'il met en œuvre le service d'accès au médium, c'est-à-dire, lorsque le médium est partagé entre plusieurs sites, le protocole MAC est l'arbitre d'accès et détermine donc le message à être transmis lorsque cela est nécessaire. En conséquence, il a des effets sur le délai de transfert des messages et sur l'ordonnancement des messages dans le réseau.

Traditionnellement, lorsqu'on parle de contraintes temporelles, on dit que le protocole MAC doit considérer les contraintes sur les différents messages. Ainsi, un message produit par une application temps réel doit être reçu dans un intervalle de temps borné. Si le délai de bout en bout (mesuré au niveau de l'application temps réel) excède cette contrainte de temps, le message est généralement considéré comme perdu quoiqu'il soit reçu avec succès [Kurose *et al.*, 1984 ; Arvind *et al.*, 1991 ; Tindell *et al.*, 1995a ; Malcolm and Zhao, 1995].

Etant donné l'importance des protocoles MAC, plusieurs études et classifications ont été proposées dans la littérature, parmi eux les protocoles sont classés selon les critères suivants :

- L'accès contrôlé et l'accès basé sur la contention [Kurose *et al.*, 1984] : les protocoles à accès contrôlé concernent les protocoles libres de collision (demande adaptative (ex : réservation et jeton) et les protocoles à assignation prédéterminée du canal (ex : TDMA)). Avec de tels protocoles, les sites sont coordonnés de telle manière que plusieurs sites n'essayent jamais de transmettre simultanément.

Les protocoles basés sur la contention fonctionnent en divisant les sites en deux ensembles, un des sites qui sont autorisés à émettre et tous les autres sites qui ne sont pas autorisés. Parmi les mécanismes qui ont été proposés comme critères pour déterminer si un site appartient à l'ensemble de sites autorisés on trouve des mécanismes probabilistes, basés sur le temps et sur l'adresse.

- Stratégies pour s'approprier le canal [Tanenbaum, 1988] : les protocoles sont classés en trois stratégies, les protocoles basés sur la contention, les protocoles libres de collision et ceux à contention limitée. La première stratégie englobe les protocoles à écoute de la porteuse, dans lesquels les sites écoutent le bus et agissent en conséquence (ex : CSMA persistant et non-persistant). Dans les protocoles à accès multiple avec détection de collision (ex : Ethernet) les sites détectent la collision, interrompent leur transmission, attendent un temps aléatoire et essayent à nouveau.

  Dans les protocoles libres de collision, il n'y a pas des collisions même pendant la période de contention (ex : mappe de bits (basic bit-map), priorités alternées (Broadcast Recognition with Alternating Priorities), multi-niveau multi-accès (Multi-Level Multi-Access), compte à rebours binaire).

  Les protocoles à contention limitée emploient la stratégie de contention lorsque la charge du réseau est faible afin de fournir un petit délai mais utilisent la technique dite libre de collision lorsque la charge est élevée afin de fournir une meilleure performance (ex : le protocole adaptatif de parcours en arbre et le protocole de l'urne).

- Méthodes d'accès au canal [Mammeri and Thomesse, 1991] : les protocoles sont classés en trois méthodes, consultation, compétition et multiplexage. Dans la première méthode, les sites se consultent les uns les autres pour décider de qui a le droit d'accès. L'autorisation peut être obtenue par échange d'information (ex : boucle à jeton, bus à jeton, scrutin) ou par le matériel (ex : lignes de contrôle et d'insertion de registres).

  Dans la méthode basée sur la compétition, les sites emploient leur propre information et l'état du canal pour décider de transmettre leurs données (ex : ALOHA, CSMA, CSMA/CD, CSMA/CA, CSMA/DCR, résolution en arbre binaire, boucle à contention). Dans la méthode de multiplexage, la bande passante est divisée entre tous les sites (ex : FDM et TDM).

- Modèles de coopération et de gestion du temps [Song and Thomesse, 1994] : les protocoles sont classés selon les modèles de coopération utilisés, client/serveur ou producteur/consommateur, et selon une gestion explicite du temps ou non (gestion qui peut être statique ou dynamique).

- Stratégies utilisées pour respecter les contraintes temporelles des messages [Malcolm and Zhao, 1995] : les protocoles sont classés selon les stratégies de garantie et de meilleur effort pour transmettre les messages synchrones et asynchrones et selon l'importance relative du processus d'arbitrage d'accès ou du processus de contrôle de transmission. Le processus d'arbitrage d'accès détermine à quel moment, *quand*, le site peut envoyer un

message sur le canal. Le processus de contrôle de transmission détermine *combien de temps* le site peut continuer à transmettre.

Dans la stratégie avec garantie, tout message accepté pour transmission est transmis en respectant ses contraintes temporelles. Dans la stratégie avec meilleur effort, tout message accepté n'est pas nécessairement transmis en respectant ses contraintes temporelles.

- Trois générations de technologie et l'évolution vers les térabits/s [van As, 1994]. Les protocoles sont examinés à travers trois générations de technologie. La première génération (ex : Ethernet, Bus à jeton et Boucle à jeton) couvre le débit de 10 Mbits/s. La deuxième génération prend en compte diverses configurations et méthodes d'accès dans l'intervalle de débits à partir de 100 Mbits/s jusqu'au multi-Gbits/s. Dans la troisième génération, diverses conceptions architecturales et expérimentales des réseaux ont l'avantage de la bande passante du médium optique qui est de plus de 20 THz.

Ce chapitre présente l'analyse et diverses classifications des protocoles MAC [Leon and Thomesse, 2000] selon les besoins temporels de l'utilisateur définis au chapitre précédent.

3.2 Caractéristiques temporelles

3.2.1 Délai du transfert des messages

Nous allons le définir comme le temps nécessaire pour transmettre un message à travers le réseau de terrain, il intègre les délais suivants : les délais dans les couches application du site émetteur et du site récepteur, le délai d'attente dans la couche MAC du site émetteur, le délai de transmission du message, le délai de propagation de bout en bout et le délai d'arrivée dans la couche MAC du site récepteur.

Le délai de transmission du message est le temps nécessaire pour transmettre physiquement le message et est égal à L/R, où L est la longueur en bits du message (trame) et R est le débit en bits/sec. Le délai de propagation de bout en bout est égal à C/P, où C est la longueur du canal en mètres et P est la vitesse de propagation en mètres/sec.

Le délai d'arrivée dans la couche MAC du site récepteur dépend de la vitesse à laquelle le processeur traite l'interruption de la couche MAC. Le délai d'attente dans la couche MAC (ou le temps d'accès au médium) du site émetteur dépend du protocole MAC [Kurose *et al.*, 1984 ; Malcolm and Zhao, 1995].

Le délai borné de transfert des messages est un besoin temporel qui a été analysé dans plusieurs articles et divers résultats ont été présentés (voir Tableau 3.1). Cependant, plusieurs suppositions doivent être faites si l'on veut de borner ce délai, par exemple : l'opération normale du réseau, transmission sans perte de messages, sans perte du jeton, fonctionnement normal du site maître, capacité illimitée du tampon à la couche MAC, charge du réseau constante, égales longueurs des messages, etc. Le délai borné de transfert des messages est donc un besoin qui doit être analysé attentivement en décrivant explicitement les suppositions faites.

Dans le but de satisfaire ce besoin, deux méthodes d'accès sont identifiées : la méthode à accès contrôlé et la méthode à accès non contrôlé mais avec résolution déterministe des collisions. Les protocoles de chaque classe sont résumés dans le Tableau 3.1 et ils sont analysés dans la Section 3.3.

3.2.1.1 Méthode à accès contrôlé

La méthode à accès contrôlé ou libre de collisions est une manière de garantir l'existence d'un seul site accédant au canal et ce à n'importe quel instant. Il y a deux approches possibles : centralisée et distribuée. Dans l'approche *centralisée*, un site spécial appelé maître ou arbitre donne le droit explicite d'accès au canal. Cette approche est connue par le modèle Maître – Esclave et est utilisée par exemple par le réseau de terrain Profibus.

Il y a un autre modèle de communication connu sous le nom de Producteur-Distributeur-Consommateur. Dans ce cas, le site maître s'adresse à chaque entité de données (adressage à la source). Le site responsable de produire l'entité, quel qu'il soit, répond à l'appel du maître en diffusant la valeur de l'entité. C'est l'approche utilisée par les réseaux de terrain WorldFIP et Foundation Fieldbus. La périodicité est gérée alors par le distributeur.

Les protocoles à accès contrôlé peuvent aussi être *distribués*. Dan ce cas, tous les sites sont placés au même niveau logique par le biais d'une technique d'allocation, c.-à-d. le temps dans le réseau est divisé en intervalles de temps égaux, appelés aussi fenêtres temporelles, qui sont assignés aux sites et pendant lesquels seulement un site peut accéder au canal.

Deux méthodes d'allocation périodique sont identifiées : le multiplexage temporel et la circulation d'un jeton. Dans certains cas l'allocation du canal n'est pas constante, c'est pourquoi divers auteurs le référent comme allocation cyclique, et non périodique. Cependant, nous utilisons le terme périodique puisque la variation de la période est prise en compte dans le besoin de contrôle de la gigue.

Dans la *méthode de multiplexage temporelle* (voir TDMA protocole, section 3.4.12) chaque fenêtre est divisée à nouveau en tranches temporelles. Selon le nombre de tranches la méthode peut être synchrone ou asynchrone [Kurose *et al.*, 1984].

Etant donné que la taille des tranches est fixée et que le nombre de sites est connu, le mécanisme de multiplexage temporel synchrone satisfait au besoin de délai borné de transfert. Néanmoins le délai d'attente dans la couche MAC du site émetteur peut être important en fonction du nombre de sites dans le réseau. De plus, ce délai a des conséquences sur la variation de la période et de la gigue qui peuvent être très grandes. D'autre part, étant donné que le nombre de tranches est limité dans la méthode asynchrone, il peut y avoir des sites bloqués en attente d'une tranche libre [Kurose *et al.*, 1984]. Ceci peut produire un délai de transfert non borné.

Dans la *méthode de passage du jeton*, la fenêtre allouée à chaque site est identifiée par un jeton mis en circulation autour d'une boucle, physique (Boucle à jeton) ou logique (Bus à jeton). Le jeton est passé d'un site vers le site adjacent et un site doit attendre le jeton pour pouvoir émettre. Le fonctionnement temporel est contrôlé en limitant le temps maximal

d'utilisation du jeton. Les protocoles FDDI et Bus à jeton utilisent cette approche. Profibus utilise également cette méthode entre les sites maîtres, ainsi que le précèdent IEC Fieldbus entre les sites maîtres appelés "Link Masters".

La méthode satisfait au besoin de délai borné de transfert si chaque site possède le jeton pendant un temps maximal d'utilisation et si le temps moyen de passage du jeton est constant (le temps requis pour passer le jeton d'un site vers son successeur). Dans ce cas, le temps moyen de rotation du jeton est constant, c.-à-d., le temps séparant deux réceptions successives du jeton.

[Moon *et al.*, 1998] ont présenté des bornes du temps moyen de passage du jeton et du temps moyen de rotation du jeton pour évaluer la dégradation de la performance de la norme IEEE 802.4 (Bus à jeton) dans un environnement bruité. Les auteurs montrent qu'un site doit attendre plus de 100 fois le temps nominal de passage du jeton lorsque le jeton est perdu, et en conséquence la boucle logique doit être réinitialisée. L'analyse montre aussi que le jeton peut être facilement perdu dans un tel environnement, c'est la raison principale de la dégradation des performances du protocole Bus à jeton. Par ailleurs, les auteurs proposent de modifier légèrement la spécification de la norme pour aider à l'immuniser contre le bruit. Les modifications proposées sont : redéfinir la condition d'occurrence du signal « noise-burst » pour le rendre peu sensible au bruit externe et reconcevoir la procédure d'initialisation de la boucle pour réduire le temps d'initialisation requis.

On peut remarquer que le protocole Boucle à jeton [IEEE 802.5] utilise la méthode de passage du jeton, néanmoins l'utilisation du mécanisme d'assignation globale des priorités aux messages empêche que l'assignation du canal soit strictement périodique. Dans ce protocole, il y a un champ de priorité dans le jeton qui contrôle le droit d'émission, chaque site l'examine lorsque le jeton passe par lui. Le site insère la priorité du message le plus prioritaire en attente d'émission si et seulement si cette priorité est supérieure à celle du jeton reçu. Un site ne peut saisir le jeton et émettre ses messages que lorsque le jeton revient avec la priorité demandée et après avoir fait un tour à travers tous les sites. De cette façon l'ordre d'émission des sites est défini par les priorités des messages et non par la place du site dans la boucle.

[Ng et Liu, 1991] ont analysé par simulation le délai moyen des messages et le délai maximal observé des messages dans trois protocoles à jeton (Bus à jeton, Boucle à jeton et Boucle à trames). Le délai moyen des messages est défini comme le temps entre l'instant d'arrivée de la trame au tampon de sortie du site émetteur et l'instant auquel la trame est reçue au site destinataire. Les auteurs montrent que la plupart du temps, le délai moyen des messages dans le protocole Bus à jeton est plus petit que ceux des protocoles Boucle à jeton et Boucle à trames. Néanmoins, le délai maximal observé des messages dans le protocole Boucle à trames est plus petit que ceux des protocoles Boucle à jeton et Bus à jeton. Les auteurs conseillent d'utiliser la boucle à trames en combinaison avec le mécanisme d'assignation global des priorités aux messages lorsque le débit de transfert augmente dans les applications temps réel.

Le protocole Boucle à trames est une variante de la méthode de passage du jeton, ici la boucle est divisée en plusieurs trames de taille égale. Chaque trame est initialement marquée comme libre et circule à travers la boucle. Lorsqu'un site veut émettre un message, il attend une trame libre, la marque comme occupée et émet son message lequel doit être plus petit ou égal à la taille de la trame. Lorsque la trame revient au site émetteur, il la marque comme libre.

Le mécanisme d'assignation globale des priorités aux messages peut être utilisé conjointement avec le protocole Boucle à trames de la manière suivante : chaque trame se voit assigner une priorité qui est égale à la priorité la plus élevée du message en attente d'émission. Un site peut utiliser une trame si la priorité de son message est supérieure ou égale à la priorité des trames en cours de circulation.

3.2.1.2 Méthode d'accès non contrôlé

La méthode est connue sous le nom d'*Accès Multiple avec Ecoute de la Porteuse* (CSMA) et est totalement distribuée. Tous les sites sont placés au même niveau logique lequel donne un caractère de modularité et de souplesse au système de communication. De plus, puisqu'il n'y a pas de messages supplémentaires tels que le jeton ou ceux qui sont produits par un éventuel maître, la surcharge du réseau est faible. Cependant, deux ou plusieurs sites peuvent transmettre au même instant et en conséquence il peut y avoir des collisions entre les messages.

Le protocole le plus populaire est le *CSMA avec Détection de Collisions* (CSMA/CD) [IEEE 802.3] lequel possède un bon comportement jusqu'à un certain niveau de charge mais à partir duquel ses performances dégradent considérablement [Stallings, 1984 ; Fine and Tobagi, 1984 ; Abeysundara and Kamal, 1991]. Dans ce protocole, le site émet dès que le canal est détecté libre. Pendant la transmission, le site écoute le canal et si une collision est détectée, les sites interrompent leur transmission, attendent une période aléatoire et retransmettent. Les sites arrêtent leurs retransmissions si le nombre de collisions a atteint un seuil fixé. Ainsi, la résolution de collisions est indéterminée et imprévisible, en conséquence il n'est pas possible de borner le délai de transfert des messages. Néanmoins, divers mécanismes supplémentaires ont été proposés dans le but de prendre en compte les contraintes temporelles des messages, quelques exemples sont les suivants : Window-Access CSMA [Kurose *et al.*, 1988 ; Zhao *et al.*, 1990 ; Znati, 1991], Virtual-Time CSMA [Molle and Kleinrock, 1985 ; Zhao and Ramamritham, 1987], PB/CSMA/CD [Ulusoy, 1995] and CSMA/LDCR [Norden *et al.*, 1999].

D'autre part, la méthode à accès non contrôlé mais avec *résolution déterministe des collisions* est une manière de borner le délai de transfert des messages. Deux protocoles sont identifiés, l'un CSMA/CA et l'autre CSMA/DCR. Dans le protocole *CSMA avec Evitement de Collision* (ex : le protocole CAN) tous les messages se voient assigner un identificateur à la configuration du réseau, l'identificateur est utilisé comme priorité pendant l'opération du réseau et notamment la phase d'arbitrage. Dès que le canal est détecté libre, les sites commencent l'émission de leur message le plus prioritaire tout en surveillant le canal. Le bit de poids plus fort du champ de l'identificateur est transmis d'abord. Si un site émet un bit récessif (1) mais écoute un bit dominant (0), cela veut dire qu'une collision s'est produite. Le site sait alors que son message n'est pas le plus prioritaire et il arrête donc son émission et attend que le canal redevienne libre. Si le site émet un bit récessif et écoute un bit récessif alors le site émet la suite du champ identificateur. S'il n'y a pas de collision dans le champ identificateur, le site émet le corps du message.

Une borne sur le délai de transfert d'un message dans le pire de cas est présentée dans [Tindell *et al.*, 1995a and 1995b]. La borne est calculée en utilisant l'analyse

d'ordonnançabilité proposé dans [Audsley *et al.*, 1993] et qui prédit le temps de réponse, dans le pire des cas, d'un ensemble de tâches périodiques et sporadiques qui se voient assigner une priorité et qui sont ordonnancées par un ordonnanceur préemptif et à priorités statiques.

Par ailleurs, une méthode analytique est présentée dans [Navet *et al.*, 2000] pour calculer la probabilité de non respect des échéances des messages dans le pire des cas.

D'autre côté, le protocole *CSMA à Résolution Déterministe des Collisions* (CSMA/DCR) est une variante déterministe de la norme Ethernet ou du protocole CSMA/CD. La variante consiste simplement à remplacer le mécanisme probabiliste de résolution de collisions par le mécanisme dit de l'arbre binaire dans le but de résoudre les collisions dans un temps borné. Le protocole utilise les adresses ou index des sites dans ce but. Les sites sont considérés comme des feuilles de l'arbre binaire, et le temps est divisé en tranches temporelles. Les sites, ayant un message à émettre, l'émettent dès que le canal est libre. Si une collision se produit, l'ensemble des sites est divisé de telle façon que seuls les sites d'une branche de l'arbre binaire peuvent ré-émettre dans la tranche temporelle suivante. S'il y a encore des collisions, le sous-ensemble de sites est divisé jusqu'à ce qu'il y n'ait qu'un seul site émetteur. Lorsqu'il n'y a plus de sites dans une branche, une tranche temporelle est détectée vide, alors l'autre moitié de l'arbre est explorée.

Etant donné que le temps de parcours de l'arbre binaire est borné, lorsque le mécanisme est implanté dans le protocole *CSMA/CD*, celui-ci satisfait au besoin d'un délai de transfert borné. Dans [Le Lann and Rivierre, 1994] une borne du délai de transfert est présentée en utilisant la technique de l'adversaire. Ce mécanisme est utilisé aussi par le protocole *DOD/CSMA/CD*, qui privilégie les messages selon leurs échéances, et par le protocole *CSMA/LDCR*, qui privilégie les messages selon leurs laxités.

Des bornes supérieures et inférieures du délai moyen de transfert des messages dans les protocoles à détection de collision basés sur l'arbre binaire sont présentées dans [Chung *et al.*, 1994].

3.2.2 Périodicité

La périodicité tient en compte le fait que la plupart des messages échangés entre les utilisateurs de l'application temps réel sont composés de données identifiées qui doivent être transmises périodiquement et ce généralement avant leur prochaine date de production. Le système de communication doit donc fournir le service pour transmettre le flot de messages périodiques parfois aussi appelé trafic synchrone. Cependant, dans ces applications il y a aussi du trafic apériodique ou asynchrone et en conséquence, le système de communication doit fournir aussi des services pour ce type de trafic.

Dans ce but, la couche MAC peut être vue comme l'ordonnanceur, c.-à-d. elle utilise une politique ou algorithme d'ordonnancement pour déterminer le message qui doit être émis à n'importe quel instant. Le problème de l'ordonnancement a été étudié de différentes façons et dans différents domaines. Notre intérêt ce n'est pas de rééditer de telles études mais d'analyser l'influence des algorithmes d'ordonnancement dans les protocoles MAC dans le but de satisfaire les besoins temporels de l'utilisateur.

Divers algorithmes d'ordonnancement des tâches, utilisés notamment dans les systèmes monoprocesseur, ont été proposés pour privilégier et ordonnancer la transmission d'un type de trafic, parmi eux les suivants : l'algorithme *rate monotonic*, l'algorithme *earliest deadline* et l'algorithme *least laxity*.

- L'algorithme *rate monotonic* [Liu and Layland, 1973] est un algorithme à priorités statiques et préemptif qui assigne des priorités aux tâches d'une façon inversement proportionnelle à la période des tâches.

On peut remarquer que l'algorithme est composé de deux parties [Audsley and Burns, 1990 ; Audsley *et al.*, 1993] : la première, le test d'ordonnancement (test suffisant mais non nécessaire [Liu and Layland, 1973] et le test suffisant et nécessaire [Lehoczky *et al.*, 1989]) décide si un ensemble de tâches est ordonnançable ; la seconde assigne un ordre prioritaire statique aux tâches.

Dans le cas des réseaux locaux, l'algorithme rate monotonic est utilisé pour décider si un ensemble des messages est transmissible (ordonnançable) [Klein *et al.*, 1994] et pour assigner des priorités aux messages synchrones, les messages de petite période se voient assigner une haute priorité [Lehoczky and Sha, 1986].

L'utilisation d'un serveur périodique a été proposée dans [Strosnider *et al.*, 1988 ; Strosnider and Marchok, 1989] pour traiter les messages asynchrones. La priorité du serveur est la plus élevée parmi les messages synchrones.

L'algorithme rate monotonic a été implanté dans le protocole Boucle à jeton en utilisant le mécanisme d'assignation des priorités aux messages [Strosnider *et al.*, 1988 ; Strosnider and Marchok, 1989]. Il a été également implanté dans les réseaux à bus en utilisant le mécanisme de compte à rebours binaire [Mok and Ward, 1979 ; Tanenbaum, 1988 ; Malcolm and Zhao, 1995]. Cependant, [Pleinevaux, 1992] considère que la synchronisation d'horloge est nécessaire entre les sites afin d'utiliser l'algorithme rate monotonic dans les applications temps réel réparties.

- L'algorithme *least laxity* [Sorenson, 1974 ; Dertouzos and Mok, 1989] est un algorithme à priorités dynamiques et préemptif qui assigne des priorités aux tâches d'une façon inversement proportionnelle à leurs laxités (la différence entre leur échéance et le temps restant de calcul). Cet algorithme est optimal pour un système uniprocesseur et multiprocesseur [Panwar *et al.*, 1988].

On utilise divers mécanismes pour implanter l'algorithme least laxity dans les protocoles MAC. En particulier sa mise en œuvre en utilisant le mécanisme d'assignation des priorités aux messages est examinée dans [Shin and Hou, 1990 ; Yao and Zhao, 1991], en utilisant le mécanisme à fenêtre temporelle est examinée dans [Kurose *et al.*, 1984 ; Kurose *et al.*, 1988 ; Malcolm *et al.*, 1990 ; Zhao *et al.*, 1990 ; Arvind *et al.*, 1991 ; Znati, 1991 ; *Lim et al.*, 1991], et en utilisant le mécanisme de temps virtuel est examinée dans [Zhao and Ramamritham, 1987].

- L'algorithme *earliest deadline first* [Liu and Layland, 1973] est un algorithme à priorités dynamiques et préemptif qui assigne des priorités aux tâches de façon inversement proportionnelle à leurs échéances.

On peut utiliser les mêmes mécanismes utilisés par l'algorithme least laxity pour une implantation de cet algorithme dans des réseaux locaux.

Dans [Lim *et al.*, 1991] cet algorithme a été proposé et les auteurs ont analysé ses performances dans les protocoles à jeton. Les auteurs concluent qu'il est important de prendre en compte l'overhead induit par la mise en œuvre des algorithmes d'ordonnancement dans la performance des protocoles MAC.

On peut remarquer que ces algorithmes ne peuvent pas être implantés exactement dans les protocoles MAC pour les raisons suivantes :

- Les algorithmes sont préemptifs, cette préemption peut être approchée en subdivisant les messages en petits paquets et en transmettant chaque paquet de façon non-préemptive.

- Quelques mécanismes utilisés ne sont pas exacts dans le but d'implanter l'arbitrage à priorité ce qui veut dire qu'ils ne peuvent toujours résoudre l'arbitrage à priorité et en conséquence de trouver le paquet de plus haute priorité dans le réseau (ex : window access et virtual-time).

- Dans le cas du mécanisme d'assignation des priorités aux messages, si le nombre de périodes ou d'échéances des messages est supérieur au nombre de niveaux disponibles de priorité, on doit donc assigner quelques messages de priorités différentes au même niveau de priorité, ce qui peut produire des inversions des priorités.

Ainsi, par rapport à la transmission de trafics périodiques et apériodiques, le service d'ordonnancement offert par la couche MAC peur être classifié de la manière suivante et il est résumé dans le Tableau 3.2.

3.2.2.1 L'ordonnancement statique

Dans ce cas, tous les besoins de l'utilisateur sont connus à l'avance et ils ne changent pas pendant l'opération du réseau. En conséquence, le test d'ordonnancement peut être fait hors-ligne, à la configuration du réseau.

On peut identifier trois techniques de base : la technique basée sur la table, celle basée sur les priorités et celle basée sur l'allocation périodique.

- Dans la *technique basée sur la table*, après avoir fait le test d'ordonnancement, une table est construite. La table est utilisée pendant l'opération du réseau par un ordonnanceur ou arbitre, qui l'exécute sur un site maître. Cette approche est utilisée pour contrôler le trafic périodique par WorldFIP, Foundation Fieldbus, Profibus entre stations maître et esclaves et P-Net.

- Dans la *technique basée sur les priorités*, le test d'ordonnancement et l'assignation de priorités aux messages sont exécutés hors-ligne. Pendant l'opération du réseau, les priorités déterminent le message à transmettre (le protocole CAN) ou le site ayant le droit d'émission (le protocole CSMA/DCR).

- La *technique d'allocation périodique* se sert de la méthode périodique d'accès au médium (le multiplexage temporel et quelques protocoles à passage du jeton) dans le but de transmettre d'abord le trafic périodique, comme c'est le cas dans FDDI, Interbus, TTP et ControlNet.

3.2.2.2 L'ordonnancement dynamique

Dans ce cas, on ne prend en compte les besoins de l'utilisateur que pendant l'opération du réseau. Le système s'occupe dynamiquement des demandes de service de communication au fur et à mesure qu'elles se présentent soit dans leur forme périodique soit apériodique. De cette façon le trafic est transmis selon son échéance ou sa laxité mais en oubliant sa nature périodique et apériodique. Nous dirons que dans ce cas le trafic est transmis indistinctement.

L'approche est très souple puisque n'importe quel changement des besoins de l'utilisateur peut être adapté, ou au moins considéré, en-ligne. Souvent, on ne fait pas de test d'ordonnancement hors-ligne, en conséquence, le service meilleur effort sert le trafic.

Fondamentalement, il existe un type d'ordonnancement dynamique basé sur des priorités dynamiques. La priorité peut être explicite ou implicite. La priorité explicite peut être encore divisée en priorité locale (Bus à jeton) et priorité globale (Boucle à jeton). La priorité implicite est mise en œuvre par des mécanismes tels que la fenêtre temporelle et le temps virtuel dans le but d'ordonnancer les messages en utilisant un algorithme à priorités dynamiques, comme c'est le cas dans le protocole CSMA/CD. Les priorités globales de la Boucle à jeton ont été utilisées également dans ce but.

On peut remarquer que le test d'ordonnancement en-ligne est nécessaire dans le but de garantir la transmission des messages à un instant précis et dans un délai spécifié. L'approche est connue sous le nom de modèle de réservation des ressources. Dans ce cas, à chaque fois qu'il y a une requête demandant l'utilisation d'une ressource, un module d'allocation de ressources est invoqué pour déterminer si le nouvel ensemble de besoins peut être satisfait et garanti. Si c'est le cas, la requête est traitée sinon, l'entité qui a fait la demande est informée de l'impossibilité de traiter sa requête. Cette démarche fait partie d'un concept plus général et qui est connu sur le nom de Qualité de Service, laquelle sera analysée dans le chapitre suivant.

3.2.3 Gigue

La gigue mesure la variation de la période dans la transmission périodique des messages, dans une optique temps réel, elle doit être confirmée entre une borne inférieure et supérieure. Ce besoin temporel de l'utilisateur ne peut être satisfait que par les protocoles fournissant une méthode d'assignation périodique du droit d'émission ou par ceux qui privilégient le transfert du trafic périodique (voir Tableau 3.3).

La gigue peut être également observée entre les différents niveaux de priorité du protocole Bus à jeton. En fait, la norme IEEE 802.4 permet d'assigner localement quatre niveaux de priorités aux messages. Chaque site possède un temporisateur qui permet de contrôler le temps maximal d'utilisation du jeton. Lorsque le site reçoit le jeton, il arme son temporisateur avec la durée maximale associée aux messages de plus haute priorité et commence son émission. Le site ne peut émettre les messages de priorité immédiatement inférieure que s'il lui reste encore suffisamment de temps pour le faire. De plus, lorsque le site passe d'une priorité à une autre, il doit réinitialiser son temporisateur avec la durée maximale associée à la nouvelle priorité. Pour permettre les transferts des quatre niveaux de priorité, il est donc nécessaire de choisir judicieusement les paramètres liés à la gestion des priorités. Néanmoins, ce choix incombe à la gestion du réseau et non à la couche MAC [Mammeri and Thomesse, 1991].

Ainsi, lorsque le site reçoit le jeton et qu'il n'a pas de messages de plus haute priorité en attente de transmission, le site peut émettre les messages de priorité inférieure jusqu'à ce que son temporisateur expire, au lieu de passer le jeton au site successeur, que lui peut avoir des messages plus prioritaires. En conséquence, si les priorités sont associées aux échéances des messages, lorsque la charge du réseau augmente, il peut y avoir des messages qui n'ont pas respecté leur échéance et qui ne sont pas des messages de priorité inférieure mais qui possèdent au contraire une priorité élevée [Ng and Liu, 1991].

D'ailleurs dans les protocoles à jeton (FDDI, Bus à jeton, Boucle à jeton, Profibus et P-Net) la gigue peut être augmentée si le temps de passage du jeton n'est pas constant comme cela a été montré dans [Moon *et al.*, 1998].

3.2.4 Cohérence temporelle

La cohérence temporelle indique si un ensemble de messages a été émis et/ou reçus dans un même intervalle de temps spécifié [ISO 184/SC5/WG2 ; Saba *et al.*, 1993 ; Thomesse, 1999]. Typiquement, les protocoles MAC ne fournissent aucun mécanisme pour signaler que ni l'émission ni la réception a été faite dans un intervalle de temps donné.

Néanmoins, dans le but de satisfaire ce besoin, une machine d'états générale a été proposée dans [Lorenz *et al.*, 1994] permettant de générer un ensemble de statuts temporels. Ceux-ci informent sur la validité temporelle d'une certaine valeur lors des étapes de production, d'émission, de réception et de consommation. Afin de générer des statuts temporels, on a défini des fenêtres temporelles qui bornent la durée de l'intervalle [requête, confirmation] chez le client et de l'intervalle [indication, réponse] chez le serveur.

3.2.5 Cohérence spatiale

La cohérence temporelle signale si les copies d'une liste dans différents sites sont égales et ce, à un instant précis ou dans un intervalle de temps spécifié. Typiquement, les protocoles MAC ne fournissent aucun mécanisme pour le signaler. Néanmoins, au niveau de la couche application, quelques réseaux de terrain offrent cette sorte de signalisation, comme par exemple WorldFIP [Saba *et al.*, 1993].

3.3 Taille des données par transaction

Divers systèmes de communication coexistent dans les applications industrielles et sont organisés hiérarchiquement selon trois niveaux selon l'architecture CIM [Pimentel, 1990], à savoir, le niveau de commande des processus/machines, le niveau de commande des cellules et le niveau de commande de l'usine.

Le niveau plus haut, la commande de l'usine, facilite l'intégration des cellules dans l'application industrielle. Ce niveau supporte la surveillance et la gestion de l'usine. Dans ce niveau, il existe de longs flux de données partagées entre l'application distribuée, lesquels sont transportés par des réseaux de but général tels que CSMA/CD, Boucle à jeton et FDDI.

Le contrôle des processus et des machines est fait au niveau de la commande des cellules. Etant donné les caractéristiques temporelles et spatiales des données, les réseaux généralistes ne peuvent pas être utilisés. A leur place, des réseaux spécialisés connus sous le nom de réseaux de terrain sont utilisés, lesquels utilisent des protocoles simples avec un overhead réduit. Ces protocoles se servent de techniques d'ordonnancement bien adaptées afin de maintenir des services de communication qui satisfont certains besoins temporels de l'utilisateur (voir Tableaux 3.1, 3.2 et 3.3).

Au niveau le plus bas, près des dispositifs de terrain, c.-à-d. les capteurs, les actionneurs et les contrôleurs, on trouve la commande des processus/machines. La taille de données échangées entre ces utilisateurs est de quelques octets et on utilise un type de réseau de terrain appelé réseau de dispositifs, quelques exemples sont Profibus/DP et Device WorldFIP, qui sont des profils réduits de leur réseau de terrain respectif, en plus de DeviceNet et Smart Distributed System (SDS) qui sont basés sur CAN. A ce niveau de l'architecture CIM, il existe autre type de réseau de terrain qui est utilisé pour interconnecter des capteurs et des actionneurs et pour transporter des données d'une taille égale à quelques bits. Ils sont appelés par réseau de capteurs ou sous-réseau de terrain comme par exemple AS-Interface (AS-I) et Seriplex.

3.4 Conclusion

Ce chapitre présente une analyse des principaux protocoles MAC et des Réseaux de Terrain à partir de deux points de vue. L'analyse prend en compte les besoins des utilisateurs identifiés dans le chapitre 2. D'autre part, l'analyse considère les caractéristiques des protocoles MAC identifiant les méthodes d'accès et les algorithmes d'ordonnancement qui satisfont aux besoins temporels des utilisateurs.

Le chapitre 3 présente des tableaux qui résument les besoins satisfaits par divers protocoles MAC. Nous pouvons en tirer la conclusion suivante : quelques-uns des protocoles MAC utilisés par les réseaux de terrain satisfont à la plupart des besoins temporels des utilisateurs, tandis que peu des principaux protocoles MAC satisfont à tous ces besoins. Nous pouvons également noter que la plupart des réseaux de terrain se servent de méthodes qui tiennent compte des besoins des utilisateurs lors de l'étape de configuration de réseau afin de satisfaire et de garantir ces besoins pendant les opérations du réseau. Néanmoins, de nombreuses des applications industrielles ont besoin de méthodes en ligne.

Résumé du Chapitre 4

Qualité de Service dans les Systèmes de Communication

Les services de type meilleur effort et de type garanti sont deux classes de service qu'un système de communication peut fournir, et ils font partie d'un concept plus vaste connu sous le nom de qualité de service (QoS), c.-à-d. l'ensemble des qualités liées à la fourniture des services, tels qu'ils sont perçus par un client ou plus généralement par un utilisateur.

Plusieurs architectures de QoS ont été proposées pour différents systèmes de communication, un aperçu de ces architectures est présenté dans [Aurrecoechea *et al.*, 1998]. Ce chapitre analyse trois architectures de QoS : le cadre de l'OSI [ISO 13236], l'architecture des Réseaux de Terrain [CEI 61158] et l'architecture de l'Internet [Braden *et al.*, 1994 ; Blake *et al.*, 1998 ; Bernet *et al.*, 1999 ; Ghanwani *et al.*, 1999 ; Yavatkar *et al.*, 2000 ; Seaman *et al.*, 1999].

4.1 Le Cadre de l'OSI

Ce cadre [ISO 13236] définit la terminologie, les concepts et fournit un modèle de QoS pour l'OSI. Ce modèle est basé sur les concepts du Modèle de Référence de l'OSI et ceux du cadre de gestion de l'OSI.

Les rapports entre les concepts de QoS sont montrés dans la figure 4.1 et sont brièvement décrits ci-dessous :

- *Les Caractéristiques de QoS* sont les aspects quantifiables de QoS. Celles-ci peuvent être génériques, spécialisées ou dérivées. Une caractéristique générique est définie indépendamment de l'objet auquel elle est appliquée (par exemple, un délai). Une caractéristique spécialisée est une spécialisation d'une caractéristique générique (ex : le délai de bout en bout). Une caractéristique dérivée est définie en tant que fonction (mathématique) d'une caractéristique spécialisée (ex : le délai moyen de bout en bout).

- *Les Besoins de QoS* sont les besoins des utilisateurs qui peuvent être quantifiés. Les besoins de QoS peuvent être exprimés en termes de paramètres de QoS (quand ils doivent être échangés entre les entités de communication) ou en termes de contexte de QoS (quand ils sont retenus dans une entité de communication). Les paramètres de QoS sont transmis à toutes les entités impliquées dans le service afin de fournir la QoS de haut en bas et de bout en bout, selon le Modèle de Référence de l'OSI. Les entités qui reçoivent des besoins de QoS les analysent et déterminent les fonctions ou les mécanismes de gestion de QoS qui sont exigés pour les satisfaire.

- *Les Catégories de QoS* représentent la politique régissant un groupe de besoins de QoS spécifiques à un utilisateur ou à une application particulière.

- *Les Fonctions de gestion et mécanismes de QoS* sont toutes les activités liées à la surveillance, au contrôle et à l'administration de la QoS. Une Fonction de Gestion de QoS (QMF) est conçue pour répondre à un ou plusieurs besoins de QoS. Les mécanismes de QoS sont les composants d'une QMF et sont exécutés par une ou plusieurs entités afin de permettre d'établir, de surveiller, d'entretenir et de contrôler une QoS. Un mécanisme de QoS peut être un traitement local ou peut impliquer la génération d'autres besoins de QoS et les communiquer à d'autres entités.

  Les QMFs sont organisées en termes de trois phases d'activité de la gestion de la QoS :

  - *Phase de prévision* : elle a pour objectif d'établir le contexte de QoS en faisant les enquêtes appropriées et l'analyse pour prévoir les caractéristiques de QoS du système.

  - *Phase d'établissement* : pendant cette phase, les entités concernées expriment leurs besoins de QoS, entrent dans un processus de négociation et de renégociation, font des accords sur la QoS à être délivrée et sur les actions à exécuter en cas de dégradation et initialisent des mécanismes pour soutenir la phase opérationnelle.

    Trois niveaux d'accord différents peuvent être négociés pour n'importe quel besoin de QoS :

    - *Meilleur effort*, toutes les parties font leur meilleur pour répondre aux besoins des utilisateurs mais il n'y a aucune garantie. Dans ce niveau, certaines valeurs des paramètres de QoS peuvent ne pas être indiquées ou quelques limites dans la représentation déterministe ou statistique peuvent ne pas être satisfaites.

    - *Garanti*, les besoins de QoS sont satisfaits en termes des paramètres indiqués de QoS. Les besoins peuvent être exprimés en tant que besoins déterministes ou statistiques. Les valeurs déterministes peuvent être indiquées comme suit : une valeur simple, une limite supérieure ou inférieure, un seuil supérieur ou inférieur, une cible d'opération, etc. Les valeurs statistiques peuvent être indiquées comme la valeur maximale, minimale, moyenne, l'écart type, etc.

    - *Obligatoire*, dans ce niveau, la QoS atteinte doit être surveillée par le fournisseur du service et celui-ci doit avorter le service si le niveau dégrade au-dessous d'un seuil. La QoS désirée n'est pas garantie et peut être délibérément dégradée pour satisfaire un accord garanti.

  - *Phase opérationnelle*, son but est d'honorer les accords faits pendant la phase d'établissement et de prendre les mesures appropriées lorsque cela n'est plus possible. Dans cette phase, les entités exécutent la surveillance, l'entretien et le contrôle de la QoS.

4.1.1 L'architecture de QoS

Le modèle de QoS pour l'OSI (comme la figure 4.2 le montre, pp. 93) considère deux classes d'entités qui participent à la gestion de QoS : les entités spécifiques à la couche et les entités

au niveau du système. Les premières sont associées à l'opération d'un sous-système-(N), elles coordonnent la réponse aux besoins de l'utilisateur du service-(N) et mettent en œuvre le contrôle des entités protocolaires. Les entités au niveau du système interagissent avec les entités spécifiques à la couche afin de surveiller et de commander le fonctionnement du système.

Les entités de QoS spécifiques à la couche sont les suivantes : la *Fonction de Contrôle de la Politique–(N) ((N)-PCF)* qui détermine la politique appliquée à l'opération d'un sous-système-(N) et les contraintes sous lesquelles toutes les autres décisions du sous-système-(N) sont faites. Ainsi, (N)-PCF modélise toute action qui doit être effectuée pour commander l'opération du sous-système-(N). La *Fonction de Contrôle de QoS-(N) ((N)-QCF)* tient en compte des besoins de QoS et choisit les entités protocolaires (ex : exerçant une influence sur l'adressage et le routage) qui participeront à la communication. *L'Entité Protocolaire-(N) ((N)-PE)* est responsable de l'opération du protocole-(N) afin de fournir le service-(N) à l'utilisateur du service-(N). En particulier, (N)-PE est responsable de négocier la QoS avec l'entité protocolaire homogène-(N), l'utilisateur du service-(N) et le fournisseur du service-(N-1).

Les entités de QoS au niveau du système (*SQE*) sont les suivantes : les *Entités de Gestion du Système* (*PME*) sont employées pour permettre de contrôler, à distance ou localement, les ressources du système. Ces entités peuvent être considérées dans deux classes : celles qui fournissent l'infrastructure de gestion et celles qui emploient l'infrastructure de gestion pour accomplir des tâches particulières de gestion. La *Fonction de Contrôle de Qualité du Système* (*SQCF*) fournit deux capacités : la capacité au niveau du système afin de régler la performance des diverses entités protocolaires impliquées et la capacité de coordination de n'importe quel besoin afin de modifier le comportement des systèmes à distance par l'intermédiaire de la gestion du système de l'OSI. Le rôle de la *Fonction de Contrôle de la Politique du Système* (*SPCF*) est semblable au rôle de la (N)-PCF spécifique à une couche, son inclusion dans SQE tient compte du fait que la politique mise en œuvre dans n'importe quelle couche dépend d'une politique qui a été établie pour le système entier.

On peut remarquer que cette architecture est basée sur les rapports client-serveur entre les couches adjacentes et entre une couche et sa couche homologue selon le Modèle de Référence OSI.

Les besoins du client dépendent de la couche considérée. Ces besoins doivent être alors évalués et exprimés en termes d'un ensemble de besoins de QoS, qui à leur tour sont exprimées en tant que paramètres de QoS et contextes de QoS. Les besoins de QoS expriment l'information pour contrôler une ou plusieurs caractéristiques de QoS, c.-à-d., les aspects quantifiables de QoS.

Les entités qui reçoivent des besoins de QoS, les analysent et déterminent la fonction de gestion de QoS et les mécanismes qui sont requis pour les satisfaire. Ceci peut impliquer la production de nouveaux besoins de QoS, typiquement plus détaillés, et les acheminer vers d'autres entités en termes de paramètres de QoS. Si une entité donnée détermine qu'elle ne peut pas répondre au besoin de QoS, l'entité renvoie la demande et la responsabilité de l'action passe à nouveau à l'entité précédente dans le flux.

4.2 L'architecture de QoS des Réseaux de Terrain

Cette section analyse l'architecture de QoS du CEI 61158, appelé système de communications de données numériques pour la mesure et le contrôle – Réseaux de Terrain pour l'usage dans les systèmes de commande industriels [CEI 61158]. Cette norme internationale est fondamentalement une collection de normes des Réseaux de Terrain, dont certaines ont été analysées dans le chapitre 3, tels que TS 61158, ControlNet, Profibus, P-Net, Foundation Fieldbus, Swiftnet, WorldFIP et Interbus. Chacun est appelé Type de Service et est numéroté de 1 à 8, respectivement.

Les prochaines sections présentent les services de liaison de données (DLS), l'architecture de QoS de la partie 3 du CEI 61158, qui est basée sur l'[ISO/CEI 8886].

4.2.1 Service de Liaison de Données (DLS)

Fondamentalement le DLS fournit des services de transfert transparent et fiable des données entre les utilisateurs du DLS, c.-à-d.:

- *Transfert transparent* de données : le DLS assure le transfert transparent des données de l'utilisateur du DLS. Il n'impose aucune restriction quant au contenu, au format ou au codage des informations, et n'a même pas besoin d'interpréter leur structure ou leur signification.

- *Transfert fiable* de données : le DLS met les utilisateurs de ce service à l'abri des pertes, insertions, et, le cas échéant, d'altération de l'ordre des données.

- *Indépendance de la couche physique* : le DLS dégage les utilisateurs du service de toute préoccupation concernant la configuration disponible (par exemple, connexion point à point) ou les moyens physiques utilisés (par exemple, transmission semi-duplex).

4.2.2 L'architecture de QoS du CEI 61158-Partie 3

Cette architecture de QoS est basée principalement sur les classes de service que la couche liaison de données doit fournir et sur certains attributs de QoS communs entre les différents types de DLS, c.-à-d., entre les normes des Réseaux de terrain. L'utilisateur du DLS peut choisir, directement ou indirectement, quelques paramètres de ces attributs afin de déterminer la qualité des services de liaison de données. La Qualité de Service se réfère alors aux aspects qui sont sous le contrôle directe du fournisseur du DLS. La QoS peut être correctement déterminée lorsque le comportement de l'utilisateur du DLS ne contraint pas ou n'empêche pas l'exécution du DLS. Les sections suivantes présentent les classes de service et les attributs de QoS.

4.2.2.1 Classes de Service de Liaison de Données

Le Type 1 de service du CEI 61158-3 définit deux classes de DLS :

- Le service en mode connexion
- Le service en mode sans connexion

Dans la première classe, avant de transmettre des données, il est nécessaire d'établir un canal logique, appelé connexion; après avoir transmis des données, la connexion est libérée. Dans la deuxième classe, aucune connexion n'est nécessaire avant la transmission des données.

Le Type 1 de service du CEI 61158-3 définit également deux autres classes de service :

- Le choix de la QoS : le DLS offre aux utilisateurs la possibilité de demander ou d'accepter une certaine QoS pour le transfert de données. La QoS est spécifiée par des paramètres de QoS exprimant des caractéristiques communes, appelées attributs, telles que la priorité, le délai maximal de confirmation, l'authentification des DLPDU, la politique d'ordonnancement et la cohérence spatiale.

- L'adressage : le DLS permet à l'utilisateur de s'identifier et d'indiquer le DLSAP (DL-point d'accès au service) à destination duquel une DLC (DL-connexion) doit être établie, chaque fois que plusieurs DLSAP sont acceptés par le fournisseur du DLS. Les adresses de liaison de données n'ont qu'une signification locale à l'intérieur d'une configuration spécifique de liaison de données, sur un support de transmission unique.

Ces services sont complètement décris dans le Chapitre 4 sous les appellations de service de gestion des DL(SAP)-adresses, des files d'attente et des tampons et le service de gestion du temps et de l'ordonnancement

On peut remarquer que ces quatre classes de DLS sont fournies selon les types de DLS, c.-à-d., l'utilisateur du DLS est limité aux classes de service fournies par la mise en œuvre du protocole choisi.

4.2.2.2 Attributs de QoS communs aux divers Types de DLS

Les aspects qui peuvent être choisis par l'utilisateur du DLS sont appelés attributs et peuvent être classifiés de la manière suivante : statiques, dynamiques et semi-statiques. Les attributs statiques de QoS sont choisis une fois pour toutes et restent inchangés pendant toute la durée de vie du service. Les attributs dynamiques de QoS sont choisis à chaque invocation de DLS. Les attributs semi-statiques de QoS sont des attributs statiques pour un type de DLS et servent de valeurs par défaut pour certains attributs dynamiques dans un autre type de DLS.

Les attributs définis par la norme sont les suivants : la priorité, le délai maximal de confirmation, l'authentification de DLPDU, la politique d'ordonnancement et la cohérence spatiale. Les quatre premiers attributs de QoS s'appliquent aux services en mode connexion et sans connexion. Le cinquième attribut ne s'applique qu'au service en mode connexion et est dynamique pour chaque DLCEP (DL-point final de connexion). Ces attributs sont résumés dans les sous-sections suivantes.

4.2.2.2.1 Priorité (attribut dynamique de QoS)

La priorité détermine la quantité maximale de données qui peut être transportée dans un DLPDU simple. Trois niveaux de priorité sont définis : urgent, normal et temps disponible, en transportant 64, 128 et 256 octets par DLPDU, respectivement. Les deux premiers sont considérés comme des niveaux de temps critique.

4.2.2.2.2 Délai maximal de confirmation (attribut dynamique de QoS)

Chaque demande d'établissement de DLCEP, et chaque réponse, indique des limites supérieures sur la durée maximale de temps autorisée pour l'achèvement de l'exécution d'une succession de primitives de connexion.

Chaque demande de service sans connexion indique une limite supérieure sur la durée maximale de temps autorisée pour l'achèvement d'une succession de primitives sans connexion.

4.2.2.2.3 Authentification du DLPDU (attribut semi-statique de QoS)

Chaque négociation d'établissement de DLCEP, et chaque transfert de données sans connexion, emploie cet attribut pour déterminer, entre autres, une limite inférieure sur la quantité d'information d'adressage utilisée dans les DLPDUs.

Trois niveaux peuvent être spécifiés : ordinaire - chaque DLPDU doit inclure la quantité minimale d'adressage, source - chaque DLPDU doit inclure une adresse source, si possible, maximal - chaque DL-adresse doit inclure la quantité maximale d'information d'adressage possible.

4.2.2.2.4 Politique d'ordonnancement (attribut semi-statique de QoS)

Cet attribut est statique pour le service sans connexion, mais est dynamique pour le service en mode connexion. Pour chacune des DLSAP-adresses, et chaque DLCEP, l'utilisateur du DLS peut opérer manuellement la politique d'ordonnancement normale (implicite) de DLL, et à sa place peut différer n'importe quelle communication demandée entre les utilisateurs du DLS par le biais de certaines primitives de service, jusqu'à ce que cet ajournement soit annulé par l'utilisateur concerné.

Les deux choix sont les suivants : implicite - toutes les communications demandées se produiront aussi tôt que possible, explicite - toutes les communications demandées se produiront seulement quand l'ajournement est explicitement annulé par l'utilisateur du DLS concerné.

4.2.2.2.5 Cohérence temporelle (attribut DLCEP dynamique de QoS)

Cet attribut s'applique seulement aux tampons de rétention, aux DLCEPs pour lesquels les tampons sont de capacités limitées, et aux primitives de service qui transfèrent des données de

l'utilisateur du DLS. Quatre types de cohérence sont définis: résidence, mise à jour, synchronisé et transparente.

La résidence est une évaluation basée sur l'intervalle de temps entre l'instant d'écriture du tampon et l'instant de sa lecture.

La mise à jour est une évaluation basée sur l'intervalle de temps entre l'instant d'occurrence d'un événement de synchronisation multi-DLE et le moment où le tampon est écrit.

La cohérence synchronisée est une évaluation basée sur les intervalles de temps et les rapports entre les instants suivants : le moment de l'occurrence d'un événement de synchronisation multi-DLE, le moment où le tampon est écrit et le moment où le tampon est lu.

La cohérence transparente se produit quand la cohérence est choisie sur un DLCEP mais aucune des évaluations ci-dessus n'est exécutée. Dans ce cas la DLC préserve n'importe quelle cohérence antérieure, mais elle-même n'invalide pas la cohérence. Quand aucune cohérence antérieure n'existe, la valeur par défaut de cohérence est vraie.

La non-cohérence se produit quand la cohérence n'est pas choisie sur un DLCEP. Dans ce cas l'attribut de cohérence est toujours faux.

4.3 L'architecture de QoS d'Internet

Le système de communication d'Internet ou le modèle de TCP/IP se compose fondamentalement des services fournis par les couches application, transport et réseau. De nos jours, cette architecture ne fournit que le service meilleur effort et c'est pourquoi le groupe de travail Internet (IETF) a proposé divers modèles de service afin de satisfaire la demande de QoS. Parmi eux, on trouve le modèle des services intégrés (IntServ) [Braden *et al.*, 1994] et le modèle des services différenciés (DiffServ) [Blake *et al.*,1998 ; Bernet *et al.*,1999].

Pour les Réseaux Locaux de type IEEE 802, le modèle de gestion de la bande passante du sous-réseau (SBM) a été proposé dans [Ghanwani *et al.*,1999 ; Yavatkar *et al.*, 2000 ; Seaman *et al.*, 1999] mais la contrainte principale est que tout le trafic doit passer par au moins un commutateur utilisant un protocole basé sur des priorités tel que l'IEEE 802.1p, [IEEE 802.1Q] ou [IEEE 802.1D]. De cette façon, les services intégrés peuvent être soutenus par les réseaux locaux en utilisant les protocoles MAC de l'IEEE 802 tels que le 802.3 et le 802.5. Dans le cas du protocole 802.3, un champ explicite de priorité sur le format de base de la trame MAC est alors requis. Les modèles de service d'Internet sont analysés dans les sous-sections suivantes et sont montrés dans la figure 4.6 (pp. 117)

4.3.1 Modèle de Services Intégrés

Le modèle d'IntServ suppose que les ressources du système de communication doivent être explicitement contrôlées afin de satisfaire aux besoins de l'application. Etant donné que les ressources sont finies, la réservation de ressources et le contrôle d'admission sont des modules

principaux de ce service. Deux autres modules sont requis : l'ordonnanceur de paquets et le classificateur.

L'application doit indiquer la QoS désirée en utilisant une liste de paramètres, appelée le « flowspec », qui est transmise par un protocole de réservation, passée vers le contrôle d'admission afin de tester et d'accepter les paramètres, et qui est finalement utilisée pour établir les paramètres du mécanisme d'ordonnancement des paquets.

Ainsi, le protocole de réservation de ressource crée et maintient l'état d'un flux spécifique dans les serveurs de bout en bout et dans des routeurs tout au long du chemin du flux. Plusieurs protocoles ont été proposés, par exemple [Ferrari and Verma, 1990 ; Topolcic, 1990 ; Zhang *et al.*, 1993], la sous-section 4.3.1.1 analyse le protocole proposé dans [Zhang *et al.*, 1993].

Le contrôle d'admission met en œuvre l'algorithme de décision qu'un routeur ou un serveur emploie pour déterminer s'il peut accorder la QoS demandée par le nouveau flux sans mettre en danger les flux précédemment acceptés et garantis. L'ordonnanceur de paquets contrôle l'expédition des différents flux de paquet en utilisant un ensemble de files d'attente. Le classificateur classifie chaque paquet entrant dans une certaine classe; tous les paquets dans la même classe obtiennent le même traitement de l'ordonnanceur de paquets.

On peut remarquer que ce modèle de service se situe au niveau de la couche transport de l'architecture d'Internet. Le modèle utilise des services fournis par la couche réseau pour réserver des ressources tout au long du chemin des paquets.

D'ailleurs, le modèle IntServ spécifie deux niveaux différents de QoS :

- *Garanti* : il fournit des limites (mathématiquement prouvables) sur les délais de bout en bout dans les files d'attente en combinant les paramètres des divers éléments du réseau dans un chemin, en plus d'assurer la disponibilité de bande passante selon les spécifications des trafics.

- *Charge contrôlée* : c'est l'équivalent du service meilleur effort sous une condition de faible charge sur le réseau. Il est par conséquent meilleur que le service meilleur effort classique qui ne précise rien sur la charge réseau.

4.3.2 Modèle de Services Différenciés

Dans ce modèle [Blake *et al.*,1998], les clients demandent un niveau de performance spécifique sur la base d'un paquet, en marquant le champ DS de l'en-tête de chaque paquet avec une valeur spécifique. Cette valeur indique le comportement par-saut (PHB) à être assigné au paquet dans le réseau. En marquant les demandes différemment et en manipulant les demandes à partir de ces marques, diverses classes différenciées de service peuvent être créées. Par conséquent, cette approche est fondamentalement un schéma de priorité relative.

Il y a de deux comportements par-saut, qui représentent deux niveaux de service (classes de trafic) :

- *Transfert expéditif* : il réduit au minimum le délai et la gigue et fournit le niveau le plus élevé de qualité de service. N'importe quel trafic qui excède le profil du trafic, défini par une politique locale, est rejeté.

- *Transfert assuré* : il délivre le trafic dans le profil avec une probabilité élevée, le trafic excessif peut être dégradé mais il n'est pas nécessairement rejeté.

On peut remarquer que les services différenciés sont sensiblement différents des services intégrés [Xiao et Ni, 1999]. D'abord, il existe seulement un nombre limité de classes de service indiquées par le champ de DS, lequel, dans le cas de l'IPv4, indique le besoin d'un service avec un délai peu élevé, une capacité de traitement élevée ou un taux de perte bas. Puisque le service est assigné à une classe, la quantité d'information d'état est proportionnelle au nombre de classes plutôt qu'au nombre de flux. En conséquence, DiffServ est extensible. De plus, les opérations de classification, marquage, surveillance et format sont seulement nécessaires à la frontière du réseau. Par conséquent, il est plus facile de mettre en œuvre et de déployer ce modèle.

4.3.3 L'architecture SBM

Fondamentalement, le modèle de gestion de la bande passante du sous-réseau est un protocole de signalisation [Yavatkar *et al.*,1999] qui permet de supporter et d'aménager les services intégrés sur les réseaux locaux partagés et commutés de [Ghanwani *et al.*,1999 ; Seaman *et al.*, 1999]. Les composants principaux de l'architecture de SBM sont les suivants :

- *Le Distributeur de bande passante (BA)* qui maintient des informations d'état sur la répartition des ressources dans le sous-réseau et effectue le contrôle d'admission. Le BA peut être centralisé ou distribué.

- *Le Module demandeur (RM)* qui réside dans chaque station d'extrémité (serveur ou routeur) et non dans les commutateurs ou ponts. Ce module est responsable de la conversion des paramètres de QoS des couches supérieures dans les niveaux de priorité de couche liaison de données.

- *Le Protocole SBM* qui est le protocole de signalisation de RM à BA ou de BA à BA afin de réserver les ressources, de questionner le BA au sujet des ressources disponibles, et de changer ou supprimer des réservations. Ce protocole permet également la communication entre les protocoles de QoS des couches supérieures et le RM.

4.4 Conclusion

Ce chapitre a présenté trois architectures de QoS à savoir : celle de l'OSI, celle de la CEI 61158 et celle de l'internet. A partir de cela plusieurs remarques peuvent être faites :

Le cadre de QoS de l'OSI définit une architecture générale basée sur les interactions entre les couches adjacentes et entre une couche et sa couche homologue. La qualité d'un service donné

est déterminée par l'information de contrôle d'une ou plusieurs caractéristiques de QoS, c.-à-d. les besoins de QoS.

L'architecture de QoS du CEI 61158, basée sur l'ISO/CEI 8886, est bien adaptée au Type de Service 1, c.-à-d., le Réseau de terrain TS 61158, simplement parce qu'il définit les quatre classes de DLS et tous les paramètres de QoS qui ont été définis comme attributs communs pour tous les Réseaux de Terrain. Néanmoins, le chapitre 4 montre que ces attributs ne sont définis que par le TS 61158.

Dans les réseaux de terrain, les utilisateurs du DLS sont habituellement les entités de couche application ou, dans certains cas, les entités d'une interface entre la couche de liaison de données et la couche d'application. Ainsi, le concept de QoS est propagé de bas en haut, c.-à-d. de la couche de liaison de données jusqu'à l'utilisateur de l'application. D'ailleurs, le fournisseur de DLS est distribué, c.-à-d., les entités de DLS peuvent être des éditeurs ou des abonnés dans les divers sites, en conséquence le modèle fournit la QoS de bout en bout.

Il est intéressant de noter que les paramètres de QoS, définis par le CEI 61158, tiennent compte de certains besoins temporels des utilisateurs comme ceux qui ont été analysés dans le chapitre 2, c.-à-d. le délai lié au transfert des messages, la différentiation entre les messages de temps-critique et non-temps-critique, la cohérence spatiale et temporelle. Néanmoins, les besoins de périodicité ne sont pas pris en compte explicitement dans cette norme, il est du ressort du concepteur de construire explicitement l'ordonnancement des messages selon la stratégie qui lui convient.

D'autre part, dans l'architecture de QoS de l'internet, le modèle dit des services intégrés (IntServ) par du principe que les services garantis ne peuvent pas être fournis sans réservation de ressources en conséquence un protocole de réservation de ressources est requis, tel que RSVP. Le modèle des services différenciés (DiffServ) utilise le fait que les paquets peuvent être différemment marqués et alors classifiés ; les paquets dans les différentes classes de trafic reçoivent différents niveaux de service.

Par ailleurs, le modèle IntServ peut être appliqué dans les réseaux locaux en utilisant l'approche de gestion de la bande passante du sous-réseau (SBM), c.-à-d. si tout le trafic traverse au moins un commutateur ou pont en utilisant un protocole basé sur des priorités tel que l'IEEE 802.1p, 802.1Q ou 802.1D.

On peut remarquer que l'architecture de QoS de l'Internet, de haut en bas, et l'architecture de QoS du CEI 61158, de bas en haut, pourraient converger au niveau de la couche liaison de données si on utilise le commutateur basé sur des priorités de l'approche SBM et l'attribut de priorité de l'approche du CEI.

Cependant, dans les chapitres 3 et 4 nous avons vu que l'attribut de priorité au niveau de la couche liaison de données n'est pas défini par tous les Réseaux de terrain. En conséquence, on a besoin de nouvelles solutions de QoS et celles-ci doivent être des solutions dynamiques.

Résumé du Chapitre 5

La Qualité de Service et les Modèles Client-Serveur

Les chapitres précédents ont montré que la plupart des réseaux de terrain fournissent certains niveaux garantis de QoS pour le trafic périodique, fondamentalement parce qu'on le connaît à l'avance et que toutes les ressources nécessaires peuvent donc être réservées pendant la configuration du réseau pour garantir des délais bornés de transfert des messages et l'ordonnancement des messages pendant l'exécution de l'application. Le chapitre 4 a également montré que certains réseaux fournissent certaines primitives de service pour demander dynamiquement un niveau de QoS.

Nous croyons que ces primitives de service peuvent être généralisées et intégrées dans une architecture de QoS afin de fournir des niveaux garantis de QoS. Spécifiquement, nous proposons d'utiliser l'approche de réservation des ressources pendant l'exécution de l'application qui est l'approche utilisée par l'ISO et par le modèle des services intégrés de l'Internet.

L'objectif de ce chapitre est maintenant de définir une architecture de QoS pour le modèle client-serveur basé sur l'approche de réservation des ressources afin de fournir dynamiquement divers niveaux de QoS. Nous avons choisi le modèle client-serveur parce qu'il permet d'établir les rapports entre les couches du modèle de référence OSI et parce qu'à l'heure actuelle il fournit seulement un service de niveau meilleur effort de QoS.

Ainsi, dans le chapitre 5 nous présentons une architecture de QoS pour le modèle client-serveur et quelques extensions de ce modèle pour le multiclient-serveur et le client-multiserveur. Nous proposons un Protocole de RéSerVation des Ressources pour le modèle client-serveur (CS-RSVP) qui est spécifié dans le Langage de Spécification et Description (SDL) et validé à l'aide de l'outil ObjectGEODE.

5.1 Introduction

Les modèles de coopération et de communication ont été proposés afin d'établir les rapports entre des entités paires. Ainsi, par exemple, la notion de client et de serveur est appliqué à différentes paires d'entités : utilisateurs, applications, processus, couches, etc.

Le modèle client-serveur est utilisé dans le modèle de référence OSI afin d'établir les interactions entre les couches adjacentes et entre une entité d'une couche et son homologue [Vega, 1996]. Néanmoins, le serveur ne fournit que des services de niveau meilleur effort, c.-à-d., le serveur effectue les opérations indiquées par la demande de service mais aucune garantie n'est donnée sur le temps de réponse, sauf si l'on fait des suppositions sur la disponibilité du serveur et des ressources [Thomesse, 1995].

Ce résumé présente l'architecture de QoS pour le modèle client-serveur et un protocole de réservation des ressources. Le chapitre 5 présente en détail l'analyse de besoins des utilisateurs, l'architecture de QoS, le protocole de réservation des ressources et les extensions pour les modèles multiclient-serveur et client-multiserveur. Les annexes A, B et C contiennent toutes les spécifications des modèles en SDL.

5.2 Modèle de QoS pour le Client-Serveur

Cette section présente un résumé du modèle de base de QoS, c.-à-d. un client et un serveur, basé sur la réservation des ressources : les ressources du serveur sont allouées aux demandes du client et ce sous une certaine politique de gestion des ressources. Avant de demander un service avec un certain niveau garanti de QoS, le client doit d'abord réserver les ressources requises dans le serveur. En conséquence, nous avons besoin d'un Protocole de RéSerVation des ressources entre le client et le serveur (CS-RSVP).

5.2.1 QoS Architecture de QoS pour le Client-Serveur

En plus des entités normales du modèle client-serveur, les composants suivants sont nécessaires (voir Fig. 5.3 pp. 129) :

- *Distributeur des ressources* (RA) qui maintient une structure d'état à propos de l'allocation des ressources dans le serveur et effectue le contrôle d'admission (CA).

- *Ordennanceur* (SC) qui ordonnance toutes les demandes de service acceptées vers l'entité du serveur.

- *Module demandeur* (RM) qui réside dans le client (RMC) et dans le serveur (RMS). Ce module est responsable de la négociation des paramètres de QoS à l'aide d'un protocole de réservation des ressources.

- *Protocole de RéSerVation des ressources entre le Client et le Serveur* (CS-RSVP), avant de demander un service avec un niveau de QoS donné, plusieurs étapes doivent être exécutées : (1) le client indique la QoS demandée, (2) les paramètres de QoS sont transmis au serveur, (3) les paramètres de QoS doivent être traduits en paramètres des ressources exigées, (4) les ressources demandées peuvent être admises, réservées et allouées ou rejetées et probablement négociées, (5) le processus d'arrêt concerne la libération des ressources.

5.2.2 Niveaux de QoS

Au moins deux niveaux différents de QoS sont nécessaires :

- Le *service garanti* : le serveur garantit que les besoins de QoS sont satisfaits. Les besoins peuvent être indiqués par des paramètres de QoS et exprimés en tant que valeurs déterministes ou statistiques [Ferrari, 1990 ; ISO 13236]. Les valeurs déterministes

peuvent être indiquées en termes d'une valeur simple, d'une limite supérieure ou inférieure, d'un seuil supérieur ou inférieur, d'un point d'opération, etc. Les valeurs statistiques peuvent être indiquées en termes d'une valeur maximale, minimale, d'un intervalle, d'une moyenne, d'un écart type, etc. Dans notre approche, ces paramètres de QoS peuvent ou non être négociés.

- Le *service meilleur effort* : le serveur ne donne aucune garantie de satisfaire les besoins de QoS, c.-à-d. des limites dans la représentation déterministe ou statistique des paramètres de QoS ne peuvent pas être satisfaites. Dans notre approche, les paramètres de QoS peuvent ou non être négociés.

5.3 Protocole de RéSerVation des Ressources

Cette section présente le résumé de la spécification du CS-RSVP en SDL et sa vérification en utilisant l'outil ObjectGEODE. CS-RSVP est spécifié en Langage de Spécification et de Description (SDL) qui est un langage orienté objets avec des concepts pour décrire les aspects logiques de structure, de données et de comportement des systèmes. SDL fournit une représentation graphique (SDL/GR) et une représentation textuelle (SDL/PR) et permet la traduction entre les deux. SDL est un langage compréhensible par les utilisateurs mais néanmoins assez formel pour soutenir l'analyse et la comparaison des comportements [Bræk, 1996].

Pour valider un modèle, on utilise le terme modèle de validation dans le monde SDL. C'est une description du système qui convient à la validation, c.-à-d. qui permet d'appliquer des techniques de validation formelle telles que le test du modèle, la validation approfondie (analyse d'accessibilité), la validation non-approfondie (analyse d'un sous-ensemble aléatoire d'états accessibles), la simulation et des techniques de validation informelles [Hogrefe, 1996].

Un modèle de validation est toujours exécutable, c.-à-d. le modèle doit être compilé et lié. Afin de construire un modèle de validation de CS-RSVP, nous avons utilisé l'outil ObjectGEODE v.4.0 de Verilog parce qu'il inclut les outils suivants : l'éditeur SDL, l'éditeur de diagrammes de séquence des messages (MSC), le vérificateur de SDL et de MSC, le simulateur interactif de SDL et de MSC, le simulateur approfondi de SDL et de MSC, le générateur de code de C et de C++, etc. [Cheng, 1996].

5.3.1.1 Définition des primitives de service CS-RSVP

Le CS-RSVP devrait fournir les primitives de service suivantes :

- *RSVcreate* est une demande de réservation de ressource. Les paramètres de QoS sont les suivants : le nom de la ressource, une variable booléenne pour indiquer si l'utilisation est périodique, si c'est le cas la valeur de la période, l'heure de départ, l'intervalle de temps de l'utilisation, l'intervalle de temps de validité de la réservation et une variable booléenne pour indiquer si la demande de réservation peut être négociée.

- *RSVaccept* est une réponse pour indiquer qu'une demande *RSVcreate* a été acceptée. Le paramètre de QoS est un identificateur de la demande acceptée.

- *RSVreject* est une réponse pour indiquer qu'une demande *RSVcreate* n'a pas été acceptée. Les paramètres de QoS sont les suivants :

  a) Si *RSVcreate* est négociable : le nom de la ressource, la variable booléenne indiquant l'utilisation périodique ou apériodique, la période que le serveur peut satisfaire, l'heure de départ que le serveur peut satisfaire, l'intervalle de temps d'utilisation de la ressource que le serveur peut satisfaire, l'intervalle de temps de validité de la réservation que le serveur peut satisfaire et la variable booléen indiquant que la demande de réservation ne peut plus être négociée.

  b) Si *RSVcreate* n'est pas négociable : les paramètres de QoS sont égaux au *RSVcreate*.

- *ServiceReq* est une demande d'utilisation d'une ressource précédemment réservée. Le paramètre de QoS est l'identificateur de la demande.

- *ServiceRsp* est une réponse pour indiquer que la demande *ServiceReq* a été effectuée. Le paramètre de QoS est l'identificateur de la demande.

- *NorequestID* est une réponse pour indiquer que l'identificateur de *ServiceReq* n'existe pas. Le paramètre de QoS est l'identificateur de la demande.

- *RSVmodify* est une demande de modification d'une réservation précédemment acceptée. Le paramètre de QoS est l'identificateur de la demande. Cette primitive de service n'est pas spécifiée.

- *RSVdeallocate* est une demande de désallocation d'une ressource précédemment réservée. Le paramètre de QoS est l'identificateur de la demande. Cette primitive de service n'est pas spécifiée.

5.3.1.2 Spécification des primitives de service CS-RSVP

Dans SDL, le comportement est toujours effectué dans le contexte d'un *système*, commençant par une description au niveau supérieur du système. La Fig. 5.5 (pp. 136) montre le système du CS-RSVP, qui est composé de trois *blocs* (client, serveur, et ServiceProvider) reliés par des *canaux* appelés c1 et c2. De plus, la figure montre les *signaux* qui passent dans chacune des directions au-dessus des canaux, comme les flèches sur les canaux l'indiquent. La figure montre également la déclaration de ces signaux ainsi que la déclaration de la structure de données (RSV) qui définit les paramètres de QoS utilisés par les primitives de service.

Dans SDL, un système est seulement défini au niveau supérieur, alors que les blocs sont éléments constitutifs d'un système. Le système est décomposé en blocs et canaux en autant de niveaux que nécessaires jusqu'à ce que les composants de base, les *processus*, soient atteints.

Les schémas 5.6 et 5.7 (pp. 137) présentent les blocs du client et du serveur, nous supposons que le bloc *ServiceProvider* représente un sous-système (N-1) de transfert fiable de données.

Un *processus* de SDL est un objet simultané avec son propre flux de contrôle, décrit par une machine d'état fini communicant (FSM), qui se compose de quatre parties principales : le port d'entrée, la FSM, les temporisateurs et les variables.

Le port d'entrée contient une file d'attente de taille illimitée pour les signaux entrants. Les signaux arrivant au processus seront mis dans le port d'entrée dans l'ordre d'arrivée, les conflits sont résolus en choisissant un ordre séquentiel arbitraire. Les signaux resteront dans le port d'entrée jusqu'à ce qu'ils soient consommés par la FSM, qui exécute une transition d'un état à un autre état pour chaque signal consommé. S'il n'y a aucun signal dans le port d'entrée, la FSM reste dans le même état jusqu'à ce qu'un signal arrive. Sur chaque transition, la FSM peut produire des signaux de sortie et peut effectuer des opérations sur les variables et les temporisateurs. Ce comportement de type état-transition de la FSM est exprimé sous la forme d'un *diagramme de processus.*

Le bloc du client, montré dans la Fig. 5.6, se compose de deux processus, le *ClientEntity* et le module demandeur du client (RMC), le premier est décrit dans le diagramme de processus de la Fig. 5.8 (pp. 138) et le deuxième dans l'annexe A.

Le bloc du serveur, montré dans la Fig. 5.7 se compose de quatre processus : le module demandeur du serveur (RMS), le distributeur des ressources (RA), l'ordonnanceur et l'entité du serveur (ServerEntity). Le processus distributeur des ressources est représenté sur la Fig. 5.9 (pp. 139). Les diagrammes des processus de tous les autres processus sont présentés dans l'annexe A.

Afin de simuler le CS-RSVP, deux procédures sont définies dans la Fig. 5.8, la procédure *GenerateRSV* et la procédure *GenerateRq*, qui sont aléatoirement exécutées par le symbole de décision étiqueté *any*. Après avoir appelé la procédure, la FSM produit le signal de sortie *RSVcreate* ou le signal *ServiceReq*, ayant comme paramètres la structure de réservation de ressource (d1) ou l'identificateur de la demande (Id1), respectivement. Après cela, la FSM attend un signal d'entrée, qui peut être soit *accepted* ou *rejected* ou soit *ServiceRsp* ou *NorequestID*. Les deux premiers sont des réponses à la demande de service *RSVcreate*. Les deux derniers sont des réponses à la demande de service *ServiceReq*.

Il faut noter que le signal d'entrée *accepted* possède deux paramètres, Id2 et d2, le premier est l'identificateur de la demande (requestID), la seconde est la structure de réservation (RSV) qui a été acceptée par le processus distributeur de ressource dans le serveur. Notons également que s'il existe un processus de négociation, les valeurs de cette structure peuvent être différentes des valeurs de la structure de réservation originelle du signal de sortie RSVcreate(d1). D'autre part, l'identificateur (requestID) devrait être stocké pour de futures demandes de service, cela n'est pas montré sur la Fig. 5.8, mais dans le but de la simulation il est aléatoirement produit par la procédure *GenerateRq*.

Le signal d'entrée *rejected* contient une structure de réservation (d2) avec les valeurs des paramètres de QoS que le processus Distributeur des ressources (RA) n'a pas pu satisfaire. Le signal d'entrée *ServiceRsp* contient l'identificateur de la demande (requestID (Id3)) envoyé

par le signal de sortie *ServiceReq*(Id1), ce signal indique que le service a été exécuté selon la QoS requise. Le signal d'entrée *NorequestID* indique que l'identificateur *requestID* envoyé par le signal *ServiceReq* n'existe pas dans le serveur.

La Fig. 5.9 montre le processus Distributeur des ressources (RA) et la déclaration de la structure de données pour les intervalles de temps réservés, nous supposons qu'il y a une seule ressource, appelée A1. Le processus RA fait appel à la procédure contrôle d'admission (CA), qui réalise le test temporel et le test d'ordonnancement pour déterminer si les paramètres de QoS peuvent être satisfaits.

Ce processus renvoie une variable booléenne, appelée *oky*. Si elle est vraie, les paramètres de QoS sont stockés et le processus RA envoie le signal de sortie *RSVaccept*, ayant comme paramètre l'identificateur de la demande pour des demandes futures de service. Autrement, si la variable de négociation (d2!neg) est vraie, le processus RA change le paramètre de l'heure de départ et envoie le signal de sortie *reject*, ayant comme paramètre la structure de réservation (d2). D'autre part, si un signal de demande de service est reçu, le processus RA détermine si l'identificateur de demande existe, si c'est le cas, il obtient les paramètres de QoS à partir de la structure de données pour les intervalles de temps réservés, et envoie le signal de sortie *RSVrequest* au processus ordonnanceur du serveur. Autrement, il envoie le signal de sortie *norequestID*.

5.4 Conclusion

Ce chapitre présente une architecture de QoS pour les modèles de communication client-serveur, multiclient-serveur et client-multiserveur. Cette architecture est basée sur l'approche de réservation des ressources utilisée dans le cadre de QoS de l'OSI et le modèle des services intégrés de l'Internet. Ce chapitre également présente un Protocole de RéSerVation des Ressources pour le Client-Server (CS-RSVP), lequel est spécifié en Langage de Spécification et Description (SDL) et validé à l'aide de l'outil ObjectGEODE v.4.0 de Verilog. Toutes les spécifications sont présentées dans les annexes A, B et C.

Le CS-RSVP permet de réserver l'utilisation des ressources par un processus de négociation entre le(s) client(s) et le(s) serveur(s). Le premier envoie un message de réservation (*RSVcreate*) contenant les paramètres de QoS suivants : le nom de la ressource, une variable booléenne pour indiquer si l'utilisation de la ressource est périodique ou pas, si c'est le cas la valeur de la période, l'heure de départ, l'intervalle de temps d'utilisation de la ressource, l'intervalle de temps de validité de la réservation et une variable booléenne pour indiquer si la demande de réservation peut être négociée ou pas.

Dans le serveur, après des tests d'admission, le distributeur des ressources (RA) répond par un message d'acceptation ou par un message de rejet. Le premier contient l'identificateur de la demande que le client emploiera pour de futures demandes de service. Le message de rejet contient les mêmes paramètres de QoS du message de réservation, mais leurs valeurs sont celles que le processus RA peut satisfaire, si la demande de réservation est négociable.

En ce qui concerne les réseaux de terrain, CS-RSVP peut être intégré dans l'approche centralisée pendant la configuration du réseau et pendant l'exécution de l'application.

Chapter 2

User Time-related Requirements Analysis

Typically, the real time systems have been only characterized as the systems that produce their results within specified time intervals. In fact, this is the main user temporal or time-related requirement, nevertheless this is not the only one, and this chapter will show it.

In general terms, the user requirements analysis allows to obtain a good insight and a deep understanding of all aspects of a given domain. Specifically, this chapter presents an user requirements analysis focused on the time-related requirements in order to establish the time-related service characteristics that a communication system should provide in the real-time systems.

The user requirements that are taken into account in this analysis are those of a special real time system used in the industrial applications where the users are the people who are in charge of the control, of the maintenance, of the production scheduling, of the technical management and so on. But also the sensors, the actuators, the production machines controllers, the transport systems, etc.

Examples of such industrial applications are as follows: automotive industry, textile machinery, factory automation, semiconductor fabrication (chip-manufacturing), electronics manufacturing, food and beverage, chemical processing, and so on.

In what concern the communication requirements, all the users have different requirements. In order to analyze them, the user requirements will be grouped into the time-related requirements and the space-related requirements. In the first group, the data exchanged among the users are characterized by the criticality, the transmission frequency, the jitter, the temporal coherence and the spatial coherence. In the second group, the data are characterized by their size.

This chapter is structured as follows: Section 2.1 presents the introduction; Section 2.2 presents the analysis of the user time-related requirements; Section 2.3 presents the user space-related requirements. Finally, some conclusions are presented.

2.1 Introduction

Industrial applications are more and more distributed and integrated, not only inside the factory itself but also between remote sites of production, of management, of product development and of product distribution. These applications are then composed of a lot of software pieces which have to cooperate, to be coordinated in order to provide the quality of service required by the different end-users. Various communication networks are then the support of this cooperation and become now the backbone of the industrial applications.

Communication systems must provide different services according to the different needs, and an important point of comparison is their capability to satisfy the different user requirements, essentially the time-related requirements.

To treat these aspects this chapter will analyze the end-users requirements and in the next chapter the main solutions will be studied considering the real-time communication networks which are essentially known as fieldbus networks.

2.2 User time-related requirements

Next paragraphs analyze the time-related requirements such as message transfer delay, periodicity, jitter, temporal and spatial coherence.

2.2.1 Message transfer delay

Typically, the real time applications interact with environments, known as the controlled object, whose current state is monitored by means of a set of data. The dynamics of the environment produces some data that have a limited temporal validity (lifetime), as it is shown in Fig. 2.1. Hence, in order to effectively act on the current state of the environment, all data processing (basically, transmission and computing) must be carried out within bounded time intervals or windows, which should be less than or equal to the data lifetime. How short such time window must be depends on the particular environment under consideration. Examples of such data are the feedback and control signals of closed loop control subsystems such as the axis position control in a robot arm or the temperature control in a chemical reaction [Almeida *et al.*, 1999b].

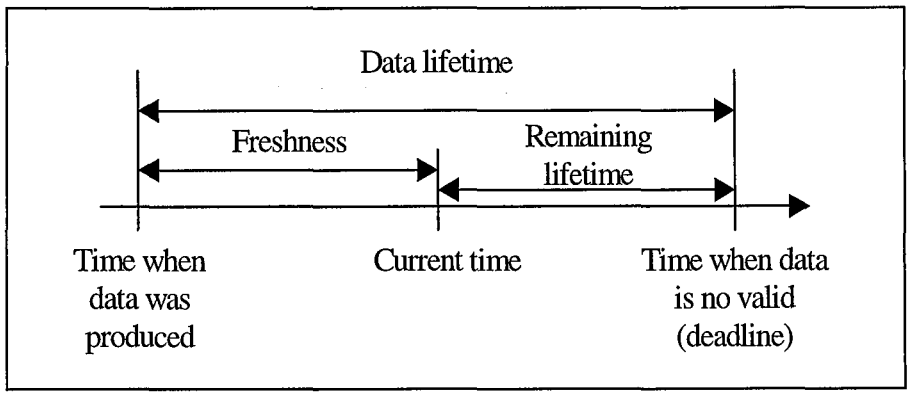

Fig. 2.1. Data lifetime, freshness and remaining lifetime characteristics.

These data are called *critical data* and can be further characterized as *soft* or *hard*. In the former case, a few data processing out of their time window can be tolerated. On the other hand, in the latter case, the environment cannot tolerate any delay in the data processing beyond the end of their lifetime, usually called *deadline*. This is the case when missing a single deadline may jeopardize the whole environment operation and even endanger people or equipment.

Besides critical data, the environment produces data whose processing time has no effect on its proper operation. These data are called *non-critical* and are normally handled by background application tasks.

In brief, with regard to these types of data, the user requirements can be summarized as follows:

- Distinction between critical and non-critical traffic is essential. The communication system should convey the data from source to destination in either type of traffic [ISO 184/SC5/WG2].

- The communication system should guarantee that all hard-critical data transfers will be completed within a bounded time interval.

- The communication system should try its best effort to transfer the soft-critical data, since minimizing the number of delayed or lost transfers will result in a proper operation of the environment in the long term, for example.

- Non-critical data transfers shall not jeopardize the delivery of the critical data.

2.2.2 Periodicity

Generally, the real time applications know the current status of their environment by means of the data produced and transmitted both periodically and randomly from the sensors or higher level devices. In the same way, the applications are in control of the environment by means of the control data transmitted to the actuators or control devices. In any case, when the transmission of such data is carried out at fixed time intervals they are called *periodic data*, otherwise *aperiodic data*. These data can be either critical or non-critical according to the previous discussion.

In some cases, it is possible to bound the minimum separation time between two consecutive aperiodic transactions of the same source, this particular sort of aperiodic data are normally known as *sporadic*.

So, in addition to the previous requirements, the user requirements with regard to these types of data can be summarized as follows:

- Distinction between periodic and aperiodic traffic. The communication system should convey the data from source to destination in either type of traffic.

- It should also be possible to distinguish the relative urgency of data, basically between periodic and aperiodic hard-critical data.

It can be noted that most of the application messages contain periodic data that must be transmitted before the beginning of their next period. Hence, *periodicity* requirement means

that the communication system must provide the service to transmit such data before the beginning of their next period.

Note also that the main difference between the periodic and aperiodic messages is the instant at which their time characteristics are known by the communication system. The time characteristics of the periodic messages may be known in advance, at network configuration time. The time characteristics of the sporadic messages are known till the message arrives at the communication system, at network operation time.

The previous classification of the data according to both their criticality and their transmission frequency is depicted in the figure 2.2. The shadow area represents the critical data, also known as *real-time data*, which can be under two different timing requirements: on one hand, the bounded transfer delay requirement (associated to its criticality), on the other hand, the periodicity requirement (associated to its transmission regularity). Nevertheless, both real-time and non-real-time data can be of a periodic, aperiodic or sporadic nature.

<table>
<tr><td>DATA</td><td colspan="2">Critical</td><td>Non-critical</td></tr>
<tr><td></td><td>Hard</td><td>Soft</td><td></td></tr>
<tr><td>Periodic</td><td colspan="2" rowspan="3">Real-time</td><td rowspan="3">Non-real-time</td></tr>
<tr><td>Aperiodic</td></tr>
<tr><td>Sporadic</td></tr>
</table>

Fig. 2.2. Temporal characteristics of data in distributed real time applications.

2.2.3 Jitter

The jitter requirement is associated with the periodicity requirement. It defines the variation in the periodic transmission of the messages which may be variable or not between a lower or an upper bound.

2.2.4 Time coherence

It is a property relating two or more event occurrences which must occur in a given time interval. It may be used to specify simultaneous actions such as production of values, transmission of data and so on [Thomesse, 1999].

In many applications, destination nodes may need several data which are both produced and transmitted from different source nodes. Hence, time coherence with regard to the message transmissions indicates whether the messages have been transmitted and/or received within a given time interval [ISO 184/SC5/WG2; Saba *et al.*, 1993].

2.2.5 Spatial coherence

Associated to a variable qualify its value. It indicates whether or not all the copies of a variable are identical at a given time or within a given time window [Saba *et al.*, 1993; Thomesse, 1999].

In many applications, the nodes typically store the values of the variables produced by one or more devices in a list, or in any data structure. When the nodes exchange these values, the copies of the list may be different due mainly to different frequencies of production, transmission and consumption, but also due to lost messages and retransmissions.

Then, spatial coherence requirement imposes on the communication system the existence of a mechanism to guarantee the consistency of the multiples copies of the list at a given time or within a given time interval. This mechanism must inform the producer and the consumer nodes of the consistency status of the list as well as to take the necessary actions to update the values of the list [ISO 184/SC5/WG2].

2.3 User space-related requirements

With regard to the size of data to be transferred per transaction, some users exchange bits, some users exchange words of data, some others exchange streams of data and long streams of data.

The bits of data are typically associated with the status of very simple devices, usually restricted to on/off status. Applications dealing with such sort of devices only require transfer a few bits of I/O data per node. However, some devices, either sensors or actuators, are more elaborated and use more information such as status, diagnostic and control. In this case, the associated applications require the transfers of words of I/O data.

Going up in the device complexity hierarchy there are the higher level devices such as control machines, robots and other equipment. The operation of such devices requires a stream of several words of data associated with status, configuration, diagnostic, calibration and control. The applications that manage these devices require then the exchange of streams of I/O data. At even higher levels there is the need to transfer files, to access databases, to gather the data associated to the whole system, to allow remote access to the system, etc. These operations commonly require the exchange of long streams of data per transaction.

It is interesting to note that, as the size of the data per transaction increases among the different end-users, the previous timing requirements become more relaxed [Almeida *et al.*, 1999b]. In fact, transactions carrying bits or words of I/O data are normally associated with sensors and actuators devices which are directly involved in the inner control loops of a subsystem of the industrial application, thus requiring frequent transmissions with low jitter and bounded as well as small message transfer delays. Streams of I/O data, normally associated with higher level devices, are involved in outer control loops, thus requiring less strict time-constrained communication services. At the manufacturing process higher levels, the applications typically require long streams of data associated with the overall control and

supervision of the whole system and the feedback actions are usually performed by human intervention, thus the respective timing requirements are even more relaxed.

Whatever the level of communication is, a timing analysis must always be considered associating attributes such as response times, deadlines and frequency with the tasks or with the messages and using a common method to specify them (temporal logic [Manna and Pnueli, 1992], timed automata [Alur and Dill, 1994], ...). The simple difference is the related time unit in which the time-related requirements are expressed.

Then, with regard to the size of data, the user requirements can be summarized as follows:

- Distinction between the size of data. The communication system should provide the services to transport the data from source to destination in whatever size of traffic.

2.4 Conclusion

This chapter has presented a user requirements analysis in order to identify the service characteristics that the communication systems must provide in the real-time systems. The analysis is focused on the time-related requirements and on the space-related requirements.

This analysis has shown that the bounded message transfer delay is a main user time-related requirement. However, there are other time-related requirements according to the users, for example in the industrial applications, these requirements are the periodicity, the jitter, the temporal coherence and the spatial coherence. Moreover, the analysis has identified the space-related requirement which takes into account the size of data per transaction.

This analysis has not taken into account other aspects of the user requirements analysis such as the performance nor dependability requirements. However, this analysis allows to organize the discussion of the communication systems of the next chapter.

Chapter 3

Communication Systems in the Real-Time Systems

Fieldbus and MAC Protocols

This chapter presents an analysis of the Fieldbus MAC protocols and some MAC protocols from two points of view, on one hand, the analysis takes into account the user time-related and space-related requirements discussed in Chapter 2 such as bounded message transfer delay, periodicity, jitter, temporal coherence, spatial coherence and the size of data per transaction. On the other hand, the analysis takes into account the characteristics of the Fieldbus networks and MAC protocols identifying the access methods and the scheduling algorithms that meet the previous requirements.

The chapter is structured as follows: Section 3.1 presents an introduction; Section 3.2 presents an analysis of the Filedbus networks and MAC protocols according to the user requirements previously discussed; Section 3.3 presents the main characteristics of the Fieldbuses and MAC protocols. Finally a summary and some conclusions are presented.

3.1 Introduction

The objective of this chapter is to analyze a special communication system called Fieldbus and the main existing Medium Access Control (MAC) protocols in order to determine which user requirements are met, particularly which of the time-related requirements analyzed in Chapter 2 are met, and how they are met.

The main reason of this approach is that there are many efforts attempting to modify existing MAC protocols in order to meet some time-related requirements. Specifically, the CSMA/CD protocol has been modified with different mechanisms such as Window-Access CSMA, Virtual-Time CSMA, PB/CSMA/CD, CSMA/LDCR, CSMA/DCR, etc. In some cases, it is said that they are well adapted for real-time systems, nevertheless it is not exactly say which time-related requirements are met.

On the other hand, *Fieldbus* are special purpose local area networks which are basically used to connect field equipment and control devices by means of a shared channel such as the bus or the ring. Fieldbuses are seen as a three or four layers structure which only includes the Physical, the Data Link (MAC and Logical Link Control) and the Application layers. Nevertheless, it is a confusing approach between conceptual and implementation models because some of the functionalities of the missing layers are still present in some fieldbuses but are merged in the application layer, e.g. support for interoperability, end-to-end control, fragmentation and reassembling of messages, routing among several fieldbuses [Thomesse, 1998].

Although many of the options taken in the protocols of each layer of the OSI Reference Model may influence the ability of the network to provide time-related communication services, the MAC layer, which is part of the data link layer, has the highest importance because it offers the access service to the communication channel, i.e. it arbitrates access to the transmission medium and then it determines what message to transmit at any given time. Therefore, it directly influences both the message transfer delay and the message scheduling of the communication system.

Note that when a set of distributed nodes share a single communication channel, the nodes must resolve the multiple-access problem [Kurose *et al.*, 1984] i.e. the channel provides the only means of communication among the nodes, and its properties are such that only a single message can be successfully transmitted over the channel at any one time. Since the nodes are distributed, they must exchange information if they wish to coordinate channel sharing. However, since there is only a single communication channel, coordination among the nodes necessarily requires use of the channel itself. Thus, there is a circular or recursive aspect of the problem. A second issue complicates the problem of channel sharing, the nodes never instantaneously know the present status of other nodes in a distributed environment; information about other nodes is always at least as old as the message propagation delay.

Moreover, in the real-time applications, the MAC layer must consider the timing requirements of individual users, such as those analyzed in Chapter 2.

The multiple-access problem can be seen as the more general problem of resource management, here the MAC layer is a scheduler which uses a policy or method to efficiently and effectively manage both the access to and the use of one resource (the communication channel) by various users (the nodes).

Given the importance of the MAC layer service, many classifications of the MAC protocols have been proposed in the literature, among which the following:

- *Controlled-access and contention-based access to the channel* [Kurose *et al.*, 1984]. Controlled-access involves collision-free protocols (demand adaptive (e.g. reservation and token protocols) and predetermined channel allocation (e.g. TDMA protocol)). That is, the nodes are coordinated in such a way that two or more nodes never attempt to transmit simultaneously. Contention-based protocols operate partitioning the nodes into two sets; a set of enabled nodes and another set of disabled nodes. Various methods are then used (e.g. probabilistic, time, address) as criteria for determining whether a node is enabled or disabled.

- *Strategies for channel acquisition* [Tanenbaum, 1988]. The protocols are classified as follows: contention-based, collision-free and limited-contention protocols. Contention-based involves both Carrier Sense protocols, in which nodes listen for a carrier and act accordingly (e.g. persistent and non-persistent CSMA), and Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol, in which nodes detect the collision, abort their transmission, wait a random period of time and try again.

  In collision-free protocols there are no collisions even during the contention period (e.g. basic bit-map, Broadcast Recognition Access Method (BRAP), Multi-Level Multi-Access

(MLMA), binary countdown). The limited-contention protocols use contention at low loads to provide low delay, but use a collision-free technique at high load to provide good channel efficiency (e.g. adaptive tree walk protocol and urn protocol).

- *Access methods to the channel* [Mammeri and Thomesse, 1991]. The protocols are classified as follows: consultation, contention and multiplexing. In the first method, nodes take advice from each other and decide who has the access right. The authorization can be got either by information exchange (e.g. token ring, token bus, slotted ring, polling) or by hardware. In the contention method, nodes use their own information and the channel stade to transmit their data (e.g. ALOHA, CSMA, CSMA/CD, CSMA/CA, CSMA/DCR, address tree searching, contention ring). In the multiplexing method the bandwidth is divided among the nodes (e.g. FDM and TDM protocols).

- *Cooperation models and time management* [Song and Thomesse, 1994]. The protocols are classified according to both the communication models (client-server and producer-consumer) and whether or not the protocols provide explicit (static or dynamic) time management.

- *Strategies used to meet message timing constrains* [Malcolm and Zhao, 1995]. The protocols are classified according to the guarantee and best-effort strategies for synchronous and asynchronous messages and depending on whether the emphasis lies on the access arbitration process or on the transmission control process. The access arbitration process determines *when* a node can send a message over the channel. The transmission control process determines *how long* a node can continue to send messages over the channel. In this work, the guarantee strategy means that an attempt is made to guarantee ahead of transmission time that the real-time messages will meet their deadlines, once a message is accepted for transmission, it is guaranteed to meet its deadline. In the best-effort strategy, the network will try to meet the message deadlines, but no guarantee are given.

- *Media access technologies: The evolution towards terabit/s LANs and MANs* [van As, 1994]. The protocols are surveyed through three technology generations. The first generation (e.g. Ethernet, Token Ring and Token bus) predominantly covers bit rates of some 10 Mbit/s. The second generation comprises different network configurations and access methods in the speed range from 100 Mbit/s up to multi-Gbit/s. In the third technology generation, architectural and experimental network designs have begun taking advantage of the tremendous transmission bandwidth of more than 20 THz of the optical medium.

This chapter presents then an analysis of the Fieldbus and MAC protocols from two points of view, on one hand, the analysis takes into account the user requirements identified in Chapter 2. On the other hand, the analysis takes into account the characteristics of the Fieldbus MAC protocols and the main existing MAC protocols in order to identify the access methods and the scheduling algorithms that meet these requirements [Leon and Thomesse, 2000].

3.2 Time-related characteristics

3.2.1 Message transfer delay

It is the required time to transmit a message through the Fieldbus network, as the figure 3.1 shows, the message transfer delay takes into account the delay in the application layers of the sending and receiving nodes (d_{AS} and d_{AR}, respectively), the waiting time in the MAC layer of the sending node (d_{MAC}), the message transmission delay (d_T), the end-to-end propagation delay (d_τ) and the arrival delay in the MAC layer of the receiving node (d_a).

The *message transmission delay* is the required time to physically transmit a message and is equal to *L/R*, where: *L* is the message length or packet length in bits, and *R* is the bit rate in bit/sec. The *end-to-end propagation delay* is equal to *C/P*, where: *C* is the channel length in meters, and *P* is the propagation speed in meters/sec.

The *arrival delay* in the MAC layer of the receiving node depends on how quickly a host processor can respond to a "message arrived" interrupt from the MAC layer. The *waiting time* in the MAC layer of the sending node depends on the MAC protocol that is used [Kurose *et al.*, 1984; Malcolm and Zhao, 1995]. Basically, this delay determines the message transfer delay.

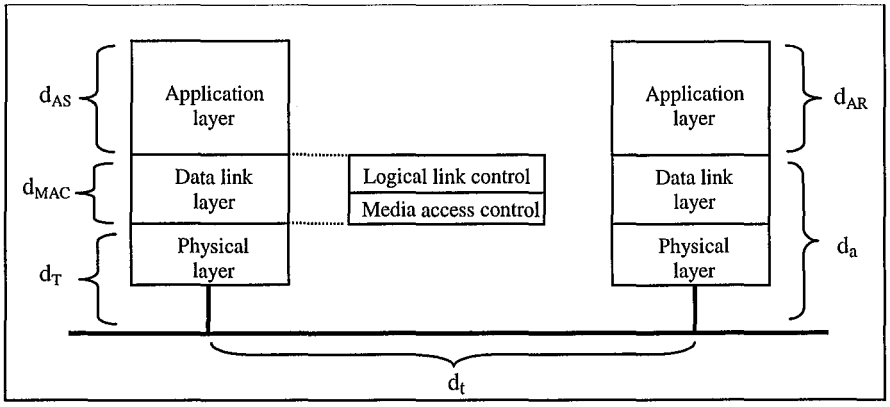

Fig. 3.1. Message transfer delay.

The message transfer delay is known under different names, for example: the transfer time, the message delay, the response time, the message delivery time, the application-to-application delay (if the delays of the missing layers are taken into account), etc. This delay is measured in different ways, for example [Joseph and Pandya, 1986]: average, maximum, worst-case, etc.

Bounded message transfer delay, between a lower and upper bound, is an user requirement that has been analyzed in a lot of papers and several good results have been presented, as are shown in Table 3.1. However, many assumptions must be made in order to delimit the message transfer delay. For example, normal network operation, no loss of messages, no loss of token, no crash of master, limited buffer capacity in the MAC layer, constant network load, fixed lengths of the messages, and so on. Bounded message transfer delay is then a requirement which must be carefully analyzed explicitly describing the assumptions made.

In order to meet this user requirement, two access methods can be identified: *Controlled access* and *uncontrolled access but with deterministic collision resolution.* The protocols within each class are analyzed in the next subsections, are summarized in Table 3.1 and will be further analyzed in section 3.3.

3.2.1.1 Controlled access or free-contention method

As a way to guarantee that there is only one node accessing the channel at any given time. There are two approaches that can be followed, centralized and distributed. In the *centralized* approach, a special node called master or arbitrator explicitly gives each node, one at a time, the right to access the medium and send some data. This model is normally known as the *Master-Slave* model and is used, for example, by *Profibus, Interbus* and *P-Net* between each master node and the respective slave nodes.

There is another communication model known as *Producer-Distributor-Consumer* model. In this case, the master node addresses a data entity, e.g. a process variable (*source addressing*). The node responsible for the production of such entity, whichever node it is, responds to the master call and broadcasts the respective value. This is the approach followed by the *WorldFIP*, the *Foundation Fieldbus* and the *TS 61158* (between the Link Active Scheduler and the publisher-subscriber elements) Fieldbus networks.

Controlled access protocols can also be *distributed.* In this case, all the nodes coexist at the same logical level. One possible technique to allow such coexistence is the periodic allocation, i.e. the global time in the network is partitioned into time windows or fixed time intervals which are assigned to the nodes and during which only one node can access the medium.

Two periodic allocation methods are identified: *time multiplexing* and *token passing*. In *TDMA* each time window is divided again into time slots. If the number of slots is equal to the number of nodes in the network, called the synchronous method, then each node periodically uses the channel during its slot and each node can only transmit during this time interval.

In the token passing protocols, the right time window for a given node to access the medium is identified by a token circulating among the nodes. Timely behavior is achieved limiting the time that a node can hold the token continuously before sending it to the next node. *Timed token* (FDDI) and *Token bus* (IEEE 802.4) protocols use this approach. *Profibus* also uses this sort of token passing among the master nodes as well as the *TS 61158* Fieldbus among special nodes called Link Masters.

In some cases the periodic allocation methods are not constant, therefore various authors refer to it as cyclic allocation, but not at all as periodic allocation. However, the term periodic is used in this work since the variation in the periodic transmission is taken into account in the jitter requirement.

It can be noted that the token passing method is also used in the *Token ring* protocol (IEEE 802.5). Timely behavior is also achieved limiting the time that a node can hold the token, nevertheless the global priority mechanism prevents a periodic allocation, i.e. even if token is

passed to the next node, it may not have the access rights because there is a priority field in the token which controls the access rights. The priority field in the token is examined by each node whenever it arrives. A node inserts the higher priority amongst its waiting messages, if and only if this priority is higher than the priority in the token. A node can only seize the token and send its message when the token comes with the requested priority and after that it has made a round through all nodes. In this way, the emission order is defined by the message priorities and is not defined by the node order in the ring as in the Token bus and the Timed token protocols.

3.2.1.2 Uncontrolled access method

These protocols are generally known as *Carrier Sense Multiple Access* (CSMA) and are fully distributed. All the nodes are considered at the same logical level which grants the communication system a high level of modularity and flexibility. Furthermore, the communication overhead is low since there are no extra messages such those issued by the master node or those required to carry tokens. However, in this case it is possible that several nodes start transmitting at the same time leading to message collisions.

The most popular of these protocols is *CSMA/CD* (IEEE 802.3) which has a good average performance up to a certain level of traffic load but, above that, the performance degrades substantially [Stallings, 1984; Fine and Tobagi, 1984; Abeysundara and Kamal, 1991]. Furthermore, the message collision resolution is indeterminate and unpredictable, therefore the message transfer delay is unbounded. Despite this, several mechanisms have been proposed in order to take into account the message timing constraints such as: Window-Access CSMA [Kurose *et al.*, 1988; Zhao *et al.*, 1990; Znati, 1991], Virtual-Time CSMA [Molle and Kleinrock, 1985; Zhao and Ramamritham, 1987], PB/CSMA/CD [Ulusoy, 1995] and CSMA/LDCR [Norden *et al.*, 1999].

On the other hand, the *uncontrolled access* but with *deterministic collision resolution* method, is a way to resolve collisions within a bounded time. Two main protocols are identified, one is *CSMA/CA* and the another is *CSMA/DCR*. In CSMA/CA and *Controller Area Network* (CAN) protocols each data entity must have a unique identifier which also sets the priority of the respective transaction. At application operation time, the nodes transmit their highest priority message while monitoring the channel. The identifier field is transmitted first, beginning with the most significant bit. The message is coded so that if a node transmits a recessive bit but monitors a dominant bit (logical 0) then a collision is detected. The node knows that its message is not the highest priority message in the network, stops transmitting and waits for the channel to become idle. Thus, at each collision, the message identifier with the smaller binary value gains access to the medium without delay.

In CSMA/DCR protocol every node has been assigned an index or address which is used to implement the binary tree search algorithm. The nodes are considered as the leaves of a binary tree and the time is slotted. The nodes transmit as soon as the channel becomes free. If there is a collision, the set of nodes is divided so that the nodes in the right branch of the binary tree are authorized to retransmit in the next slot. If there is still a collision, the subset of nodes is subdivided until only one node transmits successfully. When a time slot is detected as free, there are no more nodes in a branch and then the left branch is explored.

MAC protocols and Fieldbuses	Controlled access		Uncontrolled access (Distributed)		Access method	Bounded message transfer delay	Reference
	Centralized	Distributed	Deterministic collision resolution	Non deterministic collision resolution			
CSMA/CD				X	Probabilistic	False	IEEE 802.3
Window-CSMA				X	Time window	False	Kurose *et al.*,84 Zhao *et al.*,90
Virtual-time CSMA				X	Virtual time	False	Molle, 85; Zhao, 87
CSMA/DCR			X		Binary tree	True (worst-case)	Le Lann, 94
PB/CSMA/CD				X	Slotted time and preemption	False	Ulusoy, 95
TDMA (single slot per node)		X			Time multiplexing	True (average)	Simonot, 97
TTP		X			Time multiplexing	True	Kopetz, 94
IEEE 802.4		X			Token and local priorities	True (average)	Moon, 98
IEEE 802.4		X			Token and local priorities	True (maximum)	Ng and Liu, 91
Token passing		X			Token	Worst case performance ratio	Lim *et al.*, 91
IEEE 802.5		X			Token and global priorities		IEEE 802.5
IEEE 802.5		X			Token and global priorities	Worst case performance ratio	Lim *et al.*, 91
IEEE 802.5		X			Token and time window	Worst case performance ratio	Lim *et al.*, 91
IEEE 802.5		X			Token and global priorities	True (maximum)	Ng and Liu, 91
Slotted ring		X			Token and global priorities	True (maximum)	Ng and Liu, 91
FDDI		X			Token	True (maximum)	Johnson, 87; Agrawal, 92
WorldFIP	X				Bus arbitrator	True (average)	Song, 91 Thomesse, 93
Profibus	X	X			Token and Master-slave	True (maximum token cycle time)	Tovar and Vasques, 99
Interbus	X	X			Master-slave Summation-frame		
CSMA/CA			X		Global priorities	True (worst-case)	Tindell, 95a; Navet, 00
P-Net	X	X			Virtual token and Master-slave	True (upper bound)	Tovar *et al.*, 99
ControlNet		X			Time multiplexing		
LON				X	Probabilistic		
Foundation Fieldbus	X				Link Active Scheduler		
TS 61158	X	X			Link Active Scheduler and token	True (maximum token circulation time)	Mnaouer *et al*, 97

Table 3.1. Bounded message transfer delay requirement.

3.2.2 Periodicity

This requirement takes into account that most of the messages exchanged among the users are composed of periodic data which must be transmitted before the beginning of their next period. Hence, the communication system must provide the service to transmit this sort of traffic, sometimes referred as *synchronous traffic*. On the other hand, the communication system must also provide the service to transmit the aperiodic traffic, also called *asynchronous traffic*.

At the MAC layer, the messages are characterized as follows [Malcolm and Zhao, 1995]:

- An arrival time, which is the time at which the message becomes available for transmission.
- A length, which is the number of unit times required to transmit the entire message.
- A deadline, which defines a time constraint for transmission of the message.
- A laxity, which defines the amount of remaining time before transmission of the message must begin if the message is to meet its deadline.

Note that the main difference between the characteristics of the periodic and aperiodic messages is the instant at which are known by the communication system. The periodic traffic characteristics are known in advance, at network configuration time. The aperiodic traffic characteristics are known till the message arrives at the MAC layer, at application run time.

In order to provide both periodic and aperiodic traffic transmission services, the MAC layer can be seen a scheduler, i.e. it utilizes a policy or algorithm to determine which message to transmit at any given time.

The scheduling problem has been studied in many different ways and in many different fields. The classical problem of job sequencing in the production management has influenced most of the work in this problem even if it is known to be NP-complete in its general form.

Obviously, this problem has been broadly studied in the real time systems. Nevertheless, most of the studies are mainly based on the following task scheduling algorithms:

- *Rate monotonic* algorithm [Liu and Layland, 1973] is a static-priority preemptive scheduling policy which assigns priorities to tasks in inverse proportion to their periods.

  It can be noted that this scheduling algorithm is composed of two parts [Audsley and Burns, 1990; Audsley *et al.*, 1993]. The first part, the schedulability test, decides whether a set of tasks is schedulable under the rate monotonic scheme; the second part assigns a static priority ordering to the tasks.

  The schedulability test states that the sum of the processor utilization percentage (ratio of computation time divided by interarrival time), for all tasks, must not exceed an utilization bound. The utilization bound expression $n(2^{1/n} - 1)$ converges to 69% for large values of n.

This schedulability test is a necessary but not a sufficient test, i.e. there are task sets that can be scheduled using a rate monotonic priority ordering but which break the utilization bound [Audsley and Burns, 1990; Audsley *et al.*, 1993]. In [Lehoczky *et al.*, 1989] a necessary and sufficient schedulability test has been proposed. This test states that it is only necessary to test that each task will meet its first deadline to know that it will never miss a deadline.

In the local area networks, the rate monotonic algorithm is used to determine whether a set of messages is schedulable [Klein *et al.*, 1994] and to assign priorities to the messages. Synchronous messages with the smaller periods are assigned to the higher priorities [Lehoczky and Sha, 1986]. Asynchronous messages are assigned to a periodic server [Strosnider *et al.*, 1988; Strosnider and Marchok, 1989].

- *Earliest deadline* algorithm [Liu and Layland, 1973] is a dynamic-priority preemptive scheduling policy in which the task with the earliest (absolute) deadline from the current moment has the highest priority.

  The necessary and sufficient schedulability test states that any independent task set is schedulable under this policy if the processor utilization by all the tasks does not exceed 100%.

  This algorithm is used then to assign priorities to the messages in inverse proportion to their deadlines.

- *Least laxity* algorithm [Sorenson, 1974; Dertouzos and Mok, 1989] is a dynamic-priority preemptive scheduling policy which assigns priorities to tasks in inverse proportion to their laxities (the difference between their deadline and their remaining computation time). This scheduling algorithm is optimal for uniprocessor and multiprocessor systems [Panwar *et al.*, 1988].

  The necessary and sufficient schedulability test states that any independent task set is schedulable under this policy if the processor utilization by all the tasks does not exceed 100%.

  This algorithm assigns then the highest priority to the message with the smallest amount of remaining time before transmission must begin, if the message must meet its deadline.

It can be noted that these scheduling algorithms are either *static* or *dynamic* [Casavant and Kuhl, 1988]. In the static-priority scheduling approach there is a fixed base priority associated with each task, although the task may temporarily acquire a higher active priority at run time. On the other hand, dynamic-priority policies assign variable priorities at run time.

A survey of the schedulability tests is presented in [Fidge, 1998], a large study of the rate monotonic approach is presented in [Klein *et al.*, 1993] and its application to the real time industrial computing in [Klein *et al.*, 1994]. A full study of the earliest deadline approach is presented in [Stankovic *et al.*, 1998]. An equivalence between tasks and messages from the scheduling point of view is presented in [Cardeira, 1994; Cardeira and Mammeri, 1995].

So, with regard to the transmission of the periodic and aperiodic traffics, the message scheduling service at the MAC layer can be classified as static and dynamic, as is analyzed below and summarized in Table 3.2.

3.2.2.1 Static scheduling

In this case, all the nodes as well as their communication needs are known in advance and will not change during the application operation. Hence, the schedulability tests can be performed off-line, at network configuration time, which will give information on whether the set of messages can be scheduled according to the used algorithm. This feature is useful for periodic traffic and even for sporadic traffic. However, the aperiodic traffic is normally associated to transient overload or emergency situations, and it is not well handled by this approach due to the unpredictability of the respective arrival instants. In brief, *static scheduling* allows a guaranteed message scheduling service in applications with a fixed set of user requirements.

There are basically three distinct forms of static scheduling: *static-table based, static-priority based* and *periodic allocation based.*

- In the *static-table based* case, after schedulability test, a schedule table or list is built (usually by the user) at network configuration time and then it is used on-line by a dispatcher or arbitrator, running on a master node, to timely initiate the respective transactions. This approach (used to manage the periodic traffic in WorldFIP, Foundation Fieldbus, TS 61158 as well as in Profibus and P-Net in each master node) results in very low processing overhead at run-time and thus it is suited to support high rate periodic transactions. Nevertheless, no on-line adaptations are allowed and thus the flexibility of the communication system is rather low.

- On the other hand, when *static-priority based* scheduling is used, the schedulability test and the priority assignment are performed off-line. At application operation time, the priorities determine either which message is transmitted or which node has the channel access rights. The first approach is used by the CAN protocol since all the messages have been assigned an identifier used as priority, the second approach is used by the CSMA/DCR protocol since the nodes have been assigned an index used as priority. Note, however, that on-line changes to the set of requirements are not allowed. In this case, the application must be halted and the schedulability test must be redone before putting the application on again.

- *Periodic allocation* makes use of the periodic access methods to the medium (e.g. the time multiplexing and some token passing protocols) in order to transmit the periodic traffic, firstly. It is the approach used by FDDI, the idea behind this timed token protocol is to control the token holding time, known as the synchronous bandwidth, which is the maximum time that a node is permitted to transmit synchronous messages every time it receives the token. A node can transmit asynchronous messages only if the token arrived earlier than expected.

The time multiplexing approach is used by the *Interbus* protocol where there is a master node which periodically send a frame through the ring, the frame is divided into slots and each slot is assigned to a node, i.e. the number of slots is equal to the number of nodes.

The *TTP* protocol also makes use of the time multiplexing approach provided by the TDMA protocol in order to synchronize the periodic data sampling in each node.

ControlNet uses a time multiplexing method called the Concurrent Time Domain Multiple Access. In this method, the network time is divided into windows or portions, known as Network Update Intervals (NUI), which are further divided into three parts. The first to transmit the time-critical data, the second part for the non time-critical data and the last part to allow a constant duration of the NUI.

3.2.2.2 Dynamic scheduling

In this case, no prior knowledge concerning the user requirements is considered. At run-time, the communication system handles the requests as they firstly appear in either form periodic or aperiodic. So, the messages are transmitted according to their timing constraints, either their deadline or their laxity, but forgetting their periodic or aperiodic nature. We refer to this fact as *indistinctly transmitted traffic*. This approach is highly flexible since any changes to the communication needs can be accommodated, or at least considered, on-line. In many situations, no pre-run-time schedulability test is performed and thus a *best-effort* approach is used: the communication system does its best to handle the current requirements but the message scheduling is not guaranteed.

There is basically one form of dynamic scheduling: *dynamic-priorities based* [Mok and Ward, 1979]. The priorities can be either explicit or implicit. Explicit priority can be further divided into local (Token bus protocol) and global (Token ring protocol). Implicit priority is basically implemented in the CSMA/CD protocol by some mechanisms such as the time window and the virtual time in order to schedule the messages using a dynamic-priority scheduling algorithm, even if the global priorities of the Token ring protocol have been also used with this goal.

To achieve guaranteed dynamic scheduling, on-line schedulability test must be used. One possible approach is to use the *resource reservation model* [Ferrari and Verma, 1990]. In this case, whenever there is a request for extra resources (e.g. new periodic or aperiodic traffic) a resource allocator module is invoked to determine whether the new set of requirements can be met and guaranteed. If yes, the request is processed, otherwise, the requesting entity is notified that it is not possible, for the moment, to satisfy the request. This feature is also part of a larger concept known as Quality of Service (QoS) which will be analyzed in the next Chapter. There is recent research work concerning the use of this approach in existing Fieldbus networks such as WorldFIP [Almeida *et al.*, 1999a] and CAN [Rößler and Geppert, 1997]. Nevertheless, QoS concept must be supported by a QoS architecture which defines the elements and their relations in order to provide different levels of QoS. So, a Fieldbus QoS architecture is then required.

MAC protocols And Fieldbuses	*Static Scheduling*			*Dynamic Scheduling*	*Transmitted traffic*	*Scheduling algorithm*	*Reference*
	Table based	*Priority based*	*Periodic allocation*	*Priority Based*			
CSMA/CD					Indistinctly		IEEE 802.3
Window-CSMA				X	Indistinctly	Least laxity	Kurose, 84; Zhao, 90
Virtual-time CSMA				X	Indistinctly	Least laxity	Molle, 85; Zhao, 87
CSMA/DCR		X					Le Lann, 94
DOD/CSMA/ CD				X	Indistinctly	Earliest deadline	Le Lann, 94
CSMA/LDCR				X	Indistinctly	Least Laxity	Norden, 99
PB/CSMA/ CD				X	Indistinctly	Earliest deadline	Ulusoy, 95
TDMA (single slot per node)			X		Indistinctly		
TTP			X		Periodic		Kopetz, 94
IEEE 802.4			X	X	User defined		IEEE 802.4
IEEE 802.5				X	User defined		IEEE 802.5
IEEE 802.5				X	Periodic and deferrable server	Rate monotonic	Strosnider,89 Pleinevaux, 92
IEEE 802.5				X	Indistinctly	Least laxity	Yao and Zhao, 91
IEEE 802.5				X	Indistinctly	Earliest deadline	Lim *et al*, 91
FDDI			X		Periodic		Johnson, 87; Agrawal, 92
FDDI			X	X	Indistinctly	RM, ED, SPF	Song, 96
WorldFIP	List of identifiers		Basic cycle		Periodic	Bus arbitrator	
Profibus	Master's data base and poll list	X	X		Periodic	Master nodes	
Interbus			Summation frame		Periodic	Master node	
CAN		X			User defined		
P-Net	Softwire List		X		Periodic	Master node	
SwiftNet			Assigned channel		Periodic		
Control Net			Scheduled window		Periodic	Round-Robin	
LON				X	Indistinctly		
Foundation Fieldbus	List of transmission times		X		Periodic	Link Active Scheduler	
TS 61158	List of periodic activities		Timed token	X	Periodic	Link Active Scheduler and token	

Table 3.2. Periodicity requirement.

3.2.3 Jitter

It measures the variation in the periodic transmission of the messages which can be variable or not between a lower or an upper bound. This timing requirement can be only met by the protocols providing a periodic access method to the medium (e.g. the time multiplexing and some token passing protocols) and by the protocols privileging the transmission of the periodic messages as the Table 3.3 summarizes.

For example, WorldFIP [Thomesse, 1993] privileges the transmission of the periodic messages and provides a periodic access method, i.e. it defines one macro-cycle which is composed of micro-cycles, themselves composed of one periodic part and one aperiodic part, and eventually one special part to adjust the constant duration of the cycle. During the periodic part, the bus arbitrator reads a list of periodic data labels and injects each data label onto the network. Consumers needing to utilize the data, alerted by the label, store and use the value broadcasted by the producer. During the aperiodic part, the bus arbitrator handles all aperiodic data that have been requested for transmission, in the same way that the periodic data. So, the duration of the special part is variable according to the number of aperiodic transactions but guaranteeing the periodic message transfers, therefore the jitter is constant.

This is also the case of ControlNet with the network maintenance window which is used as a "guardband" to allow the constant duration of the Network Update Intervals (NUI).

The period in the periodic access methods can be very high if there are many nodes in the network, for instance, in the synchronous TDMA protocol, i.e. a single slot by node by cycle, all the nodes have been assigned a slot even if a node does not have any message to send or if the node is crashed. In this case, the slot will be unused, the jitter will be constant but the period will be high.

In the token passing protocols (e.g. Timed token, Token bus, Token ring, Profibus and P-Net) the jitter can be increases if the token passing time, i.e. the time between the emission and the reception of the token between two adjacent nodes, is not constant. For example, in [Moon *et al.*, 1998] it is shown that the token can be easily lost in a noisy environment, like the industrial applications. So, when the token is lost, the token passing time is increased by more than 100 times the nominal token passing time of a noise-free environment.

The jitter can also be observed among the different levels of priority of each node in the Token bus protocol [Ng and Liu, 1991]. In this case, there are four priority levels which are assigned to the messages. Every node has a timer which controls the token time used by each priority level. When the node receives the token, it sets its timer with the maximum duration associated with the higher priority and then the node begins to transmit. The node may send lower priority messages only if it has still sufficient time to do it. When the node passes from one priority to the other it must reset its timer with the maximum duration associated with the new priority. Note that if a node does not have any higher priority message when it receives the token, the node can transmit the next lower priority messages even though its successor node may have higher priority messages. So, when the network load increases, the messages that start to miss their deadlines are not necessarily the lowest priority messages, instead they can be any of the higher priority classes.

MAC protocols and Fieldbuses	*Periodic Allocation*	*Privileged traffic*	*Jitter*	*Reference*
CSMA/CD		Indistinctly	False	IEEE 802.3
Window-CSMA		Indistinctly	False	
Virtual-time CSMA		Indistinctly	False	
CSMA/DCR				
DOD/CSMA/CD		Indistinctly	False	
CSMA/LDCR		Indistinctly	False	
PB/CSMA/CD		Indistinctly	False	
TDMA (single slot per node)	X	Indistinctly	True	
TTP	X	Periodic	True	Kopetz, 94
IEEE 802.4	X	User defined	True	IEEE 802.4
IEEE 802.5		User defined		IEEE 802.5
IEEE 802.5		Periodic and deferrable server		Strosnider, 89 Pleinevaux, 92
FDDI	X	Periodic	True	Johnson, 87 Agrawal, 92
WorldFIP	X	Periodic	True	
Profibus	X	Periodic	True	
Interbus	X	Periodic	True	
CAN		User defined		
P-Net	X	Periodic	True	
SwiftNet	X	Periodic	True	
Control Net	X	Periodic	True	
LON		Indistinctly	False	
Foundation Fieldbus	X	Periodic	True	
TS 61158	X	Periodic	True	

Table 3.3. Jitter requirement.

3.2.4 Time coherence

Section 2.2.4 has defined the time coherence requirement as a mechanism to indicate whether the messages have been transmitted and/or received within a given time interval [ISO 184/SC5/WG2; Saba *et al.*, 1993; Thomesse, 1999].

Typically, the MAC protocols do not provide any mechanism to notify that the transmission and/or the reception of several messages have been made within a given time interval, as it will be shown in Section 3.4.

However, in order to meet this timing requirement, a general state machine has been proposed in [Lorenz *et al.*, 1994] to generate a set of temporal status. It informs of the temporal validity of the values at time production, at time emission, at time reception and at time consumption. In order to generate the temporal status, several time windows are defined to bound both the time interval [request, confirmation] at the client side and the time interval [indication, response] at the server side.

3.2.5 Spatial coherence

Section 2.2.5 has defined the spatial coherence requirement as a mechanism to guarantee the consistency of multiples copies of a list at a given time or within a given time interval.

Typically, the MAC protocols do not provide any mechanism to generate the consistency status of any list, as it will be shown in Section 3.4. Nevertheless, at the application layer, some Fieldbuses do it, for example WorldFIP [Saba *et al.*, 1993].

3.3 Space-related characteristic

When considering real-time applications, the user requirements identified in the previous Chapter have to be taken into account in order to choose the most appropriate solution. However, within any real time application, such needs vary either in terms of the associated time units or in terms of the amount and type of information that must be exchanged [Almeida *et al.*, 1999b]. For example, in the industrial applications, the communication needs of a machine-tool controller are different from those of a cell controller and also different from those of the human users.

In order to satisfy these different needs, several communication systems may coexist in the industrial applications and are hierarchically organized according to levels in what is known as the Computer Integrated Manufacturing (CIM) architecture [Pimentel, 1990]. In this architecture, three main levels are usually considered: floor plant control, cell control and process/machine control. For each of these levels different services are required.

The higher level, *floor plant control*, promotes the integration of all cells within the industrial application. It supports the global factory management and monitoring. The long streams of data shared by the applications at this level are normally conveyed by *local area general-purpose networks* such as CSMA/CD, Token Ring and FDDI.

The simultaneous control of several processes or machines is carried out at the *cell control level*. Due to the temporal and size properties of the data transactions at this level, general-purpose networks cannot be used. Instead, special purpose networks known as *fieldbus* are used which rely on simpler protocols with reduced communication and processing overhead. These protocols make use of scheduling techniques particularly adapted to support time-related communication services. Examples of such field networks are Profibus/FMS, WorldFIP, P-Net, Interbus and Foundation Fieldbus.

At the lowest level, close to the field devices, i.e. sensors, actuators and controllers, is the *process/machine control level* which includes, for example, the position control of axis in a robot arm or in a control machine, or the temperature and pressure control in a chemical reaction. The requirements of the applications at this level are even more demanding in what concerns timing services. Also, the data exchanged per transaction is short. In these cases, standard fieldbuses as those referred above may not cope with the application requirements. In such case, a particular type of fieldbus is used which are known as *device networks*. Examples are Profibus/DP and Device WorldFIP (DWF), which are reduced profiles of the

respective fieldbus standards, DeviceNet and Smart Distributed System (SDS), which are based on the CAN protocol.

At this low level there is also another type of networks which use very simple protocol solutions to provide fast interconnection of simple sensors and actuators. They are known as *sensor networks* or also as *sub-fieldbuses* due to their simplicity. Examples of these are AS-Interface (AS-I) and Seriplex.

Above all these levels, the wide-area distribution of large sectors of the industrial application such as production, management and product development should be considered, and it is bringing a new dimension to the factory upper layer. The interconnection of the different industrial sites is carried out with wide-area networks (WANs) which rely upon services provided by telecommunications operators. At this level either dedicated leased lines or ATM services are typically used so that some real-time services can be achieved.

3.4 Fieldbus and MAC Protocols Analysis

Next subsections analyze the main existing MAC protocols and Fieldbus networks.

3.4.1 CSMA/CD protocol

In the *Carrier Sense Multiple Access with Collision Detection* protocol [IEEE 802.3], each node first listens for a carrier to determine if another transmission is in progress. If the channel is idle, the node can transmit. During transmission, the node listens for any collisions. If a collision is detected, the nodes abort their transmissions, wait for a random period of time (determined according to a binary exponential back-off algorithm) and try again. The nodes stop their retransmissions if the collision number reaches a fixed threshold.

This protocol has a good average performance up to a certain level of traffic load but, above that, the performance degrades substantially [Stallings, 1984; Fine and Tobagi, 1984; Abeysundara and Kamal, 1991]. Furthermore, since resolution of the message collisions is indeterminate and unpredictable, the message transfer delay is unbounded, and worst, there can be messages not transmitted. On the other hand, the protocol does not privilege any type of traffic for transmission, it indistinctly manages the synchronous and the asynchronous traffics, therefore it does not meet the periodicity requirement.

3.4.2 CSMA/DCR protocol

The *CSMA with Deterministic Collision Resolution* protocol [Le Lann and Rolin, 1984; Boudenant *et al.*, 1987; Le Lann and Rivierre, 1994] is a deterministic variant of the CSMA/CD protocol. The variation simply consists in replacing the binary exponential back-off algorithm with a deterministic binary tree search algorithm to resolve the collisions within a bounded time.

At network configuration time each node has been assigned an index or address which is used by the protocol to implement the binary tree search algorithm. As long as no collision occurs, CSMA/DCR behaves exactly like CSMA/CD. Upon collision occurrence, CSMA/DCR repeatedly divides the set of transmitting nodes, i.e. the set of the nodes is divided into two: winners and losers. The former is the set formed by all the nodes having addresses higher than or equal to the average of the addresses. The loser nodes cease emitting until all frames of the winner nodes are transmitted.

The division process continues until all branches of the tree are analyzed: each branch of the address tree is analyzed until a node successfully answers (without collision) or until no node answers (a branch is abandoned as soon as no node answers at the end of a given time). After search of a given branch, the algorithm DCR passes to the branch immediately to the left.

The upper and lower bounds of the average packet delay of the tree protocols with collision detection have been proposed in [Chung *et al.*, 1994].

3.4.3 DOD/CSMA-CD protocol

The *Deadline Oriented Deterministic CSMA/CD* protocol [Le Lann and Rivierre, 1994] is similar to CSMA/DCR, the major difference come from using indices that are dynamically computed on-line, rather than statically allocated to sources. The indices represent deadline equivalence classes. All the messages that are assigned to the same index have comparable deadlines.

3.4.4 CSMA/CA and CAN protocols

In the *CSMA with Collision Avoidance* and *Controller Area Network* protocols [Bosch, 1992; ISO 11898] all the messages have been assigned an unique identifier which is used as priority, the length of the identifier is equal to 11 bits in the standard format and is equal to 29 bits in the extended format. At network operation, when the channel is detected as idle, each node begins to transmit its highest priority message whilst monitoring the channel. The most significant bit of the identifier field is transmitted first. The message is coded so that if a node transmits a recessive bit (logical 1) but monitors a dominant bit (logical 0) then a collision is detected. The node knows that its message is not the highest priority message, stops transmitting and waits for the channel to become idle, as the Fig. 3.2 shows. Meanwhile, the node becomes a receiver of the message. Each receiver node performs an acceptance test to determine whether the data are relevant. If so, the data are accepted, otherwise are ignored. If there is no a collision in the least significant bit, the node transmits the body of the message.

An analysis that limits the response time of all the CAN messages including the lowest priority message is presented in [Tindell *et al.*, 1995a and 1995b]. Other analysis providing an analytical method for computing the worst case deadline failure probability is presented in [Navet *et al.*, 2000]. These analyses allow to limit the message transfer delay. Nevertheless, the *CAN* protocol does not make any distinction between periodic and aperiodic traffic, it is an user responsibility.

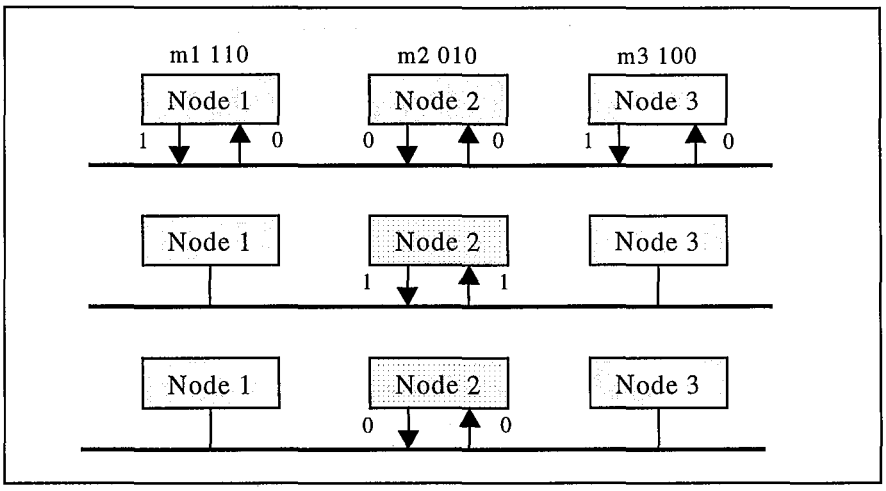

Fig. 3.2. CSMA/CA protocol.

DeviceNet and **SDS** (*Smart Distributed System*) are two CAN-based control Fieldbus networks, as well as the **Eibus** and **Batibus** Fieldbus networks. The *European Installation Bus* (Eibus) is a control system for all related applications in home and buildings. It defines a data communication system for use at the automation level in heating, ventilating and air conditioning and related building management applications.

3.4.5 Window-CSMA protocol

In this protocol [Kurose *et al.*, 1984; Kurose *et al.*, 1988; Zhao *et al.*, 1990; Znati, 1991], the window is a time interval (*low*, *high*), where *low* is the current time in the network (t_0) and *high* is an upper bound ($t_0 + w$).

In the CSMA/CD networks, each node stores a local copy of the window. Whenever the channel becomes idle, each node with a message in the current window starts its transmission. The channel state is used to determine the number of transmitting nodes. If the channel remains idle, it means that any node has a message in the current window, then the window will be enlarged, with *low* equal to *high*, and *high* is update. If a collision occurs, there are two or more messages in the window, then the window size will be decreased, with *high* equal to middle point of the old window, as is shown in Fig. 3.3.

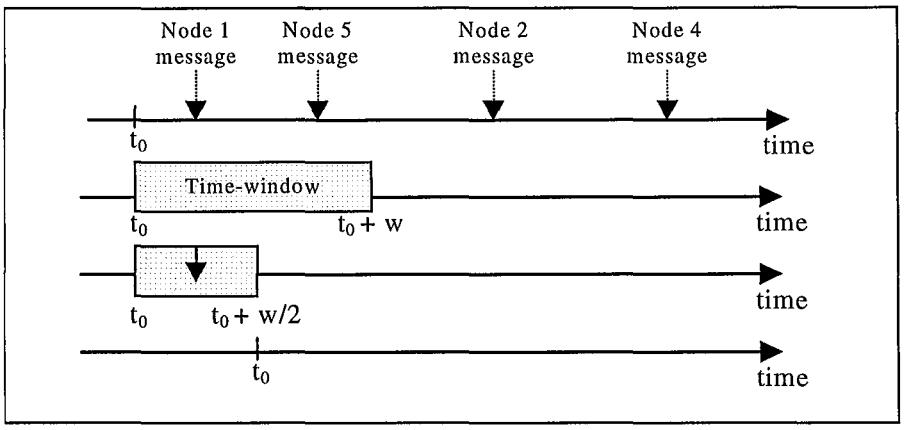

Fig. 3.3. Window-CSMA protocol.

It can be noted that the window mechanism is used to implement the least laxity scheduling algorithm in the network. Some improvements to this protocol have been proposed in [Znati, 1991] resulting in faster collision resolution but no any bound on the message transfer delay is presented. Furthermore, when two or more messages have identical laxities, the window cannot be further divided and the messages will be then transmitted with a given probability. Therefore, this protocol cannot always guarantee bounded message transfer delay. Note also that the periodic and aperiodic messages are indistinctly transmitted.

3.4.6 Virtual-time CSMA protocol

In this protocol [Molle and Kleinrock, 1985; Zhao and Ramamritham, 1987], when the channel becomes idle, each node with a message to send waits for a time interval proportional to the priority of its highest priority pending message. There is a fixed proportionality constant which is used to determine the waiting time. At the expiration of the waiting time, the node checks the channel state. If it is busy, the node waits for another idle period. Otherwise, the node starts to transmit its highest priority pending message. However if there is a collision, the nodes involved either retransmit the message with a probability p or wait for another time interval proportional to the message priority.

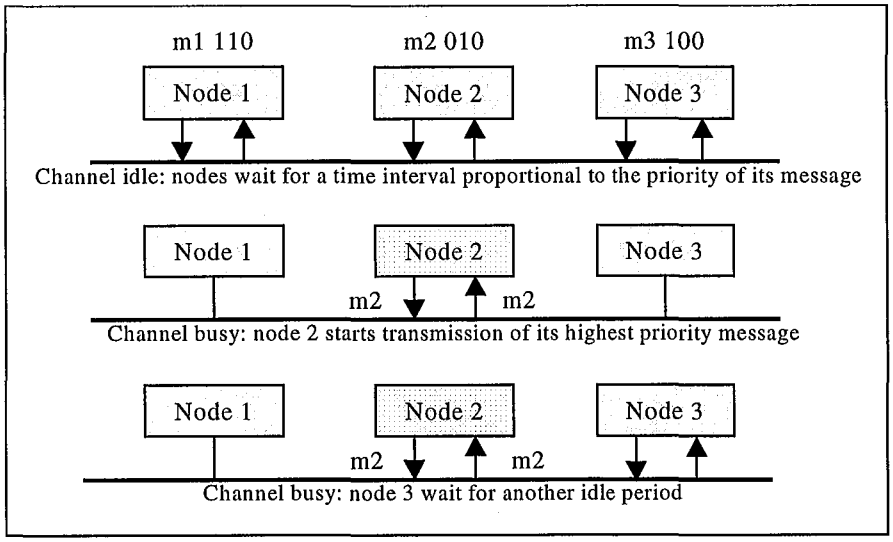

Fig. 3.4. Virtual-time CSMA protocol.

It can be noted that the virtual-time mechanism is used to implement the least laxity scheduling algorithm in the network, but no any bound of the message transfer delay is presented in the previous papers. Moreover, the periodic and aperiodic messages are indistinctly transmitted.

3.4.7 PB/CSMA/CD protocol

In the *Preemption Based CSMA/CD* protocol [Ulusoy, 1995], the messages are classified into two groups based on both their laxities and a predefined threshold. At any given time, a message is considered to be *critical* if its laxity is less than or equal to the threshold, otherwise it belongs to the class of *non-critical* massages.

Message laxities are continuously evaluated and thus at any time a non-critical message can be transferred to the class of critical messages. If the laxity of a message is negative then the message is considered as a lost message and is dropped from the system.

The time is slotted and the nodes only can begin the message transmission at the beginning of each slot. The length of the slot is considered to be equal to the end-to-end propagation delay of the channel. When the channel is idle, the transmission of a critical message can start at the beginning of the next slot. On the other hand, a non-critical message has to wait for the duration of the next slot and its transmission starts at the beginning of the slot after next (if the channel is still idle). This scheme aims to give privilege to critical messages in accessing the channel and to prevent the collision of critical with non-critical messages.

The transmission of a critical message starts by broadcasting a short control message to all other nodes to notify them that the channel is going to be occupied by a critical message and to ensure that its transmission cannot be preempted. However, if the critical message collides (with another critical message) it is retransmitted immediately with probability P_i.

If the message being transmitted is no critical, it is preempted to allow the critical message to be sent. To realize preemption, the message transmission can be interfered by a short noise burst on the channel, so that its node will stop sending the message. Following a collision, a non-critical message is suspended for the duration of a slot and if that slot is not occupied by a critical message then the non-critical message is retransmitted with probability P_i.

3.4.8 CSMA/LDCR protocol

The *CSMA with Laxity based Deterministic Collision Resolution* protocol [Norden *et al.*, 1999] dynamically assigns priorities to the messages based on their laxity (higher priorities are assigned to the messages having smaller laxities). Each node maintains a queue of messages sorted on the increasing order of laxity. Nevertheless, there can be messages with negative laxity, which are then removed from the queue. Therefore the waiting time in the MAC layer is indeterminate and unpredictable and there can be messages not transmitted.

This protocol consists of the following three phases: contention mode, collision resolution mode and least laxity first (LLF) mode.

Contention mode:

a) Each node examines the message at the head of its queue. If its laxity is negative, the message is dropped. Otherwise, the node transmits a notifier of the message, which indicates both the number of slots required by the message and its deadline.
b) If the notifier transmission is successful, the message is transmitted, otherwise if a collision occur all the nodes enter into collision resolution mode by traversing the binary tree in pre-order.

Collision resolution mode:

a) When a node gets its turn, it transmits its notifier. Thus, every node is informed of the message requirement.

b) If the laxity of the first message at the queue of a node is less than a threshold (but non negative) then the message is transmitted, otherwise the message is deferred and pre order traversal continues.
c) Even after the collision resolution tree is completely traversed, if some deferred messages are still to be transmitted, the protocol enters into the LLF mode.

LLF mode:

a) All the deferred messages are transmitted, based on the least laxity order of messages among the nodes.
b) If all the messages are either transmitted or dropped, the next epoch begins.

3.4.9 Token bus protocol

The nodes are connected to a bus but they form a logical ring [IEEE 802.4], i.e. each node knows the address of its adjacent nodes in the ring. A special frame, called *token*, circulates amongst the nodes in such a manner that only one node possesses the token at any time and during a bounded time.

This protocol may meet the bounded message transfer delay requirement if several assumptions are made, for example the token rotation time will be constant if both the token holding time in each node is fixed and the average token passing time is constant. The *token holding time* is the time that a node possesses the token, the *token passing time* is the time between the emission and the reception of the token between two adjacent nodes, the *token rotation time* is the time between two successive receptions of token in one node.

Nevertheless, in [Moon *et al.*, 1998] it is shown that the token can be easily lost in a noisy environment. So, when the token is lost, the token passing time is increased by more than 100 times the nominal token passing time of a noise-free environment. The authors derive a mean token passing time and the mean token rotation time from the detailed token passing operation to evaluate the performance degradation of this protocol.

On the other hand, the IEEE 802.4 protocol allows to locally assign four levels of priority to the messages. Each node has a timer which allows it to control the token time used by each priority level, i.e. when the node receives the token, it sets its timer with the maximum duration associated with the higher priority and begins the transmission of these messages. The node may send the messages of immediately lower priority only if it has still sufficient time to do it. Moreover, when the node passes from one priority to the other it must reset its timer with the maximum duration associated with the new priority.

Ng and Liu [Ng and Liu, 1991] have analyzed by simulation the average message delay and the observed maximum message delay of three token protocols such as the token bus, the token ring and the slotted ring. The *average message delay* is defined as the time between the instant at which the data frame arrives at the output buffer of the sending station and the instant at which the data frame is actually received at the destination station. This delay is calculated varying the data transfer rate while maintaining the total network utilization as a constant for all transfer rate. This is done by increasing the network load according to the

network transfer rate by a load factor. It is assumed that there is no error in transmission and that there is an infinite buffer in each station.

The authors show that the average message delay in the token bus is almost always the smallest among the token ring and the slotted ring protocols. However, the observed maximum message delay of the slotted ring protocol is smaller than those of the token bus and token ring protocols. Because of this, and because the average performance of the slotted ring protocol remains good as the size of the transfer rate of the network increases, the authors recommend to use this protocol with a global priority assignment mechanism in hard real-time systems.

The *slotted ring* protocol is a variant of the token passing protocol by dividing the ring into several fixed size slots. All slots are initially marked as empty, when a node has a message to send, it waits until a slot marked as empty arrives, it marks the slot as occupied and sends the message. After the slot has traversed the ring, the node releases the slot by marking it empty. The slotted ring protocol may use a global priority assignment mechanism with a slight modification. Each slot has a priority which is equal to the highest priority of all the messages waiting to be sent. At the time when the slot is marked as empty, a node can use the slot only when it has a message with the priority equal to or higher than the priority of the slot.

3.4.10 Token ring protocol

The basic idea of the [IEEE 802.5] protocol is as follows: there is a token to grant channel access rights to a node. At network operation, the token is passed from one node to its adjacent node around the ring in one direction. A node must wait for a "free" token before it can transmit. Once a free token is received, it is marked as "busy" and the message transmission is started. At the end of transmission, the sending node releases a new free token, i.e. each sending node manages the current token.

In fact, there is a priority field in the token (the last three bits of the Access Control (AC) byte, as is shown in Fig. 3.5) where the protocol tries to maintain the highest priority of the message that is waiting to be transmitted on the network. A node can seize the token only if it has a message to transmit and if its priority (the last three bits of the Frame Control field of a LLC frame, i.e. the user priority bits) is higher than or equal to the priority of the priority field in the token.

Eight levels of priority are defined in the basic configuration of the token ring (the first three bits of the AC field). The priority of a token is indicated by these bits called the token priority bits. Each level of priority has its own token. At the initial operation of the ring, a token having the lowest priority is put in circulation. At any given time, the ring operates on only one level of priority, called the current priority. Then the current priority token is either in circulation or is blocked by a node transmitting its data.

A node can reserve the next token, if it has data with a priority higher than that required by other nodes, modifying the reservation bits of the token in circulation. When the token returns to its node source, this node tests the reservation bits. If the required priority is lower than or equal to that of the current token, the node continues to transmit its data, otherwise it ceases

transmission and puts in circulation a token with the highest priority requested. A node which creates a token with a priority higher than that of the token seized is responsible for this token and must eliminate it when it is not used any more. The sending node may also ceases transmission if its time of channel utilization has finished or if it does not have any more messages whose priority is higher than or equal to the current priority.

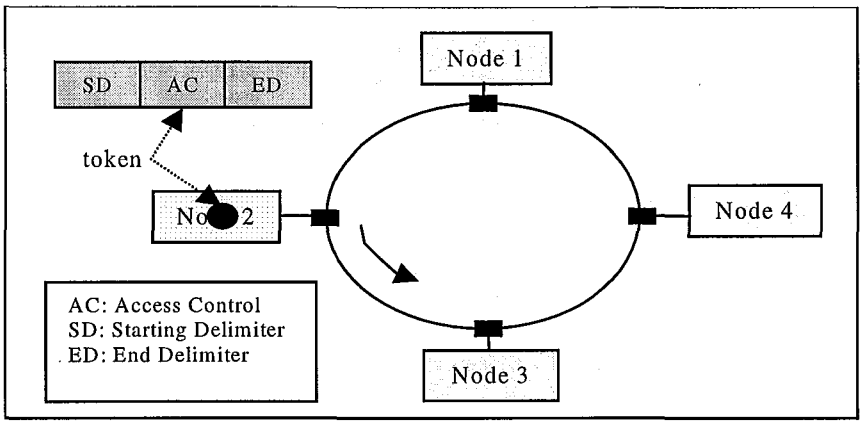

Fig. 3.5. Token Ring protocol.

In order to meet the deadline constraints of synchronous messages and to improve the response time of asynchronous messages, Strosnider *et al.* [Strosnider *et al.*, 1988; Strosnider and Marchok, 1989] have proposed to use a Deferrable Server and the rate monotonic scheduling algorithm. The Deferrable Server assigns higher priority to the asynchronous messages up until the point where the synchronous messages would start to be late.

The authors have also proposed an IEEE 802.5 schedulable unit model. The model assumes that the free token makes a full rotation before being claimed. This allows the establishment of a constant schedulable frame rate which greatly simplifies a priori timing analysis. Nevertheless, two corrections that must be made to the Strosnider's formulae are presented in [Pleinevaux, 1992]. The first correction concerns the duration of the schedulable unit. The second correction pertains to the time that a node must wait for transmitting an alarm (asynchronous traffic). Furthermore, Pleinevaux has drawn the reader's attention on the two following problems that must be handled when implementing this solution: the clock synchronization and the management of soft real time traffic transmission time.

On the other hand, in [Lim *et al.*, 1991; Yao *et al.*, 1995] a worst case analysis of the performance of three token ring protocols is compared. Such protocols are: token passing, priority driven and window protocol. In the *token passing* protocol, a free token is circulating around the ring. If the node wishes to send a message, it captures the token and sends its message. Upon the completion of a message transmission, the node releases the token to the node downstream and the protocol continues recursively.

In the *priority driven* protocol, the token contains a priority field which regulates node's access to the ring, like the token ring protocol. If the number of messages is less than or equal to the number of priority levels, then the messages with distinct deadlines will have distinct priorities. The smaller the deadline, the higher the priority. Clearly, the protocol implements

the Earliest Deadline First (EDF) scheduling algorithm for transmission. Nevertheless, if the number of messages is larger than the number of priorities, then it is possible that several messages having different deadlines are mapped into a single priority. As a result, this protocol can only implement the EDF transmission policy approximately.

In the *window* protocol, the message deadline axis is partitioned into several windows. While the token is circulating around the ring for the first time, information about the number of messages in each window is collected by means of counters in the token. After the first token circulation, if the first non-empty window contains only one message, the node is allowed to send this message, otherwise the window will be further divided into many smaller windows and the protocol recursively uses the token to locate the earliest deadline message.

The *worst case performance ratio* of these protocols shows that no protocol can always dominate the others, and each protocol has its own applicable area in the parameter space defined by the network attributes and the application parameters. Furthermore, each protocol incurs a scheduling overhead proportional to the token node-to-node delay and therefore has deteriorating performance when the token node-to-node delay increases. In addition, the performance of a protocol also depends on the scheduling policy it employs. Potentially, the closer it is to the EDF policy, the better its performance.

The worst case performance ratio [Lim *et al.*, 1991; Yao *et al.*, 1995] is defined as the minimum performance ratio, i.e. the performance ratio of a protocol is defined as the ratio between the number of messages sent by a token ring protocol and an ideal protocol having perfect knowledge about nodes in the ring without experiencing any time delay and using the EDF scheduling algorithm.

In [Yao and Zhao, 1991] an extended IEEE 802.5 protocol suitable for transmitting time constrained messages is studied. In this protocol the laxities of the messages are mapped into priorities. The message with the highest priority is transmitted first. As a result, this protocol approximates the optimal minimum-laxity-first transmission policy. It is found that in the worst case the extended IEEE 802.5 protocol can send at least 50% of the messages sent by the optimal one. Simulation results show that the average performance of the protocol improves as the number of priorities increases.

3.4.11 Timed token protocol (FDDI)

The *Fiber Distributed Data Interface* protocol [ANSI, 1990] has been incorporated into several standards, for example [Malcolm *et al.*, 1996]: the Token bus and the Survivable Adaptable Fiber Optic Embedded Networks (SAFENET).

The idea behind the timed token protocol is to control the token rotation time. As part of the ring initialization process, the nodes negotiate a value for the Target Token Rotation Time (TTRT), which specifies the expected token rotation time. Each node requests a value that is fast enough to support its synchronous traffic needs. The shortest requested time is assigned to a network parameter which specifies the operational TTRT. Each node is assigned a portion of TTRT, known as its synchronous bandwidth (H_i), which is the maximum time that a node is permitted to transmit synchronous messages every time it receives the token. A node

can transmit asynchronous messages only if the token arrived earlier than expected. In FDDI terminology, the traffic that is assigned guaranteed bandwidth is called synchronous traffic; all other traffic is called asynchronous.

In [Johnson, 1987; Sevcik and Johnson, 1987] it is formally proved that when the token rotates quickly enough to prevent initiation of recovery unless there is failure of a physical resource or unless the network management entity within a station initiates the recovery process, the maximum token rotation time between two consecutive visits to a node is bounded by twice the Target Token Rotation Time, i.e., 2 TTRT. This result has been extended in [Agrawal *et al.*, 1992] to any v ($v \geq 2$) consecutive visits to a particular node, i.e., v TTRT. A tighter bound has been proposed in [Zhang and Burns, 1995]

The synchronous bandwidth allocation scheme is an algorithm which allocates values for each synchronous bandwidth (H_i) [Agrawal *et al.*, 1992; Agrawal *et al.*, 1993]. The schemes can be divided into two classes based on the type of information used: local allocation schemes and global allocation schemes. The former uses only information available locally to a node in allocating H_i. A global synchronous bandwidth allocation scheme uses global information in allocating synchronous bandwidth to a node.

In a global scheme, if the parameters of a node change, it may be necessary to compute again the synchronous bandwidths at all nodes. Thus, a global scheme might not be well suited to a dynamic environment. Furthermore, the extra information used would create more overheads in the network than a local scheme. On the other hand, in a local scheme, if the parameters of a node change, only the synchronous bandwidth of this node needs to be recalculated. This makes a local scheme flexible and more suitable for use in dynamic environments.

A local scheduling mechanism for each FDDI node has been proposed in [Song and Simonot, 1996]. Three local scheduling policies were examined: non-preemptive earliest deadline first, non-preemptive rate monotonic and smallest period first using overwrite prioritized buffers. The simulation results show that the smallest period first has good performance and is simple to implement.

3.4.12 TDMA protocol

In the *Time Division Multiple Access* protocol, the time is divided into fixed length intervals or frames; each frame is further subdivided into slots. The method may be synchronous or asynchronous according to the number of slots [Kurose *et al.*, 1984].

In the *synchronous time division* method, the number of slots is equal to the number of nodes in the network, so that each node periodically uses the channel during its slot and only can transmits during this time interval. If a node does not have any message to send, or if the node is not in the network, its slot will be unused. Given that the slot size is fixed and that the number of nodes is known, this method meets the bounded message transfer delay requirement. However, the waiting time in the MAC layer of the sending node may be very long according to the number of nodes in the network.

In the *asynchronous time division* or *slot-switched* method, the number of slots is fewer than the number of nodes, the slot allocation may be centralized or distributed. It is centralized if there is a privileged node which allocates a slot for each node, the slot is requested during an allocation phase. It is distributed if there is not a privileged node, so each node uses the channel during a pre-allocated slot. Once a node is assigned to the *i*th slot in a frame, the node has sole access to the channel during the *i*th slot of every subsequent frame until it releases explicitly the slot [Kurose *et al.*, 1984]. This method does not meet the bounded message transfer delay requirement because there can be blocked nodes waiting for free slots.

Several performance issues of the TDMA protocol are presented in [Simonot and Song, 1997], specially message sojourn time and message overflow. The authors show how to use these results to design protocol parameters in order to meet the soft real-time requirements. The message sojourn time only depends on the number of messages that the average message finds upon arrival. The average message is defined as a message randomly selected from the set of messages waiting for transmission. The message overflow can occur for limited buffer capacity in each node. So, the percentage of messages that are transmitted successfully (prior to their deadlines) can be obtained through the probability distribution of the sojourn time.

3.4.13 TTP protocol

The *Time-Triggered Protocol* [Kopetz and Grünsteidl, 1994] is used in the time-triggered architecture where the information about the behavior of the system is known a priori - at design time - by all the nodes (dispatcher table). Access to the communication channel is determined by a time-division multiple-access (TDMA) method. Each node is allowed to send messages only during a predetermined time duration, called its TDMA slot. The sequence of periodic TDMA slots is called a TDMA cycle.

3.4.14 P-NET

It uses a hybrid medium access method, i.e. a distributed method according to the principle of virtual token among the master nodes and a centralized method according to the master-slave principle: a master sends a request and the addressed slave returns a response [DS 21906; EN50170-1].

The *virtual token* method is as follows: There are two counters in each master node, an idle-bus-bit-period-counter and an access-counter. The former counts the number of "1"s on the bus and is cleared when a "0" appears on the bus. The access-counter increments when the idle-bus-bit-period-counter is equal to 40,50,60.... and is set to 1 when the maximum number of masters is reached, as is shown in Fig. 3.6. When the access-counter becomes equal to the node address of a master, this master is allowed to access the bus (gets the token) in the interval between 2 and 7 bit periods after the match. Each master is allowed to send only one request each time it gets the token (cyclic priority). After a given master's bus access, the token is passed to the next master in a cyclic fashion. The passing of the virtual token takes place within 10 bit periods, and no data is actually sent over the bus.

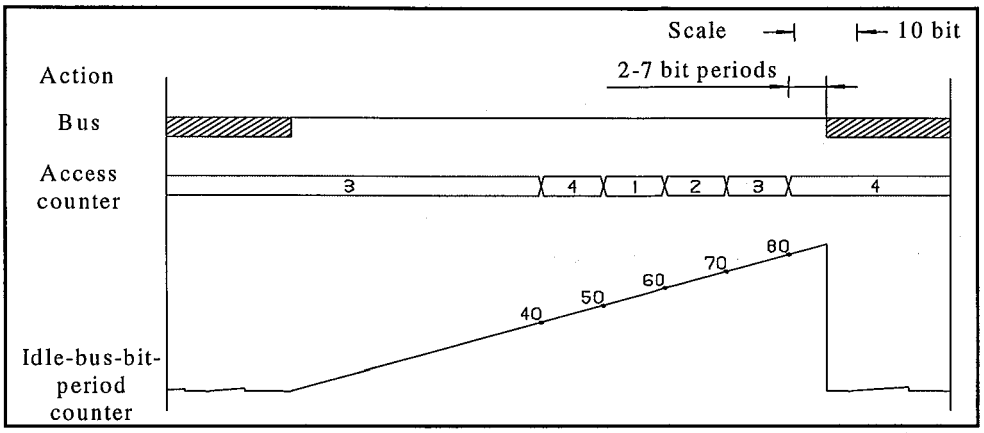

Fig. 3.6. P-NET protocol.

All global variables in a master node are accessed through a Softwire List (SW list). This also applies to all global variables in the slave nodes. All communication to a slave node is based on the SW list located in the slave. The SW list in the slave node describes both the data types (byte, integer, real, etc.) and the physical addresses of the global variables. Variables are addressed with a logical address called the SoftWire number (SWNo) which is an entry in the SW list in the slaves. When an entry in the SW list of a master node defines an external variable, the SW list holds both the node addresses and the SWNo. The SW list is generated from the global variable declaration performed in the application program when it is compiled.

In [Tovar *et al.*, 1999] the pre-run-time schedulability conditions for supporting real-time traffic with P-NET networks are established. Essentially, formulae to evaluate the upper bound of the end-to-end communication delay in P-NET messages are provided. Using this upper bound, a feasibility test is then provided to check the timing requirements for accessing remote process variables.

3.4.15 PROFIBUS

The *PROcess FIeld BUS*, Fieldbus Message Specification (FMS) and Decentralized Peripherals (DP) [DIN 19245-1,2,3 and 4; EN50170-2], uses a hybrid medium access method: a distributed method according to the principle of token passing among the master nodes and a centralized method according to the master-slave principle. The nodes are then divided into master and slave as follows:

- *DP-Master* (class 1), which cyclically polls the assigned DP-Slaves according to the Master's Data Base and handles the user data exchange.
- *DP-Master* (class 2), which acyclically communicates with the DP-Slaves and interacts as a configuration or diagnostic tool.
- *DP-Slave*, which may be associated with both types of DP-Master, i.e. class 1 and class 2.
- *FMS-Master*, which cyclically or acyclically polls the associated *FMS-Slaves* according to its Poll List.

The hybrid media access allows master-slave communication as well as master-master communication. The latest is used for data transfer between DP-Master (class 1) and DP-Master (class 2). The initiator of a master-slave communication is always a DP-Master. The initiator of a master-master communication is always a DP-Master (class 2). No communication is defined between DP-Masters of the same class.

The exchange of messages takes place in cycles. A *message cycle* consists of both a master station' action frame (request or send/request frames) and the associated acknowledgement frame or response frame of either a master or a slave station. User data may be transmitted in both the action frame (send frame) and the response frame. The acknowledgement frame does not contain any user data. The complete message cycle is only interrupted for token transmission and for transmission of data without acknowledgement (e.g. necessary for broadcast messages). All stations except the respective token holder shall in general monitor all requests. The stations acknowledge or respond only when they are addressed.

In Profibus, the modes of transmission operation define the time sequence of the message cycles. Four modes are identified and are described below: (1) Token handling; (2) Acyclic request or send/request; (3) Cyclic send/request, polling; (4) Registration of stations.

3.4.15.1 Token handling mode

The token is passed from master station to master station in ascending numerical order of station addresses and thus determines the instant at which each master station may access the medium. A DP-Slave station needs a master-request to exchange information.

Profibus-DP protocol defines two priority levels for the messages: high and low. All request messages are sent with high priority (master-slave communication). If there is new diagnostic information in a DP-Slave the response messages, in the polling mode, will be replied with high priority. In all other cases, the replies are sent as low priority messages.

In the service classes for the message cycles, the user of the FDL interface (i.e. the application layer of Profibus-FMS) may choose two priorities: low and high. The priority is passed to the FDL with the service request.

When a master station receives the token, it always performs all available high priority message cycles first and then the low priority ones. If at token reception time, the Real Rotation Time (T_{RR}) is equal to or greater than the Target Rotation Time (T_{TR}), only one high priority message cycle including retries, in the case of an error, may be performed. Then the token shall be passed to next station immediately.

In general, after token reception or after the first high priority message cycle, the following is to be considered: high priority or low priority message cycles may be carried out only if at the beginning of the execution T_{RR} is less than T_{TR} and thus the Token Holding Time $T_{TH} = T_{TR} - T_{RR}$ is still available. Once a high or low priority message cycle is started, it is always completed, including any required retry, even if T_{RR} reaches or exceeds the value of T_{TR} during the execution. The prolongation of the T_{TH} automatically causes a shortening of transmission time for message cycles at the next token reception.

An accurate evaluation of the maximum Profibus token cycle time is presented in [Tovar and Vasques, 1999] considering that all types of traffic are allowed. It is the basis for the setting of the T_{TR} parameter in order to guarantee the timing requirements of the high priority messages. In this work, it is shown that the maximum token cycle time is a consequence of the overrun of the token holding timer in a master, and it is also shown how such overrunning impacts the cycle time properties of this protocol.

3.4.15.2 Acyclic request or send/request mode

In this mode, single message cycles are conducted sporadically. A master station initiates this mode due to a local user's request upon receipt of the token. If there are several requests, this mode of operation may be continued until the maximum allowed token rotation time expires.

3.4.15.3 Cyclic send/request mode

When polling, the master station cyclically addresses stations with a given request according to a predefined sequence, called the Poll List. All slave and master stations to be polled are marked in this list. After receipt of the token, handling of the Poll List (Poll Cycle) is only started after all requested high priority message cycles have been carried out. After each complete Poll Cycle the requested low priority message cycles are performed in turn. The order in which the message cycles are performed obeys the following rules:

If the Poll Cycle is completed within the Token Holding Time (T_{TH}), i.e. there is still T_{TH} available, the requested low priority message cycles are carried out in turn as far as possible within the remaining T_{TH}. A new Poll Cycle starts at the next receipt of the token which has T_{TH} for low priority message cycles available.

If at the end of a Poll Cycle there is not any further T_{TH} available, the requested low priority message cycles are as far as possible processed at the next token receipt that has available T_{TH} for low priority message cycles. After that, a new Poll Cycle starts as described above.

If a Poll Cycle takes several T_{TH}, the Poll List is processed in segments, but without inserting requested low priority message cycles. Low priority message cycles are carried out only at the end of a complete Poll Cycle, as described above. The Poll Cycle time, i.e. the maximum station delay time, depends on the duration of a message cycle, on the token rotation time, on the length of the Poll List and on the underlain low priority message cycles.

3.4.15.4 Registration of stations mode

At token receipt the mode starts if the local user requests a Live List via management and after performing any previously requested low priority message cycles. During polling the mode is performed between Poll Cycles. According to the given address space each possible station is addressed once, except the master stations registered in the List of Active Stations (LAS). The correctly responding stations, i.e. those which acknowledge positively and the master stations of the LAS are entered in the Live List as existing master or slave stations.

3.4.16 WorldFIP

The *World Factory Instrumentation Protocol* [Galara and Thomesse, 1984; EN50170-3] defines a Bus Arbitrator (BA) which "gives permission to speak" to each information producer. The application layer sends to the BA, at the data link layer, a set of service request primitives that describe the execution of both the basic cycles and the macro cycles (the chaining of one or more basic cycles) [Thomesse, 1993].

A basic cycle is composed of at least one window, called the periodic window, and at most of the following four windows: a periodic window, an aperiodic variable window, an aperiodic message window and possibly a synchronization window to adjust the constant duration of the basic cycle, as are shown in Fig. 3.7.

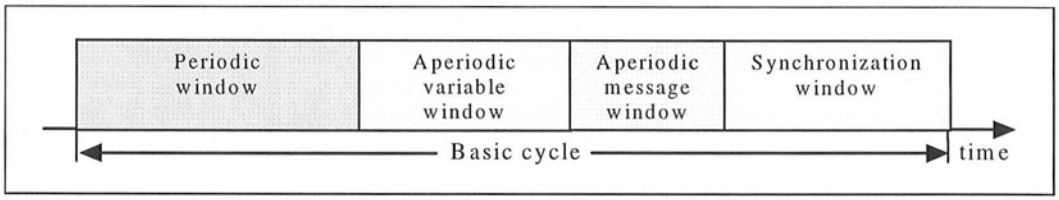

Fig. 3.7. Basic cycle of the WorldFIP protocol.

Each variable is identified by an unique identifier and each node can be a producer (P) and/or consumer (C) of one or more variables. During the periodic window, the BA reads a list of periodic variables and injects each variable identifier onto the network (Fig. 3.8.a), each variable has only one producer (Fig. 3.8.b). Consumers needing to utilize the variable, alerted by the identifier, store and use the value broadcasted by the producer (Fig. 3.8.c).

Whenever an aperiodic variable or an aperiodic message are produced, the producer utilizes the response, of a received periodic variable identifier, to request its transmission. The bus arbitrator stores the identifier, that is carried by the request, into the appropriate queue (Fig. 3.8.c). After completing the current periodic window, the requested aperiodic variables and the requested aperiodic messages are then handled in the same way that the periodic variables within the proper windows, where the relevant producers reply current values.

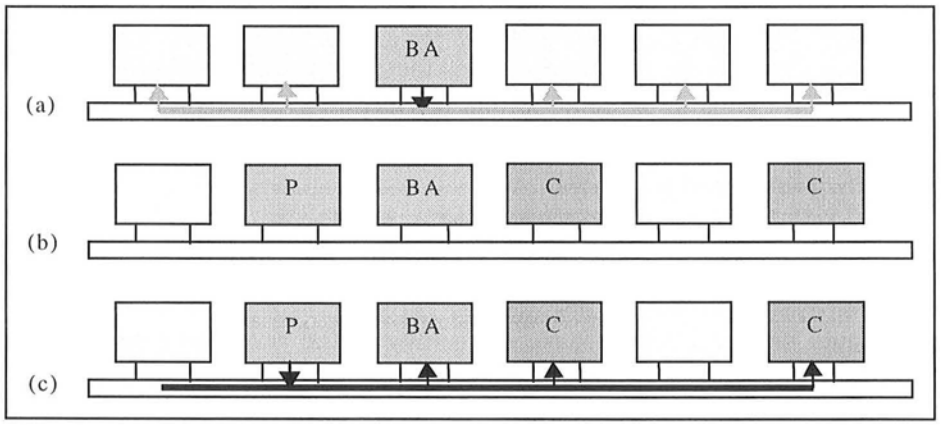

Fig. 3.8. WorldFIP protocol.

In [Song, 1991] a performance evaluation of FIP is presented using the following approaches: analytic and simulation. In the former approach, the services of the data link layer are modeled by a queuing system which is composed of two stages in parallel-series with intermittent servers. In the second approach, a simulator describing the data link layer in simulation language QNAP2 was developed. From this work, it is possible to compute the message transfer delay, to build the list of periodic variables and to compute the size of the windows in a basic cycle.

It is to be noted that *WorldFIP* is a typical example of *centralized* and *static* resource reservation method, i.e. there is one node, the bus arbitrator, having different lists of identifiers that define at any moment the communication requirements of all the nodes. These requirements are known in advance and will not change during system operation. Hence, it is possible to assign an identifier to each variable and message.

3.4.17 Foundation Fieldbus

It uses a centralized access method where there is a bus scheduler, called the Link Active Scheduler (LAS). Two types of nodes are defined: Link Masters and Basic Devices. The Link Masters are capable of becoming the LAS, the Basic Devices do not have this capability.

The LAS has a list of transmission times for all periodic messages in all the nodes. When it is time for a node to send a periodic message, the LAS transmits a Compel Data message to the node (Fig. 3.9.a). Upon reception of this message, the "Publisher" node (P) broadcasts or "publishes" the data stocked in its buffer to all the nodes. Any node configured to receive the data is called a "Subscriber" node (S) (Fig. 3.9.b).

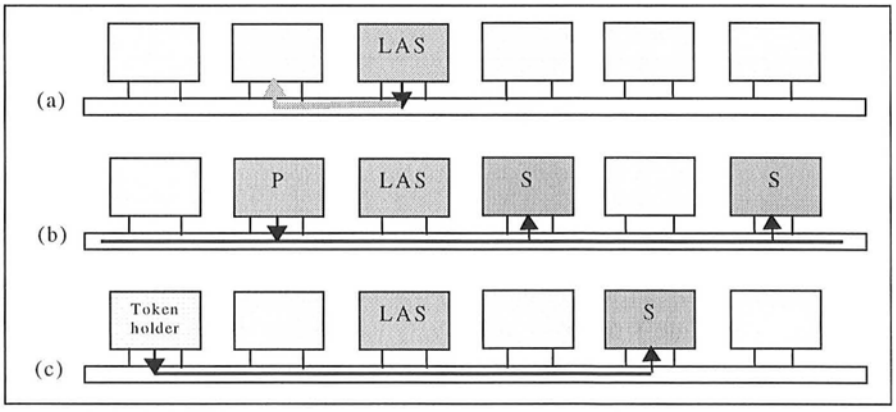

Fig. 3.9. Foundation Fieldbus protocol.

If there is time to do something before the next Compel Data message, the LAS may do as follows, otherwise transmits idle messages while waiting:

- The LAS passes a token message to each node. When a node receives it, the node is allowed to send aperiodic messages until it has finished or until the maximum token holding time has expired, whichever is the shorter time (Fig. 3.9.c). If the node has nothing to send, it immediately returns the token to the LAS.

- All the nodes, properly responding to the pass token message, compose a list called the Live List. The LAS, after completion of a pass token cycle, sends Probe Node messages to the node addresses that are not in the Live List, if a node answers, the LAS adds the node to the Live List and confirms its addition.

- The LAS periodically broadcasts a Time Distribution message. So, all the nodes have the same data link time.

3.4.18 ControlNet

It uses the producer-consumer model and the Concurrent Time Domain Multiple Access (CTDMA) method. In this, the network time is divided into time windows, called Network Update Intervals (NUI), which are further divided into the following three windows or portions: scheduled, unscheduled and network maintenance, as are shown in Fig. 3.10.

During the scheduled window, each configured note transmits one and only one time-critical data in a given interval. During the unscheduled window, the nodes transmit one or more non time-critical data according to the network load. In this window, the transmission order is rotated (round robin) in each NUI. Finally, the network maintenance window is used as "guardband" to allow the constant duration of the NUI.

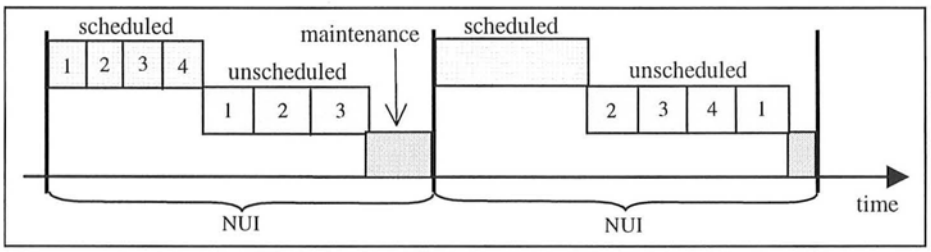

Fig. 3.10. ControlNet protocol.

3.4.19 SwiftNet

It uses the producer-consumer model and the TDMA protocol. SwiftNet was created to satisfy Boeing Commercial Airplane's need for a truly synchronous data bus.

Time on the bus is divided into equal length intervals called *slots*. Each node listens to traffic and may send traffic only during a slot assigned to it. A local link has 2 BL slots during one bus cycle (typically, BL = 16).

Bus access, data transmission and data reception is coordinated by each node based on a shared perception of which channel and slot occupy now the bus. Each data (variable measured by a sensor or stimulus sent to an actuator) uses a dedicated *channel*, which is configured for one or more slots per bus cycle. When one of the slots, assigned to one of the channels in a node, is noted by that node's bus-cycle clock, the node either transmits or receives (as assigned) the data. Each slot may be assigned to only one channel. Each channel may be scanned one or many times during each bus-cycle.

3.4.20 INTERBUS

It uses the master-slave model and an access method called summation-frame. This method combines the data of all the devices into one message, which is sent to all devices in the ring, as is shown in Fig. 3.11, it guarantees that the process image is consistent between all devices since all input data stems from the same scan time and all output data from the devices are received at the same time.

The summation-frame can be seen as a special class of TDMA protocol, this means that every device has been allocated a time slot. Additional time slots can be defined for connection-mode data blocks "on demand". In this way, data blocks of several Mbytes can be transmitted without altering the cyclically short scanning reference for the process data types.

The summation-frame is realized by means of a register structure. Every device joins the ring via a register, the length of which is determined by the amount of data points of the device. By coupling all the devices together, a ring is created, the length and structure of which corresponds exactly to the configuration of the user data field in the summation-frame message. The output data for the devices is placed in the output buffer of the master according to the physical order of the connected output devices. A transmission cycle begins with a data sequence. This data sequence contains the loop-back word on the transmission side, followed by the output data.

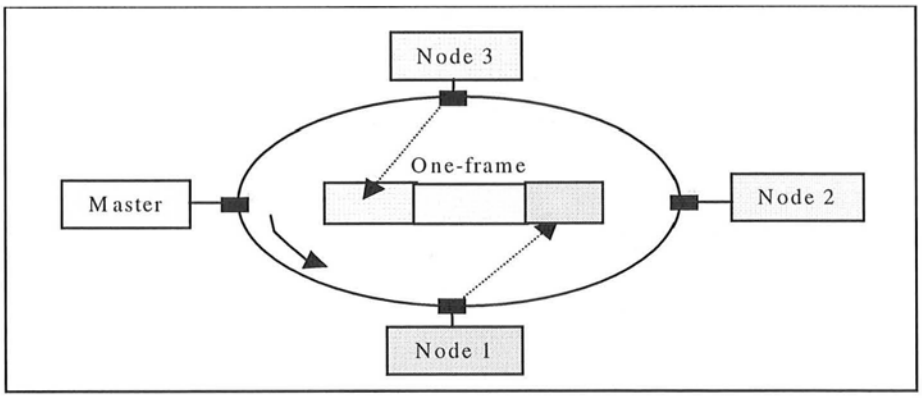

Fig. 3.11. Interbus protocol.

During data output, the return flow of process information is simultaneously entered into the input buffer of the master as input data. After output of the entire summation-frame and simultaneous input of the same frame, all output data is correctly positioned in the individual devices.

In addition to the data cycle for the transmission of user data as described above, an identification cycle is also defined in the data link layer of Interbus. This cycle acts as the bus management. Each bus device has a layer 2 identification code, which gives information as the type of device and its range of user data. The independent configuration of the bus system is carried out by a sequence of identification cycles, which are initiated by the bus master in order to read the identification of the remote devices. Using the identification code read, the summation-frame configuration for the data cycle can be established.

3.4.21 TS 61158

The TS 61158 MAC protocol is based on a centralized access method and uses a timed-token mechanism to organize the access to the common bus. The nodes are classified into the following three functional classes: basic class, link master class and bridge class. The last two classes enable a node to assume the Link Active Scheduler (LAS) role, each other node is called Data Link Entity (DLE), which can be a publisher or subscriber node.

The LAS has an initial schedule, set by the management system, in which periodic activities are scheduled in the same way that Foundation Fieldbus (Fig. 3.9), i.e. the LAS transmits a Compel Data message to the publisher node. Upon reception, the publisher broadcasts or publishes the data to all the nodes. Any node configured to receive the data is called subscriber.

On the other hand, the LAS uses the notion of Delegated Token (DeT) to transfer the right to transmit to another DLE and to execute either its periodic or aperiodic services for a specified duration time (Fig. 3.12.a). The DLE which owns the DeT becomes the only master of the link (Fig. 3.12.b) and itself can delegate this right to another DLE, creating a Reply Token (ReT), with which it can compel the sending of a data or an acknowledgment. When the delegated duration expires, the DeT holding node has to send back the token to the LAS (Fig. 3.12.c).

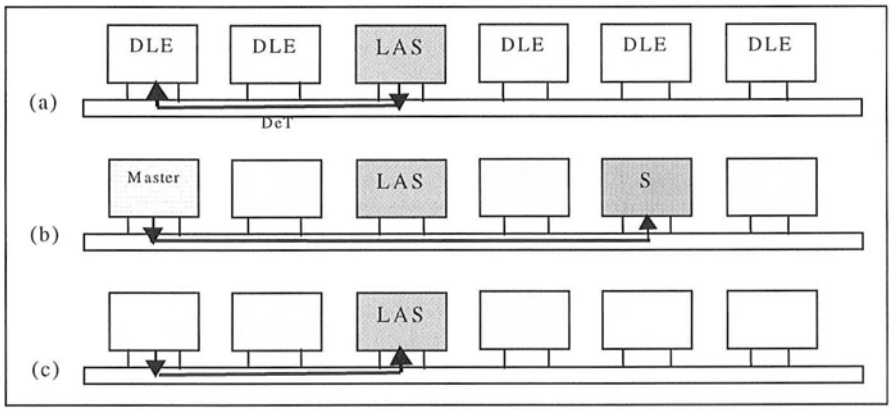

Fig. 3.12. TS 61158 MAC protocol.

The LAS uses the time between the end of the actual periodic delegation and the next scheduled periodic activity to allow the execution of management related services as well as the execution of asynchronous activities by all the active nodes. To do that, the LAS first executes the link-related management activities for a predefined duration and then emulates a "circulated token" mechanism by sending a DeT to all active nodes, in turn, starting from itself. Each time a node receives the DeT, it has to give it back to the LAS after either expiration of the delegation duration or completion of its asynchronous activities execution.

The LAS uses an array of Maximum Token Holding Time (MTHT), with an entry for each node specifying its MTHT during one cycle of token circulation. When it is time for the next scheduled activity, the LAS interrupts the token circulation, executes the scheduled activity and then resumes the token circulation process.

This protocol provides the following three priority levels: urgent, normal and time-available. Usually, urgent priority is assigned to time critical data such as alarms. Normal priority is assigned to event based data, diagnosis data, parameter settings, etc. Time-available priority is assigned to trend data, program download data and so on. The permissible priority level is set at the beginning of each token circulation cycle and is used in each delegation. At the beginning of a new cycle, the LAS uses the Target Token Rotation Time (TTRT) configuration parameter and measures the Actual Token Rotation Time (ATRT), i.e. the time elapsed between two successive token delegations, to adjust the priority level. If the ATRT is bigger than the TTRT, the priority level is increased, otherwise it is decreased. The protocol defines a minimum delegation time, common to all the nodes, below which a delegation does not take place.

In [Mnaouer *et al.*, 1997] an asynchronous bandwidth allocation mechanism is proposed in order to ensure both that the maximum delivery bounds of urgent data are not violated and that the maximum token circulation time is kept within bounds. In this approach, first the allocable asynchronous bandwidth is derived and then a modified version of the normalized proportional allocation scheme is proposed in order to determine the bandwidth shared by each node, i.e. the maximum token holding time. This approach is evaluated by analytical and simulation analysis. The results show that the asynchronous bandwidth allocation scheme and the parameter setting policy can ensure both, in one hand, that the delivery delay bounds of urgent messages are not violated and that the maximum token cycle time is kept within bounds, and on the other hand, the results show that parameter tuning can further lead to improve the protocol performance.

3.5 Conclusion

As it was mentioned in the previous chapter, in the real time applications each user (human users, sensors, actuators and so on) has different communication requirements, specifically time-related requirements. On the other hand, the communication systems have been designed in order to satisfy some user requirements.

The goal of this chapter is to identify which time-related requirements are met, and how they are met, by each MAC protocol. This chapter presents then an analysis of the main Fieldbus MAC protocols and MAC protocols from two points of view, on one hand, the analysis takes into account the user time-related requirements such as the bounded message transfer delay, the periodicity, the jitter, the temporal coherence and the spatial coherence. On the other hand, the analysis takes into account the characteristics of the Fieldbuses and MAC protocols identifying the access methods and the scheduling algorithms that fulfill these requirements.

A summary is presented in Table 3.4, where *True* indicates that the time-related requirement is met, and *False* otherwise. *Periodic access* indicates whether the protocols periodically access to the medium. *Access method* summarizes the methods used to access the medium. *Jitter* indicates whether the variation in the periodic access is bounded. *Privileged traffic* indicates whether the periodic messages are privileged for transmission or whether the periodic and aperiodic messages are indistinctly transmitted. *Scheduling algorithm* summarizes the policies to transmit the messages: Rate monotonic (RM), Earliest deadline (ED), Least laxity (LL) and Smallest Period First (SPF) policy which is equal to RM.

Bounded message transfer delay indicates whether or not it is bounded. *Reference* indicates either the paper where the bound of the message transfer delay is presented or the paper where the scheduling algorithm is proposed. Note that only the first author is given but the complete reference is given in the Bibliography section.

From table 3.4 it can be noted that some Fieldbus networks meet most of the user time-related requirements defined in the Chapter 2, whereas few of the MAC protocols meet these requirements. Moreover, the user requirements are met within different levels, for example if the bounded message transfer delay requirement is met, the table shows that the published bound can be an average, a maximum or the worst-case. This table should be extended to identifier the different levels within which the periodicity and the jitter requirements are met.

It can also be noted that most of the Fieldbus networks make use of methods that allow take into account the user requirements at network configuration time in order to meet them and guarantee them at network operation time, as it is shown in the Tables 3.1, 3.2 and 3.3. Nevertheless, if the user requirements change, it is necessary to stop the industrial application in order to determine whether the new set of requirements can be met and guaranteed. If yes, the user requirements are configured and the industrial application is restarted. However, many of these applications cannot be stopped without great losses and then on-line methods are required. One possible solution is the resource reservation method, which is part of a larger concept known as Quality of Service and it will be analyzed in the next Chapter.

MAC protocols and Fieldbuses	*Periodic Access*	*Access method*	*Jitter*	*Privileged Traffic*	*Scheduling Algorithm*	*Bounded Message transfer delay*	*Reference*
CSMA/CD	False	Probabilistic	False	Indistinctly		False	IEEE 802.3
Window-CSMA	False	Time window	False	Indistinctly	Least laxity	False	Kurose,84; Zhao,90;Znati,91
Virtual-time CSMA	False	Virtual time	False	Indistinctly	Least laxity	False	Molle,85; Zhao,87
CSMA/DCR	False	Binary tree	False	Indistinctly		True (worst-case)	Le Lann, 94
DOD/CSMA/ CD	False	Binary tree	False	Indistinctly	Earliest deadline	True	Le Lann, 94
CSMA/LDCR	False	Binary tree	False	Indistinctly	Least laxity	False	Norden, 99
PB/CSMA /CD	False	Slotted time and preemption	False	Indistinctly	Least laxity	False	Ulusoy, 95
TDMA (single slot per node)	True	Time multiplexing	True	Indistinctly		True (average)	Simonot, 97
TTP	True	Time multiplexing	True	Periodic	Dispatcher table	True (average)	Kopetz, 94
IEEE 802.4	True	Token and local priorities	True	Indistinctly		True (average)	Moon, 98
IEEE 802.4	True	Token and local priorities	True	Indistinctly		True (maximum)	Ng and Liu, 91
IEEE 802.5	False	Token and global priorities	False	Periodic and deferrable server	Rate monotonic	True	Strosnider, 89; Pleinevaux, 92
IEEE 802.5	False	Token and global priorities	False	Indistinctly	Earliest deadline	Worst case performance ratio	Lim *et al.*, 91
IEEE 802.5	False	Token and time window	False	Indistinctly	Earliest deadline	Worst case performance ratio	Lim *et al.*, 91
IEEE 802.5	False	Token and global priorities	False	Indistinctly	Least laxity		Yao and Zhao,91
IEEE 802.5	False	Token and global priorities	False	Indistinctly		True (maximum)	Ng and Liu, 91
Slotted ring	False	Token and global priorities	False	Indistinctly		True (maximum)	Ng and Liu, 91
FDDI	True	Token	True	Periodic		True (worst-case)	Johnson, 87; Agrawal, 92; Zhang, 95
FDDI	True	Token	True	Periodic	Local: ED, RM, SPF	True (worst-case)	Song, 96
WorldFIP	True	Bus arbitrator	True	Periodic		True (average)	Thomesse, 93 Song, 91
Profibus	True	Token and master-slave	True	Periodic		True (maximum token cycle time)	Tovar and Vasques, 99
Interbus	True	Master-slave and one frame	True	Periodic		True	
CSMA/CA (CAN)	False	Global priorities	False	User defined		True (worst-case)	Tindell, 95a; Navet, 00
P-Net	True	Master-slave and virtual token	True	Periodic (Softwirelist)		True (upper bound)	Tovar *et al.*, 99
Eibus Batibus	False	Global priorities (CAN)	False	User defined		True (worst-case)	
Control Net	True	Time multiplexing	True	Periodic		True	
LON	False	CDMA/CD	False	Indistinctly		False	
Foundation Fieldbus	True	Link Active Scheduler	True	Periodic			
TS 61158	True	Link Active Scheduler and token	True	Periodic		True (maximum token cycle time)	Mnaouer *et al.*, 97

Table 3.4. Characteristics of the services provided by the Fieldbuses and MAC protocols.

Chapter 4

Quality of Service in the Communication Systems

Best-effort services and guaranteed services are two types of service that a communication system must provide, and are part of a larger concept known as *Quality of Service* (QoS), i.e. the set of qualities related to the provision of a service, as they are perceived by a client.

Several QoS architectures have been proposed for different communication systems, a survey of these architectures is presented in [Aurrecoechea *et al.*, 1998]. This chapter analyzes three QoS architectures: The OSI QoS Framework [ISO 13236], the Fieldbus QoS architecture [IEC 61158] and the Internet QoS architecture [Braden *et al.*, 1994; Blake *et al.*, 1998; Bernet *et al.*, 1999; Ghanwani *et al.*, 1999; Yavatkar *et al.*, 2000; Seaman *et al.*, 1999].

4.1 OSI QoS Framework

The OSI QoS Framework [ISO 13236] defines terminology, concepts and provides a model of QoS for OSI. This model is based on the concepts of the OSI Reference Model and those of the OSI Management Framework.

The relationships among the QoS concepts are shown in Fig. 4.1 and are briefly described below:

- *QoS characteristics*, which are the quantifiable aspects of QoS. The characteristics can be generic, specialized or derived. A *generic* characteristic is defined independently of what it is applied (e.g. time delay). A *specialized* characteristic is a specialization of one generic characteristic that may or must be applied in order to make the characteristic concrete and usable in practice (e.g. end-to-end time delay). A *derived* characteristic is defined as a (mathematical) function of others (e.g. mean end-to-end time delay).

  The [ISO 13236] defines the followings QoS characteristics of a system: temporal, performance, coherence, capacity, integrity, safety, security and reliability.

- *QoS requirements*, which are the quantified user requirements. The QoS requirements can be expressed as either the QoS parameters (when the requirements need to be conveyed between communication entities) or the QoS context (when they are retained in a communication entity). The QoS parameters are conveyed to all the entities involved in providing the service in order to provide both the top-to-bottom QoS and the end-to-end QoS. Entities that receive QoS requirements analyze them and determine the QoS management functions or mechanisms that are required to meet them.

- *QoS categories*, which represent a policy governing a group of QoS requirements specific to a particular user or application.

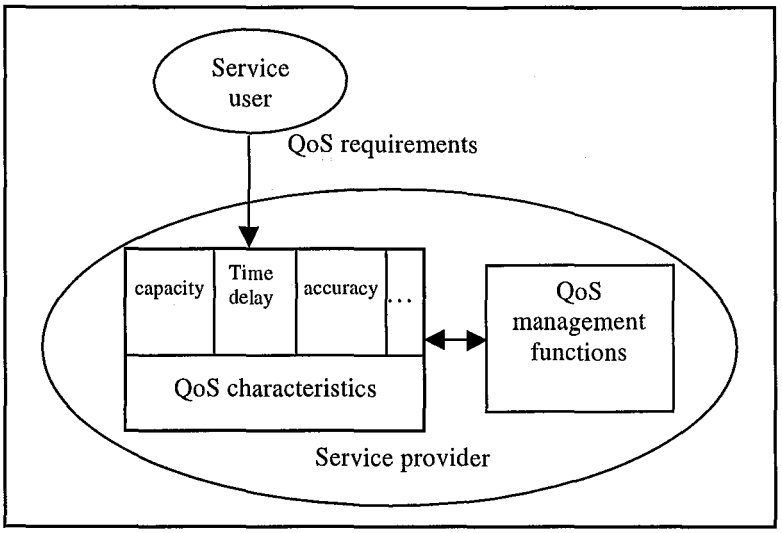

Fig. 4.1. Relationships among the OSI QoS concepts.

- *QoS management functions and QoS mechanisms*, which are all the activities related to the monitoring, the control and the administration of QoS. A QoS Management Function (QMF) is designed to meet one or more QoS requirements. QoS mechanisms are the components of a QMF and are performed by one or more entities in order to support establishment, monitoring, maintenance, control and inquiry. The operation of a QoS mechanism may be either local processing or it may involve further generation of QoS requirements and then to communicate them to other entities.

  The QMFs are organized in terms of three phases of QoS activity, described below:

  - *Prediction phase* is concerned with establishing the QoS context by making relevant inquiries and performing analysis to predict the QoS characteristics of the system.

  - *Establishment phase*, during this phase the concerned entities express requirements for QoS, enter into negotiations and re-negotiations, make agreements on the QoS to be delivered and on the actions to be performed if it degrades, and initiate mechanisms to support the operational phase.

    Three different levels of agreement can be negotiated for any QoS requirement:

    - *Best effort*, all the parties do their best to meet the user requirements but there is no assurance that the QoS will be provided. In this level, some QoS parameter values may not be specified or some bounds in the deterministic or statistical representation cannot be satisfied.

    - *Compulsory*, in this level the achieved QoS must be monitored by the provider and the service is aborted if it degrades below a compulsory level. The desired QoS is not guaranteed and may be deliberately degraded to meet a guaranteed agreement.

- *Guaranteed*, the QoS requirements are met, as they are specified through the QoS parameters and expressed as either deterministic or statistical requirements. Deterministic values can be given as follows: single value, an upper or lower limit, an upper or lower threshold, an operating target, etc. Statistical values can be given as maximum, minimum, range, mean, variance and standard deviation, etc.

- *Operational phase*, its purpose is to honour the agreements made during the establishment phase and to take the appropriate actions whenever it is not further possible. In this phase, the entities perform QoS monitoring, QoS maintenance and QoS inquiries.

4.1.1 OSI QoS architecture

The model of QoS for OSI (shown in Fig. 4.2) considers two classes of entities that take part in the management of QoS. These classes are as follows: layer-specific and system-wide entities. The *layer-specific QoS entities* are associated with the operation of a (N)-subsystem, they coordinate the response to the requirements of the (N)-service user and implement direct control of the protocol entities. The *System-wide QoS entities* interact with the layer QoS entities to monitor and to control the performance of the system.

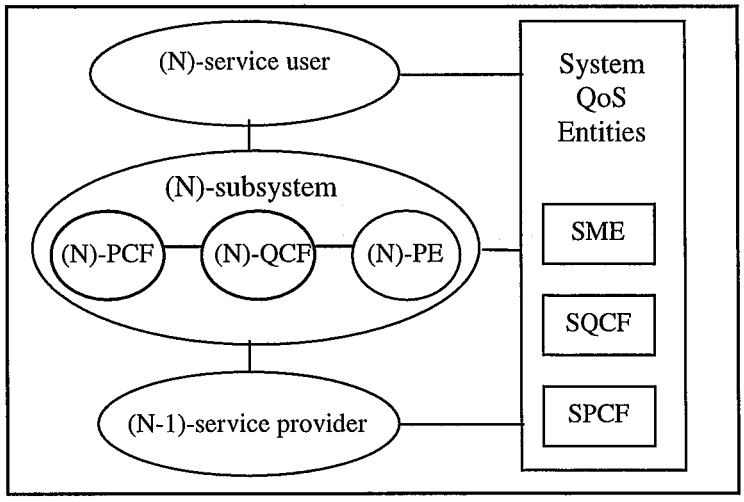

Fig. 4.2. Model of QoS for OSI.

The layer-specific QoS entities are as follows: The *(N)-Policy Control Function ((N)-PCF)* which determines both the policy that is applied to the operation of a (N)-subsystem and the constraints under which all other decisions of the (N)-subsystem are made. Thus, it models any action that must be performed to control the remaining operation of the (N)-subsystem. *The (N)-QoS Control Function ((N)-QCF)* takes into account the QoS requirements and selects the protocol entities (e.g. exerting an influence on addressing and routing) that will participate in the communication. *The (N)-Protocol Entity ((N)-PE)* is responsible for operating the (N)-protocol in order to provide the (N)-service to the (N)-service-users. In

particular, it is responsible for negotiating QoS with its peer (N)-PE, the (N)-service-user and the (N-1)-service-provider.

The System-wide QoS Entities (SQE) are as follows: The *System Management Entities (SME)* are used to enable system resources to be remotely and locally managed. These entities may be regarded as falling into two classes: those that provide the management infrastructure and those that use the management infrastructure to perform particular management tasks. The *System Quality Control Function (SQCF)* provides the followings two capabilities: a system-wide capability for tuning the performance of the various protocol entities that are in operation, and a coordination of any requirement to modify the behavior of remote systems via OSI systems management. The role of the *System Policy Control Function (SPCF)* is similar to the role of the (N)-PCF in a layer-specific, its inclusion as SQE takes into account that any policy implemented at any layer depends on a policy which has been established for the whole system.

It can be noted that this architecture is based on the client-server relationships between the adjacent layers and between a layer and its peer according to the OSI Reference Model. Note that in this model, every layer is a client of its underlying layer, and every layer is a server of its upper layer.

Client requirements depend on the layer being considered, it can be any of the layered architecture. These requirements must be then quantified and expressed as a set of QoS requirements, which in turn are expressed as either QoS parameters or QoS context. QoS requirements express information to manage one or more QoS characteristics, i.e. the quantifiable aspects of QoS.

Entities that receive QoS requirements analyze them and determine the QoS management functions or mechanisms that are required to meet them. This may involve generate further, typically more detailed, QoS requirements and convey them to other entities as QoS parameters such as a maximum value, a target, a threshold, and so on. If a given entity determines that it cannot meet a QoS requirement, there is a reverse flow to indicate it, in this case, the responsibility for action passes back to the previous entity in the flow.

4.2 Fieldbus QoS architecture

This section analyzes the QoS architecture of the IEC 61158, Digital Data Communications for Measurement and Control – Fieldbus for use in Industrial Control Systems [IEC 61158]. This International Standard basically is a collection of Fieldbus Standards, some of which have been analyzed in the Chapter 3, such as TS 61158, ControlNet, Profibus, P-Net, Foundation Fieldbus, Swiftnet, WorldFip and Interbus. They are called *type* and are numbered from 1 to 8, respectively.

Next subsections present the Data Link Services (DLS), the QoS architecture of the IEC 61158 Part 3, which is based on the [ISO/CEI 8886], and finally analyze the QoS architecture with regard to the Fieldbus standards.

4.2.1 Data Link Services

Basically the DLS provide both transparent and reliable data transfer among DLS-users, i.e.:

- Transparent data transfer. It does not restrict the content, format or coding of the information, nor does it ever need to interpret the structure or meaning of the information. It may, however, restrict the amount of information that can be transferred as an indivisible unit.

- Reliable data transfer. The DLS relieve the DLS-user of concerns regarding insertion or corruption of data, or, if it is requested, loss, duplication or misordering of data, which may occur.

- Independence from the underlying Physical Layer. The DLS relieve DLS-users of all direct concerns regarding which configuration is available and which physical facilities are used.

Before to analyze the QoS architecture, next subsection presents several definitions of DLS.

4.2.1.1 Common Data Link Service definitions

The relationships among the different components of the DLS are shown in Fig. 4.3 and the following points can be noted:

- the DLS-user entities access the service at the boundary between the two adjacent layers, called *DL Service Access Point* (DLSAP),
- a DLS-user entity can access one or more DLSAP,
- a *DL Entity* (DLE) can be connected to one or more DLS-user entities,
- a DLSAP links one DLS-user entity and one DL entity,
- the DLSAP addresses are depicted as designating small gaps (points of access) in the DL layer portion of a DLSAP,
- a *DL Connection-End-Point* (DLCEP)-address also designates a specific point of information flow (its DLCEP) within the DLSAP,
- a single DL-entity may have multiple DLSAP-addresses and group DL-addresses associated with a single DLSAP.

The IEC 61158 defines the following terms:

The *DLSDU* is a DL-service-data-unit.

An *extended link* is a DL subnetwork consisting of the maximal set of links interconnected by DL-relays, sharing a single DL-name (DL-address) space, in which any of the connected DL-entities may communicate, one with another, either directly or with the assistance of one or more DL-relay entities.

The *DLSAP* is a distinctive point at which DL-services are provided by a single DL-entity to a single higher-layer entity.

An *(individual) DLSAP-address* is a DL-address that designates only one DLSAP within the extended link. A single DL-entity may have multiple DLSAP-addresses associated with a single DLSAP.

A *group DL-address* is a DL-address that potentially designates more than one DLSAP within the extended link. A single DL-entity may have multiple group DL-addresses associated with a single DLSAP. A single DL-entity also may have a single group DL-address associated with more than one DLSAP.

A *DL(SAP)-address* is either an individual DLSAP-address, designating a single DLSAP of a single DLS-user, or a group DL-address potentially designating multiple DLSAPs, each of a single DLS-user.

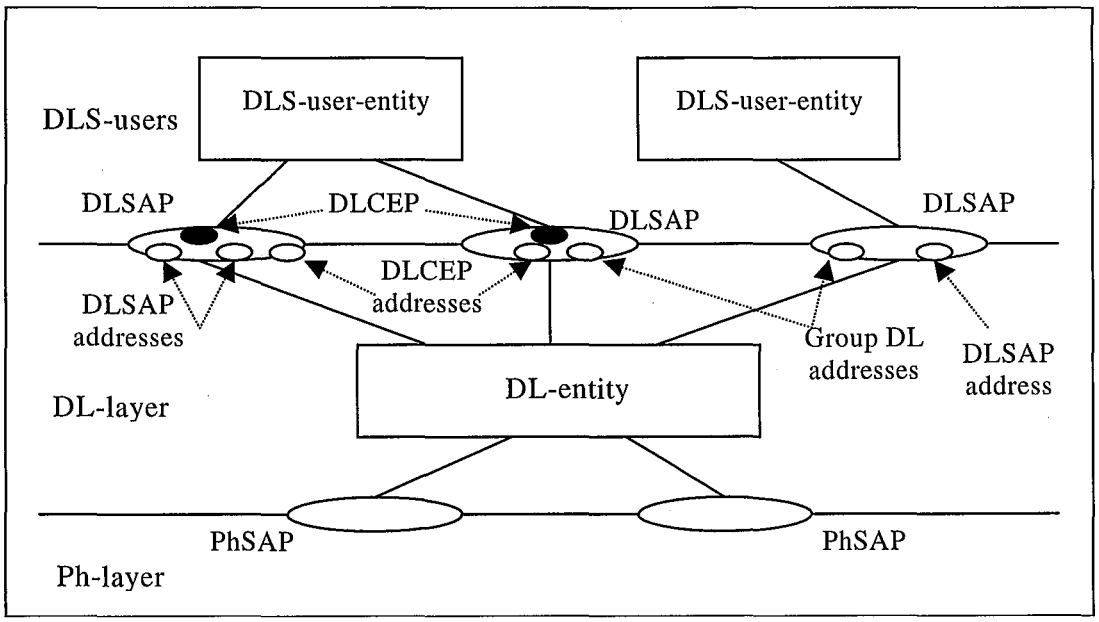

Fig. 4.3. Relationships among the components of the Data Link Service.

A *peer DLC* is a point-to-point DL-connection offering DL-duplex-transmission between two peer DLS-users where each can be a sending DLS-user, and each as a receiving DLS-user may be able to exert flow control on its sending peer. A peer DLC is negotiated to provide either symmetrical service or asymmetrical service. A peer DLC may also be negotiated to provide only DL-simplex service.

A *multi-peer DLC* is a centralized multi-end-point DL-connection offering DL-duplex-transmission between a single distinguished DLS-user, known as the *publisher* or *publishing* DLS-user, and a set of peer but undistinguished DLS-users, known collectively as the *subscribers* or *subscribing* DLS-users, where the publishing DLS-user can send to the subscribing DLS-users as a group (but not individually), and the subscribing DLS-users can

send to the publishing DLS-user (but not to each other). A multi-peer DLC always provides asymmetrical service. It may also be negotiated to provide only DL-simplex service either from the publisher to the subscribers or from the subscribers to the publisher.

A *DLCEP-address* is a DL-address which designates either one peer DL-connection-end-point or one multi-peer publisher DL-connection-end-point, and implicitly the corresponding set of subscriber DL-connection-end-points.

A *DLSEP-address* is a DL-address which designates a DL-scheduling-end-point within a DLE.

A *DLPDU* is a DL protocol-data-unit, another term for frame which is a DLPDU consisting of a source MAC ID, zero or more Lpackets, and FCS, and transmitted or received by an associated PhE.

4.2.2 IEC 61158-Part 3 QoS architecture

This QoS architecture is mainly based on the classes of service that the Data Link Layer must provide as well as on the QoS attributes which are common among the different types of DLS, i.e. the Fieldbus Standards. A DLS-user may select, directly or indirectly, some parameters of these attributes in order to determine the quality of the data link services. The term *Quality of Service* refers then to those aspects that are under the direct control of the DLS-provider. QoS can only be properly determined when DLS-user behavior does not constrain or impede the performance of the DLS. Next subsections present the classes of service and the QoS attributes.

4.2.2.1 Classes of Data Link Service

The IEC 61158-3 Type 1, based on the [ISO/CEI 8886], define two classes of DLS:

- a connection-mode data transfer service
- a connectionless-mode data transfer service.

In the first class, before data transmission, it is required to establish a logical channel, called connection; after data transmission, the connection is released. In the second class, no any connection is required before data transmission.

This International Standard also defines other two classes of service such as:

- a DL(SAP)-address, queue and buffer management service
- a time and transaction scheduling service.

The DL(SAP)-address, queue and buffer management service defines the interactions between the DLS-user and the DLS-provider that take place at a DLSAP. Information is passed between the DLS-user and the DLS-provider by means of DLS primitives that convey

parameters. These primitives are used to provide a local service between a DLS-user and the local DLE. Remote DLEs and remote DLS-users are not directly involved in this service.

The time service provides DLS-users with a means to indirectly synchronize and schedule their activities with a shared sense of time. Scheduling service allows to complete a deferred primitive, to distribute the current value of a buffer and to start an exchange service.

It can be noted that these four classes of DLS will be provided according to the types of DLS, i.e. the DLS-user is limited to those classes of service supported by the selected DL protocol implementation. The following paragraphs briefly describe the classes of DLS.

4.2.2.1.1 Connection-mode data transfer service

This service provides DLS-user with the followings facilities, as is shown in Fig 4.4:

a) A means to establish either:

1) a peer DLC between two DLS-users for exchanging DLSDUs between the two DLS-users, or

2) a multi-peer DLC between a single publishing DLS-user and a set of subscribing DLS-users for sending DLSDUs
   i) from the publishing DLS-user to the set of subscribing DLS-users, and
   ii) optionally, from any of the subscribing DLS-users to the publishing DLS-user.

b) A means of establishing an agreement for a certain QoS associated with a DLC, between the initiating DLS-user, the responding DLS-user(s) and the distributed DLS-provider.

c) A means of transferring DLSDUs of limited length on a DLC. The transfer of DLSDUs is transparent and there are no constraints on the DLSDU content imposed by the DLS.

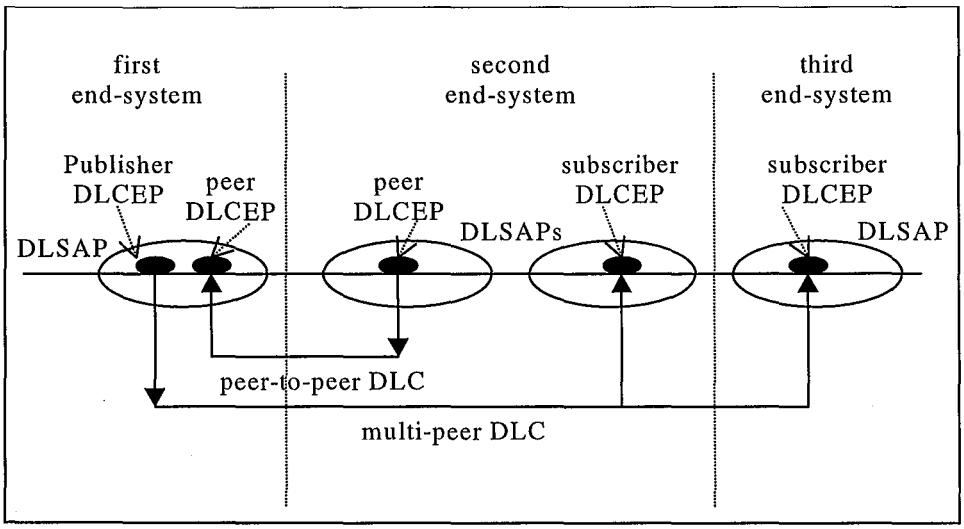

Fig. 4.4. Peer-to-peer and multi-peer DLCs and their DLCEPs.

d) A means of conveying timeliness information about those DLSDUs and their conveyance by the (distributed) DLS-provider in certain modes of DLC operation.

e) A means by which the receiving DLS-user at a peer DLCEP may control the flow rate at which the sending DLS-user may send DLSDUs, if supported by the DLC's QoS.

f) A means by which a DLCEP, and possibly a DLC, can be returned to a defined state and the activities of the DLS-users resynchronized by use of a RESET primitive.

g) A means by which the publishing DLS-user of a multi-peer DLC can query whether there are any subscribing DLS-users of the DLC.

h) A means for the unconditional, and therefore possibly destructive, release of a DLCEP and possibly a DLC, by one of the DLC's DLS-users or by the DLS-provider.

4.2.2.1.2 Connectionless-mode data transfer service

This DLS provides DLS-user with the following facilities:

a) A means for transferring DLSDUs of limited length from one source DLSAP to a destination DLSAP or group of DLSAPs, without establishing and later releasing of a DLC. The transfer of DLSDUs is transparent and there are no constraints on the DLSDU content imposed by the DLS. QoS for this transmission can be selected by the sending DLS-user.

b) A means by which the status of delivery at the destination DLSAP can be returned to the source DLSAP.

c) A means by which previously submitted DLSDUs of limited length can be exchanged between two DLSAPs, and the status about the exchange can be provided to the DLS-users, without establishing and later releasing of a DLC. The transfer of the DLSDUs is transparent and there are no constraints on the DLSDU content imposed by the DLS.

d) A means by which a DLS-user can be notified that an exchange was attempted, but that no DLSDUs were available for the exchange.

e) A means by which a DLS-user can query if there are any DLS-users that can receive DLSDUs sent to a specified (usually group) DL(SAP)-address.

4.2.2.1.3 DL(SAP)-address, queue and buffer management service

It provides DLS-user with the following facilities:

a) A means for creating and deleting a retentive buffer, or a non-retentive buffer, or a FIFO queue of specified depth, for use:

1) in the communication of DLS-user-data between a DLS-user and the DLS-provider.
2) in redistributing received DLS-user data without DLS-user intervention.
3) in facilitating DLS-user supervision of the timing of DLSDU-conveyance to peer DLS-users.

b) A means for associating an individual DLSAP-address or group DL-address with, and disassociating a DL(SAP)-address from, the DLSAP at which the request is made.

c) A means by which DLSDUs of limited size are written to or read from a buffer, or read from a FIFO queue.

4.2.2.1.4 Time and transaction scheduling service

It provides DLS-user with the following facilities:

a) A means by which a DLS-user can request the current value of DL-time from the DLS-provider. This DLS provides DLS-users with a means to indirectly synchronize and schedule their activities via this shared sense of DL-time.

b) A means by which a DLS-user can compel the DLS-provider to complete one already-issued DLS-request primitive whose execution has been deferred at the issuing of a DLS-user's request, and which was issued by the DLS-user itself with a local DLSAP-address or from a local DLCEP.

c) A means by which a local DLS-user can compel the DLS-provider to complete one already-issued DLS-request primitive whose execution has been deferred at the remote DLS-user's request, and which was issued by that remote DLS-user that is the peer or publisher connected to the local DLS-user's peer or subscriber DLCEP.

d) A means by which a publishing or subscribing DLS-user of a multi-peer DLC can compel the DLS-provider to distribute the current value of the associated buffer.

e) A means by which a DLS-user can compel the DLS-provider to initiate an exchange service of a specified priority between two different DLSAP-address.

f) A means by which a DLS-user can group one or more requests of the types enumerated in b), c), d) or e) into a sequence, and schedule that sequence for either one-time or periodic or repetitive DLS.

g) A means by which a DLS-user can cancel a scheduled sequence of the type specified in f).

h) A means by which a DLS-user can dynamically select a subset of a previously-scheduled sequence of the type specified in f).

4.2.2.2 QoS attributes common to multiple types of Data Link Service

The aspects that may be selected by a DLS-user are called *attributes* and can be classified as follows: static, dynamic and semi-static. *Static* QoS attributes are selected once for an entire type of DLS. *Dynamic* QoS attributes are selected independently at each DLS invocation. *Semi-static* QoS attributes are static attributes for one type of DLS, and serve as defaults for corresponding dynamic attributes in another type of DLS. Most of the QoS attributes have default values which can be set by DL-management and then overridden on a per-DLSAP-address basis by the DLS-user.

The attributes defined by the IEC 61158 are as follows: DLL priority, DLL maximum confirm delay, DLPDU authentication, DL-scheduling-policy and DL-timeliness.

The first four QoS attributes apply conceptually to both connection-mode and connectionless operation. The DLS-user may specify values for these attributes when binding a DLSAP-address to the DLS-user's DLSAP; any unspecified attributes will assume the default values set by DL-management. The fifth attribute applies only to the connection-mode operation, and is dynamic for each DLCEP. Next paragraphs describe these QoS attributes.

4.2.2.2.1 DLL priority (dynamic QoS attribute)

All DLCEP establishment requests and responses, all connectionless data transfer requests, and many DL-scheduling requests, specify an associated DLL priority used in scheduling DLL data transfer services. This DLL priority also determines the maximum amount of DLS-user-data that can be conveyed in a single DLPDU. This maximum is determined by the DL-protocol specification.

The DL-protocol should support three DLL priority levels, each of which should be capable of conveying a specified amount of DLS-user data per appropriate DLPDU. The three DLL priorities with their corresponding ranges of conveyable DLS-user-data (per DLPDU) are, from highest priority to lowest priority:

- *URGENT* — capability of conveying up to 64 DLS-user octets per DLPDU
- NORMAL — capability of conveying up to 128 DLS-user octets per DLPDU
- TIME-AVAILABLE — capability of conveying up to 256 DLS-user octets per DLPDU.

URGENT and NORMAL are considered *time-critical* priority levels; TIME-AVAILABLE is considered a *non-time-critical* priority level. The default QoS value can be set by DL-management; when not so set its value is TIME-AVAILABLE.

4.2.2.2.2 DLL maximum confirm delay (dynamic QoS attribute)

Each DLCEP establishment request, and each response, specifies upper bounds on the maximum time duration permitted for the completion of each related instance of a sequence of connection-oriented DLS primitives, which can be as follows:

a) the common maximum time for completion of (shown in Fig. 4.5):
   - a related sequence of DL-CONNECT primitives
   - a related sequence of DL-RESET primitives
   - a related sequence of DL-SUBSCRIBER-QUERY primitives.

b) the maximum time for completion of a related sequence of DL-DATA primitives.

Each connectionless service request specifies an upper bound on the maximum time duration permitted for the completion of each related instance of a sequence of connectionless DLS primitives, which can be as follows:

a) the maximum time for completion of a related sequence of locally-confirmed DL-UNITDATA primitives

b) the common maximum time for completion of:
   - a related sequence of remotely-confirmed DL-UNITDATA primitives,
   - a related sequence of DL-LISTENER-QUERY primitives, or
   - an instance of the DL-UNITDATA-EXCHANGE service.

Each parameter either has the value *UNLIMITED* or specifies an upper bound, in a given time unit, for example 1 ms, from 1 ms to 60 s, and so on. The default QoS values can be set by DL-management; when not so set the value for each of these QoS parameters is UNLIMITED.

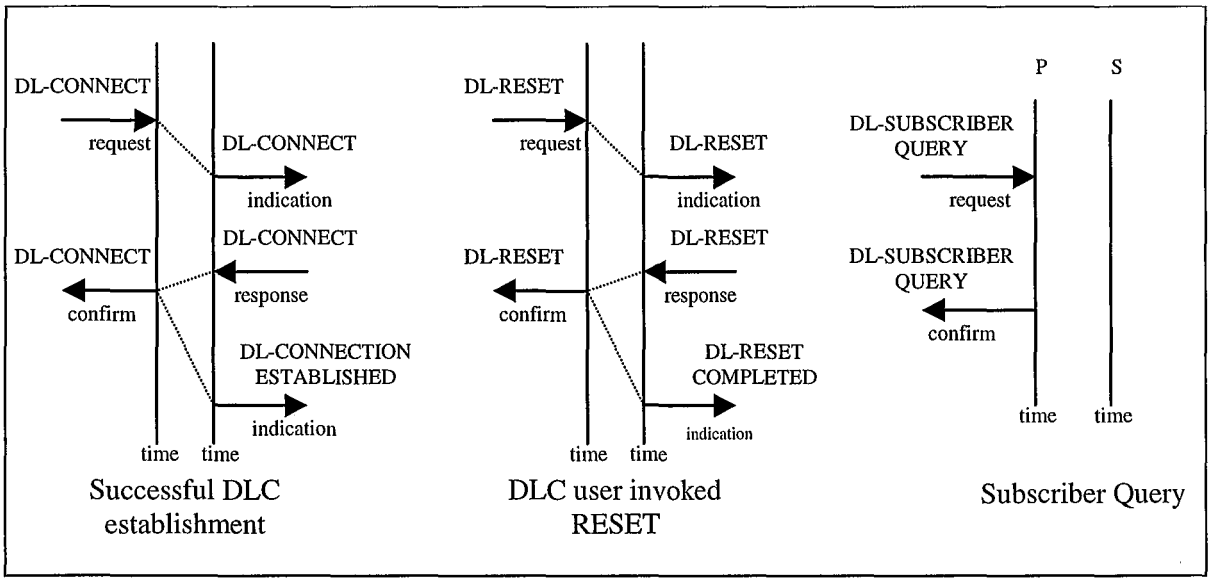

Fig. 4.5. Time sequence diagrams.

4.2.2.2.3 DLPDU authentication (semi-static QoS attribute)

Each DLCEP establishment negotiation, and each connectionless data transfer, uses this attribute to determine the following:

a) a lower bound on the amount of DL-addressing information used in the DLPDUs that provide the associated DLL data transfer services,

b) whether the current state of a sending peer or publisher DLCEP should be sent at low-frequency to the DLC's peer or subscriber DLCEP(s) even when there are no unconfirmed DLS-user requests outstanding at the sending DLCEP, and

c) whether all related scheduling actions should be executed locally.

The three levels specifiable, with their amounts of DL-addressing information, are as follows:

- *ORDINARY* — each DLPDU should include the minimum permitted amount of addressing information.

- *SOURCE* — each DLPDU should include a source DL-address where possible.

- *MAXIMAL* — each DL-address should include the maximal amount of addressing information possible. Also, all related scheduling actions should be executed locally; and each sending peer or publisher DLCEP of the DLC should maintain a low-frequency report of state information when there is no DLS-user activity.

The default QoS value can be set by DL-management; when not so set its value is ORDINARY. DLCEP establishment may negotiate ORDINARY to SOURCE to MAXIMAL.

4.2.2.2.4 DL-scheduling-policy (semi-static QoS attribute)

This attribute is static for connectionless services, but is dynamic for connection-mode services. For each DLSAP-address, and each DLCEP, the DLS-user can override the normal (implicitly-scheduled) DLL policy, and instead can defer any inter-DLS-user communication required by a DL-DATA or DL-UNITDATA request DLS-primitive until that deferral is cancelled by an involved DLS-user. A DL-COMPEL-SERVICE request, specifying the affected DLSAP-address or DLCEP, permits the continued execution of just a single deferred in-process request or response DLS-primitive. Only DL-services that provide DLS-user intercommunication are affected by this attribute.

The two choices are as follows:

- *IMPLICIT* — any required communications with peer DLS-user(s) from this DLSAP-address, or from this DLCEP, will occur as soon as possible.

- *EXPLICIT* — any required DATA or UNITDATA communications with peer DLS-user(s) from this DLSAP-address, or from this DLCEP, will occur only when the deferral is explicitly cancelled by an involved DLS-user.

The default QoS value cannot be set by DL-management; its value is always IMPLICIT.

4.2.2.2.5 DL-timeliness (dynamic DLCEP QoS attributes)

This attribute only applies to retentive DL-buffers, to DLCEPs at which DL-buffers are bound, and to those DLS-primitives which transfer DLS-user data to or from DL-buffers at such DLCEPs.

Each DLCEP establishment request, and each response, can specify DL-timeliness criteria which are applied to information sent from, or received into, retentive buffers at that DLCEP. Four types of DL-timeliness can be supported: RESIDENCE timeliness, UPDATE timeliness, SYNCHRONIZED timeliness, and TRANSPARENT timeliness.

- *RESIDENCE* timeliness is an assessment based upon the length of time that a DLS-user datum has been resident in a buffer, which is the time interval between the moment when the buffer is written and the moment when the buffer is read.

  So, DL-timeliness = $0 \leq (R_T - W_T) < \Delta T$

- *UPDATE* timeliness is an assessment based upon the time interval between the moment of occurrence of a multi-DLE synchronizing event and the moment when the buffer is written.

  Then, DL-timeliness = $0 \leq (W_T - S_T) < \Delta T$

- *SYNCHRONIZED* timeliness is an assessment based upon the time intervals and timing relationships among:

  1) the moment of occurrence of a multi-DLE synchronizing event (S_T), and

  2) the moment when the buffer is written, and

  3) the moment when the buffer is read

  So, DL-timeliness = $0 \leq (W_T - S_T) \leq (R_T - S_T) < \Delta T$

- *TRANSPARENT* timeliness occurs when timeliness is selected on a DLCEP but none of the above assessments are performed. In such a case the DLC preserves any prior buffer timeliness, but does not itself invalidate that timeliness. When no prior buffer timeliness exists, the default timeliness value is TRUE.

- *NO* timeliness occurs when timeliness is not selected on a DLCEP. In such a case the DL-timeliness attribute of DLS-user data always is FALSE.

The DL-time when the original buffer is written by a DL-PUT request primitive can also be conveyed to DLS-users which read a copy of the buffer. This DL-time is not available when the buffer timeliness is FALSE.

4.2.3 Service Type 1 (TS 61158) QoS architecture

This section and the remaining sections analyze the previous QoS architecture with regard to the Fieldbus Standards. This section discusses the TS 61158 Fieldbus network which is presented in the section 3.4.21.

4.2.3.1 Connection-mode service

The term Quality of Service refers to certain characteristics of a DLC as observed between the connection end-points. Once a DLC is established, the DLS-users at the DLCEPs have the same knowledge and understanding of the actual QoS of the DLC.

4.2.3.1.1 Quality of connection-mode service

QoS is described in terms of QoS parameters. These parameters give DLS-users a method of specifying their needs, and give DLS-provider a basis for protocol selection. All the QoS parameters are selected on a per-connection basis during the establishment phase of a DLCEP, or of a DLC and DLCEP. Once the DLC is established, throughout the lifetime of the DLC, the agreed QoS values are not reselected at any point in time. QoS parameters are as follows:

4.2.3.1.1.1 DLCEP class

Each DLC/DLCEP establishment request specifies the class of the DLCEP. Three choices for DLCEP class are possible:

- *PEER* — the DLS-user can exchange DLSDUs with one other peer DLS-user.
- *PUBLISHER* — the DLS-user can send DLSDUs to a set of zero or more associated subscribing DLS-users, and may be able to receive DLSDUs from any of those subscribing DLS-users.
- *SUBSCRIBER* — the DLS-user can receive, and request, DLSDUs from the associated publishing DLS-user, and may be able to send DLSDUs to that publishing DLS-user.

The default QoS value for the DLCEP class is PEER.

4.2.3.1.1.2 DLCEP data delivery features

Both members of a peer DLC, or the publishing DLS-user of a multi-peer DLC, specify the data delivery features of the DLC's DLCEP(s). The available choices of DLCEP data delivery features depends on the class of DLS provided. The five choices for DLCEP data delivery features, and their effects, are as follows:

- *CLASSICAL* — the DLS-user can send data which will be delivered without loss, duplication or misordering to the receiving DLS-user(s). All relevant DLS-users will be notified of any loss of synchronization on the DLC.

- *DISORDERED* — the DLS-user can send data which will be delivered immediately upon receipt to the receiving DLS-user(s), without duplication but potentially in a different order than the order in which it was originally sent. All relevant DLS-users will be notified of any unrecoverable loss of DLS-user-data or loss of synchronization on the DLC.

- *ORDERED* — the DLS-user can send data which will be delivered immediately upon receipt to the receiving DLS-user(s), without duplication or misordering, but with potential loss of some DLS-user-data. Loss of DLS-user-data will not be reported, and recovery of DLS-user data lost prior to the last-reported DLS-user data will not be attempted.

- *UNORDERED* — the DLS-user can send data which will be delivered immediately upon receipt to the receiving DLS-user(s). Loss, duplication and misordering of DLS-user-data will not be detected or reported. No attempt will be made by the DLS-provider to recover from such events.

- *NONE* — the DLS-user cannot send data in this direction of data transfer.

So, there are four classes of connection-mode DLS:

1. CLASSICAL, ORDERED and UNORDERED peer and multi-peer DLCs are all supported; DISORDERED peer and multi-peer DLCs may be supported.

2. only ORDERED and UNORDERED peer and multi-peer DLCs, and CLASSICAL peer DLCs, are supported; CLASSICAL and DISORDERED multi-peer DLCs are not supported; DISORDERED peer DLCs may be supported.

3. only ORDERED and UNORDERED peer and multi-peer DLCs are supported; CLASSICAL and DISORDERED peer and multi-peer DLCs are not supported.

4. only UNORDERED peer and multi-peer DLCs are supported; ORDERED, CLASSICAL and DISORDERED peer and multi-peer DLCs are not supported.

On a peer DLC, the QoS value for the DLCEP data delivery features may be chosen independently for each direction of data transfer. The default QoS value in each direction for the DLCEP data delivery features is UNORDERED.

On a multi-peer DLC, the QoS value for the DLCEP data delivery features for the subscribers-to-publisher direction of data transfer is restricted to UNORDERED and NONE.

The default QoS value for the DLCEP data delivery features in the publisher-to-subscribers direction is UNORDERED. The default QoS value for the DLCEP data delivery features in the subscribers-to- publisher direction is NONE.

4.2.3.1.1.3 DLL priority

This parameter is defined and its default value is specified in section 4.2.2.2.1.

4.2.3.1.1.4 DLL maximum confirm delay

This parameter is defined and its default values is specified in section 4.2.2.2.2. Failure to complete a DL-CONNECT or DL-RESET request within the specified interval results in a DLS-provider initiated release of the DLCEP, and possibly of the DLC.

4.2.3.1.1.5 DLPDU authentication

This parameter is defined and its default value is specified in section 4.2.2.2.3. The DLS-user may override that default value when establishing a DLCEP.

4.2.3.1.1.6 DL-scheduling-policy

This parameter is defined and its default value is specified in section 4.2.2.2.4. The DLS-user may override that default value when establishing a DLCEP. When the DLCEP is bound as sender to a buffer, then the DLS-user should ensure that this parameter has the value EXPLICIT.

4.2.3.2 Connectionless-mode service

4.2.3.2.1 Quality of connectionless-mode service

The term QoS refers to certain characteristics of a connectionless-mode data transmission as observed between the DLSAPs. QoS describes aspects of a connectionless mode data transmission that are attributable solely to the DLS-provider.

A basic characteristic of a connectionless-mode service is that no long-term dynamic association is set up between the involved parties. Thus, associated with each DL-connectionless mode data transmission, certain measures of QoS are requested by the sending or initiating DLS-user when the primitive action is initiated. QoS parameters for connectionless service are as follows:

4.2.3.2.1.1 DLL priority

Connectionless data transfer and data exchange primitives specify the priority of the transferred DLSDU(s). This parameter is defined and its default value is specified in section 4.2.2.2.1.

4.2.3.2.1.2 DLL maximum confirm delay

This parameter is defined and its default value is specified in section 4.2.2.2.2.

4.2.3.3 DL(SAP)-address, queue and buffer management Data Link Service

The provided services are as follows: Buffer or Queue Creation, Buffer or Queue Deletion, DL(SAP)-address Activation, DL(SAP)-address Deactivation, Update Buffer and Copy Buffer or Dequeue.

4.2.3.3.1 Buffer or Queue Creation

It allows to specify the following parameters:

4.2.3.3.1.1 Queuing policy

This parameter specifies whether to create either:

- *BUFFER-R* — a retentive buffer, whose contents are not affected by being read, which can be overwritten (as a single atomic action) by either the DLS-provider or DLS-user, and which can be used only with the connectionless UNITDATA-exchange and connection-oriented data services; or

- *BUFFER-NR* — a non-retentive buffer, which is set empty after being read, which can be overwritten (as a single atomic action) by either the DLS-provider or DLS-user, and which can be used only with the connectionless UNITDATA-exchange service; or

- *QUEUE* — a FIFO queue of maximum depth *K* which contains between zero and *K* DLSDUs, which will reject attempts to remove DLSDUs when empty and to append DLSDUs when full, and which can be used with the connectionless UNITDATA-transfer and connection-oriented data services, and for DLSDU-receipt with the connectionless UNITDATA exchange service.

4.2.3.3.1.2 Maximum queue depth

This parameter is present when the Queuing Policy parameter has the value QUEUE. When present, this parameter specifies K, the maximum number of items in the associated queue. The supported values for this parameter should include the values one, two, three and four.

4.2.3.3.1.3 Maximum DLSDU size

This parameter specifies an upper bound on the size (in octets) of DLSDUs that can be put into the buffer or queue. The maximum size permitted for such DLSDUs may be constrained by a companion DL-protocol specification and by DL-management.

4.2.3.3.2 DL(SAP)-address Activation

The bind DL(SAP)-address DLS primitive is used:

a) to associate a DL(SAP)-address with the DLSAP at which the primitive is invoked,
b) to establish the DL(SAP)'s role, if any, when participating in the DL-UNITDATA and DL-UNITDATA-exchange connectionless services at that DL(SAP)-address,
c) to associate up to six previously created retentive buffers or non-retentive buffers or limited-depth FIFO queues with the various priorities and directions of potential data transfer at the specified DL(SAP)-address, and
d) to specify default values for some QoS attributes for connection-mode and connectionless data transfer services using the specified DL(SAP)-address.

A DL(SAP)-role parameter constrains the DL-connectionless DL-UNITDATA and DL-UNITDATA-EXCHANGE service primitives that can be issued with this DL(SAP)-address at the local DLSAP (to which this DL(SAP)-address is being bound). It also constrains whether the DL(SAP)-address may have an associated DLCEP and the permitted types of bindings (implicit queue, explicit queue or explicit buffer) to the DL(SAP)-address.

The unbind DL(SAP)-address DLS primitive is used to unbind a DL(SAP)-address from the invoking DLSAP. Any buffers or queues that were explicitly bound to the DL(SAP)-address are also unbound from that DL(SAP)-address at the same time.

4.2.3.4 Time and transaction scheduling service

The DLS provides the same facilities defined in section 4.2.2.2.4.

There can be many classes of time-related DLS according to the granularity of the clock, for example 1 μs, 10 μs, 100 μs and so on. These classes will provide an aggregate measure of local and reference DLE clock resolution and drift rate, and of the frequency of time distribution on the local link.

4.2.4 Service Type 2 (ControlNet) QoS architecture

Section 3.4.18 presented the ControlNet MAC protocol and shown that the network time is divided into Network Update Intervals (NUI), which are further divided into scheduled, unscheduled and network maintenance windows. At the Logical Link Control layer, the first two windows can be seen as levels of QoS using the priority attribute defined by the IEC 61158. Note, however, that the duration time of the windows is fixed and that each node only can transmit one message during the scheduled window and one or more messages during the unscheduled window, according to the network load.

From the IEC 61158 point of view, the available QoS options for the connection-mode and the connectionless-mode services are sending priority and timing. The choice of sending priority implicitly selects the timing characteristics of the DLS supplier execution of the transmission. Three alternative priorities are available: scheduled, high and low priorities.

4.2.4.1 *Scheduled priority*

This QoS provides accurate time-based cyclic and acyclic sending of DLSDUs. The execution timing for this scheduled service can be accurate and repeatable.

4.2.4.2 *High priority*

This QoS provides acyclic sending of DLSDUs with a bounded upper time for the sending delay. Data on this priority is sent only when all scheduled data has been sent and a non-scheduled sending opportunity is available.

4.2.4.3 *Low priority*

This QoS provides sending of DLSDUs only on a time-available basis. Data on this priority is sent only when all other priorities of data have been sent and a non-scheduled sending opportunity is available.

4.2.5 Service Type 3 (Profibus) QoS architecture

Section 3.4.15 presented the Profibus MAC protocol and shown that the nodes are organized as masters and slaves. Master nodes form a logical ring where there is a token which is passed from one master node to the next master node, thus the token determines the instant at which each master may access the medium. A slave node needs a master-request to exchange information.

Four transmission operation modes were identified: Token handling, acyclic request or send/request operation, cyclic send/request operation or polling and registration of stations. So, from the IEC 61158 point of view, the three latest modes are seen as connectionless DL services, as is discussed below. Note that the priority levels, high and low, provided by the Profibus FMS were not taken into account in the IEC 61158. Note also that the second and third transmission modes can be seen as levels of QoS.

The services of Profibus address the following characteristic requirements for Data Link and management services:

a) *Acyclic Data Transfer.* Send Data with/without Acknowledge (SDN/SDA); Send and Request Data with Reply (SRD).

b) *Polling Data Transfer.* Cyclic Send and Request Data with Reply (CSRD).

c) *Layer Management.* Initialization, configuration and event handling.

4.2.6 Service Type 4 (P-NET) QoS architecture

Section 3.4.14 analyzed the P-NET MAC protocol and shown that each master node cyclically transmit only one message each time it gets the virtual token. So, from the IEC 61158 point of view the P-Net QoS architecture provides two types of DLS:

a) a connectionless-mode data transfer service, providing confirmed and unconfirmed data transfer,

b) a management service. The type 4 management service provides services for reading and writing managed objects.

4.2.7 Service Type 6 (Swiftnet) QoS architecture

Section 3.4.19 presented the Swiftnet MAC protocol and shown that it is a slotted TDMA protocol where each data uses a dedicated channel, i.e. one or more slots per bus-cycle. The bus configuration is totally end-user defined. Therefore the levels of QoS are statically guaranteed. From the IEC 61158 Standard point of view, Swiftnet provides the followings DL services.

4.2.7.1 Connection-mode Data Transfer Service

The IEC 61158 Standard argues that the attribute values of the type 1 (TS 61158) may be mapped by the Calling and Called DLEs into and from the values set by the Bus Configuration DLS-user. But unlike the Type 1 DLS, DLCEP establishment, negotiation and connectionless data transfer do not use DLPDU authentication attribute to determine a lower bound on the address length of each DLPDU. Connection mode transfers are identified by a DLC-Identifier.

Type 6 has additionally some inherent QoS attributes for real time and connection-mode services. The current bus configuration of the Type 6 bus primarily determines the values achieved for these attributes. Thus the QoS of these services is *inherent* rather than static, dynamic or semi-static. None of the inherent QoS parameters are visible at the DLS-user. These QoS parameters are:

- *Scan Rate*. The rate at which a specific I/O device (or a group of devices) is sampled. For any data rate the maximum total scan rate of any bus configuration is effectively fixed. This total scan rate can be divided in many ways among a diverse population of channels to meet user needs.

- *Jitter*. Any deviation from a regular synchronous pattern in either a) sampling a device b) transmission of data values c) imposition of a trigger signal. Jitter should be measured from time to time within one node on the length and also across multi-nodes on length.

4.2.7.2 *Connectionless-mode Data Transfer Service*

This service is a subset of the Type 1 of DLS. The restrictions imposed on the connectionless service by the Type 6 are that the DLSAPs used for the connectionless service support and use the Explicit Address Channel method and the connectionless UNCONFIRMED "Channel class".

Each of these DLSAP recognizes one or more DLSAP-addresses and possibly contained DLCEP addresses. The length of the Net-ID portion of these addresses is a configuration issue. The attribute values of the Type 6 connectionless services are a subset of the values available from Type 1. Thus formal DLS-user interaction with the DLS relative to the connectionless services and the QoS parameters can be mapped onto a subset of the Type 1 services.

4.2.8 Service Type 7 (WorldFIP) QoS architecture

Section 3.4.16 analyzed the WorldFIP MAC protocol and shown that the basic cycles are composed of at most four windows: a periodic window, an aperiodic variables window, an aperiodic messages window and a synchronization window. Note that in each basic cycle there at least one periodic window, therefore it can be seen as a guaranteed level of QoS. Nevertheless from the IEC 61158 point of view, the windows are seen as the following two types of data transmission services:

a) the first handles connection-oriented buffer transfers between pre-established point-to-multi-point DLCs on the same local link,

b) the second handles acknowledged or unacknowledged connectionless message transfers between single DLSAPs, or unacknowledged message transfers from a single DLSAP to a group of DLSAPs on the extended link.

Note that for purposes of clarity, the expressions "buffer transfer" and "message transfer" are used to distinguish between the two types of communications services, connection-oriented and connectionless, respectively, which are offered by the DLS Type 7.

There are also two types of buffer transfer services:

1) *Cyclical buffer transfer.* Variable names and periods are defined when the system is configured, and are based on application needs. Cyclical exchanges are automatically triggered by the communications system without the user requests.

2) *Explicit request for buffer transfer.* Upon user request the value(s) of one or more variables are circulated.

The message transfer service also has two forms:

1) *Cyclical messages transfer.* Resources and periods are defined when the system is configured and are based on application needs. Cyclical transfers are automatically triggered by the communications system without the user requests.

2) *Aperiodic message transfer.* Upon user request one or more messages are circulated.

4.2.9 Service Type 8 (Interbus) QoS architecture

Section 3.4.20 presented the Interbus MAC protocol and shown that there is a master node that sends a message to all the slaves in the ring; whenever each slave receives this message, the slave reads from and writes to its assigned slot into the message and then pass the message to the next slave in the ring. This access method is called summation-frame. Note that it provides guaranteed levels of QoS.

From the IEC 61158 point of view, Interbus provides three classes of DLCEPs:

a) *PEER* — the DLS-user can exchange DLSDUs with one other peer DLS-user.

b) *PUBLISHER* — the DLS-user can send DLSDUs to a set of zero or more associated subscribing DLS-users.

c) *SUBSCRIBER* — the DLS-user can receive DLSDUs from the associated publishing DLS-user.

All buffers and queues are pre-created and bound to DLCEPs. The DLS-user cannot directly create, delete, bind or unbind buffers or queues. DLCEPs of class PEER always use queues; DLCEPs of classes PUBLISHER and SUBSCRIBER always use buffers which are bound to them.

4.3 Internet QoS architecture

Internet communication system or the TCP/IP model is basically composed of the services provided by the application, transport and internet (network) layers. Nowadays, this architecture only provides best-effort services and this is why the Internet Engineering Task Force (IETF) has proposed many service models and mechanisms to meet the demand for QoS.

Two services models will be discussed in this section: Integrated Services (IntServ) [Braden *et al.*, 1994] and Differentiated Services (DiffServ) [Blake *et al.*, 1998; Bernet *et al.*, 1999]. There are other service models such as Multiprotocol Label Switching (MPLS), Traffic Engineering and Constraint-based Routing but are not discussed here, a survey of them can be found in [Xiao and Ni, 1999].

For the IEEE 802-style LANs, a Subnet Bandwidth Management (SBM) has been proposed [Ghanwani *et al.*, 1999; Yavatkar *et al.*, 2000; Seaman *et al.*, 1999] but its main requirement is that all traffic must pass through at least one switch using the IEEE 802.1p, [IEEE 802.1Q], or [IEEE 802.1D]. In this way, Integrated Services can be supported by shared and switched LANs using the IEEE 802 MAC protocols such as 802.3, 802.5 and 802.12. For the 802.3, an explicit user-priority field on top of the basic MAC frame format is then required. The Internet service models are analyzed in the following subsections and are shown in Fig. 4.6.

4.3.1 Integrated Services model

IntServ model assumes that the resources of the communication system (e.g. bandwidth and buffers) must be explicitly managed in order to meet the application requirements. Given that the resources are finite, resource reservation and admission control are key building blocks of this service. Two other building blocks are required: packet scheduler and classifier.

In order to state the resource requirements, an application must specify the desired QoS using a list of parameters, called *flowspec*. The flowspec is carried by the reservation protocol, passed to the admission control for testing acceptability, and finally used to set the parameters of the packet scheduling mechanism.

So, the *resource reservation* protocol creates and maintains flow-specific state in the endpoint hosts and in routers along the path of a flow. Several of these protocols have been proposed, for example [Ferrari and Verma, 1990; Topolcic, 1990; Zhang *et al.*, 1993], and the next subsection describes one. The *Admission control* block implements the decision algorithm that a router or host uses to determine whether a new flow can be granted the requested QoS without impacting earlier guarantees. The *Packet scheduler* block manages the forwarding of different packet streams using a set of queues. The *Classifier* blcok maps each incoming packet into some class; all packets in the same class get the same treatment from the packet scheduler.

It can be noted that this service model is situated at the transport layer of the Internet architecture. The model makes use of the services which are furnished by the network layer (routing and packet scheduling) to reserve resources along the path of the packets.

4.3.1.1 Resource ReSerVation Protocol (RSVP) [Zhang *et al.*, 1993]

Before or when each data source starts transmitting, it sends a *path message* containing the flowspec of the data source. When a router receives a path message, it checks to determine whether it already has the path state for the receiver(s); if not it creates the path state for that target. The router then obtains the outgoing interface of the path message from the routing protocol in use, and updates its table of incoming and outgoing links accordingly. This path message is forwarded immediately only if it is from a new source or indicates a change in routes.

When a receiver receives a path message, it sends a *reservation message* using the (possibly modified) flowspec that came in the incoming path message. The reservation message is guided along the reverse route of the path message to reach the data source. If any router, along the way, rejects the reservation, an reject message is sent back to the receiver and the reservation message is discarded.

Once the reservation is established, the receiver periodically sends *reservation refresh messages*, which are identical to the original request. As the reservation requests are forwarded along the path tree, the router merge the requests for the same multicast group by pruning those that carry a request for reserving a smaller or equal amount of resources than some previous requests. When a sender (receiver) wishes to terminate the reservation-connection, the sender (receiver) sends out a path *teardown message* to release the path state or reserved resources.

4.3.1.2 Levels of QoS

Two different levels are specified by this model:

- *Guaranteed*: It provides firm (mathematically provable) bounds on end-to-end queuing delays by combining the parameters from the various network elements in a path, in addition to ensure the bandwidth availability according to a traffic specification.

- *Controlled Load*: This is equivalent to the best effort service under unloaded conditions. Hence, it is better than best effort level.

4.3.2 Differentiated Services model

In this model [Blake *et al.*, 1998], customers request a specific performance level on a packet by packet basis, by marking the DS field (Type of Service (TOS) byte for the IPv4 header and Traffic Class byte for the IPv6 header) of each packet with a specific value. This value specifies the Per-Hop Behavior (PHB) to be allotted to the packet within the provider's network. Thus, operations such as classification, marking, policing, and shaping need to be implemented at network boundaries or hosts. By marking the requests differently and handling the requests based on these marks, several differentiated service classes can be created. Therefore, this approach is essentially a relative priority scheme.

Providers and customers negotiate agreements with respect to the services to be provided at each customer/provider boundary. The agreements are commonly referred to as Service Level Agreements (SLAs); the subset of the SLA which provides the technical specification of the service is referred to as the Service Level Specification (SLS).

The SLA basically specifies both the service classes supported and the amount of traffic allowed in each class. A SLA can be static or dynamic. Static SLA are negotiated on a regular (e.g., monthly or yearly) basis. Customers with dynamic SLA must use a signaling protocol (e.g., RSVP) to request services on demand.

Customers can mark DS fields of individual packets to indicate the desired service. At the ingress of the ISP networks, packets are classified, policed, and possibly shaped. The classification, policing and shaping rules used at the ingress routers are derived from the SLAs, as well as the amount of buffering space needed for this operations. When a packet enters one domain from another domain, its DS field may be remarked as determined by the SLA between the two domains.

Classification is the process of sorting packets based on the content of packet headers according to defined rules. *Policing* is the process of handling out-of-profile traffic (e.g. discarding excess packets). *Shaping* is the process of delaying packets within a traffic stream to cause it to conforms to some defined traffic profile.

It can be noted that Differentiated services is significantly different from Integrated services [Xiao and Ni, 1999]. First, there are only a limited number of service classes indicated by the DS field which, in the case of the IPv4, indicates the need for low-delay, high throughput or low-loss-rate service. Since the service is allocated to one class, the amount of state information is proportional to the number of classes rather to the number of flows. Therefore, Diffserv is more scalable. Second, sophisticated classification, marking, policing and shaping operations are only needed at the boundary of the networks. The Internet Service Provider core routers need only to have behavior aggregate classification, i.e. the process of sorting packets based only on the contents of the DS field. Therefore, it is more easy to implement and deploy this model.

4.3.2.1 Levels of QoS

There are two Per Hop Behaviors, which represent two service levels (traffic classes):

- *Expedited Forwarding*: It minimizes delay and jitter and provides the highest level of aggregate quality of service. Any traffic that exceeds the traffic profile, defined by a local policy, is discarded.

- *Assured Forwarding*: It delivers the traffic within profile with a high probability, excess traffic can be demoted but not necessarily dropped.

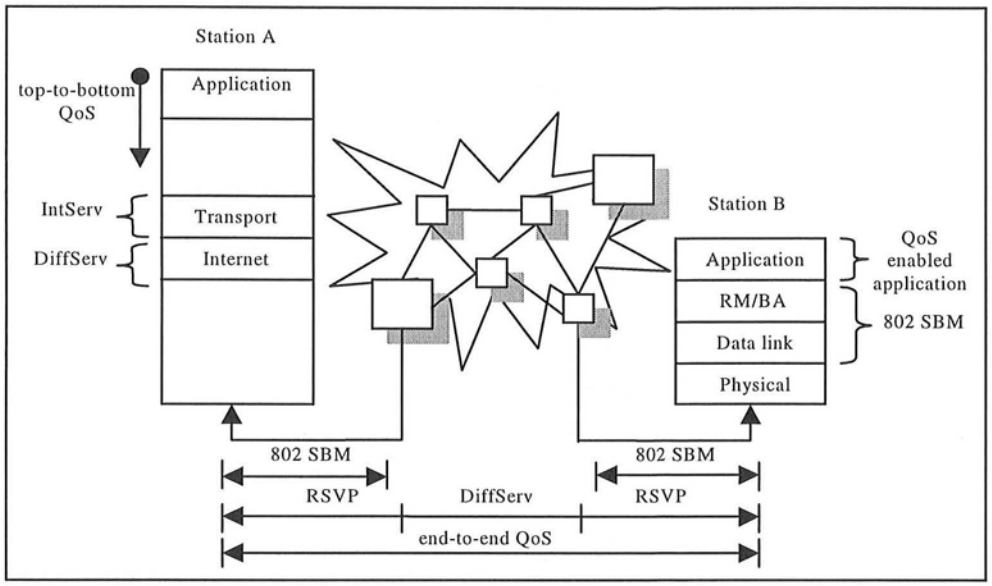

Fig. 4.6. Internet QoS architecture.

4.3.3 SBM architecture

Basically, the Subnet Bandwidth Management is a signaling protocol [Yavatkar *et al.*, 1999] that allows Integrated Services over shared and switched LAN networks [Ghanwani *et al.*, 1999] and enables Integrated Services mapping on these networks [Seaman *et al.*, 1999]. The main components of the SBM architecture are as follows:

- *Bandwidth Allocator* (BA), which maintains state information about allocation of resources on the subnet and performs admission control. BA can be centralized or distributed.

- *Requester Module* (RM), which resides in every end-station (host or router) and not in any switch. This module is responsible for mapping the higher-layer QoS parameters into the data link layer priority levels.

- *SBM protocol,* which is a RM-to-BA or BA-to-BA signaling protocol for reserving resources, querying a BA about available resources, and changing or deleting reservations. This protocol also allows the communication between higher-layer QoS protocols and the RM. The primitives could be implemented as an API for an application to invoke functions of the BM via the RM.

4.4 Conclusion

This chapter has presented three QoS architectures and after that several points can be noted as follows:

The OSI QoS Framework defines a general architecture based on the interactions between the adjacent layers and between a layer and its peer. The quality of a given service is determined by the information to manage one or more QoS characteristics, i.e. the QoS requirements.

The IEC 61158 QoS architecture is based on the ISO/CEI 8886 and is well adapted to the DLS type 1, i.e. the TS 61158 Fieldbus, simply because it defines the four classes of DLS as well as all the QoS parameters for the defined common attributes. Nevertheless, the main argument is clear: a DLS-user select the desired quality of DL service.

In the Fieldbus networks, the DLS-users usually are the application layer entities or, in certain cases, the entities of an interface between the data link and the application layers. So, the QoS concept is propagated to the user application from bottom-to-up. Moreover, the DLS-provider is distributed, i.e. the DL entities may be either publishers or subscribers at the different nodes, therefore end-to-end QoS is provided.

It is interesting to note that the QoS parameters, defined by the IEC 61158, take into account some user time-related requirements such as those analyzed in Chapter 2, i.e. bounded message transfer delay requirement, differentiation between time-critical and no time-critical message requirement, temporal coherence requirement. Nevertheless, the periodicity and the jitter requirements are not taken into account. In other words, the bounded message transfer delay requirement is taken into account by the DLL maximum confirm delay attribute which is applied to connection-mode data transfer service.

The differentiation between time-critical and no time-critical message requirement is taken into account by the DLL priority attribute and the DL scheduling-policy attribute. Temporal coherence requirement is taken into account by the DL timeliness attribute which is only applied to the connection-mode data transfer service. Note that the DL scheduling policy only provides DL user with a means to implicitly or explicitly control any required communication but it does not provide any means to control any periodic communication. Therefore, periodicity requirement is not taken into account.

On the other hand, in the Internet QoS architecture, IntServ model argues that the guaranteed services cannot be given without resource reservation and then a resource reservation protocol is required, such as RSVP. Diffserv model argues that the packets can be differently marked and then classified, packets in different classes receive different levels of service.

IntServ could be supported by the LANs using the Subnet Bandwidth Management (SBM) approach, if all the traffic passes through at least one priority switch using the IEEE 802.1p, 802.1Q, or 802.1D.

It can be noted that the top-to-bottom Internet QoS architecture and the bottom-to-top IEC 61158 QoS architecture could converge at the data link layer if the priority switch of the SBM

approach and the DLL priority attribute of the IEC approach are used. However, as the Chapters 3 and 4 have analyzed, the DLL priority attribute is not defined by all the Fieldbus networks. Therefore, new QoS solutions are required and these must be dynamic solutions.

In fact, from the resource reservation point of view, WorldFIP is a typical example of *centralized* and *static* resource reservation mechanism, i.e. there is one node, the BA, having different lists of identifiers that define at any moment the communication requirements of all the nodes. These requirements are known in advance and will not change during application operation. Hence, it is possible to assign an identifier to each variable and message. Moreover, dynamic resource reservation is possible within a Macro Cycle since aperiodic variables and messages can be handled in their respective windows. So, the fact that the response frame is used to request a future transmission can be seen as a basic dynamic resource reservation primitive.

In the same way, the acyclic request or send/request operation and the cyclic send/request (polling) operation of Profibus can be seen as two levels of QoS, where the response is also used to request future transmissions.

The windows of ControlNet also can be seen as levels of QoS, given that the NUI is cyclically repeated; the scheduled window provides a cyclic and acyclic guaranteed service, unscheduled window provides a best-effort service possibly with a bounded upper time if only one message per node is transmitted in this window. However, it can be noted that ControlNet is a *distributed* and *static* resource reservation mechanism, i.e. all the nodes must be configured in advance, at network configuration time in order to have one slot into the scheduled window and, on the other hand, each node must know the time critical messages to be transmitted.

Next chapter presents then a dynamic resource reservation protocol that may be used by the centralized Fieldbus networks.

Chapter 5

Quality of Service and Client-Server Models

The previous chapters have shown that most of the Fieldbus networks provide different guaranteed levels of QoS mainly for the periodic traffic, basically because it is known in advance and therefore all the required resources could be reserved at network configuration time in order to guarantee bounded message transfer delays and message scheduling at network operation time. The chapter 4 also shown that some of these networks provide certain service primitives to request dynamically some levels of QoS.

We believe that these service primitives must be generalized and integrated into a Fieldbus QoS architecture in order to provide guaranteed levels of QoS at network operation time. Specifically, We propose to use the resource reservation approach at network operation time, this is the approach used by the OSI QoS Framework and the Integrated Services model of the Internet QoS architecture.

The objective of this chapter is then to define a QoS architecture for the client-server model based on the resource reservation approach in order to provide dynamically different levels of QoS. The client-server model has been chosen because it allows to establish the relationships between the OSI layers and because nowadays it only provides a best effort level of QoS.

Thus, this chapter presents a QoS architecture for the client-server model and some extensions for the multiclient-server and client-multiserver models. In addition, a Client-Server Resource ReSerVation Protocol is presented, which is specified in Specification and Description Language (SDL) and then validated with the aid of ObjectGEODE.

5.1 Introduction

The cooperation and communication models have been introduced to establish the relationships between entity pairs [Vega and Thomesse, 1995]. Thus, for example, the notion of client and server is applied to different pairs of entities: users, applications, process, layers, and so on.

In the client-server model, two entities are in relationship: the client and the server. The former sends to the server a service request (Req). The server, according to its resources, processes the indication (Ind) and sends back a response (Rsp), which is received by the client as a confirmation (Cnf). A client can only request services that conform to the client interface for a given server. A *client interface* [Adler, 1995] specifies the individual services or operations supported by the server.

A client can be seen as a triggering entity, and a server as a reactive entity [Andrews, 1991]. Client makes requests which trigger reactions from the server. Thus, a client initiates an

activity, at the time of its choosing; its behavior may be either *blocking* or *nonblocking*. In the former case, the client blocks until it receives a confirmation. In the second case, the client proceeds but it must poll a local queue until confirmation become available. On the other hand, the server waits for requests to be made, then reacts to them.

The client-server model is used in the OSI Reference Model to establish the relationships between adjacent layers and between a layer and its peer. In the OSI model, each layer is a client of its underlying layer and each layer is a server of its upper layer [Vega, 1996], as Fig. 5.1 shows.

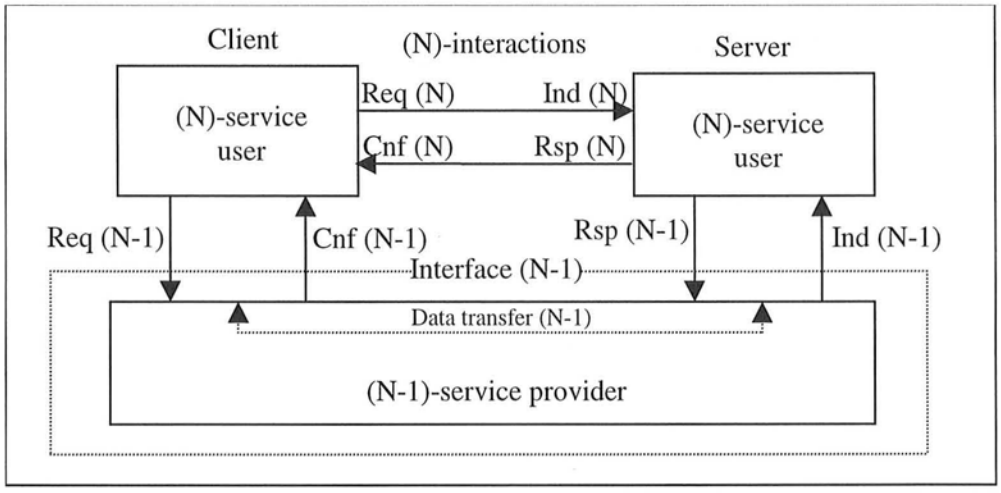

Fig. 5.1. Relationships between adjacent layers and between a layer and its peer.

Client-server model promotes modular, flexible and extensible system design [Adler, 1995]. Nevertheless, the server only provides best-effort services, i.e. the server performs the operations indicated by the service request, but no guarantee is given in the response time, except if some assumptions are made on the availability of the server and its resources [Thomesse *et al.*, 1995].

An approach to provide guaranteed services argues that each service request has been assigned a priority. Server entity will process the requests according to a scheduling policy. However, this schema alone may not provide different QoS levels. It can be argued a preemptable and priority based policy for the services using either priority inheritance [Sha *et al.*, 1990] or priority ceiling [Sha and Goodenough, 1990] protocols. Nevertheless, the execution time in a server is usually longer than the execution time in a critical region, so that this schema alone may not solves the priority inversion problem [Nakajima and Tokuda, 1998].

Priority inversion problem refers to situations where the server is undertaking actions on behalf of a low priority service while preempting a high priority service. Thus, it is impossible to compute the worst case blocking time of the higher priority service accessing a shared server, and therefore, best effort services can only be given.

Furthermore, in [Wedde *et al.*, 94] it has been proved that, even for the subclass of client-server programs, computing the worst-case blocking time for execution paths of tasks that use shared resources is NP-hard.

Another approach allows the server to have several threads for processing requests in order to improve preemptability of server [Nakajima and Tokuda, 1998; Graham and Majumdar, 1999]. When an indication arrives at the server, a thread is selected for processing it and the service is started if the thread is idle. The duration of priority inversion is determined by the thread selected for processing requests. Nevertheless this approach is dependent on the implementation, i.e. the thread execution must be support by an operating system.

This chapter presents then the resource reservation approach as a means to provide guaranteed levels of QoS. This chapter is structured as follows: Section 5.2 analyzes the client QoS requirements. Section 5.3 presents the QoS client-server model. Section 5.4 presents the QoS multiclient-server model. Section 5.5 presents the QoS client-multiserver model, and section 5.6 presents the conclusion. Appendix A, B and C present all the specification of these models in SDL.

5.2 Client QoS requirements

Client requirements depend on the type of client entity, it can be an end-user, an application, a process, a layer and so on. Whichever is the client, its requirements must be quantified and expressed as a set of QoS parameters, which express information to manage one or more QoS characteristics, i.e. quantifiable aspects of QoS, according to the OSI QoS Framework.

Chapter 2 presented an analysis of the user time-related requirements in terms of message transfer delay, periodicity, jitter, time coherence and spatial coherence. These requirements will be taken into account in section 5.4 to define the QoS parameters of the client-server QoS model.

The message transfer delay requirement must take into account that there is a 2-way message exchange, sometimes called latency time, i.e. one from *client to server one way message transfer delay* (t_{CS}), which is the time interval between the request and the indication, and other from *server to client one way message transfer delay* (t_{SC}), which is the time interval between the response and the confirmation, as Fig. 5.2 shows.

This distinction is required because the latency time is often computed by dividing the time elapsed during a round-trip message exchange by 2 (i.e. average one-way message transfer delay). This can give misleading results in distributed environments where the one-way message transfer delay in each direction of a 2-way message exchange may be quite different [Irey *et al.*, 1998].

So, at the client side, the *transaction time* (t_T) is the time interval between a request and its confirmation.

In addition to the previous requirements, in the client-server model, another user time-related requirement must be taken into account: the *service time* (t_S), which is the time interval between an indication and the response, at the server side.

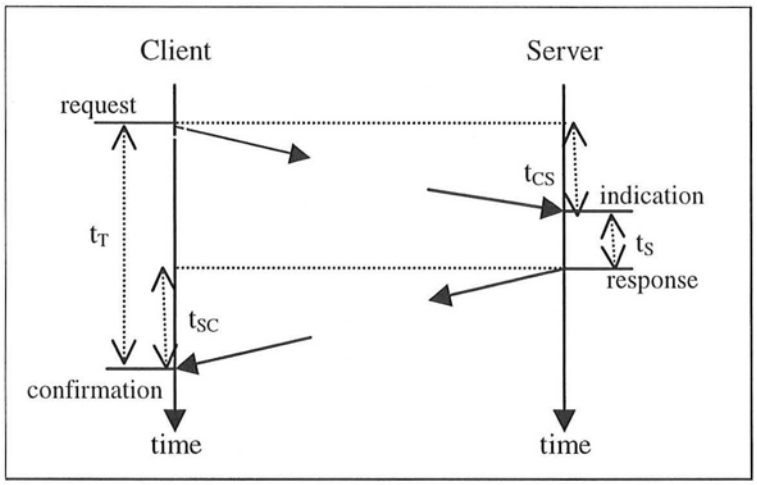

Fig. 5.2. Client-Server Model.

5.3 Client-Server QoS model

This section presents the basic QoS model, one client and one server. The client-server QoS model is based on the *resource reservation* approach, i.e. the server resources are apportioned to the client QoS requests, and subject to a management policy. Before any use of a service with some guaranteed level of QoS, the client must first reserve server resources. So, a Resource ReSerVation Protocol is required between the client and the server (CS-RSVP).

The client-server QoS model has the architecture presented in next subsection, provides the levels of QoS discussed in subsection 5.3.2 and uses the CS-RSVP described in section 5.4.

5.3.1 Client-Server QoS Architecture

Section 4.4.1 shown that given that the resources are finite, resource reservation and admission control are key building blocks of a QoS architecture. Moreover, the basic insight behind client-server model is separating the aspects of access and management of resources [Andreoli and Pareschi, 1996]. So, in addition to normal entities into the client-server model, i.e. the Client Entity (CE) and the Server Entity (SE), the following components are required and are shown in the figure 5.3:

- *Resource Allocator* (RA), which maintains a state structure about allocation of resources on the server and performs Admission Control (AC).

- *Scheduler* (SC), which schedules the accepted service requests towards the server entity accordingly to meet the QoS parameters.

- *Requester Module* (RM), which resides in both the client (RMC) and the server (RMS). This module is responsible for negotiating QoS parameters between them using a resource reservation protocol. This module will reside in every client and every server according to the multiclient-server and client-multiserver models, which will be analyzed in sections 5.5 and 5.6.

- *Client-Server Resource ReSerVation Protocol* (CS-RSVP), which is a reservation protocol between the client and the server. Before any use of a service with a level of QoS, the following steps must be executed: (1) The client specifies the required QoS; (2) The QoS parameters are conveyed to the server; (3) QoS parameters must be mapped to the required resources; (4) Required resources may be either admitted, reserved and allocated or rejected and possibly negotiated; (5) The close-down procedure concerns resource deallocation.

  Basically, these steps remain without change in the QoS multiclient-server model with the exception of the step (4), which must add a queue to process several requests from the clients.

  For the QoS client-multiserver model, some steps remains without change but others must be extended as follows: (2) The QoS parameters are conveyed along the path between the client and the servers; (2a) The QoS parameters among different servers must be translated if their representation is different; (4) The required resources may be either admitted/reserved/allocated or rejected/possibly negotiated along the path between the client and the servers; (5) The close-down procedure concerns resource deallocation along the path of the reservation.

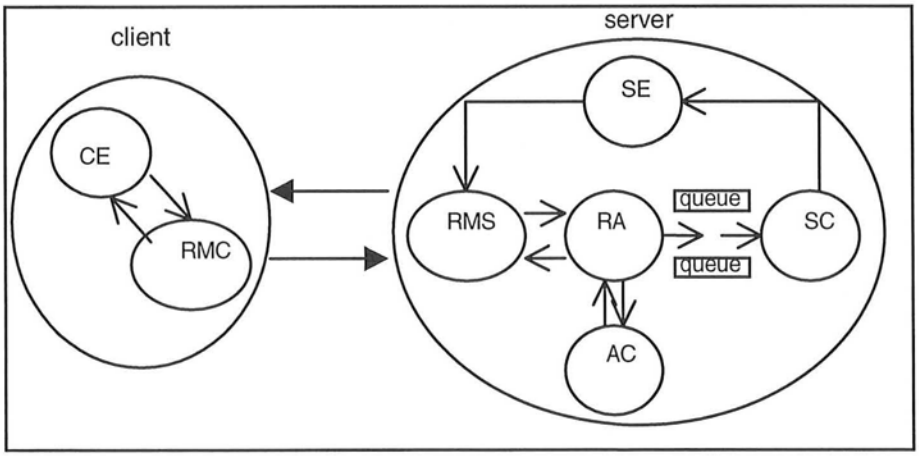

Fig. 5.3. Client-Server QoS Architecture.

In the client-server QoS architecture shown in the figure 5.3, the Client Entity (CE) passes its QoS parameters to its requester module (RMC) which creates a Resource ReSerVation (RSV) message sending it to the server. The Requester Module of the Server (RMS) receives it, passing it to the Resource Allocator (RA), which invokes the Admission Control (AC) mechanism, it performs admission tests (the two following tests are proposed: temporal and schedulability tests, discussed in section 5.4.2.2) to determine whether the QoS parameters

embedded into the RSV message are met. If yes, RA classifies, puts the RSV message in a specific queue, updates its state structure, and sends to the client an accept response. Otherwise, RA creates a reject response sending it to the RMC. Once received, RMC may start a negotiating process changing the QoS parameters according to the recommendations made by the RA and contained into the reject response. Finally, SE performs the requested services and sends the service response through the RMS.

Note that the scheduler (SC) may use different scheduling algorithms, e.g. rate monotonic, least laxity or earliest deadline. Note also that the schedulability test is performed by the AC mechanism and that the scheduling policy is performed by the SC mechanism.

Note that this description uses the term RSV message, nevertheless the service primitives are defined in section 5.4.3.1. This description will be further refine in the following sections.

5.3.2 Levels of QoS

At least two different levels of QoS are required:

- *Guaranteed services*, the server guarantees that the QoS requirements are met, as they are specified through the QoS parameters, and expressed as either deterministic or statistical requirements [Ferrari, 1990; ISO 13236]. Deterministic values could be given as follows: single value, an upper or lower limit, an upper or lower threshold, an operating target, etc. Statistical values could be given as maximum, minimum, range, mean, variance and standard deviation, etc.

- *Best effort services*, there is no assurance that the QoS levels will be provided. Some bounds in the deterministic or statistical representation cannot be satisfied.

5.4 Client-Server Resource ReSerVation Protocol

This section analyzes the main design goals and design principles of the CS-RSVP as well as presents its specification in Specification and Description Language (SDL) and its verification using ObjectGEODE.

5.4.1 CS-RSVP Design goals

This protocol will must take into account the existence of multiple and heterogeneous clients, every with its own QoS requirements, even though some clients may request the same service. Then, CS-RSVP must allow clients to specify their QoS requirements.

A key point in this specification is the regularity of the requests, i.e. if for each service request sent by a client, it must invoke the RSVP, it is clear that this may give rise to an explosion in protocol overhead when the number of clients is big and also when a negotiation process exists. Thus, CS-RSVP must allow clients to specify the regularity of their requests as either *periodic* or *aperiodic*. If the request will be sent at fixed time intervals, it will be periodic,

otherwise aperiodic. Note that in this way the periodicity requirement analyzed in Chapter 2 is taken into account.

Furthermore, in the case of periodic requests, the reservation protocol must also allow to specify its validity, called *reservation validity*, i.e. the time interval during which the resources will be reserved, e.g. one day, one month, one year, etc. This approach is called reservations in advance.

In addition to the regularity, the CS-RSVP must allow to specify the *starting time* and the *duration of the client requests*, as it is suggested in [Degermark *et al.*, 1997; Ferrari *et al.*, 1997] for advance reservations. Finally, CS-RSVP must allow to specify whether the QoS parameters are *negotiable* or not.

The CS-RSVP is a signaling protocol to reserve resources, it is not a MAC protocol, not even a routing or transport protocol, and should avoid replicating any access control or routing function. CS-RSVP' task is to establish and maintain resource reservation between the client(s) and the server(s).

In summary, CS-RSVP is a means used by clients to communicate their QoS requirements to the server in a efficient way, independent of the specific requirements.

5.4.2 CS-RSVP Design Principles

5.4.2.1 Client initiated reservation versus Server initiated reservation

An important design goal is to decide whether the client(s) or the server(s) should have responsibility for initiating reservations. A client knows the qualities and quantities of the service requests that it can send, while a server knows the qualities and quantities of the available resources. Next paragraphs analyzes both approaches: client-oriented and server-oriented protocols.

In a *client-oriented protocol*, the client sends to the server a RSV message which indicates the QoS parameters of the service request. Upon receiving this message, the Resource Allocator (RA) invokes the Admission Control (AC) mechanism, which performs admission tests to determine whether the QoS parameters are met. If yes, RA creates a request identifier (requestID), stocks the QoS parameters into a state structure, puts the requestID into the appropriate queue, and sends to the client the requestID as an accept response. So, the client will use this identifier in future service requests.

If the QoS parameters cannot be met, RA creates a reject response sending it to the client, if the QoS parameters are negotiable the response contains the QoS parameters that could be met. Upon reception, the client may initiate a negotiating process. Whereas client decides to accept or reject these parameters, their values must be locked by the AC mechanism waiting for either a reservation message or a lock timer expiration.

Note that the refresh messages used by the Integrated Service model of Internet are not further required, i.e. the client does not have to send periodically messages to refresh its QoS

parameters. The server knows the regularity of the request and the time interval during which it is valid. When the request is no longer valid, the server invokes a close-down procedure, which concerns resource deallocation.

However, it can be noted that one or more clients may temporally be down or disconnected. In any case, the server reserves the resources at right time, but if the request (requestID) does not arrive within a given time, the server must release the resources. This procedure may be repeated *n* times before resource deallocation.

On the other hand, in the *server-oriented protocol*, once any resource is available for a time interval, the server broadcasts a *WHO_message* specifying both the resource and the QoS characteristics. Upon receiving this message, clients, requiring the resource, send a RSV message specifying the QoS parameters of their service requests. This protocol may continue then as a client-oriented protocol. However in this approach, the server requires a resolution method if several clients answer at the same time. Moreover, since the server starts the reservation process whenever a time interval is available for a given resource, an explosion in protocol overhead may exist if the time intervals are very small. On the other hand, if the time intervals are very high then the resource may be useless for long periods. Therefore, this time interval must be carefully defined.

Note that in this approach the server is not longer a reactive process, it triggers the reservation process; and clients react to it, even if the server still waits for service requests.

The CS-RSVP specified in the next subsection is a client oriented protocol. In fact, the description presented in subsection 5.3.1 is a client-oriented protocol.

5.4.2.2 Admission Control mechanism

The AC mechanism must perform admission tests to determine whether the QoS parameters contained into the RSV messages could be met.

The proposed client-server QoS model should perform two types of test: temporal test and schedulability test. The *temporal test* determines whether the temporal parameters could be met, e.g. the starting time, the time interval of resource utilization, the period, and the time interval during which the reservation is valid. The *schedulability test* determines whether the service requests could be schedule according to the policy used by the scheduler, e.g. rate monotonic, earliest deadline first, and so on.

Temporal test is required because several reservation requests may have identical temporal parameters, i.e. several reservation requests may request the same starting time, or an interception exists either in the time intervals of resource utilization or in the periods. This temporal test may be based on, for example, a dynamic storage allocation algorithm, which reserves and frees variable-size blocks of memory from a larger storage area. There is a lot of such algorithms and their discussion is out of the scope of this chapter. Nevertheless, in the client-server QoS model, the AC mechanism may reserve and free variable-size blocks of time intervals from one RSVP invocation to the next one.

Thus, in order to accommodate and negotiate future RSV requests, the AC mechanism may manage a data structure (e.g., a list) to link together all the available time intervals. Each item contains the starting time, the size or length of the free time interval, and the starting time of the next free time interval. The free blocks can be linked in order of starting times.

The AC mechanism may also manage a data structure to link all the reserved time intervals assigning to each item one or more timers to control its periodicity and its validity time intervals.

Once a RSV message arrives at the AC mechanism, it locates the starting time of the request within the free time interval list. If the item exists, AC compares the time interval of resource utilization with the length of the free time interval, if the first is smaller than or equal to the second, then AC calculates the first period of the request, if it is periodic, and then locates the appropriate item into the list to compare the time interval of resource utilization with the length of the free time interval. This procedure is repeated until reservation validity time.

If all the tests are true, The AC mechanism cuts all the appropriate blocks of the free time interval list, and creates a item within the reserved time interval list. Otherwise, if any test is false, AC rejects the RSV request. If it is negotiable, AC may modify the temporal parameters of the RSV request according to the *best-fit* free time interval. The AC mechanism is not specified in the current version of CS-RSVP, but it is a possible extension.

5.4.3 Specification and validation of CS-RSVP

The essence of a protocol is the behaviour of both the communicating protocol entities and the underlying medium, i.e. the set of possible sequences of events that can occur when the entities execute [Pehrson, 1990].

There are different approaches to specify the protocols and their properties. Behaviours are often specified in terms of some kind of transition system, e.g. state machines, nets, traces and so on.

Specifically, the ISO and the International Telecommunication Union (IUT) have standardized the Formal Description Languages (Estelle [ISO 9074], LOTOS [ISO 8807] and SDL [ITU Z.100]) to allow the user to express all the needed details in an implementation specification. Estelle and SDL are based on the transition system approach, while LOTOS is based on the family of process algebra [Bolognesi and Brinksma, 1987].

The CS-RSVP is specified in Specification and Description Language (SDL) because it is an object-oriented language with concepts for describing the logical structure, data and behaviour aspects of systems. SDL provides both a graphical representation (SDL/GR) and a prose (text) representation (SDL/PR) and allows translation between both. SDL is a language that is intelligible to human beings, formal enough to support analysis and comparison of behaviours [Bræk, 1996].

SDL is especially suited for describing reactive and discrete systems, i.e. reactive systems are those whose behaviour is dominated by interactions between action inputs to the system and

reaction outputs by the system. Discrete systems are those whose interactions appears at discrete points and by means of discrete events. SDL is only intended to describe functional properties and it is not intended for describing non-functional properties of systems like capacity, weight, power consumption, performance and so on.

For validation purposes, the term validation model is used in the SDL world. It is a description of the system which is suitable for validation, i.e. to apply validation techniques such as testing the formal model, exhaustive validation (reachability analysis), non-exhaustive validation (analysis of a random subset of the reachable states), simulation and informal validation techniques (checklist) [Hogrefe, 1996]. Fig. 5.4 presents the validation scheme for a validation model specified in SDL, where ASN.1 may be used to define the data structures of the system for storing and exchanging data, Message Sequence Charts (MSC) [ITU Z.120] are normally used to illustrate the most significant system executions in terms of message flow through the system and to and from its environment.

A validation model is always executable, so, in order to construct a validation model of CS-RSVP the toolset ObjectGEODE v.4.0 of Verilog is used. It includes the following tools: SDL editor, MSC editor, SDL & MSC checker, SDL & MSC interactive simulator, SDL & MSC exhaustive simulator, C & C++ code generator, C run-time library and design tracer [Cheng, 1996].

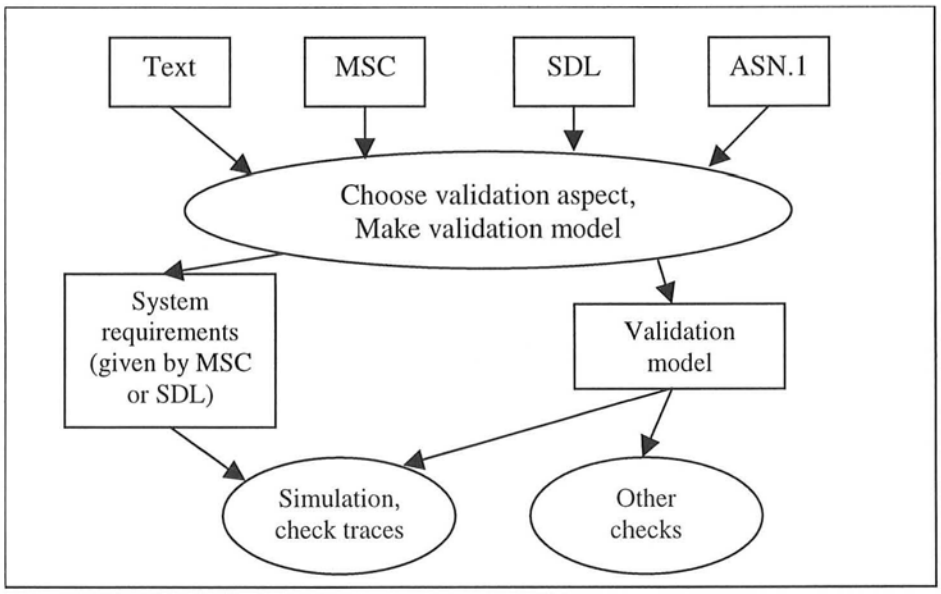

Fig. 5.4. The SDL validation scheme.

5.4.3.1 Definition of CS-RSVP service primitives

The CS-RSVP should provide the following service primitives:

- *RSVcreate*, which is a request to reserve a resource. The QoS parameters are as follows: the resource name, a boolean variable to indicate whether the resource utilization is periodic, the period value if it is the case, the starting time, the time interval of resource

utilization, the time interval of reservation validity, and a boolean variable to indicate whether the reservation request may be negotiated.

- *RSVaccept*, which is a response to indicate that a RSVcreate is accepted. Its QoS parameter is a request identifier.

- *RSVreject*, which is a response to indicate that a RSVcreate is not accepted. Its QoS parameters are as follows:

  a) If the RSVcreate is negotiable, i.e. the boolean variable is equal to TRUE: the resource name, the boolean variable indicating periodic or aperiodic resource utilization, the period that could be met, the starting time that could be met, the time interval of resource utilization that could be met, the time interval of reservation validity that could be met, and the boolean variable is equal to FALSE to indicate that the reservation request cannot be further negotiated, i.e. only one negotiation exchange is allowed.

  b) If the RSVcreate is not negotiable, i.e. the boolean variable is equal to FALSE: the QoS parameters are equal to the RSVcreate parameters.

- *ServiceReq*, which is a request to use a previously reserved resource. Its QoS parameter is a request identifier.

- *ServiceRsp*, which is a response to indicate that a ServiceReq is done. Its QoS parameter is the request identifier.

- *NorequestID*, which is a response to indicate that a ServiceReq identification does not exist. Its QoS parameter is the request identifier.

- *RSVmodify*, which is a request to modify a previously reserved resource. Its QoS parameter is a request identifier. This service primitive is not specified in the current version of the protocol.

- *RSVdeallocate*, which is a request to deallocate a previously reserved resource. Its QoS parameter is a request identifier. This service primitive is not specified in the current version of the protocol.

5.4.3.2 Specification of CS-RSVP service primitives

In SDL, behaviour is always performed in the context of a *system*, beginning with a top level description of the system. Fig. 5.5 shows the CS-RSVP system, which is composed of three *blocks* (client, server, and ServiceProvider) interconnected through *channels* called c1 and c2. In addition, the figure shows the *signals* passed in each direction over the channels, as indicated by the arrows on the channels. Moreover, the figure shows the declaration of these signals as well as the declaration of the data structure (called RSV), which defines the QoS parameters used by the service primitives.

In SDL, a system only occurs at the top level, while blocks only occur inside. System is decomposed into blocks and channels recursively over as many levels as desired until the basic components, called *processes*, are reached.

Figures 5.6 and 5.7 present the client and server blocks, we suppose that the ServiceProvider block represents a reliable (N-1) data transfer subsystem.

One *SDL process* is a concurrent object with its own control flow, described by an Extended Communicating Finite State Machine (FSM), which is composed of the following four main parts: input port, FSM, timers and variables.

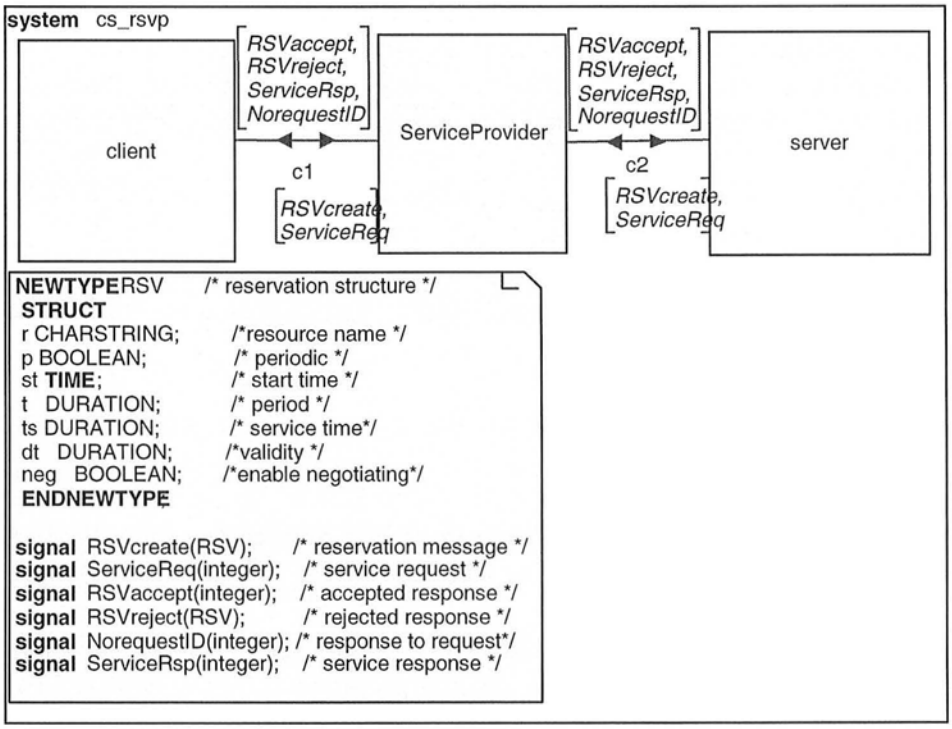

Fig. 5.5. CS-RSVP system.

The input port contains an unbounded queue of incoming signals. Signals arriving at the process will be merged into the input port in the order they arrive, conflicts are resolved by selecting an arbitrary sequential order. Signals will remain in the input port until they are consumed by the FSM, which performs one transition from one state to another state for each consumed signal. This transition takes a short but undefined time. If there are no signal in the input port, FSM remains in the same state until a signal arrives. On each transition, FSM may generate output signals, perform operations on the variables and timer operations. This FSM state-transition behaviour is expressed in terms of a *process diagram.*

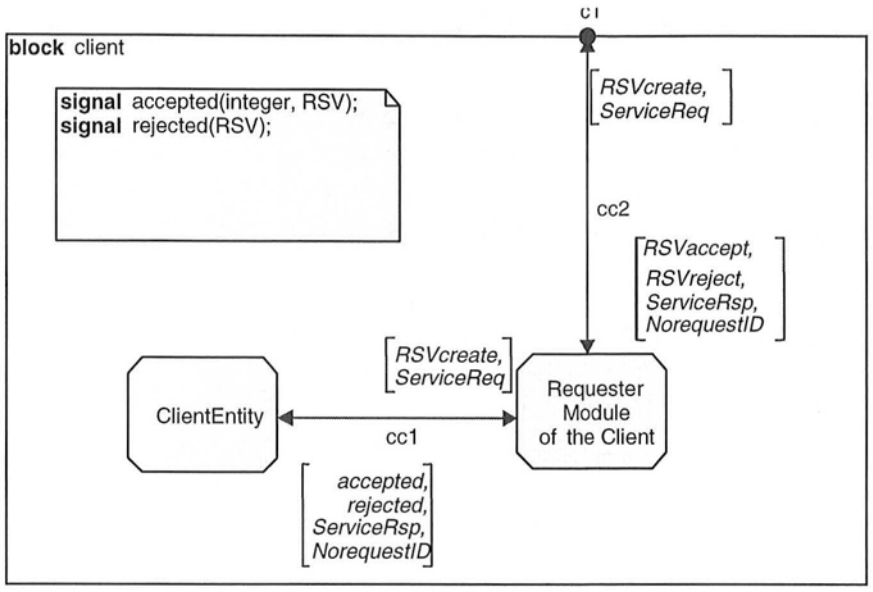

Fig. 5.6. Client block.

The client block, shown in the Fig. 5.6, consists of the following two processes: ClientEntity and Requester Module of the Client (RMC), the former is described in the process diagram of the Fig. 5.8, the RMC process diagram is presented in the Appendix A (Fig. A.4).

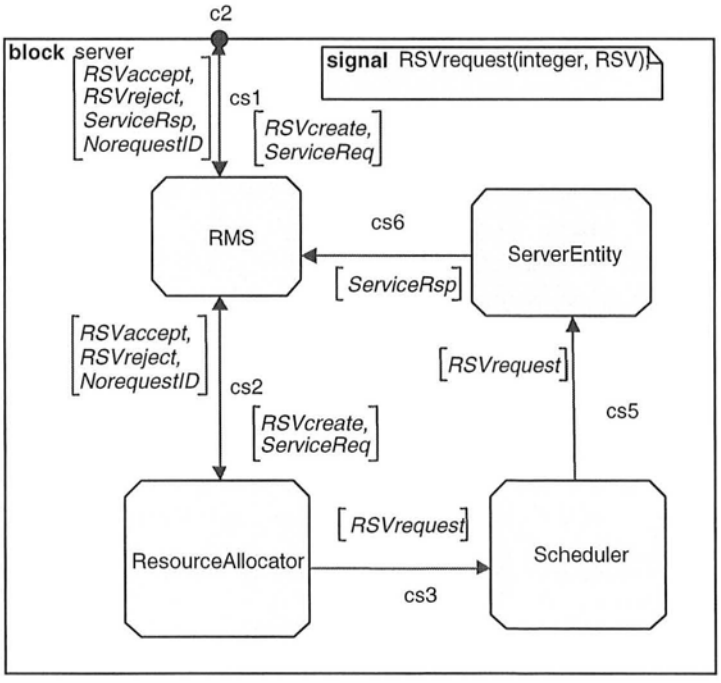

Fig. 5.7. Server block.

The server block, shown in the Fig. 5.7, is composed of the following four processes: Requester Module of the Server (RMS), Resource Allocator, Scheduler and ServerEntity. The Resource Allocator process is shown in the Fig. 5.9, the process diagrams of the remaining processes are presented in the Appendix A.

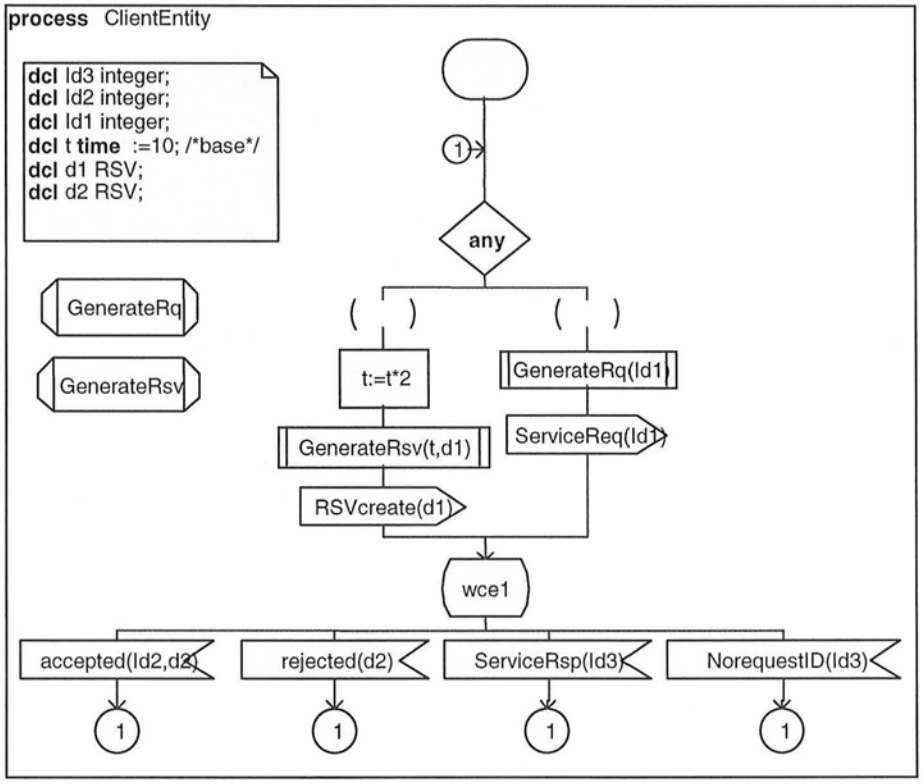

Fig. 5.8. ClientEntity process diagram.

In order to construct a validation model of CS-RSVP, the following two procedures are defined in the Fig. 5.8: GenerateRSV and GenerateRq, which are randomly invoked by the decision symbol tagged as *any*. After procedure call, FSM generates either an output signal RSVcreate or an output signal ServiceReq, having as parameters a resource reservation structure (d1) or a request identifier (Id1), respectively. After that, FSM waits for an input signal, which can be either accepted or rejected, or either ServiceRsp or NorequestID. The first two are two responses to a reservation create request (RSVcreate). The latest two are two responses to a service request.

Note that the input signal accepted(Id2, d2) has two parameters, the first is a request identifier and the second is the reservation data structure (RSV) which has the QoS parameters that have been accepted by the Resource Allocator process at the server. Note also that if a negotiating process exists, the values into this data structure may be different from the values into the original reservation data structure of the output signal RSVcreate(d1). On the other hand, the request identifiers should be stocked for future service requests, it is not shown in the Fig. 5.8, but are randomly generated by the GenerateRq procedure for simulation purposes.

The input signal rejected(d2) contains as QoS parameter a reservation data structure whose values are those that the Reservation Allocation process cannot meet. The input signal ServiceRsp(Id3) contains as QoS parameter the request identifier, which is sent by the output signal ServiceReq(Id1), this signal indicates that the service was performed according to the

desired QoS. The input signal NorequestID(Id3) indicates that the request identifier sent by the ServiceReq(Id1) does not exist at the server.

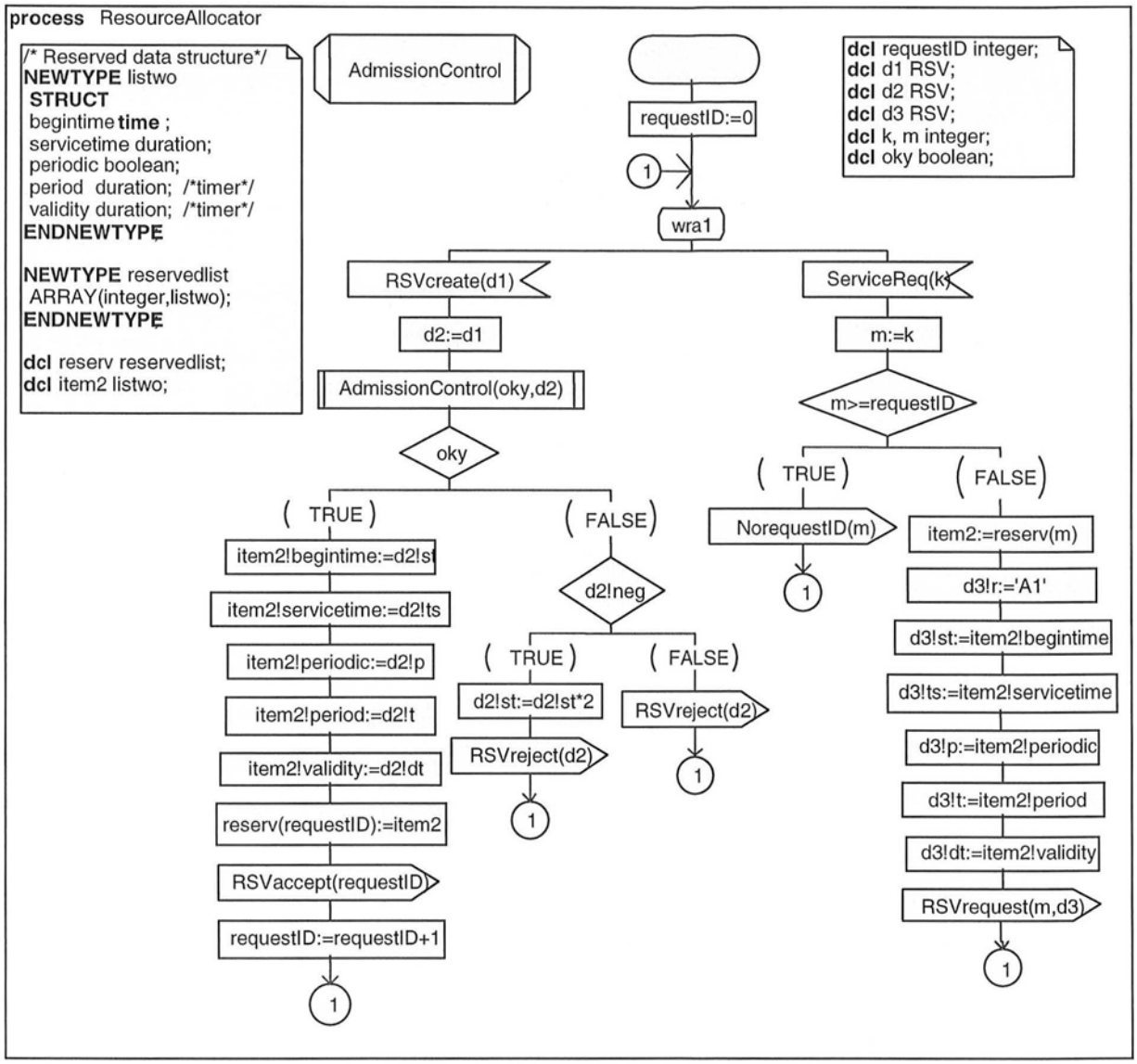

Fig. 5.9. Resource Allocation process diagram.

The figure 5.9 shows the Resource Allocation process, as well as the data type structure for the reserved time intervals, we suppose that there is only one resource in each server, in this case the resource is called A1. The RA process calls the AdmissionControl procedure, which performs both the temporal and the schedulability tests to determine whether the QoS parameters could be met.

The AdmissionControl procedure returns a boolean variable, called oky. If it is equal to TRUE, the QoS parameters are accepted and stoked, then the RA process sends the output signal RSVaccept(requestID) which has as parameter a request identifier for future service requests. Otherwise, if the negotiating variable (d2!neg) is equal to TRUE, the RA process changes the

starting time parameter value, for simulation purposes only, and sends the output signal reject(d2), having as parameter the reservation data structure.

On the other hand, if a service request signal (ServiceReq) is received, RA determines whether the request identifier exists, if it is the case, gets the QoS parameters from the reserved data structure and sends the output signal RSVrequest to the Scheduler process of the server. Otherwise, RA sends the output signal NorequestID.

5.5 MultiClient-Server QoS model

Usually a server provides services to more than one client, then the server can be seen as a entity running in an infinite loop accepting requests from the clients. So, the multiclient-server QoS model is simply a generalization of the client-server QoS model presented in the section 5.3, where all the clients are composed of a ClientEntity and a Requester Module of the Client, and, on the other hand, the server remains without change, as is shown in Fig. 5.3.

For generalization and re-use, SDL uses type concepts for systems, blocks, processes, procedures, services, signals and data. So, new types can be defined by (single) inheritance from other types, i.e. SDL is a full blown object-oriented language [Bræk, 1996]. In this way, the previous CS-RSVP is easily extended for the MultiClient-Server (MCS-RSVP) and Client-MultiServer (CMS-RSVP) QoS models, as the Fig. 5.10 and 5.13 show. The overall specification of these protocols is presented in Appendix B and C, respectively.

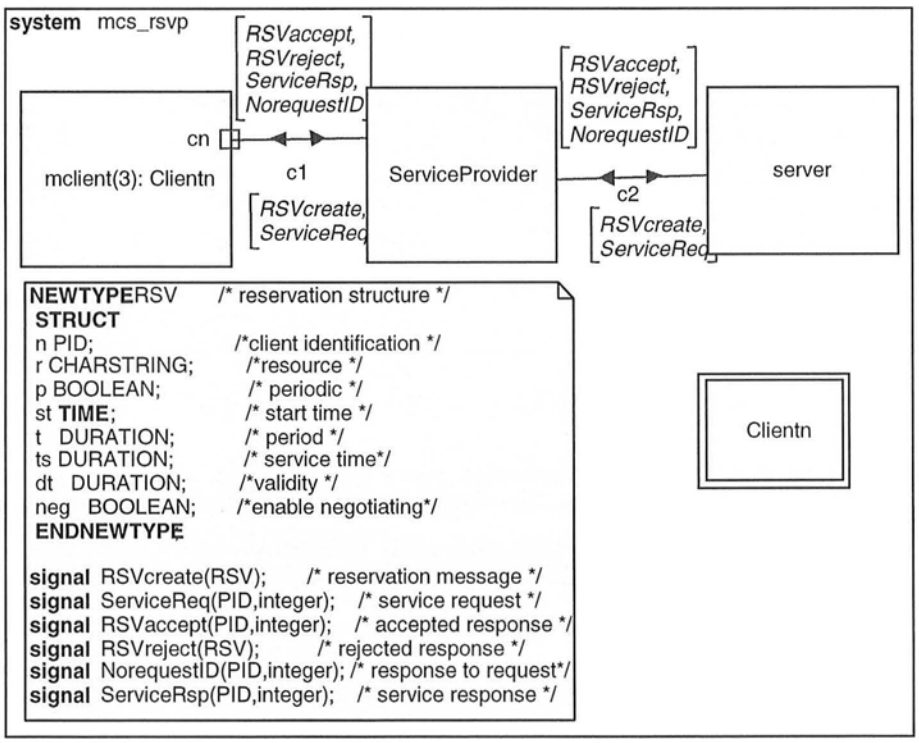

Fig. 5.10. MCS-RSVP system.

The Fig. 5.10 presents the top level description of the MCS-RSVP *system*, which is composed of three *blocks* (mclient, server, and ServiceProvider) interconnected through the *channels* c1 and c2. Note that the *block* mclient(3) defines three instances of the *block type* Clientn, whose declaration is made at system level and is defined as *block type*. Then, the blocks mclient are defined by inheritance from block type Clientn, which is shown in the Fig. 5.11.

Note that the *block* server remains without change, as is specified in the Fig. 5.7. Nevertheless, the *block* ServiceProvider must now manage the signals from and to three blocks mclient. In SDL, the *gate* concept allows to connect each instance of a given type block to a channel, so the gate called *cn* is attached outside the block mclient symbol. A gate is like the signal route symbol and is used to ensure that the instances of the block type are connected properly.

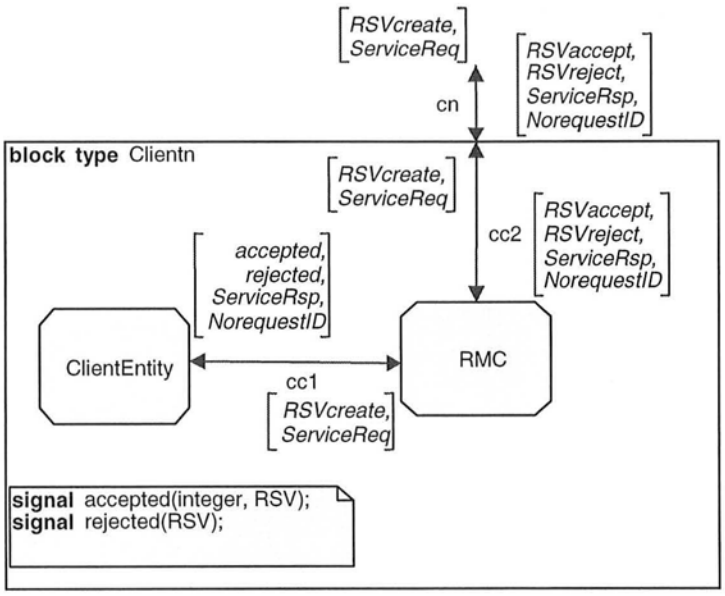

Fig. 5.11. Clientn block type.

5.6 Client-MultiServer QoS model

In this model, a client requests one or more services which are provided by multiple servers. The services can be the same on all the servers, *duplicate server model*, or the service can be divide into multiple (dependent or independent) subservices, *parallel server model*. The first model is well adapted for fault tolerance and to service multiple clients. The second model is well suited for subservices parallel processing [Adler, 1995].

A client can localize and interact with other servers *directly or indirectly* through one or more resource servers, also called main servers. Moreover, a client can request multiple services either *explicitly or implicitly*, shown in the figure 5.12. In the first case, the client issues distinct requests for each desired service. In the second case, the client sends a single request to a main server.

Once an indication has been received, the main server localizes the service and sends to secondary servers the requests for each subservice. At indication time of subservice, the secondary servers perform the service and send back a response. The main server assembles all results into a unified response and sends it to the client [Dakroury and Elloy, 1989; Dakroury, 1990].

Note that the behavior of the proposed CS-RSVP is still valid for the direct client-multiserver model. In this case, the client must reserve resource utilization in each server. On the other hand, some minimal changes are required for the indirect client-multiserver model. First, the main server only needs to know the localization of the resources, the main server does not manage the resources of the secondary servers. Second, the main server distributes the reservation requests among the secondary servers, waits for subresponses, and sends an accordingly response to the client. This activity can be performed by the Requester Module of the Main Server, and its behavior is like a client sending reservation requests and receiving responses.

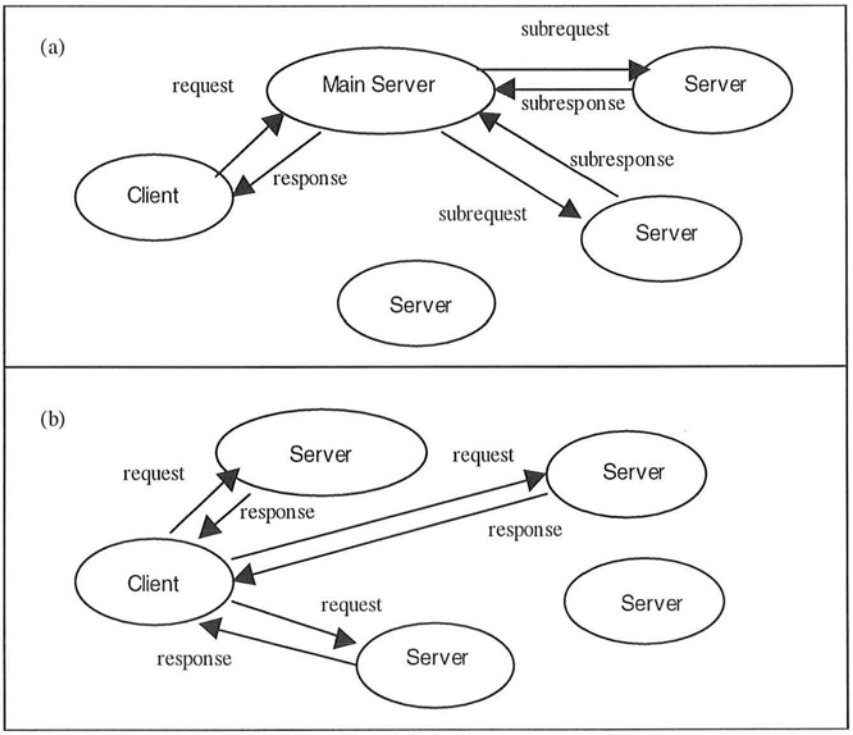

Fig. 5.12. Client-multiserver models: (a) indirect; (b) direct.

The Fig 5.13 presents the Client-MultiServer system, it can be noted that three Secondary Servers have been defined and only one Main Server. The Secondary Servers are instance of the type block SS, which is defined in the block type of Fig. 5.14.

It can also be noted that some other service primitives have been defined in order to configure the resources of the secondary servers in the main server and to reserve and access their services. They are as follows:

- *confresource*, which is a request from the secondary server to the main server in order to have an identifier of the resource. Its parameter is a SDL process identification.

- *configured*, which is a response from the main server to the secondary server to assign a resource identifier. The parameters are the SDL process identification and the resource identifier.

- *subRSVcreate*, *subRSVaccept*, *subRSVreject*, *subServiceReq*, *subServiceRsp* and *subnoServiceReqID* are service primitives equal to the service primitives of CS-RSVP but are used between the Main Server and the Secondary Servers.

The service primitives between the client and the Main Server remain without change.

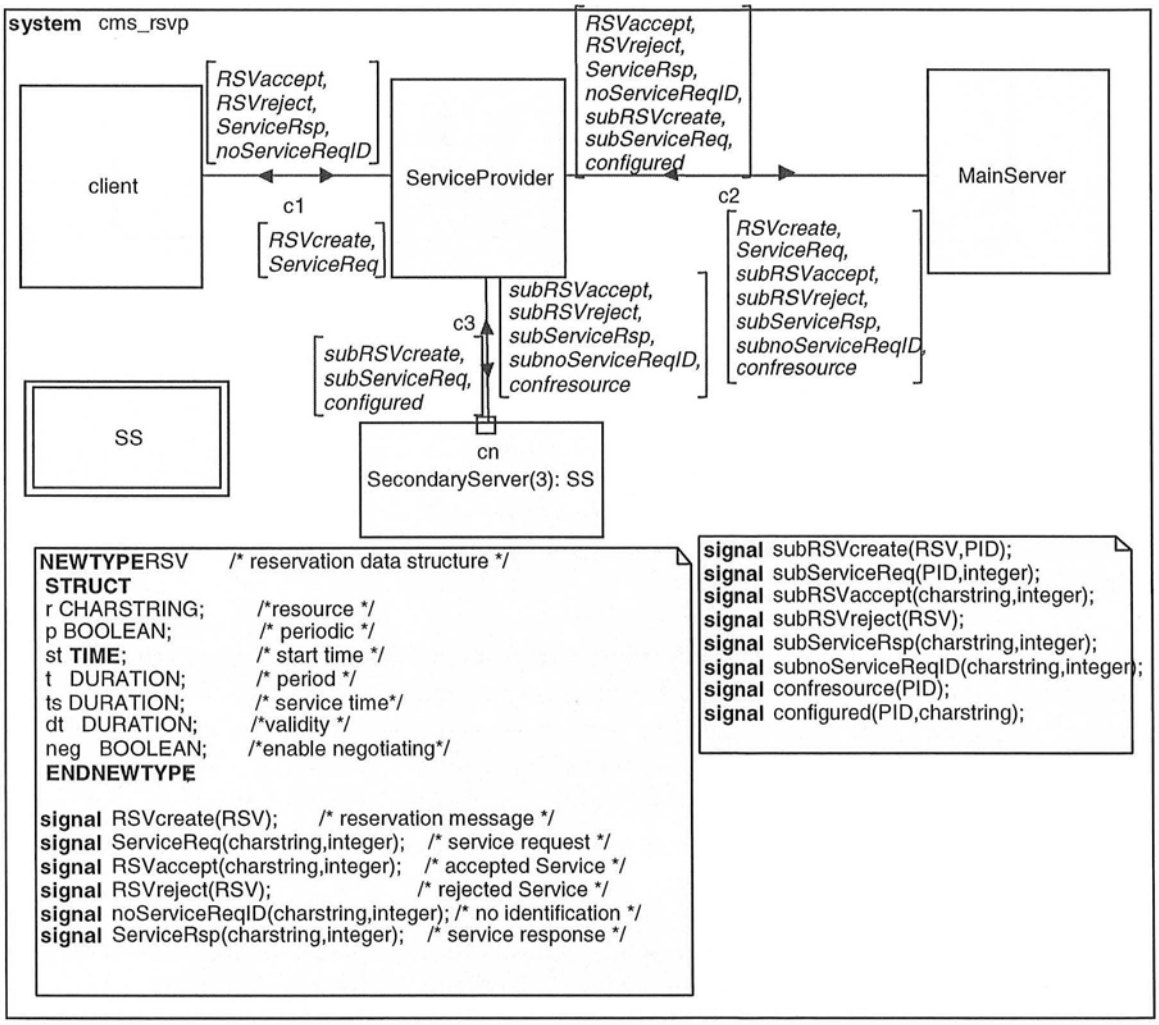

Fig. 5.13. CMS-RSVP system.

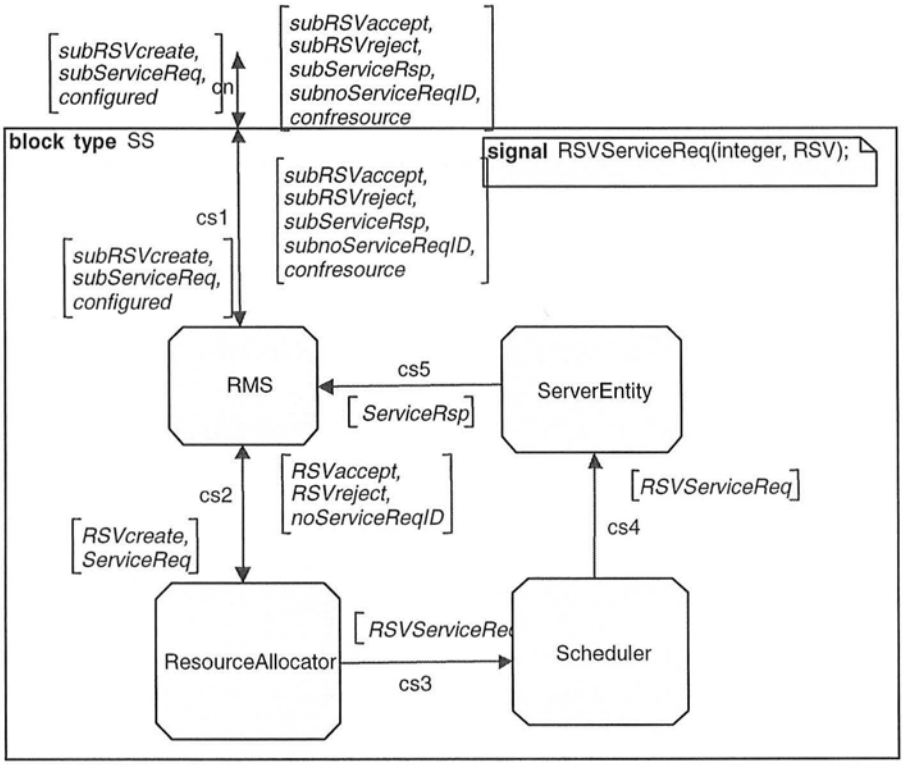

Fig. 5.14. Secondary Server block type.

5.7 Conclusion and Future Work

This chapter has presented a QoS architecture for the client-server model, for the multiclient-server and for the client-multiserver models. This architecture is based on the resource reservation approach used in the OSI QoS Framework and the Internet Integrated Service model.

This chapter also has presented a Client-Server Resource ReSerVation Protocol (CS-RSVP) which is specified in Specification and Description Language (SDL), and then validated with the aid of ObjectGEODE v.4.0 of Verilog. This protocol is then extended for the MultiClient-Server and Client-MultiServer models. Their specifications are completely presented in the Appendices A, B and C as well as some validation results in terms of Message Sequence Charts (MSC).

CS-RSVP allows to reserve resource utilization through a negotiation process between the client(s) and the server(s). The former sends a reservation message (RSVcreate) containing the following parameters: the resource name, a boolean variable to indicate whether the resource utilization is periodic, the period, the starting time, the resource utilization time interval, the reservation validity time interval, and a boolean variable to indicate whether the reservation request could be negotiated.

At the server, after admission tests, the Resource Allocator (RA) responds either with an RSVaccept or RSVreject message. The first contains a request identification that the client will use for future service requests, the second contains the same parameters of the reservation request, but their values are those that RA can meet, if the reservation request is negotiable. Whereas client decides to accept or reject these parameters, their values must be locked by the server waiting for either a reservation request or a lock timer expiration.

This basic reservation protocol will be extended with service primitives such as RSVmodify and RSVdelete.

It can be noted that some SDL processes must be further specified, e.g. Scheduler, ServerEntity and Admission Control. Up to now, the CS-RSVP implementation only takes into account the service requests in a FIFO order and the resource reservation requests are randomly accepted.

On the other hand, Chapter 4 also presented the Internet Differentiated Service model, this approach is essentially a relative priority scheme. So, the client-server QoS model presented in this chapter can be extended as the *Prioritized Client-Server QoS model* as follows: the clients may mark their requests according to a desired class of service. By marking the requests differently and handling the requests based on these marks, several differentiated service classes can be created.

In order to receive differentiated services from a server, each client must have a Service Level Agreement (SLA) with the server. SLA basically will specify the service classes supported and the amount of requests allowed in each class. Next paragraph analyzes the static case.

When a client wishes to send a request, it will be passed to the RMC (requester module) which will have responsibility for marking the request, according to the SLA, and also for controlling the number of requests in each class, request exceeding a threshold must be dropped. When the server receives the request, it will be passed to the resource allocator (RA), which directly will classify and put the request into a queue according to the desired service classes. These queues may require a management scheme that can be performed by the AC. In any case, a request will follow the procedure described in the QoS client-server architecture, presented in subsection 5.3.2.

With regard to the Fieldbus networks, CS-RSVP can be integrated to the centralized approaches as follows:

At network configuration time, a master or bus arbitrator node has knowledge of all the user requirements, generally the network configuration is made by the user. So, MCS-RSVP can be used as a configuration protocol, where the server performs as the bus arbitrator and the clients performs as the slaves or the producers/consumers nodes.

At network operation time, MCS-RSVP can be used in order to provide different levels of QoS. Nevertheless, its integration into the Fieldbus architectures such as WorldFIP, Foundation Fieldbus or Profibus is part of our future research work.

Conclusion et perspectives générales

Les nombreux réseaux de terrain existants ont été définis depuis presque 20 ans pour satisfaire les besoins de communication entre capteurs, actionneurs et dispositifs de commande dans divers contextes. Tous ces réseaux présentent des protocoles de MAC qui tentent de garantir une certaine qualité de service temporelle. La plupart des concepteurs a opté pour des configurations statiques, pour des solutions souvent centralisées, pour des réservations de bande passante ou de fenêtres temporelles, de façon à ce que, en cas de bon fonctionnement, le respect des instants d'émission soit garanti.

Cette thèse a présenté une solution pour rendre dynamique certaines de ces solutions statiques en particulier dans le cas où les accès au médium sont contrôlés. Nous nous sommes appuyés pour ce faire sur les concepts qui sont en cours de développement sur l'internet.

Cette solution est issue de deux analyses complémentaires qui sont présentées dans les premiers chapitres. D'une part nous avons analysé les besoins temporels des applications temps réel des systèmes automatisés (chapitre 2), d'autre part les différents protocoles MAC existants, normalisés ou non ont été étudiés et comparés (chapitre 3). Cette comparaison n'est certainement pas exhaustive mais elle intègre les normes actuelles et beaucoup de propositions récentes de nouveaux mécanismes.

Dans le chapitre 2 nous avons présenté une analyse des besoins des utilisateurs afin d'identifier les caractéristiques des services que les systèmes de communication doivent fournir dans les systèmes temps réel. L'analyse s'est concentrée sur les besoins ayant rapport au temps et à l'espace et ne prend pas en compte d'autres besoins parfois importants des utilisateurs tels que la performance ni la sûreté de fonctionnement. Cependant, l'analyse permet d'organiser la discussion des systèmes de communication du chapitre 3.

Cette analyse a montré que le principal besoin des utilisateurs est le délai borné du transfert des messages. Cependant, il y a d'autres besoins temporels selon les utilisateurs, par exemple dans les applications industrielles, ces besoins sont la périodicité, la gigue, les cohérences temporelles et la cohérence spatiale.

Le besoin de périodicité provient du fait que la plupart des données échangées entre les utilisateurs sont périodiquement échantillonnées, ainsi le système de communication doit fournir le service de transmission pour ce type de messages. La transmission des données doit être également périodique, la gigue mesure son éventuelle variabilité.

La cohérence temporelle de transmission ou de réception indique si les messages ont été transmis et/ou reçus dans un intervalle de temps donné. La cohérence spatiale indique si toutes les copies d'une variable sont identiques à un instant donné ou dans une fenêtre de temps donnée.

Le chapitre 3 présente une analyse des principaux protocoles MAC et Réseaux de Terrain à partir de deux points de vue. L'analyse prend en compte les besoins des utilisateurs identifiés dans le chapitre 2. D'autre part, l'analyse considère les caractéristiques des protocoles MAC

identifiant les méthodes d'accès et les algorithmes d'ordonnancement qui satisfont aux besoins des utilisateurs.

Ce chapitre présente des tableaux qui résument les besoins satisfaits par divers protocoles MAC. Nous pouvons en tirer la conclusion suivante :

Quelques-uns des protocoles MAC utilisés par les réseaux de terrain satisfont à la plupart des besoins temporels des utilisateurs, tandis que peu des principaux protocoles MAC satisfont à tous ces besoins. D'ailleurs, les besoins des utilisateurs sont satisfaits dans différentes valeurs, par exemple dans le cas du besoin de délai borné lié au transfert des messages le tableau 4.1 montre que la borne publiée dans divers articles peut être une valeur moyenne, maximale ou de pire cas. Ainsi, les tableaux 4.2 et 4.3 doivent être étendus afin de prendre en compte les diverses valeurs dans lesquels les besoins de périodicité et les contraintes de gigue sont satisfaits.

Nous pouvons également noter que la plupart des réseaux de terrain se servent de méthodes qui tiennent compte des besoins des utilisateurs lors de l'étape de configuration de réseau afin de satisfaire et de garantir ces besoins pendant les opérations du réseau. Néanmoins, si les besoins changent, il est nécessaire d'arrêter l'application industrielle afin de déterminer si le nouvel ensemble de besoins peuvent être satisfaits et garantis. Si c'est le cas, les nouveaux besoins sont intégrés et l'application industrielle est remise en marche.

Cependant, de nombreuses des applications industrielles ne peuvent être arrêtées sans perturbations importantes et des méthodes en ligne sont alors requises. Une solution possible est la méthode de réservation des ressources, qui fait partie d'un plus concept connu sous le nom de qualité de service et qui est analysée dans le chapitre 4.

Le chapitre 4 a présenté trois architectures de QoS à savoir : celle de l'OSI, celle de la CEI 61158 et celle de l'internet. A partir de cela plusieurs remarques peuvent être faites :

Le cadre de QoS de l'OSI définit une architecture générale basée sur les interactions entre les couches adjacentes et entre une couche et sa couche homologue. La qualité d'un service donné est déterminée par l'information de contrôle d'une ou plusieurs caractéristiques de QoS, c.-à-d., les besoins de QoS.

L'architecture de QoS du CEI 61158, basée sur l'ISO/CEI 8886, est bien adaptée au Type de Service 1, c.-à-d., le Réseau de terrain TS 61158, simplement parce qu'il définit les quatre classes de Service de la couche Liaison de Données (DLS) et tous les paramètres de QoS qui ont été définis comme attributs communs pour tous les Réseaux de Terrain. Néanmoins, le chapitre 4 montre que ces attributs ne sont définis que par le TS 61158.

Dans les réseaux de terrain, les utilisateurs du DLS sont habituellement les entités de couche application ou, dans certains cas, les entités d'une interface entre la couche de liaison de données et la couche d'application. Ainsi, le concept de QoS est propagé de bas en haut, c.-à-d. de la couche de liaison de données jusqu'à l'utilisateur de l'application. D'ailleurs, le fournisseur de DLS est distribué, c.-à-d., les entités de DLS peuvent être des éditeurs ou des abonnés dans les divers sites, en conséquence le modèle fournit la QoS de bout en bout.

Il est intéressant de noter que les paramètres de QoS, définis par le CEI 61158, tiennent compte de certains besoins temporels des utilisateurs comme ceux qui ont été analysés dans le chapitre 2, c.-à-d. le délai lié au transfert des messages, la différentiation entre les messages de temps-critique et non-temps-critique, la cohérence spatiale et temporelle. Néanmoins, les besoins de périodicité ne sont pas pris en compte explicitement dans cette norme, il est du ressort du concepteur de construire explicitement l'ordonnancement des messages selon la stratégie qui lui convient.

En d'autres termes, le besoin lié au délai borné de transfert des messages est pris en compte par l'attribut du délai maximal de confirmation qui est appliqué aux services en mode connexion.

Le besoin de différencier les messages temps-critique et non temps-critique est pris en compte par l'attribut de priorité et l'attribut de politique d'ordonnancement. Les besoins de cohérence spatiale et temporelle sont pris en compte par l'attribut de cohérence appliqué au service en mode connexion. Noter que l'attribut politique d'ordonnancement fournit à l'utilisateur un moyen implicite ou explicite de contrôler n'importe quelle communication autre que la communication périodique. En conséquence, le besoin de périodicité n'est pas pris en compte par cette architecture de QoS de même que le besoin de contrôle de la gigue.

D'autre part, dans l'architecture de QoS de l'internet, le modèle dit des services intégrés (IntServ) par du principe que les services garantis ne peuvent pas être fournis sans réservation de ressources en conséquence un protocole de réservation de ressources est requis, tel que RSVP. Le modèle des services différenciés (DiffServ) utilise le fait que les paquets peuvent être différemment marqués et alors classifiés ; les paquets dans les différentes classes de trafic reçoivent différents niveaux de service.

Par ailleurs, le modèle IntServ peut être appliqué dans les réseaux locaux en utilisant l'approche de gestion de la bande passante du sous-réseau (SBM), c.-à-d. si tout le trafic traverse au moins un commutateur ou pont en utilisant un protocole basé sur des priorités tel que l'IEEE 802.1p, 802.1Q ou 802.1D.

On peut remarquer que l'architecture de QoS de l'Internet, de haut en bas, et l'architecture de QoS du CEI 61158, de bas en haut, pourraient converger au niveau de la couche liaison de données si on utilise le commutateur basé sur des priorités de l'approche SBM et l'attribut de priorité de l'approche du CEI.

Cependant, dans les chapitres 3 et 4 nous avons vu que l'attribut de priorité au niveau de la couche liaison de données n'est pas défini par tous les Réseaux de terrain. En conséquence, on a besoin de nouvelles solutions de QoS et celles-ci doivent être des solutions dynamiques.

En fait, du point de vue de la réservation des ressources, WorldFIP est l'exemple d'un mécanisme centralisé et statique de réservation des ressources, c.-à-d. il y a un site, l'arbitre du bus, ayant différentes listes d'identificateurs qui définissent à tout moment les besoins de communication de tous les sites. Ces besoins sont connus à l'avance et ne changent pas pendant l'exécution de l'application. Dans le cas des données apériodiques, l'utilisation de la réponse à une demande périodique peut être vue comme une sorte de réservation dynamique

des ressources puisque la demande apériodique de transmission des variables et des messages est manipulée dynamiquement dans leurs futures fenêtres respectives.

La réponse à une requête est également employée pour demander de futures transmissions dans Profibus, c'est le cas des opérations acycliques et cycliques du type SDR (Send and Request Data with Reply).

Les fenêtres de ControlNet peuvent être vues comme des niveaux de QoS, étant donné que le NUI est répété de façon cyclique ; la fenêtre ordonnancée fournit un service cyclique et acyclique garanti, la fenêtre non-ordonnancée fournit un service meilleur effort probablement avec un temps supérieur borné si un seul message par site est transmis dans cette fenêtre. Cependant, on peut remarquer que ControlNet est un mécanisme distribué et statique de réservation des ressources, c.-à-d., tous les sites doivent être connus à l'avance, à l'instant de configuration du réseau afin d'avoir une tranche de temps dans la fenêtre ordonnancée et, d'autre part, chaque site doit connaître les messages temps-critiques à transmettre.

Le chapitre 5 présente un protocole dynamique de réservation des ressources qui peut être employé par les réseaux de terrain avec une configuration centralisée.

Le chapitre 5 présente une architecture de QoS pour les modèles de communication client-serveur, multiclient-serveur et client-multiserveur. Cette architecture est basée sur l'approche de réservation des ressources utilisée dans le cadre de QoS de l'OSI et le modèle des services intégrés de l'Internet.

Ce chapitre également présente un Protocole de RéSerVation des Ressources pour le Client-Server (CS-RSVP), lequel est spécifié en Langage de Spécification et Description (SDL) et validé à l'aide de l'outil ObjectGEODE v.4.0 de Verilog. Toutes les spécifications sont présentées dans les annexes A, B et C.

Le CS-RSVP permet de réserver l'utilisation des ressources par un processus de négociation entre le(s) client(s) et le(s) serveur(s). Le premier envoie un message de réservation (*RSVcreate*) contenant les paramètres de QoS suivants : le nom de la ressource, une variable booléenne pour indiquer si l'utilisation de la ressource est périodique ou pas, si c'est le cas la valeur de la période, l'heure de départ, l'intervalle de temps d'utilisation de la ressource, l'intervalle de temps de validité de la réservation et une variable booléenne pour indiquer si la demande de réservation peut être négociée ou pas.

Dans le serveur, après des tests d'admission, le distributeur des ressources (RA) répond par un message d'acceptation ou par un message de rejet. Le premier contient l'identificateur de la demande que le client emploiera pour de futures demandes de service. Le message de rejet contient les mêmes paramètres de QoS du message de réservation, mais leurs valeurs sont celles que le processus RA peut satisfaire, si la demande de réservation est négociable.

Pendant que le client décide d'accepter ou de rejeter ces paramètres, le serveur bloque les valeurs en attendant un message de réservation ou l'expiration d'un temporisateur.

On peut remarquer que quelques-uns des processus doivent être encore spécifiés, par exemple, l'ordonnanceur, l'entité du serveur (ServerEntity) et le contrôle d'admission. Jusqu'à

présent, la spécification du CS-RSVP tient compte seulement des demandes de service dans l'ordre où elles sont faites, c.-à-d. premier-arrivé premier-servi, et les demandes de réservation des ressources sont aléatoirement acceptées.

D'autre part, le chapitre 4 présente le modèle de services différenciés de l'Internet, cette approche est essentiellement un schéma basé sur des priorités. Ainsi, le modèle de QoS pour le client-serveur qui est présenté dans le chapitre 5 peut être étendu afin de prendre en compte le schéma basé sur des priorités de la manière suivante : les clients peuvent marquer leurs demandes de service selon une classe de service désirée. En marquant les demandes différemment et en manipulant les demandes à partir de ces marques, diverses classes différenciées de service peuvent être créées.

Afin de recevoir un service différencié de la part du serveur, chaque client doit établir un accord sur le niveau de service (SLA) avec le serveur. SLA indique fondamentalement les classes de service soutenues et la quantité de demandes permises dans chaque classe. Les paragraphes suivants analysent le cas statique. Noter que le SLA statique se comporte comme un distributeur de ressources sur le côté client selon l'architecture de QoS de la Fig. 5.3.

Quand un client souhaite envoyer une demande de service au serveur, la demande est passée vers le RMC (module demandeur du client) qui est responsable de marquer la demande selon le SLA, et qui est responsable de contrôler le nombre de demandes dans chaque classe, une demande excédant un seuil doit être annulée.

Quand le serveur reçoit la demande, elle est dirigée vers le distributeur des ressources (RA), qui classifie et met la demande dans une file d'attente selon les classes désirées de service. Ces files d'attente peuvent avoir besoin d'un schéma de gestion, lequel peut être exécuté par le CA. Dans tous les cas, une demande suit le processus décrit dans l'architecture de QoS du client-serveur de la sous-section 5.3.2. Il faut noter que le processus de contrôle d'admission (CA) n'est plus appelé par le RA, à moins qu'il existe un processus de négociation.

En ce qui concerne les réseaux de terrain, CS-RSVP peut être intégré dans l'approche centralisée de la manière suivante :

- Pendant la configuration du réseau, l'utilisateur stocke généralement la configuration du réseau dans un site, l'arbitre du bus ou le maître, sous forme d'une liste ou d'une table ou plus généralement d'une certaine structure de données. L'information stockée représente tous les besoins de communication des utilisateurs tels que les actionneurs, les capteurs, etc. Ces besoins ont été en général traduits sous forme d'une structure d'ordonnancement garantie par des tests d'ordonnançabilité.

- Ainsi, le CS-RSVP peut être employé comme protocole de configuration avec un serveur qui l'exécute sur le site arbitre du bus et un client qui s'exécute sur les sites esclaves ou les sites producteurs/consommateurs.

- Pendant l'exécution de l'application, le CS-RSVP peut être employé pour fournir différents niveaux de QoS.

- Son intégration dans les architectures centralisées telles que WorldFIP, Foundation Fieldbus et Profibus (entre autres) fait partie de notre travail de recherche futur.

Finalement, pour conclure ce travail, nous pouvons énoncer quelques remarques et perspectives :

- Les réseaux de transmission de données sont nés il y a maintenant plus de 40 ans avec le développement des ordinateurs et de l'informatique,

- La commutation de paquets née au début des années 70 a vu s'affronter les concepts de datagramme et de circuit virtuel, la notion de qualité de service était déjà à la base de ce débat, mais en termes de fiabilité et de sécurité de transmission,

- Les réseaux industriels ont fait apparaître le besoin de satisfaire des contraintes de temps, nouveaux paramètres ou nouvelles caractéristiques de la qualité de service,

- Les réseaux de terrain nés au début des années 80 ont mis en exergue ces contraintes du point de vue à la fois de la sécurité et de la garantie de délivrance des messages, mais aussi du point de vue temporel,

- Les années 90 avec le développement d'Internet vers le multimédia ont conduit à intégrer des transferts contraints par le temps sur de vastes réseaux, la notion de qualité de service voisine de celle des réseaux de terrain a été alors introduite dans les grands réseaux,

- Les réseaux de terrain vu leur domaine restreint ont pu se permettre d'introduire des solutions ad hoc,

- Internet a cherché à développer des solutions dynamiques pour satisfaire les besoins de nouveaux trafics, des solutions statiques n'étaient pas envisageables compte tenu de la diversité des utilisateurs et de leur dynamique propre,

- Une certaine convergence de besoins et de solutions apparaît entre les réseaux temps réel et l'internet ; les techniques d'ordonnancement, de réservation de ressources, de contrôle d'admission, connues depuis longtemps dans les systèmes d'exploitation, puis dans les réseaux locaux et les réseaux de terrain, font leur apparition dans les grands réseaux,

- Le principal problème à résoudre est maintenant celui de la connaissance impossible d'un état global instantané, et celui de la prise de décision répartie.

Sur ce dernier point, on pourrait étudier si certaines variables d'état du système de communication ne pourraient pas être considérées comme des variables globales au moins à une échéance donnée, pour des algorithmes d'admission et d'ordonnancement distribués.

Enfin on peut penser à des profils de communication relativement standards qui, indépendamment des domaines d'application, intégreraient les mécanismes de gestion de la qualité de service, des algorithmes d'ordonnancement afin de respecter au mieux les contraintes de temps imposées par les applications.

Appendix A

SDL Specification of the Client-Server Resource ReSerVation Protocol

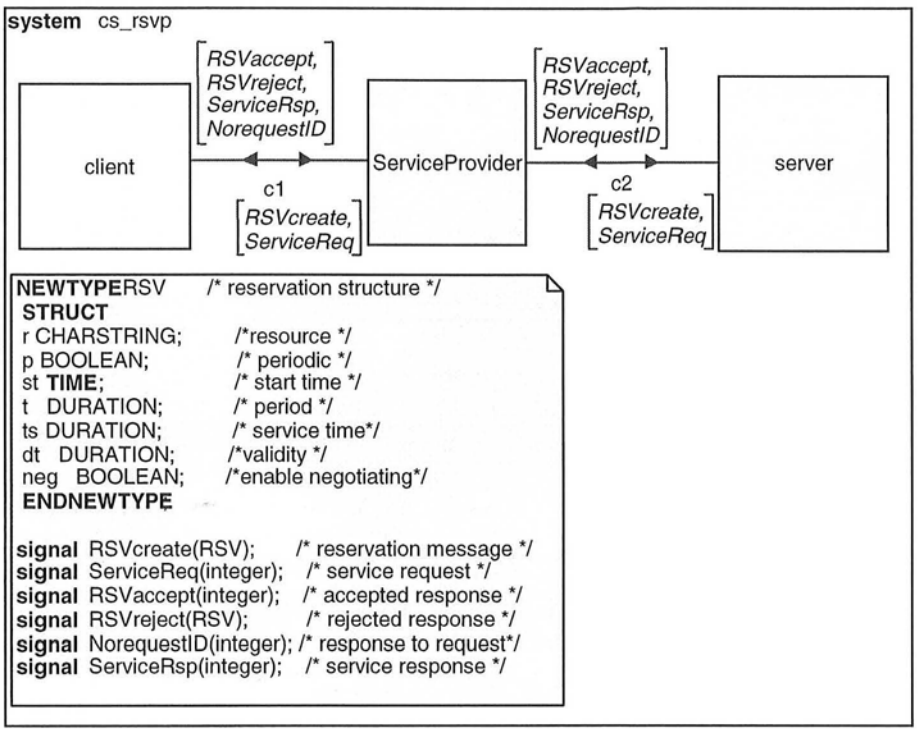

Fig. A.1. CS-RSVP system.

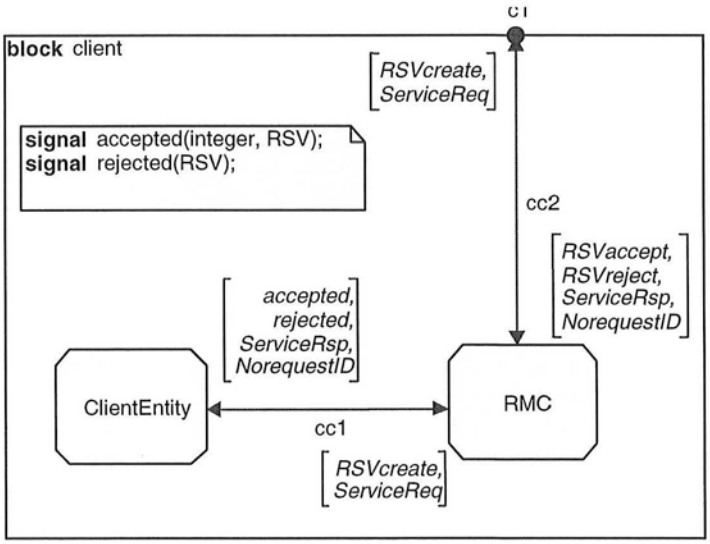

Fig. A.2. Client Block

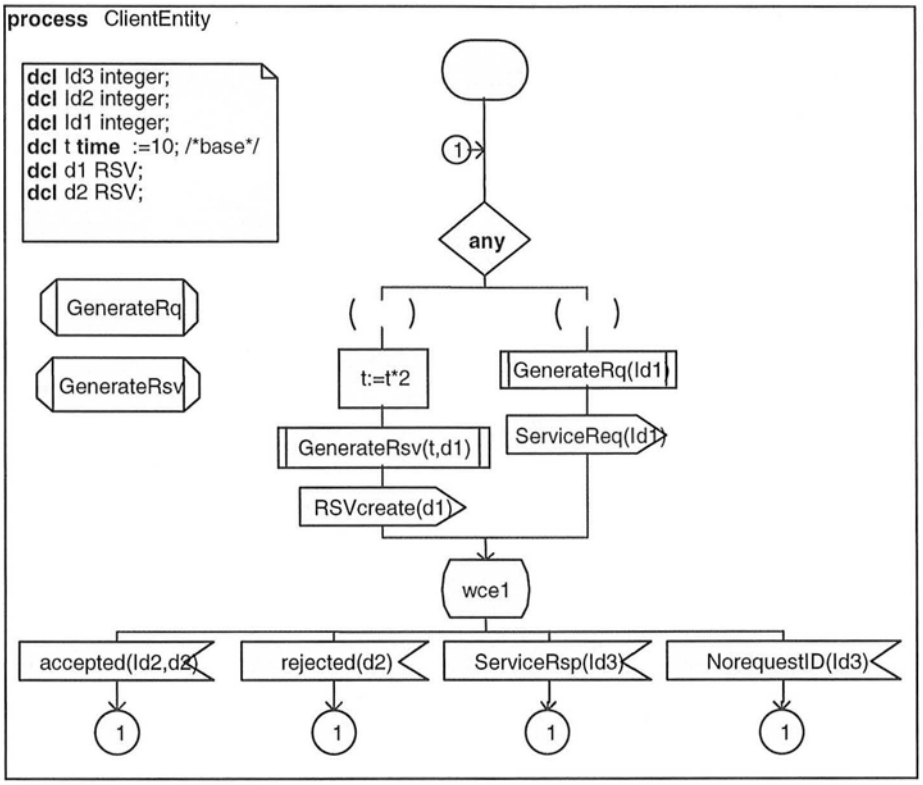

Fig. A.3. ClientEntity process diagram

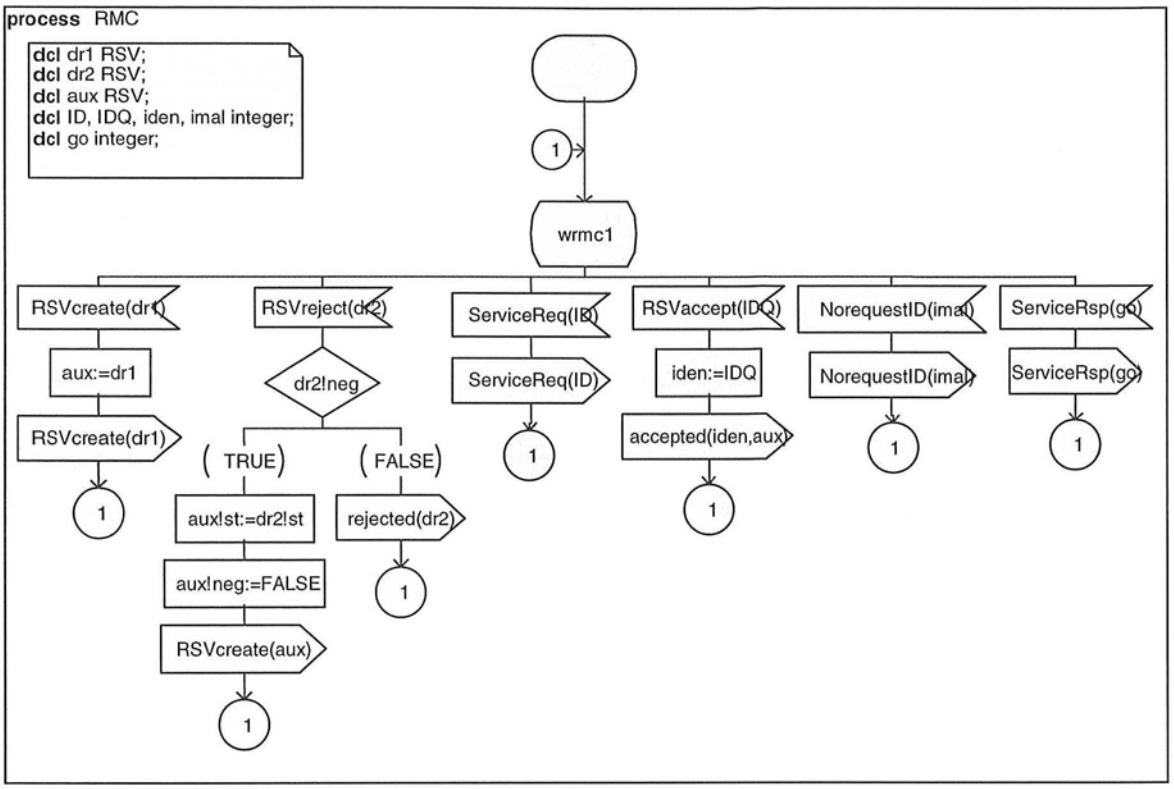

Fig. A.4. RMC process diagram

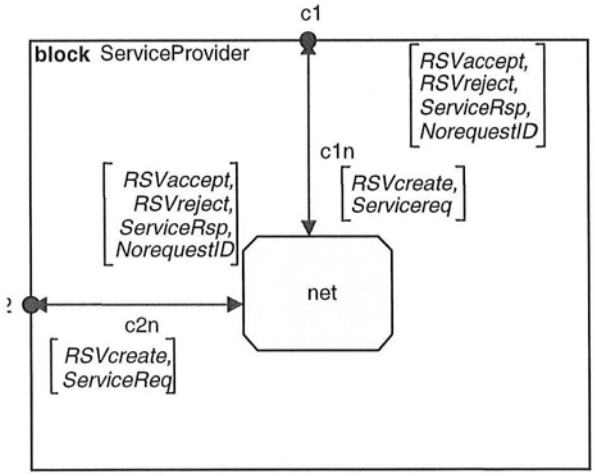

Fig. A.5. ServiceProvider Block

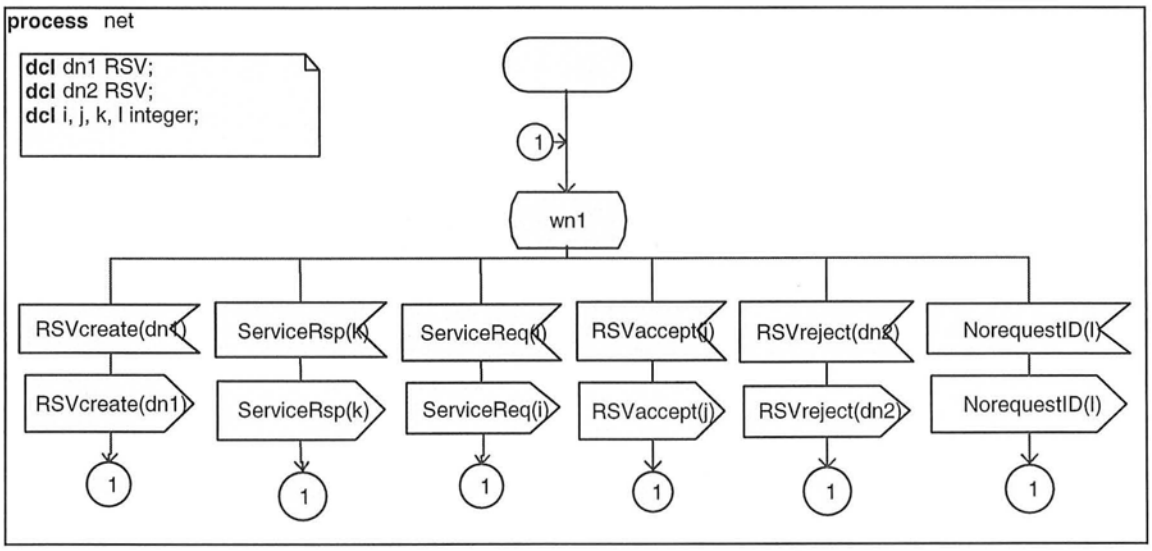

Fig. A.6. net process diagram

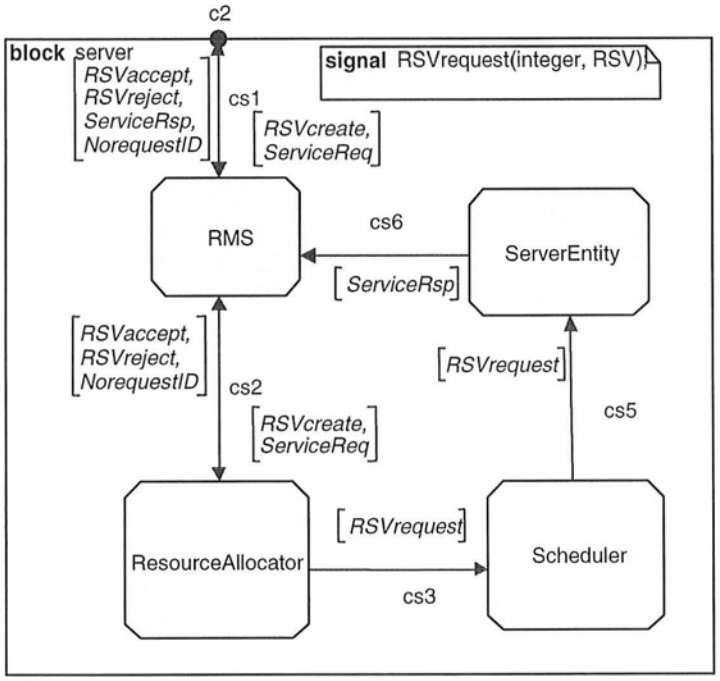

Fig. A.7. Server Block

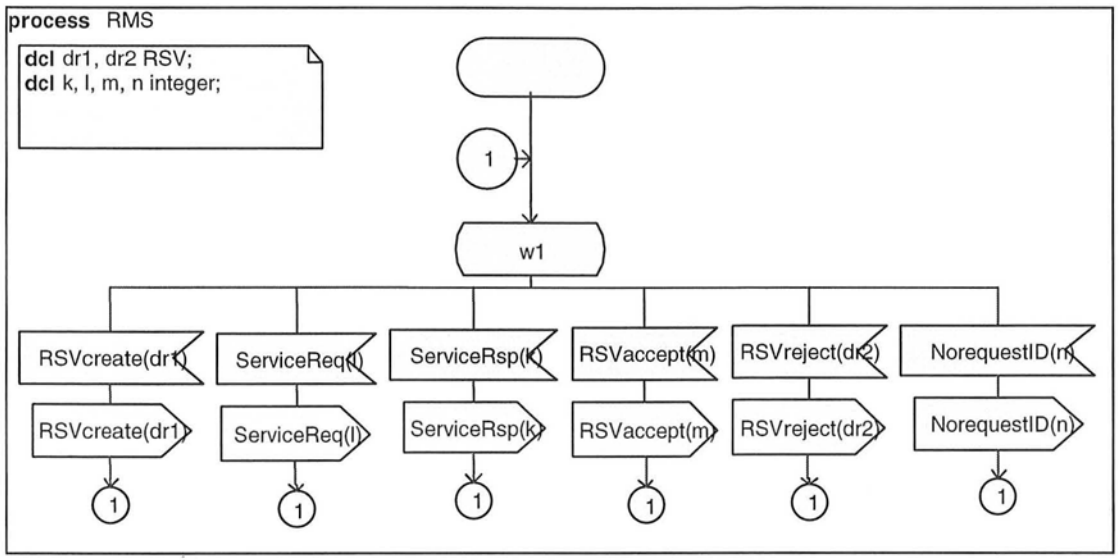

Fig. A.8. RMS process diagram

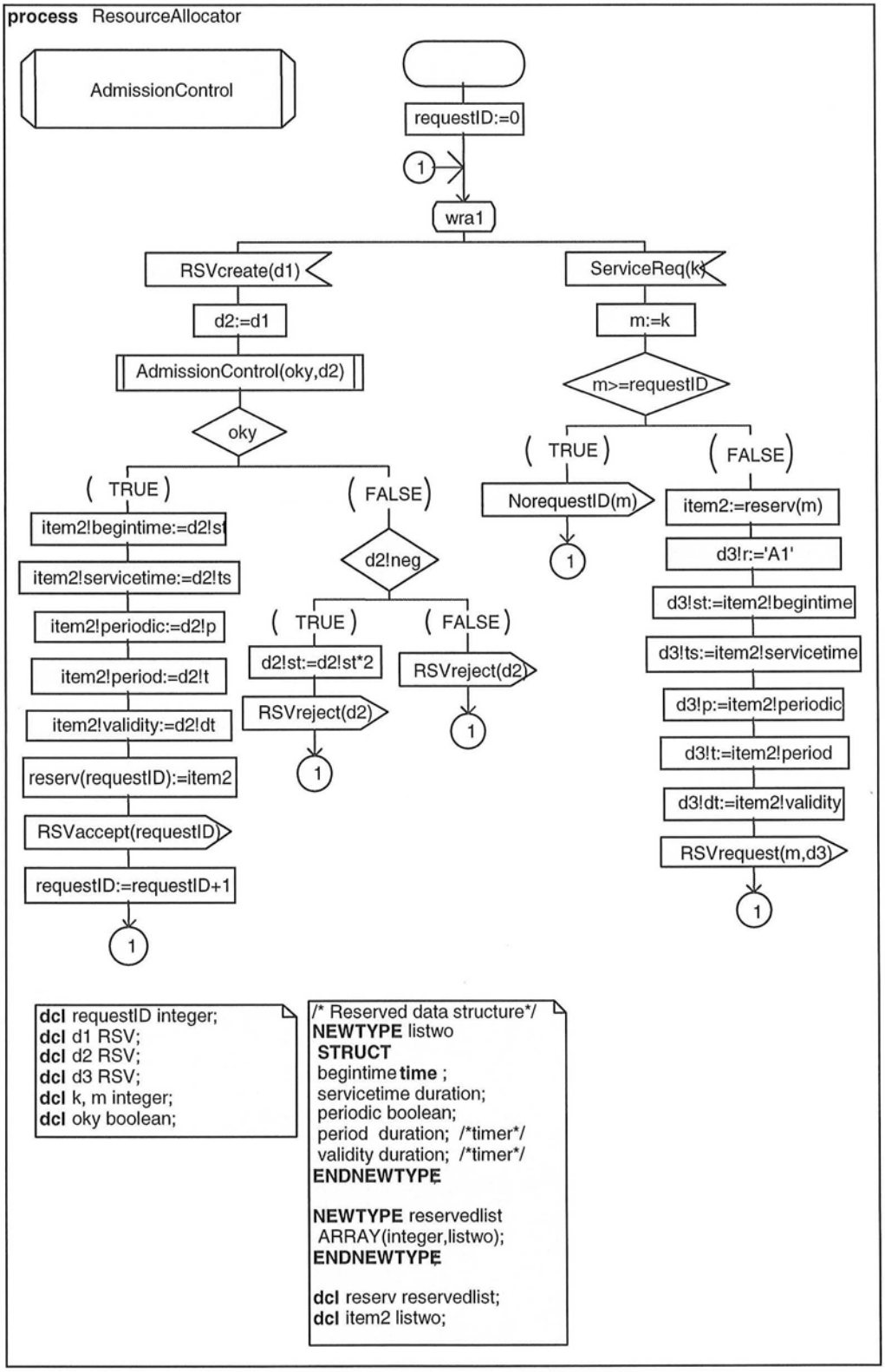

Fig. A.9. ResourceAllocator process diagram

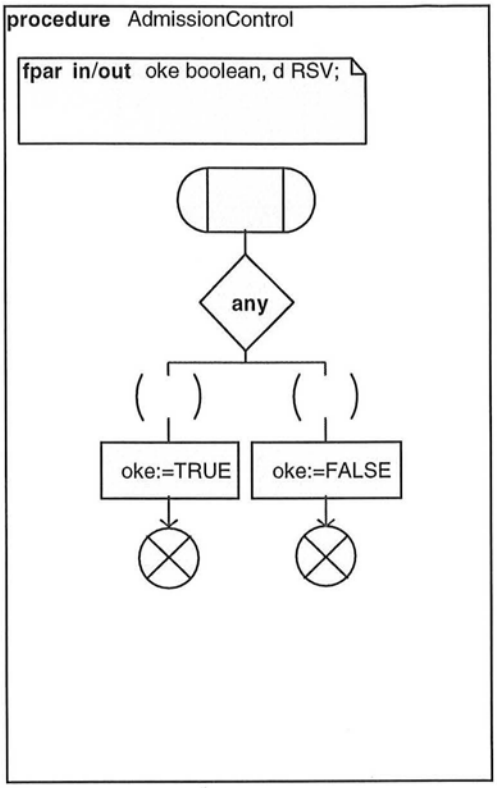

Fig. A.10. AdmissionControl process diagram

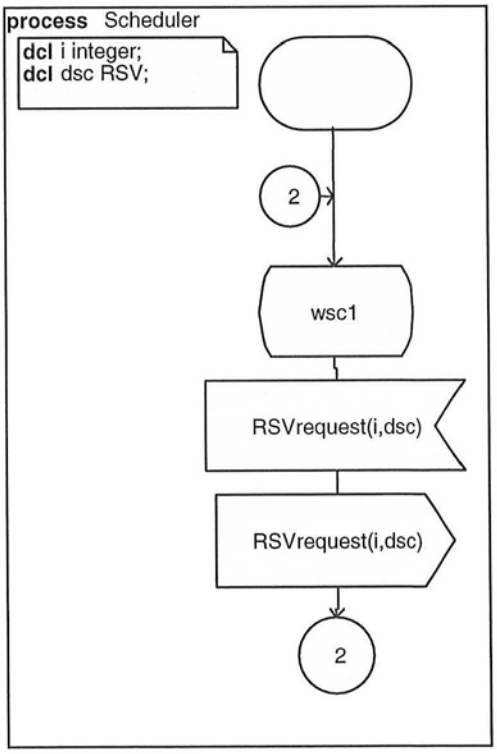

Fig. A.11. Scheduler process diagram

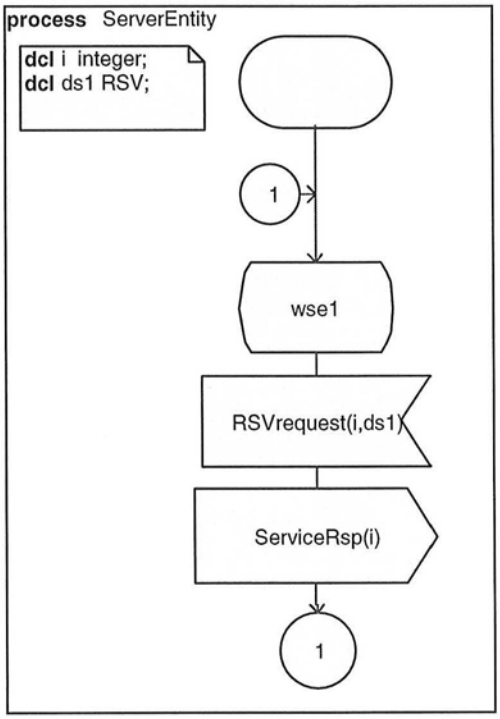

Fig. A.12. ServerEntity process diagram

Appendix B

SDL Specification of the MultiClient-Server Resource ReSerVation Protocol

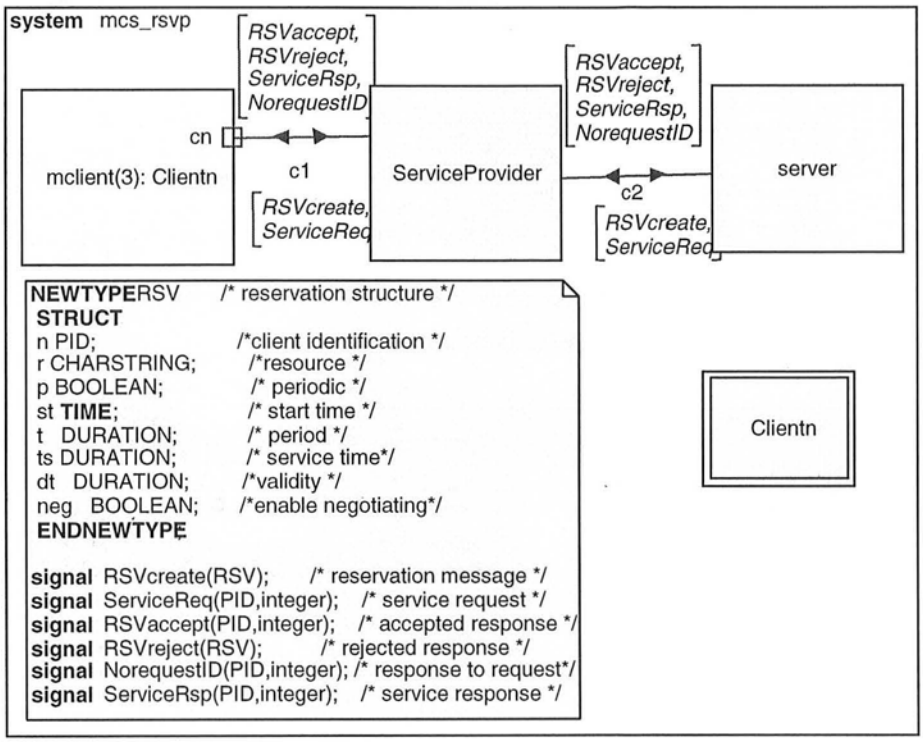

Fig. B.1. MCS-RSVP system.

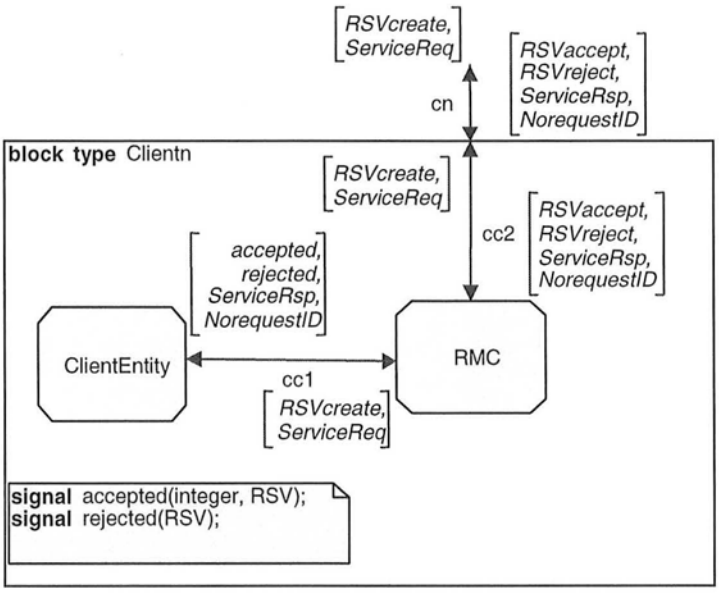

Fig. B.2. Block type Clientn

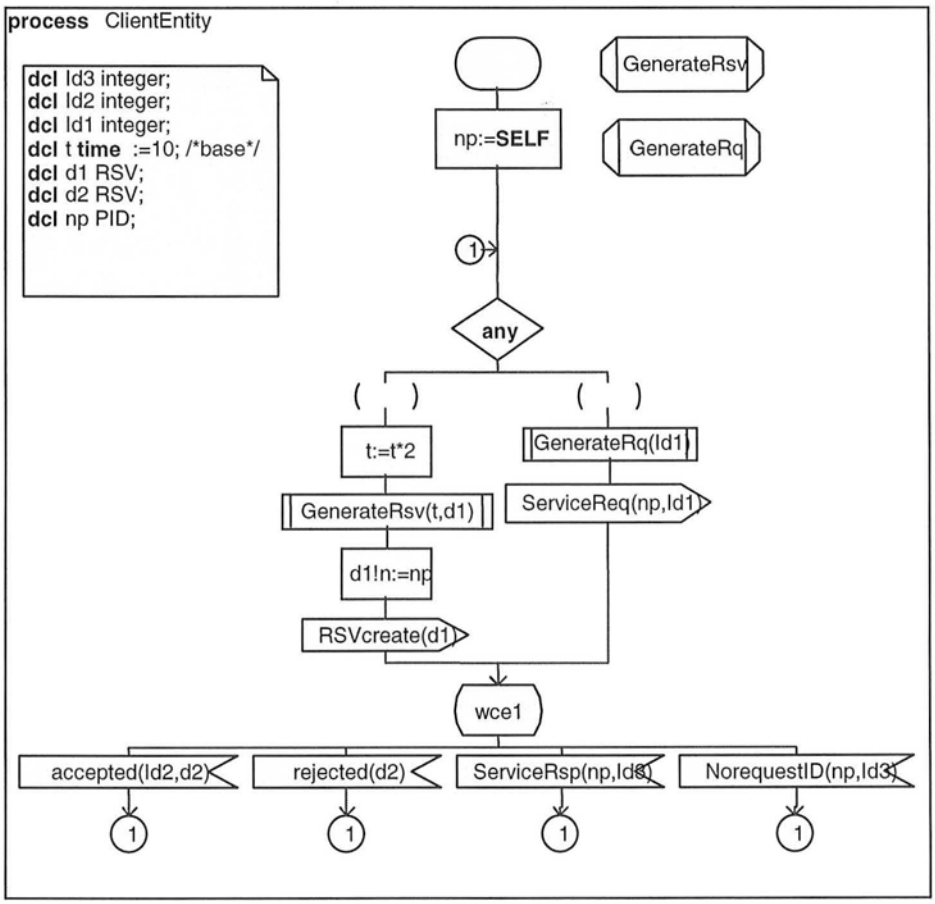

Fig. B.3. ClientEntity process diagram

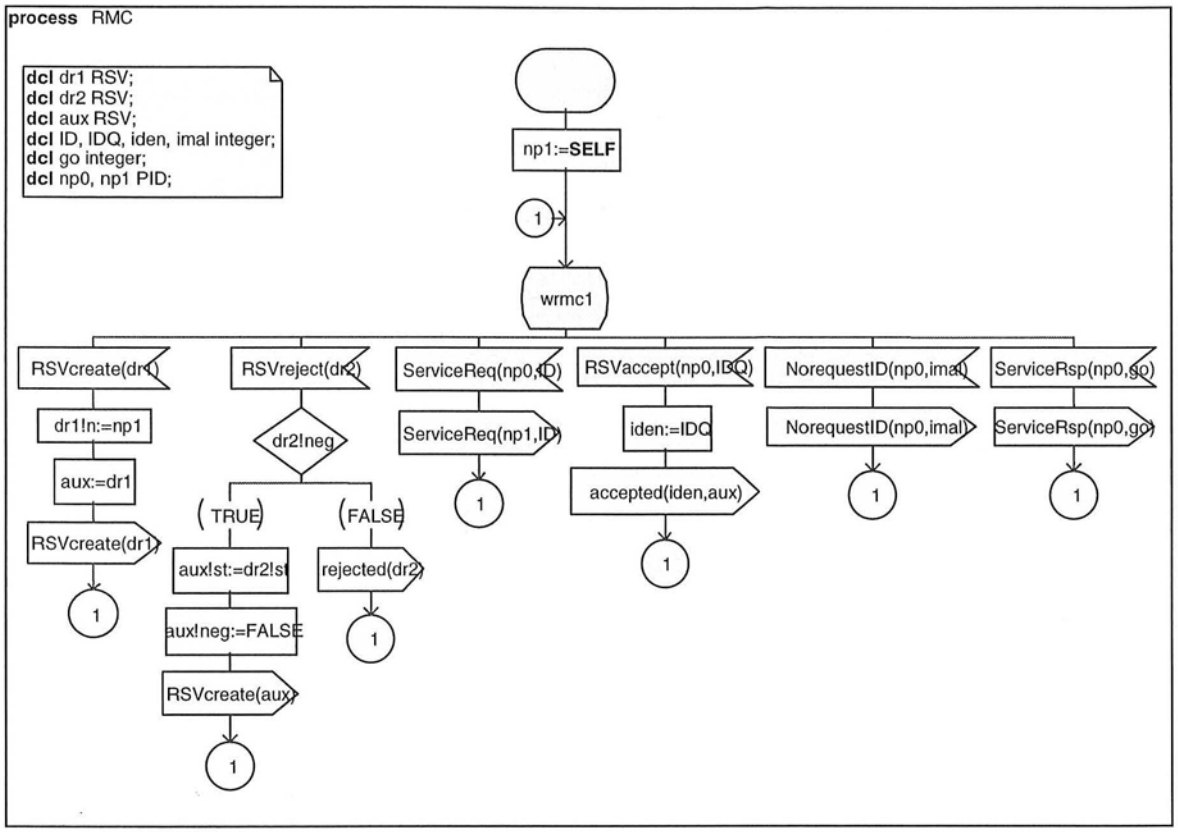

Fig. B.4. Requester Module of the Client process diagram

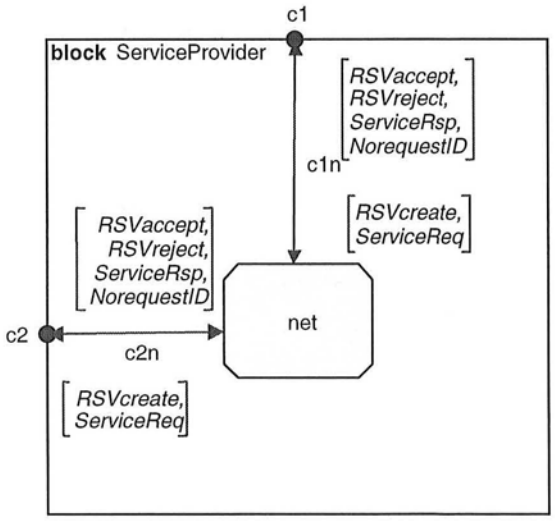

Fig. B.5. ServiceProvider block

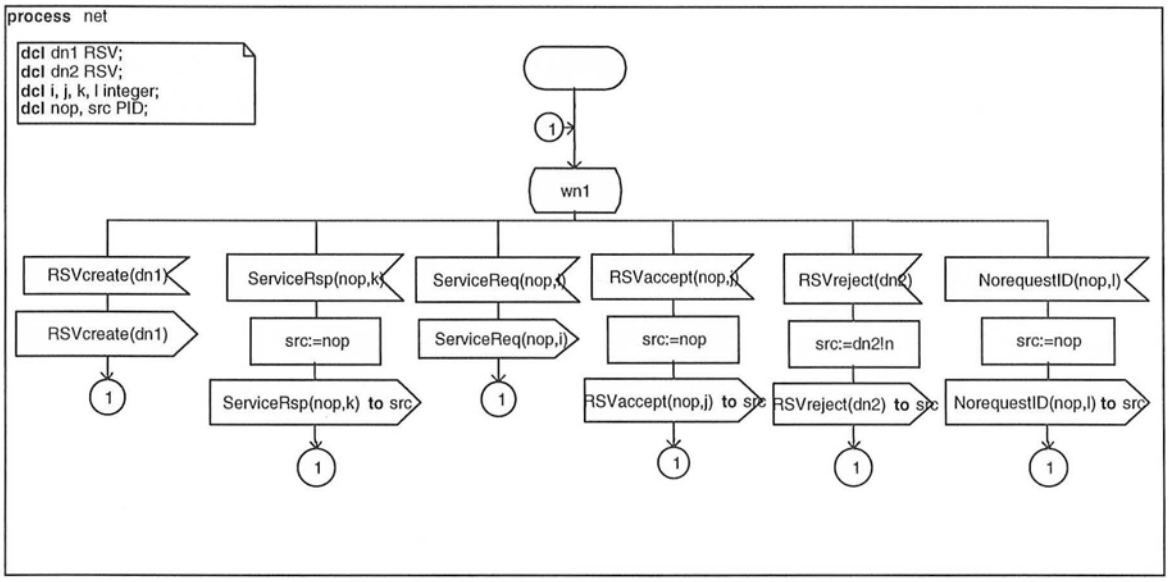

Fig. B.6. Net process diagram

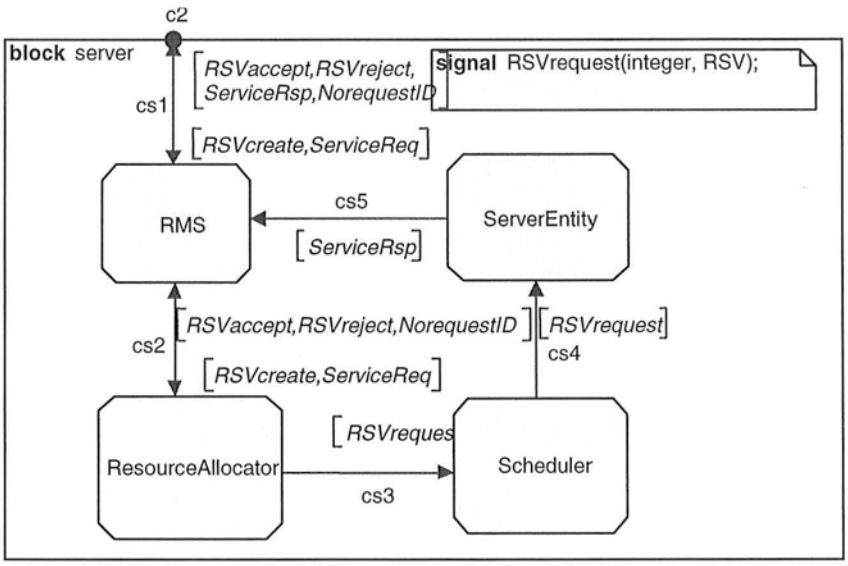

Fig. B.7. Server block

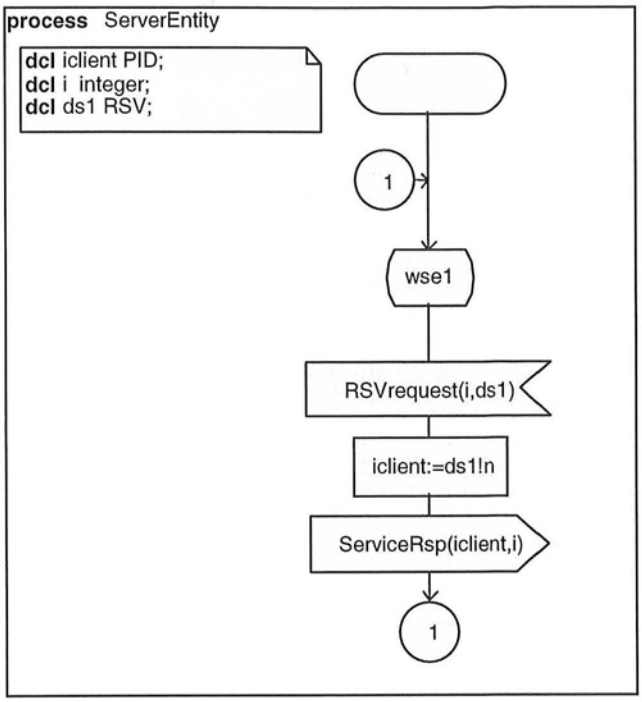

Fig. B.8. ServerEntity process diagram

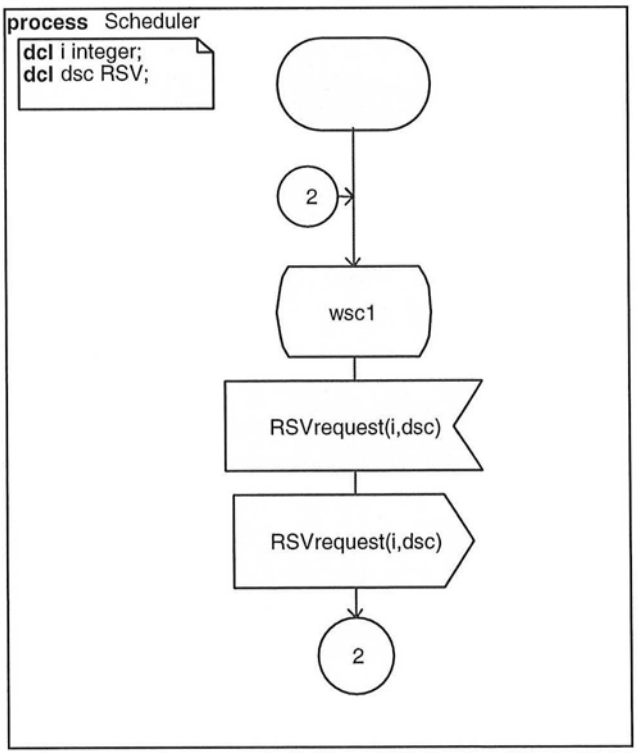

Fig. B.9. Scheduler process diagram

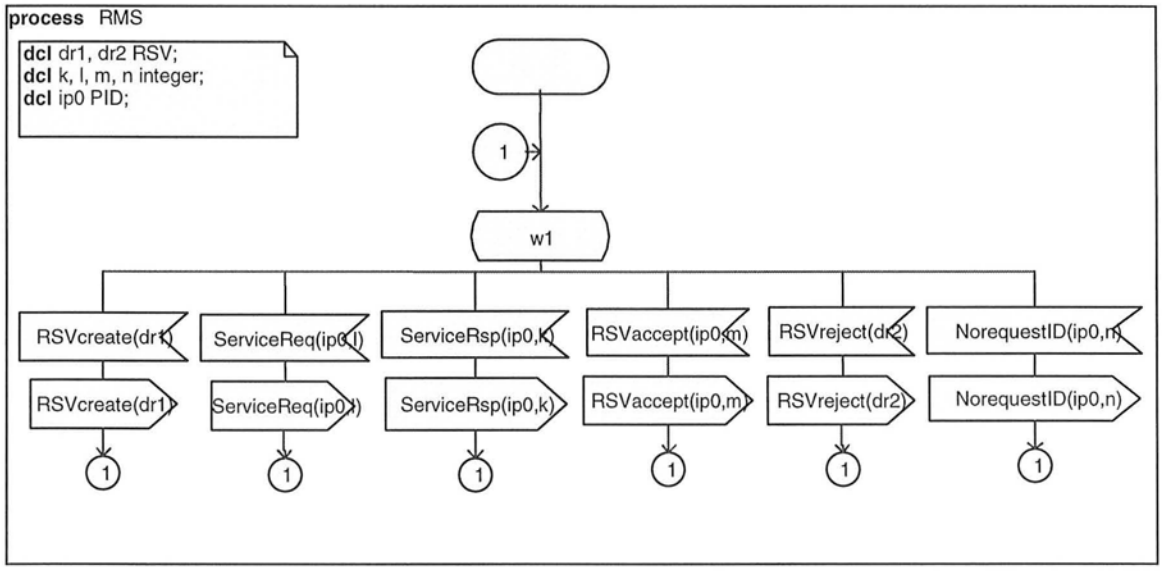

Fig. B.10. Requester Module of the Server process diagram

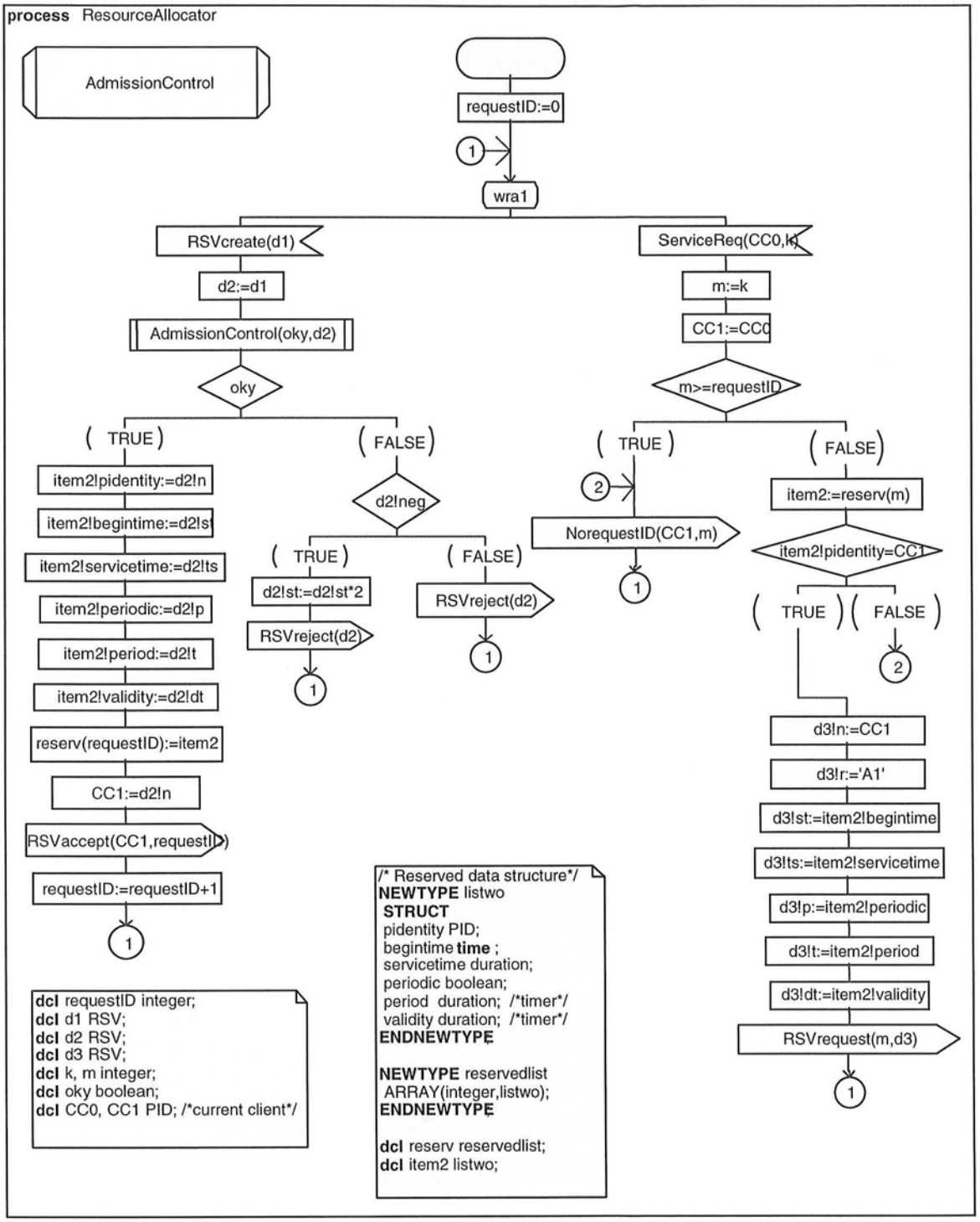

Fig. B.11. Resource Allocator process diagram

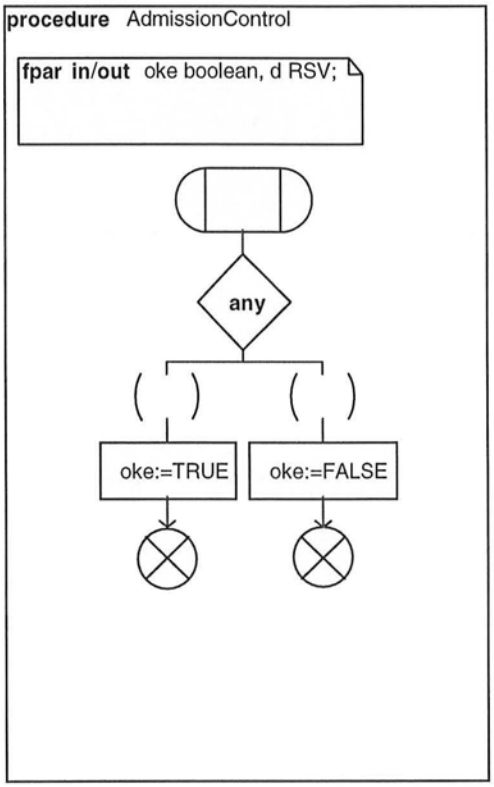

Fig. B.12. AdmissionControl process diagram

Appendix C

SDL Specification of the Client-MultiServer Resource ReSerVation Protocol

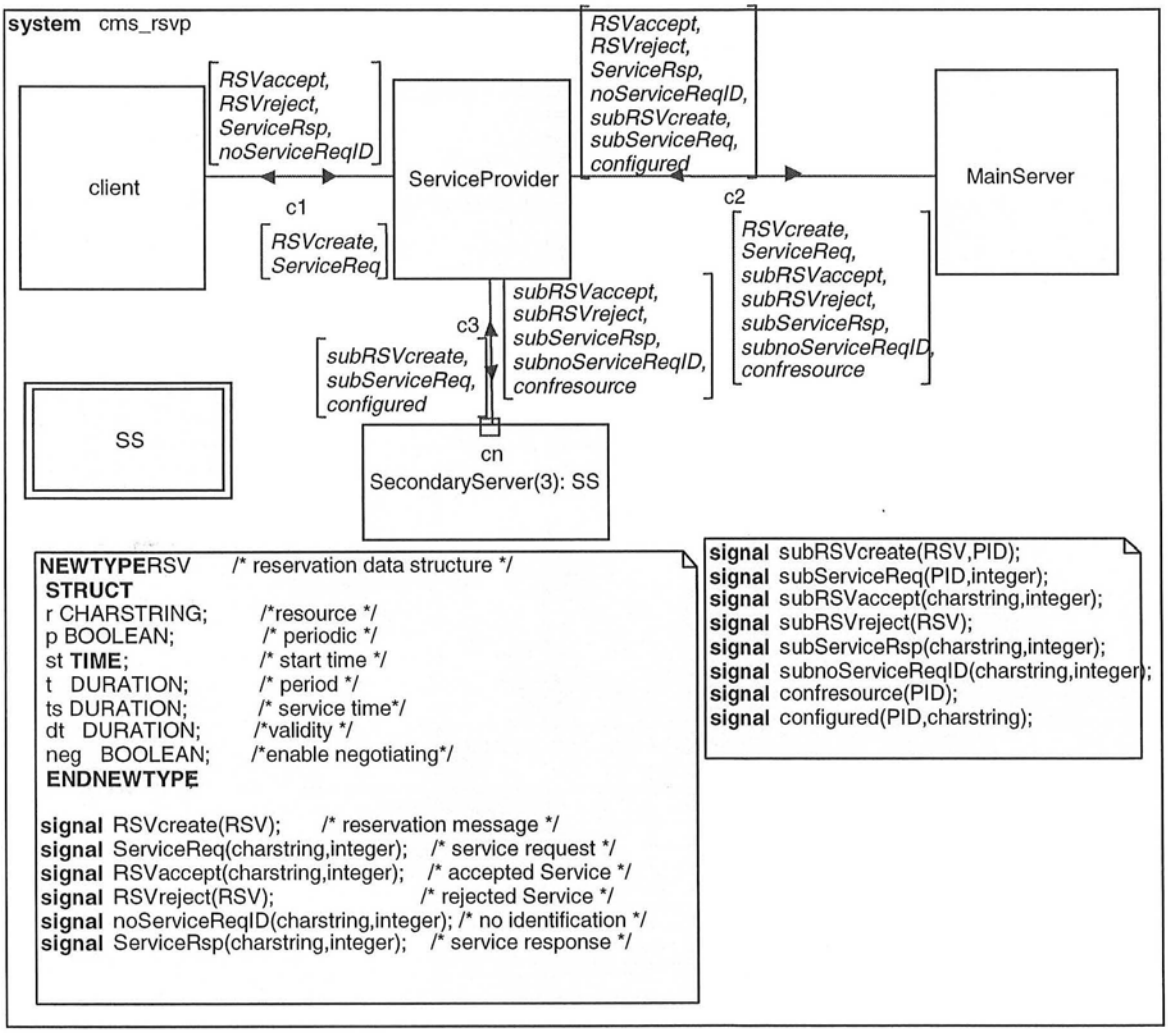

Fig. C.1. CMS-RSVP system.

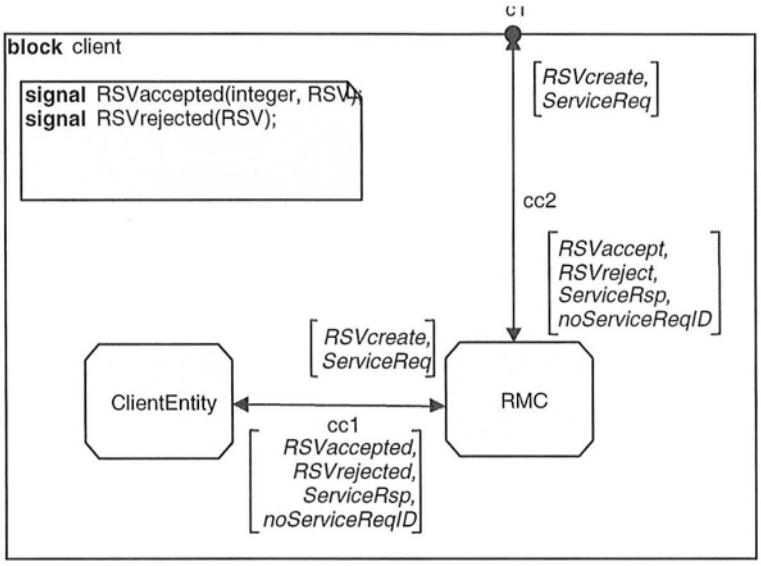

Fig. C.2. Client Block

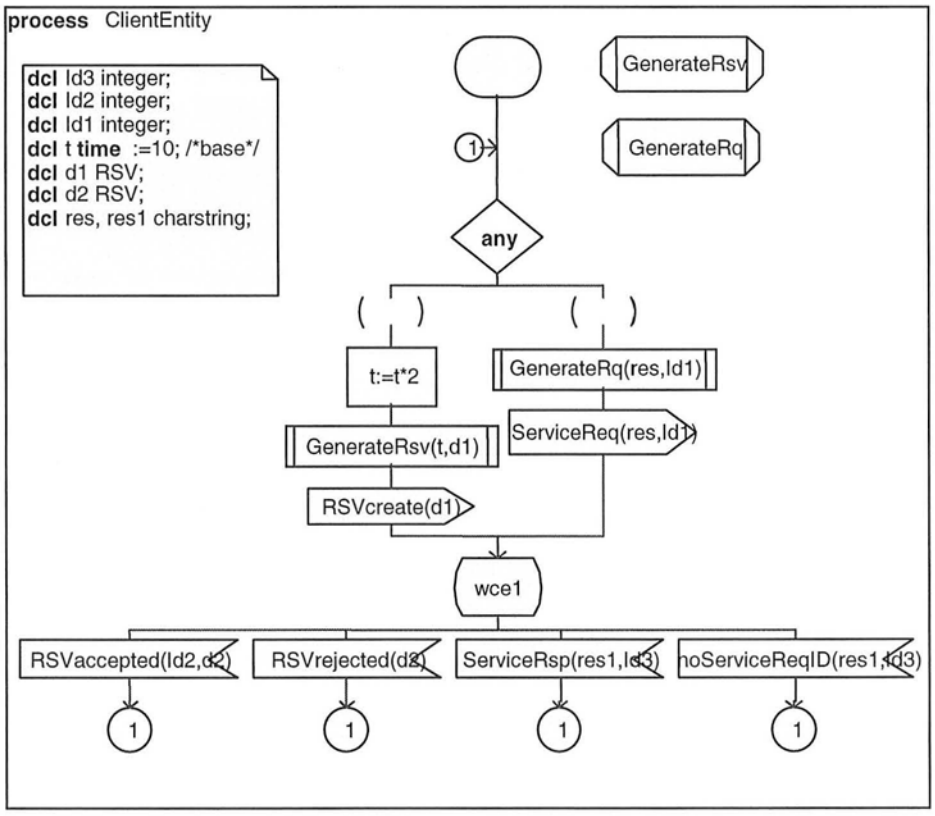

Fig. C.3. ClientEntity process diagram

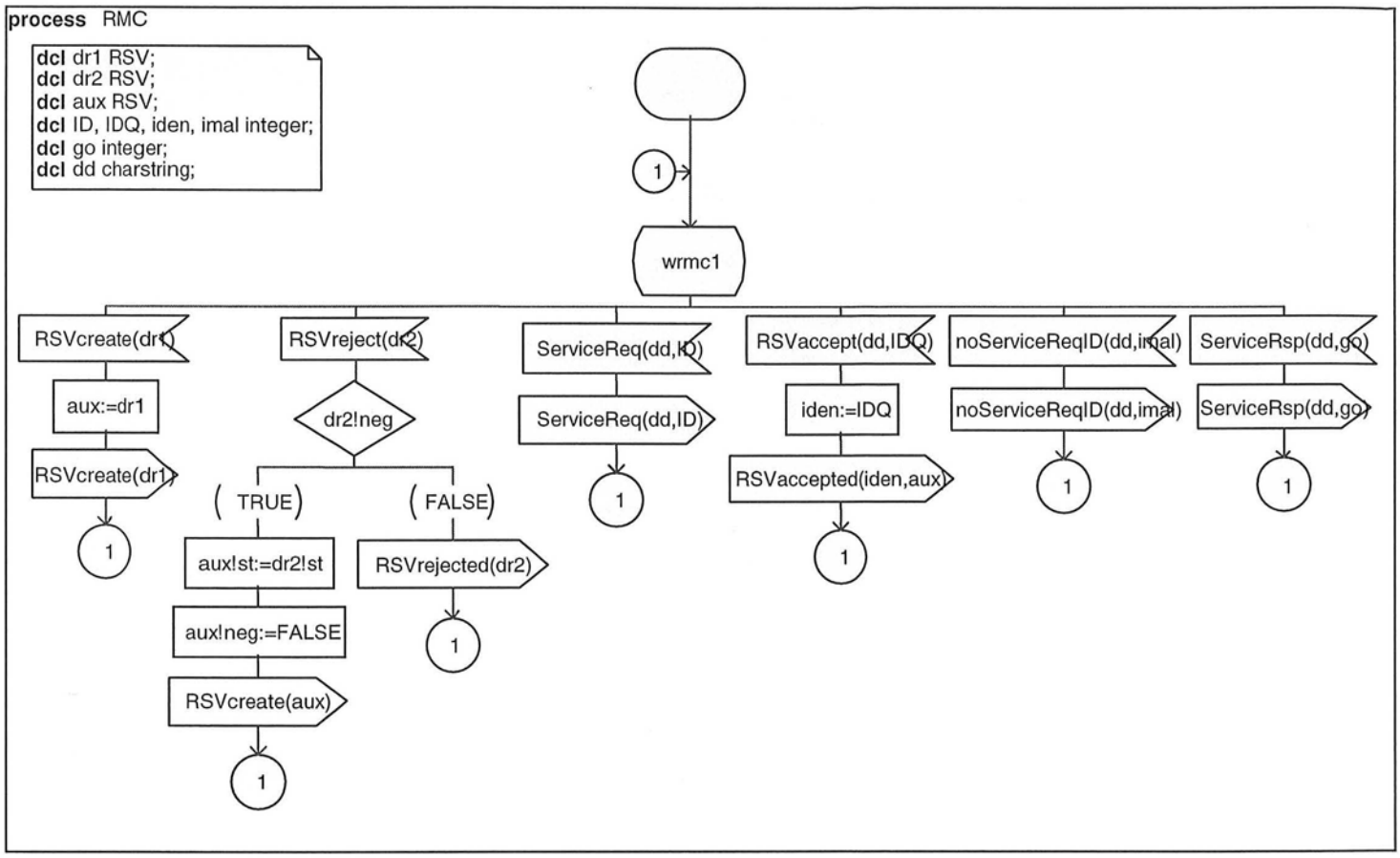

Fig. C.4. RMC process diagram

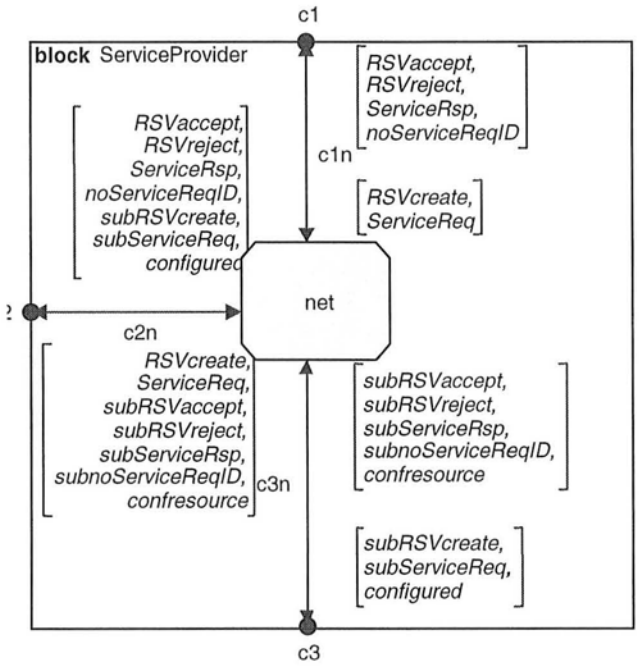

Fig. C.5. ServiceProvider Block

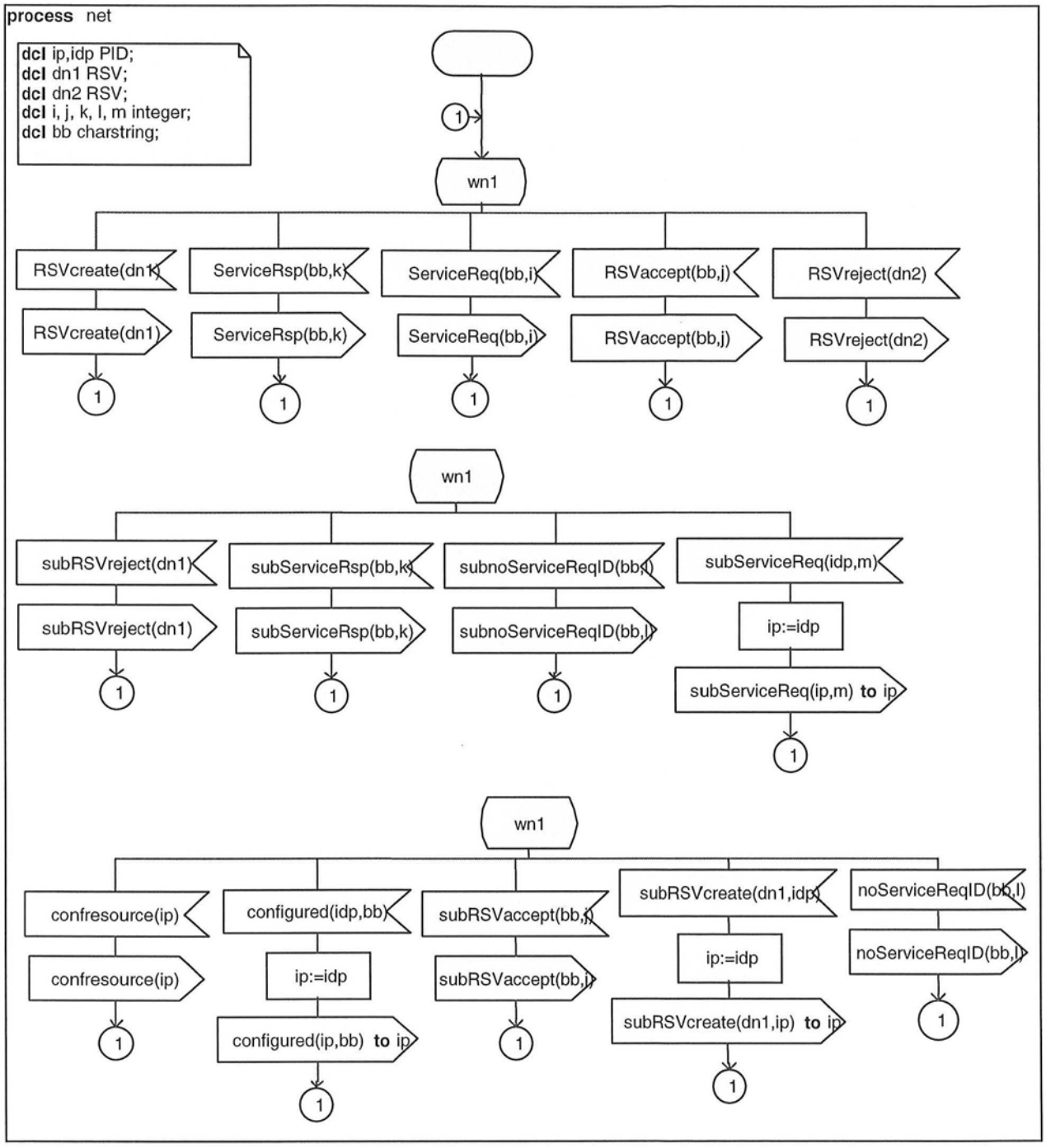

Fig. C.6. net process diagram

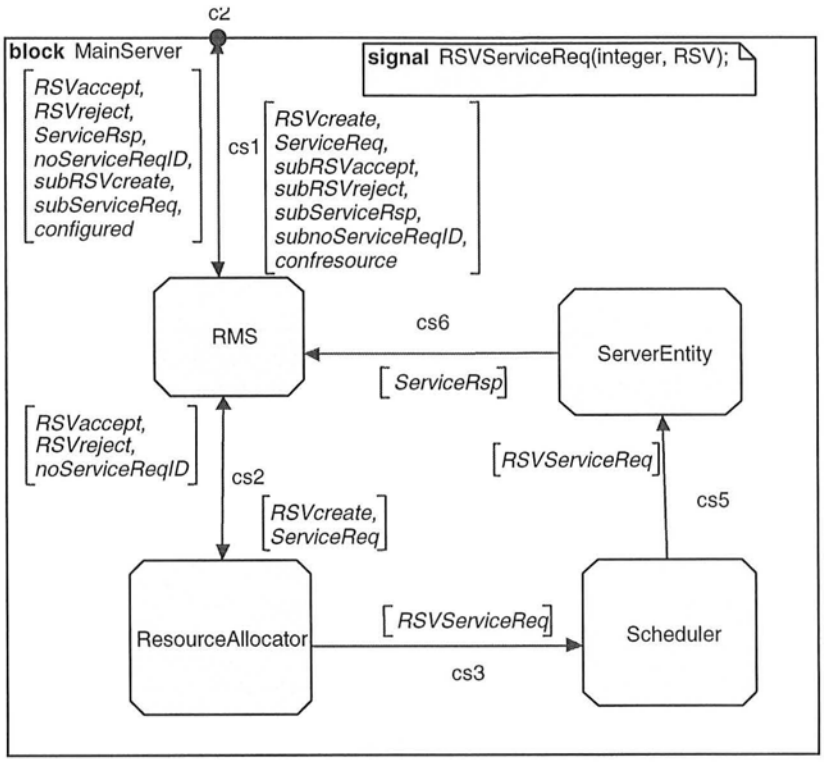

Fig. C.7. MainServer Block

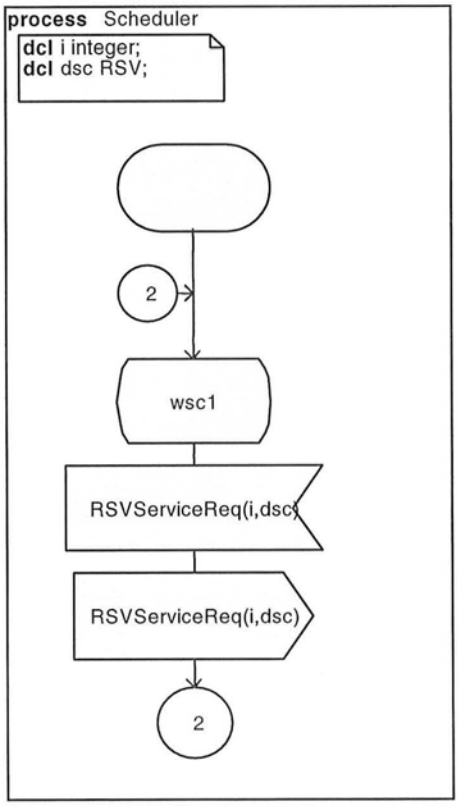

Fig. C.8. Scheduler process diagram

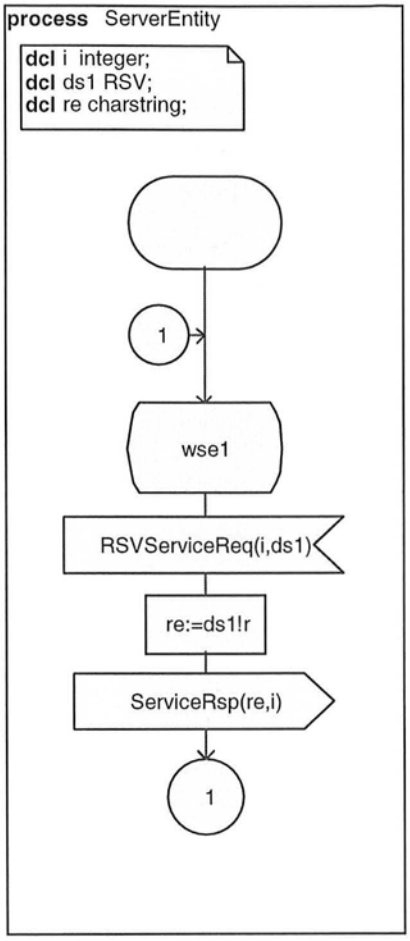

Fig. C.9. ServerEntity process diagram

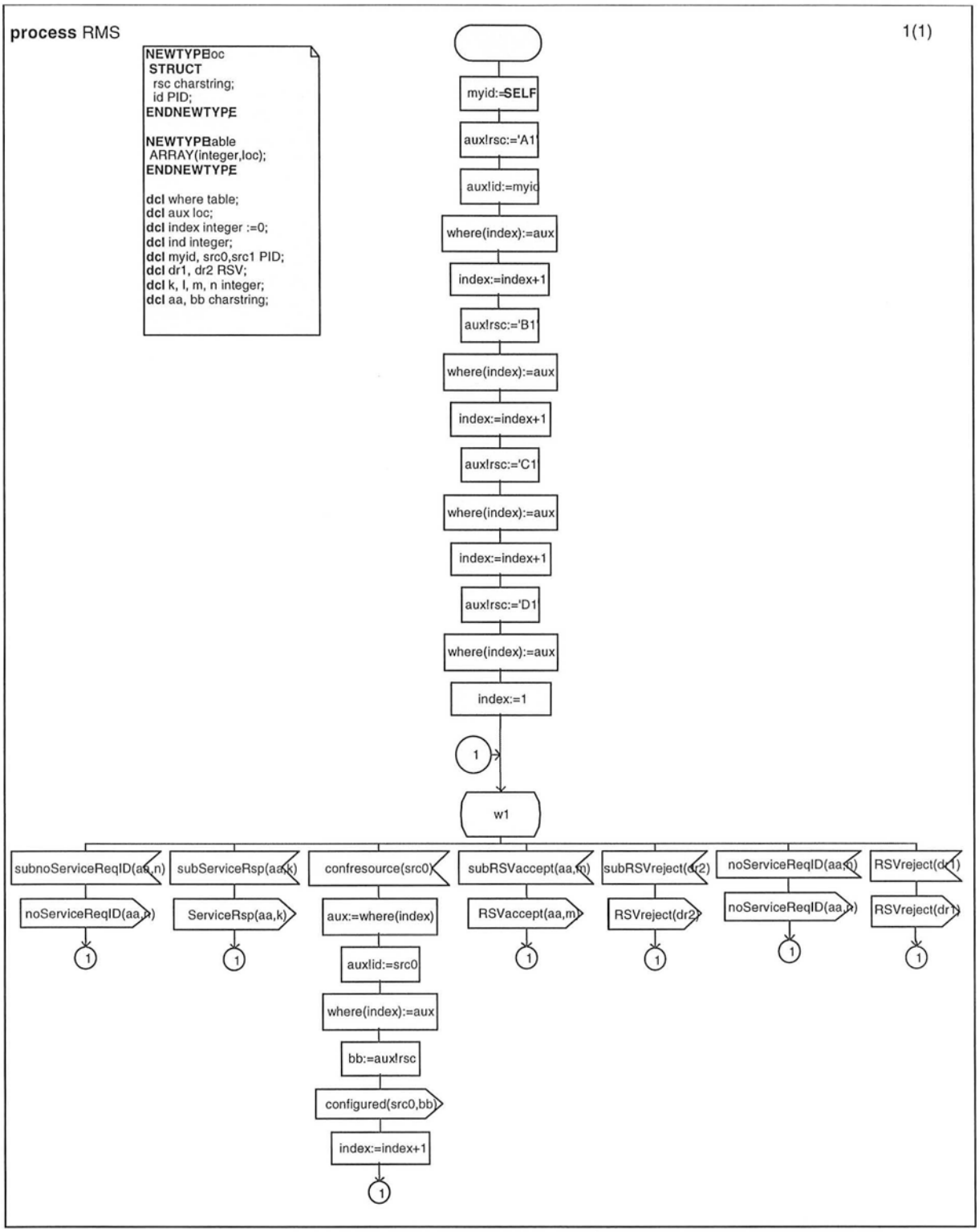

Fig. C.10. RMS process diagram

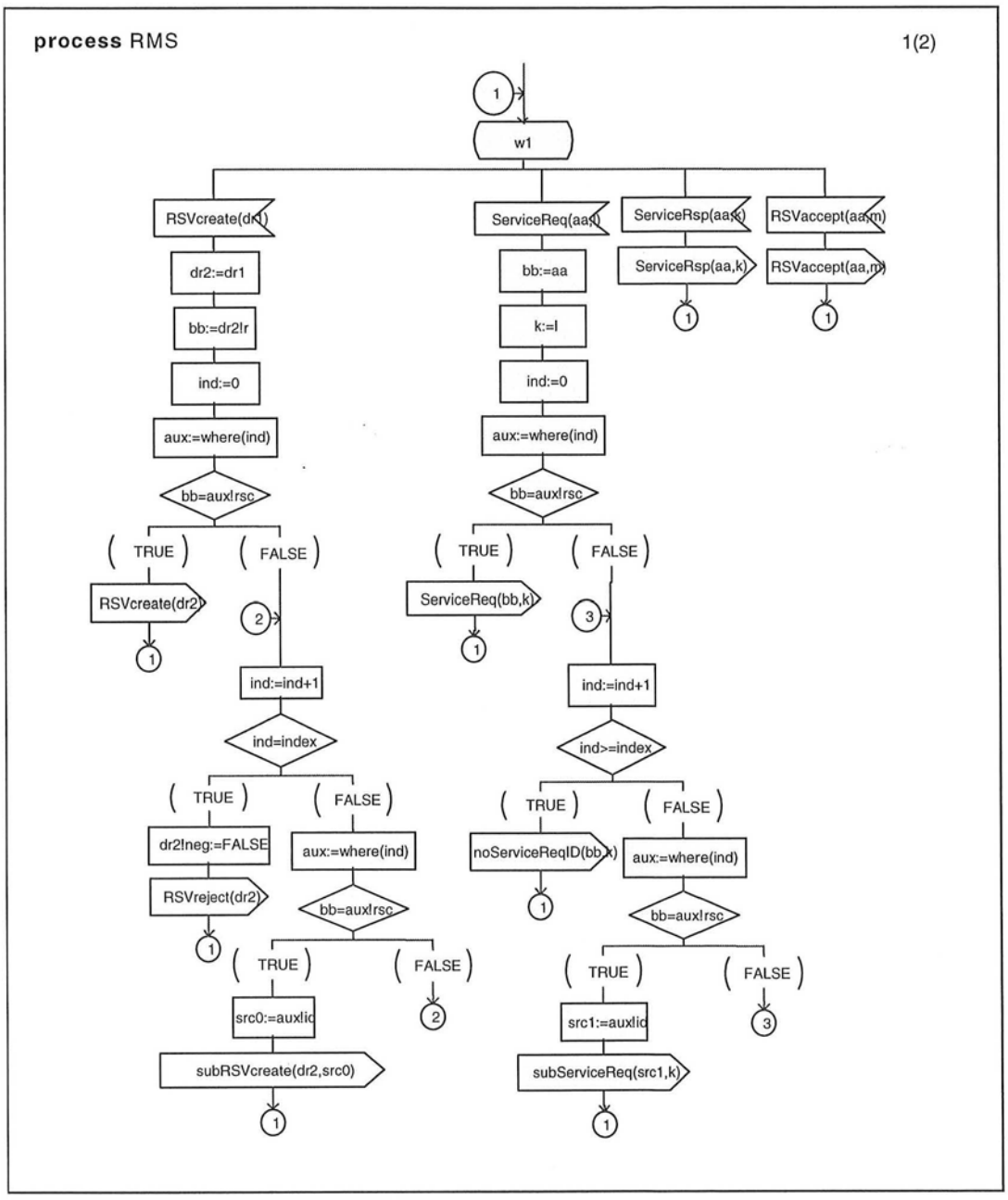

Fig. C.11. RMS process diagram

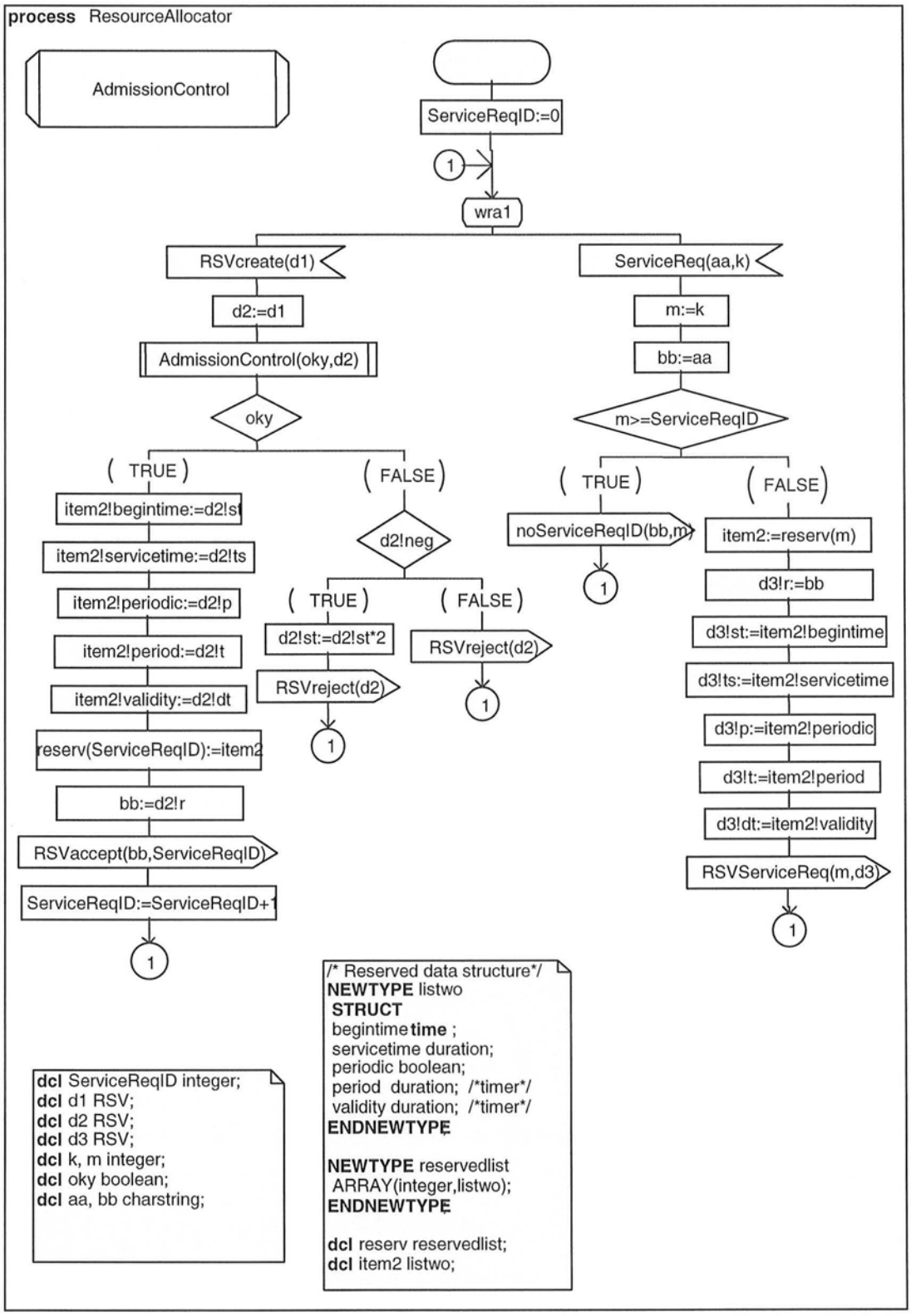

Fig. C.12. ResourceAllocator process diagram

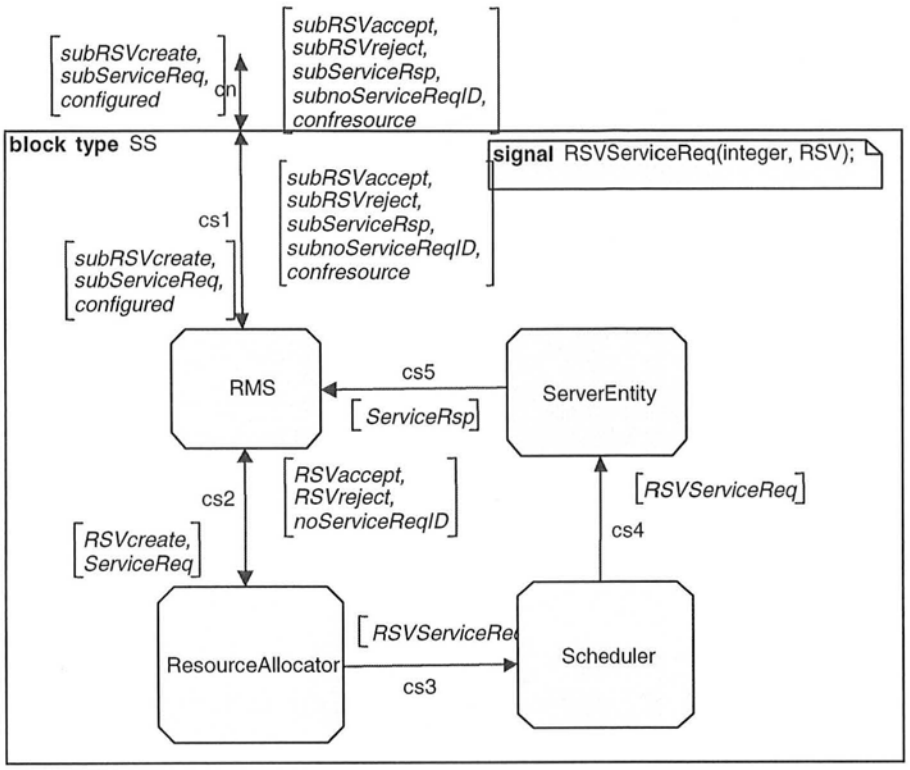

Fig. C.13. SecondayServer Block type

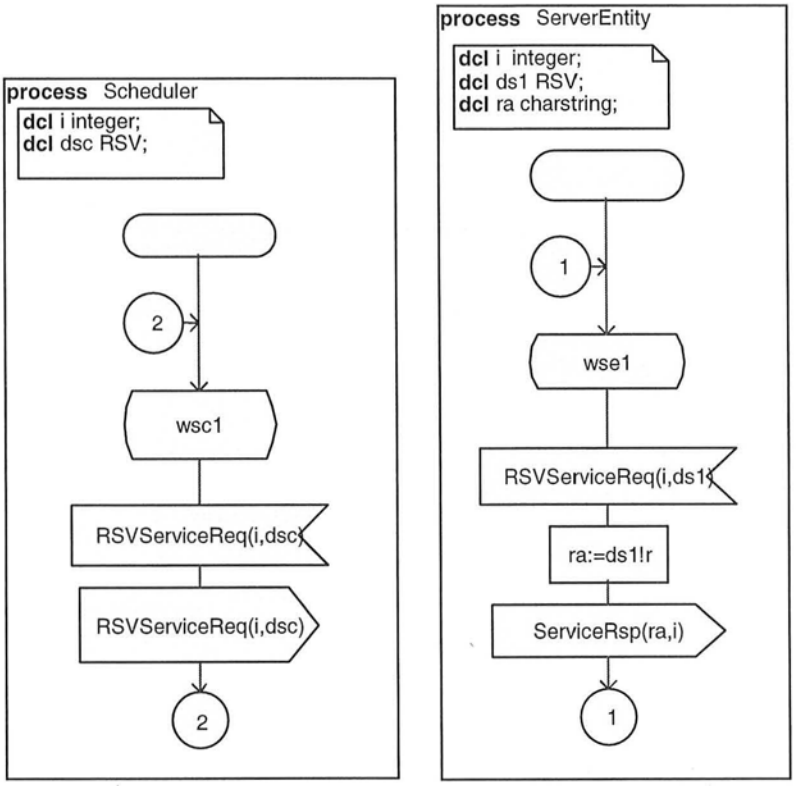

Fig. C.14. Scheduler and ServerEntity process diagrams

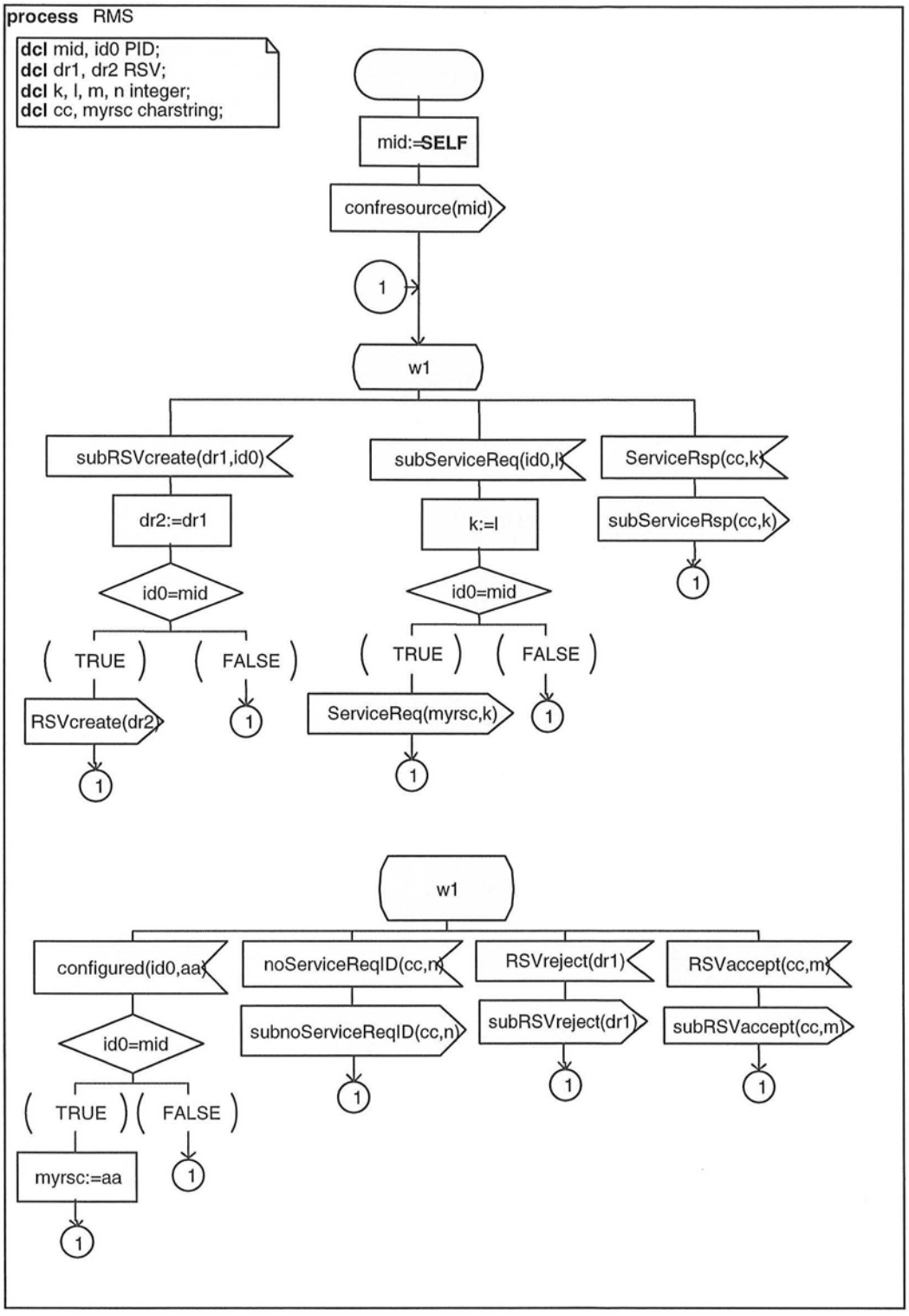

Fig. C.15. RMS process diagram

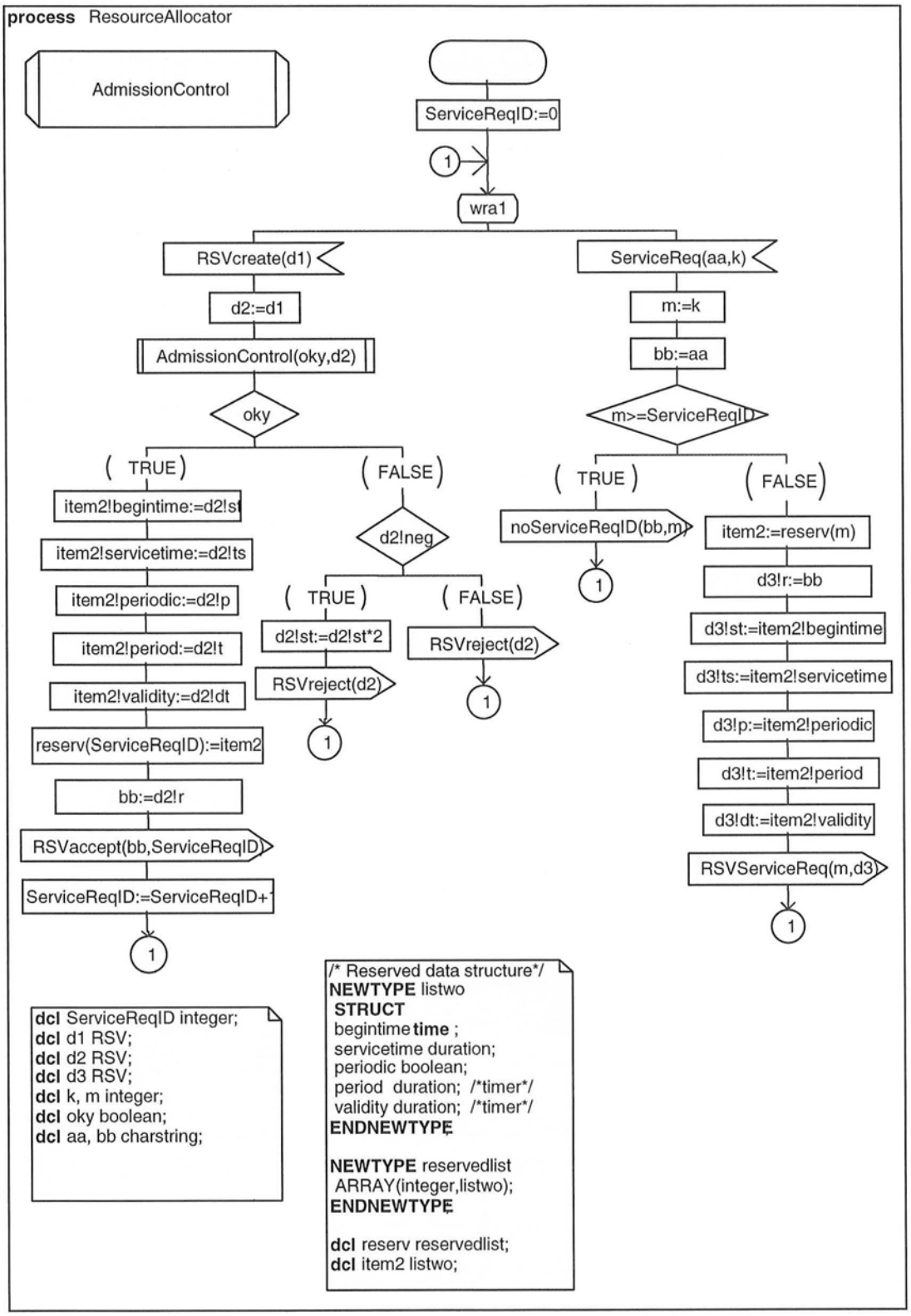

Fig. C.16. ResourceAllocator process diagram

Table des figures et des tableaux

Liste des abréviations utilisées

AC	Admission Control
BA	Bandwidth Allocator
CAN	Controller Area Network
CE	Client Entity
CEI	Commission Électrotechnique Internationale
CMS-RSVP	Client-MultiServer Resource ReSerVation Protocol
CSMA/CA	CSMA with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CSMA/DCR	CSMA with Deterministic Collision Resolution
CSMA/LDCR	CSMA with Laxity based Deterministic Collision Resolution
CS-RSVP	Client-Server Resource ReSerVation Protocol
DiffServ	Differentiated Services
DL	Data Link
DLC	DL-Connection
DLCEP	DL Connection-End-Point
DLE	Data Link Entity
DLPDU	DL Protocol Data Unit
DLS	Data Link Service
DLSAP	DL Service Access Point
DLSDU	DL Service Data Unit
DLSEP	DL Scheduling End Point
DOD/CSMA/CD	Deadline Oriented Deterministic CSMA/CD
EDF	Earliest Deadline First
FDDI	Fiber Distributed Data Interface
FSM	Finite State Machine
IEC	International Electrotechnical Commission
IEEE	Institue of Electrical and Electronics Engineers
IntServ	Integrated Services
ISO	Organisation International de Normalisation
LL	Least Laxity
LLC	Contrôle de la Liaison Logique
MAC	Contrôle d'Accès au Médium
MCS-RSVP	MultiClient-Server Resource ReSerVation Protocol
MSC	Message Sequence Charts
(N)-PCF	(N)-Policy Control Function
N-PDU	Unité de données du protocole-N
(N)-PE	(N)-Protocol Entity
(N)-QCF	(N)-QoS Control Function
NUI	Network Update Intervals
OSI	Interconnexion des Systèmes Ouverts
PB/CSMA/CD	Preemption Based CSMA/CD
Ph	Physical
PhSAP	Physical Service Access Point
PROFIBUS	PROcess FIeld BUS

QoS	Quality of Service
QMF	QoS Management Function
RA	Resource Allocator
RM	Requester Module
RMC	Requester Module of the Client
RMS	Requester Module of the Server
RSVP	Resource ReSerVation Protocol
SBM	Subnet Bandwidth Management
SC	Scheduler
SDL	Langage de Spécification et Description
SE	Server Entity
SME	System Management Entities
SPCF	System Policy Control Function
SQCF	System Quality Control Function
SQE	System QoS Entities
TDMA	Time Division Multiple Access
TTP	Time-Triggered Protocol
WorldFIP	World Factory Instrumentation Protocol

Bibliographie

[Abeysundara and Kamal, 1991] Abeysundara B.W. and A.E. Kamal. High-Speed Local Area Networks and their Performance: A Survey. *Computing Surveys*, 23(1): 221-264, 1991.

[Adler, 1995] Adler, R.M. Distributed coordination models for client/server computing. *Computer - IEEE Computer Magazine*, 28(4): 14-22, 1995.

[Adrion *et al.*, 1982] Adrion, R., M. Branstad and J. Cherniavsky. Validation, Verification, and Testing of Computer Software. *Computing Surveys*, 14(2): 159-192, 1982.

[Agrawal *et al.*, 1992] Agrawal, G., B. Chen, W. Zhao and S. Davari. Guaranteeing synchronous message deadlines with the timed token protocol. In *Proc. of 12th International Conference on Distributed Computing Systems*, pp. 468-475, Japan, 1992.

[Agrawal *et al.*, 1993] Agrawal, G., B. Chen and W. Zhao. Local Synchronous Capacity Allocation Schemes for Guaranteeing Message Deadlines with the Timed Token Protocol. In *Proceedings of INFOCOM'93*, pp. 186-193, 1993.

[Almeida *et al.*, 1999a] Almeida, L., R. Pasadas and J.A. Fonseca. Using the Planning Scheduler to Improve Flexibility in Real-Time Fieldbus Networks. *Control Engineering Practice*, 7(1): 101-108, 1999.

[Almeida *et al.*, 1999b] Almeida, L., M. Leon, J.A. Fonseca and J.P. Thomesse. Real-time Communications in Factory. In *World Multiconference on Systemics, Cybernetics and Informatics - SCI'99*, 119-125, Orlando-USA, 1999.

[Alur and Dill, 1994] Alur, R. and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126: 183-235, 1994.

[Andreoli and Pareschi, 1996] Andreoli, J-M. and R. Pareschi. Integrated Computational Paradigms for Flexible Client-Server Communication. *Computing Surveys*, 28(2): 297-299, 1996.

[Andrews, 1991] Andrews, G.R. Paradigms for Process Interaction in Distributed Programs. *Computing Surveys*, 23(1): 49-90, 1991.

[ANSI, 1990] ANSI. FDDI Media Access Control (MAC). *ANSI Standard X3T9.5/88-139, Rev. 4.0*, 1990.

[Arvind *et al.*, 1991] Arvind, K., K. Ramamritham and J. Stankovic. A Local Area Network Architecture for Communication in Distributed Real-Time Systems. *The J. of Real-Time Systems*, 3, 115-147, 1991.

[Audsley and Burns, 1990] Audsley, N. and A. Burns. Real-time System Scheduling. *Technical Report YCS 134*, Department of Computer Science, University of York, 1990.

[Audsley *et al.*, 1993] Audsley, N., A. Burns, M.F. Richardson, K. Tindell and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5): 284-292, 1993.

[Aurrecoechea, 1998] Aurrecoechea, C., A. Cambell and L. Hauw. A survey of QoS architectures. *Multimedia Systems*, 6, 138-151, 1998.

[Bayart and Simonot-Lion, 1995] Bayart, M. and Simonot-Lion. Impact de l'émergence des réseaux de terrain et de l'instrumentation intelligente dans la conception des architectures des systèmes d'automatisation de processus. Contrat MESR–92-p-239, Centre de Recherche en Informatique de Nancy, Février 1995.

[Bernet *et al.*, 1999] Bernet, Y., *et al.* A Framework for Differentiated Services. *Internet draft*, draft-ietf-diffserv-framework-02.txt, Feb., 1999.

[Blake *et al.*, 1998] Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss. An Architecture for Differentiated Services. *Internet RFC 2475*, Dec., 1998.

[Boehm, 1976] Boehm, B. Software Engineering. *IEEE Transactions on Computers*, C-25(12): 1226-1241, 1976.

[Bolognesi and Brinksma, 1987] Bolognesi, T. and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14, 25-59, 1987.

[Bosch, 1992] Bosch,R. GmbH. *CAN Protocol Specification V2.0 (A,B)*, 1992.

[Boudenant *et al.*, 1987] Boudenant, J., B. Feydel and P. Rolin. An IEEE 802.3 compatible deterministic protocol. In *IEEE INFOCOM'87*, pp. 573-579, Sn. Francisco, USA, 1987.

[Braden *et al.*, 1994] Braden, R., D. Clark and S. Shanker. Integrated Services in the Internet Architecture: an Overview. *Internet RFC 1633*, June, 1994.

[Bræk, 1996] Bræk, R. SDL Basics. *Computer Networks and ISDN Systems*, 28(12): 1585-1602, 1996.

[Cardeira, 1994] Cardeira, C. Ordonnancement temps réel par réseaux de neurones. *Ph.D. Thèse*, INPL, France, 1994.

[Cardeira and Mammeri, 1995] Cardeira, C. and Z. Mammeri. A schedulability analysis of tasks and networks traffic in distributed real-time systems. *Measurement*, 15: 71-83, 1995.

[Casavant and Kuhl, 1988] Casavant, T.L. and J.G. Kuhl. A Taxonomy of Scheduling in general-purpose Distributed Systems. *IEEE Transactions on Software Engineering*, 14(2):141-154, 1988.

[Cheng, 1996] Cheng, K.E. A requirements definition and assessment framework for SDL tools. *Computer Networks and ISDN Systems*, 28(12): 1703-1716, 1996.

[Chung *et al.*, 1994] Chung, W.S., C.K. Un and B.C. Shin. Analysis of the delay bounds of a tree protocol with collision detection. *Computer Communications*, 17(4): 288-296, 1994.

[Dakroury and Elloy, 1989] Dakroury, Y. and J.P. Elloy. A new multi-server concept for the MMS Environment. In *Proc. of 9th IFAC Workshop on DCCS*, 1989.

[Dakroury, 1990] Dakroury, Y. Specification et validation d'un protocole de messagerie multi-serveurs pour l'environnement MMS. *Ph.D. Thèse*, ENSM, Nantes, France. 1990.

[Dasarathy, 1985] Dasarathy, B. Timing Constraints of Real-Time Systems: Constructs for expressing Them, Methods of Validating Them. *IEEE Transactions on Software Engineering*, 11(1): 80-86, 1985.

[Degermark *et al.*, 1997] Degermark, M., T. Köhler, S. Pink and O. Schelén. Advances reservations for predictive service in the Internet. *Multimedia Systems*, 5: 177-186, 1997.

[Dertouzos and Mok, 1989] Dertouzos, M. and A. Mok. Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks. *IEEE Transactions on Software Engineering*, 15(12): 1497-1506, 1989.

[DIN 19245-1,2,3] German Standards 19245-1 to 19245-3. PROFIBUS, Process fieldbus, *DIN 19245-1,2 and 3*, 1990

[DIN 19245-4] German Draft Standard 19245-4. PROFIBUS-PA, Profibus for Process Automation, *DIN 19245-4*, 1996.

[DS 21906] DS, Danish Standard, DS 21906. *P-Net, Multi-master, multi-net fieldbus for sensor, actuator and controller communication*, 1990.

[EN 50170-1] CENELEC EN 50170-1. P-NET. General Purpose Field Communication System, *EN 50170-1*, 1995.

[EN 50170-2] CENELEC EN 50170-2. PROFIBUS. General Purpose Field Communication System, *EN 50170-2*, 1995.

[EN 50170-3] CENELEC EN 50170-3. WorldFIP. General Purpose Field Communication System, *EN 50170-3*, 1995.

[Ferrari, 1990] Ferrari, D. Client Requirements for Real-Time Communication Services. *Internet RFC 1193*, Nov., 1990.

[Ferrari and Verma, 1990] Ferrari, D. and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3): 368-379, 1990.

[Ferrari *et al.*, 1997] Ferrari, D., A. Gupta and G. Ventre. Distributed advances reservation of real-time connections. *Multimedia Systems*, 5, 187-198, 1997.

[Fidge, 1998] Fidge, C. Real-Time Schedulability Tests for Preemptive Multitasking. *Real Time Systems*, 14(1): 61-93, 1998.

[Fine and Tobagi, 1984] Fine, M. and F.A. Tobagi. Demand Assignment Multiple Access Schemes in Broadcast Bus Local Area Networks. *IEEE Transactions on Computers*, C-33(12): 1130-1159, 1984.

[Galara and Thomesse, 1984] Galara, D. and J.P. Thomesse. FIP : Proposition d'un système de transmission série multiplexée pour les échanges d'information entre capteurs, des actionneurs et des automates réflexes. *Ministère de l'Industrie et de la Recherche*, France, 1984.

[Ghanwani *et al.*, 1999] Ghanwani, A., J. Wayne, V. Srinivasan, A. Smith and M. Seaman. A Framework for Integrated Services over shared and switched IEEE 802 LAN Technologies. *Internet draft,* draft-ietf-issll-is802-framework-07, June 1999.

[Graham and Majumdar, 1999] Graham, W.C., and S. Majumdar. Performance of scheduling strategies for client-server systems. *Journal of Parallel and Distributed Computing*, 58(3): 389-424, 1999.

[Hogrefe, 1996] Hogrefe, D. Validation of SDL systems. *Computer Networks and ISDN Systems*, 28(12): 1659-1668, 1996.

[IEC 61158] International Electrotechnical Commission. Digital Data Communications for Measurement and Control – Fieldbus for use in Industrial Control Systems. *IEC 61158*, 1999.

[IEEE 802.1D] IEEE Standard, Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks -Common Specifications - Media Access Control (MAC) bridges, *IEEE 802.1D*, 1998.

[IEEE 802.1Q] IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridge Local Area Networks, *IEEE 802.1Q*, 1998.

[IEEE 802.3] IEEE Standard 802.3. Carrier Sense Multiple Access with Collision Detect (CSMA/CD) Access Method and Physical Layer Specifications, 1985.

[IEEE 802.4] IEEE Standard 802.4. Token-Passing Bus Access Method and Physical Layer Specifications, 1985.

[IEEE 802.5] IEEE Standard 802.5. Token Ring Access Method and Physical Layer Specifications, 1985.

[Irey *et al.*, 1998] Irey IV, P.M., R.D. Harrison and D.T. Marlow. Techniques for LAN Performance Analysis in a Real-Time Environment. *Real Time Systems*, 14(1): 21-44, 1998.

[ISO 7498] International Standardization Organization. Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. *ISO/IEC 7498*, 1994.

[ISO 8807] International Standardization Organization. Information Processing Systems, Open Systems Interconnection, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. *ISO 8807*, 1989.

[ISO/CEI 8886] International Standardization Organization. Information technology – Open Systems Interconnection – Data link service definition. *ISO/CEI 8886*, 1996.

[ISO 9074] International Standardization Organization. Information Processing Systems, Open Systems Interconnection, Estelle – A Formal Description Technique Based on an Extended State Transition Model. *ISO 9074*, 1989.

[ISO 11898] International Standardization Organization. Road Vehicles-Interchange of Digital Information-Controller Area Network (CAN) for High Speed Communication. *ISO DIS 11898*, 1992.

[ISO 13236] International Standardization Organization. Quality of Service – Framework. *ISO/IEC DIS 13236*, 1996.

[ISO 184/SC5/WG2] International Standardization Organization. Architecture and Communications, User requirements for system supporting time critical communications. *ISO TC 184/SC5/WG2*, 1992.

[ITU Z.100] ITU-T, Recommendation Z.100 - Specification and Description Language (SDL). *International Telecommunication Union*, 1993.

[ITU Z.120] ITU-T, Recommendation Z.120 – Message Sequence Chart (MSC), *International Telecommunication Union*, Geneva, 1994.

[Joseph and Pandya, 1986] Joseph, M. and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5): 390-395, 1986.

[Johnson, 1987] Johnson, M.J. Proof that timing requirements of the FDDI token ring protocols are satisfied. *IEEE Transactions on Communications,* COM-35(6): 620-625, 1987.

[Klein *et al.*, 1993] Klein, M., T. Ralya, B. Pollak, R. Obenza and M. Harbour. A Practitioner's Handbook for Real-Time Analysis. *Kluwer Academic Publishers*, 1993.

[Klein *et al.*, 1994] Klein, M., J.P. Lehoczky and R. Rajkumar. Rate-monotonic analysis for real-time industrial computing. *Computer – IEEE Computer Magazine*, 27(1): 24-33, 1994.

[Kopetz, 1991] Kopetz, H. Event-triggered versus time-triggered real-time systems. *Lecture Notes in Computer Science*, 563, 87-101, 1991.

[Kopetz and Grünsteidl, 1994] Kopetz, H. and G. Grünsteidl. TTP - A Protocol for Fault-Tolerant Real-Time Systems. *Computer – IEEE Computer Magazine*, 27(1): 14-23, 1994.

[Kopetz, 1997] Kopetz, H. Real-Time Systems, Design Principles for Distributed Embedded Applications. *Kluwer Academic Publishers*, 1997.

[Kurose *et al.*, 1984] Kurose, J.F., M. Schwartz and Y. Yemini. Multiple-Access Protocols and Time Constrained Communication. *Computing Surveys*, 16(1): 43-70, 1984.

[Kurose *et al.*, 1988] Kurose, J.F., M. Schwartz and Y. Yemini. Controlling window protocols for time constrained communication in multiple access networks. *IEEE Transactions on Communications*, 36(1): 41-49, 1988.

[Laprie *et al.*, 1989] Laprie, J.C., B. Courtois, M.C. Gaudel and D. Powel. Sûreté de fonctionnement des systèmes informatiques. *Dunod informatique*, 1989.

[Laprie, 1992] Laprie, J.C. Dependability: Basic Concepts and Terminology. *Springer-Verlag, J.C. Laprie 5 of Dependable Computing and Fault Tolerance*, Vienna, Austria. 1992.

[Lehoczky and Sha, 1986] Lehoczky, J.P. and L. Sha. Performance of Real-Time Bus Scheduling Algorithms. *ACM Performance Evaluation Review*, Special Issue, 14(1), 1986.

[Lehoczky *et al.*, 1989] Lehoczky, J.P., L. Sha and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings 10th IEEE Real-Time Systems Symposium*, pp. 166-171, 1989.

[Le Lann and Rolin, 1984] Le Lann, G. and P. Rolin. Procédé et dispositif pour la transmission de messages entre différentes stations à travers un réseau local à diffusion. *French patent, No. 8416957*, 1984.

[Le Lann and Rivierre, 1994] Le Lann, G. and N. Rivierre. Real-Time Communications over Broadcast Networks: the CSMA-DCR and the DOD/CSMA-CD Protocols. In *Actes des Conférences de RTS'94*, 67-84, Paris-France, Janvier 1994.

[Leon and Thomesse, 2000] Leon, M. and J.P. Thomesse. Fieldbuses and Real-Time MAC Protocols. Accepted paper In *4th IFAC International Symposium on Intelligent Components and Instruments (SICICA 2000)*, Buenos Aires, Argentina, September, 2000.

[Lim *et al.*, 1991] Lim, C-C., L-j. Yao and W. Zhao. A comparative study of three token ring protocols for real-time communication. In *Proceedings of the 11th IEEE International Conference on Distributed Computing Systems*, pp. 308-317, 1991.

[Liu and Layland, 1973] Liu, C. and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. of the Association for Computing Machinery*, 20(1): 46-61, 1973.

[Lorenz, 1994] Lorenz, P., Z. Mammeri and J.P. Thomesse. A State-Machine for Temporal Qualification of Time-Critical Communication. In *26th IEEE Southeastern Symposium on System Theory*, pp. 654-658, Ohio, USA, 1994.

[Malcolm *et al.*, 1990] Malcolm, N., W. Zhao and C. Barter. Guarantee Protocols for Communication in Distributed Hard Real-Time Systems. In *Proceedings of the IEEE INFOCOM'90*, pp. 1078-1086, 1990.

[Malcolm and Zhao, 1995] Malcolm, N. and W. Zhao. Hard Real-Time Communication in Multiple-Access Networks. *Real Time Systems*, 8(1): 35-77, 1995.

[Malcolm *et al.*, 1996] Malcolm, N., S. Kamat and W. Zhao. Real-Time Communication in FDDI Networks. *Real Time Systems*, 10(1): 75-107, 1996.

[Mammeri and Thomesse, 1991] Mammeri, Z. and J.P. Thomesse. Réseaux Locaux, Couche Physique et Couche Liaison de Données. *TEKNEA*, 1991.

[Manna and Pnueli, 1992] Manna, Z. and A. Pnueli. The temporal logic of reactif and concurrent systems: Specification. *Springer-Verlag*, 1992.

[Mnaouer *et al.*, 1997] Mnaouer, A.B., T. Ito, H. Tanaka, W-K. Yoo and T.S. Sekiguchi. Asynchronous Bandwidth Allocation and Parameter Setting in the Fieldbus Protocol. *Transactions of the Institute of Electrical Engineers of Japan*, 117-C(7): 962-970, 1997.

[Mok and Ward, 1979] Mok, A. and S. Ward. Distributed Broadcast Channel Access. *Computer Networks*, 3, 327-335, 1979.

[Molle and Kleinrock, 1985] Molle, H.L. and L. Kleinrock. Virtual time CSMA: Why two clocks are better than one. *IEEE Transactions on Communications*, COM-33(9): 919-933, 1985.

[Moon *et al.*, 1998] Moon, H-j., H.S. Park, S.C. Ahn and W.H. Know. Performance degradation of the IEEE 802.4 token bus network in a noisy environment. *Computer Communications*, 21(6): 547-557, 1998.

[Nakajima and Tokuda, 1998] Nakajima, T. and H. Tokuda. User-level real-time network on microkernel-based operating systems. *Real Time Systems*, 14(1): 45-60,1998.

[Navet *et al.*, 2000] Navet, N., Y-Q. Song and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over controller area network. *Journal of Systems Architecture*, 46: 607-617, 2000.

[Ng and Liu, 1991] Ng, J.K.Y. and J.W.S. Liu. Performance of Local Area Network Protocols for Hard Real-Time Applications. In *Proceedings of the 11th IEEE International Conference on Distributed Computing Systems*, pp. 318-326, 1991.

[Norden *et al.*, 1999] Norden, S. and S. Balaji, G. Manimaran, and C.R.M. Siva. Deterministic protocols for real-time communication in multiple access networks. *Computer Communications*, 22(2): 128-136, 1999.

[Panwar *et al.*, 1988] Panwar, S.S., D. Towsley and J.K. Wolf. Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service. *J. of the Association of the Computing Machinery*, 35(4): 832-844, 1988.

[Perhson, 1990] Perhson, B. Protocol Verification for OSI. *Computer Network and ISDN Systems*, 18(3): 185-201, 1990.

[Pimentel, 1990] Pimentel, J.R. Communications Networks for Manufacturing. *Prentice-Hall, Englewood-Cliffs*, 1990.

[Pleinevaux, 1992] Pleinevaux, P. An improved hard real-time scheduling for the IEEE 802.5. *The J. of Real-Time Systems*, 4, 99-112, 1992.

[Rößler and Geppert, 1997] Rößler, F. and B. Geppert. Applying Quality of Service Architectures to the Field-Bus Domain. *In 2th IEEE International Workshop on Factory Communications Systems – WFCS'97*, pp. 39-48, 1997.

[Saba *et al.*, 1993] Saba, G., J.P. Thomesse and Y.Q. Song. Space and Time Consistency Qualifications in a Distributed Communication System. *IMACS/IFAC*, pp. 383-391, 1993.

[Seaman *et al.*, 1999] Seaman, M., A. Smith, E. Crawley and J. Wroclawski. Integrated Service Mappings on IEEE 802 Networks. *Internet draft,* draft-ietf-issll-is802-svc-mapping-04, June, 1999.

[Sevcik and Johnson, 1987] Sevcik, K.C. and M.J. Johnson. Cycle time properties of the FDDI token ring protocol. *IEEE Transactions on Software Engineering*, SE-13(3): 376-385, 1987.

[Sha and Goodenough, 1990]Sha, L., and J.B. Goodenough. Real-time scheduling theory and Ada. *Computer - IEEE Computer Magazine*, 23(4): 53-62, 1990.

[Sha *et al.*, 1990] Sha, L., R. Rajkumar and J.P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9), 1990.

[Shin and Hou, 1990] Shin, K.G. and C.J. Hou. Analysis of three contention protocols in distributed real-time systems. In *Proceedings 11th IEEE Real-Time Systems Symposium*, pp. 136-145, 1990.

[Simonot and Song, 1997] Simonot, F. and Y-Q. Song. Real-time communications using TDMA-based multi-access protocol. *Computer Communications*, 20(6): 435-448, 1997.

[Song, 1991] Song, Y-Q. Etude de Performance de FIP, Aide au Dimensionnement d'Applications. *Ph.D. Thèse*, INPL, France, 1991.

[Song and Thomesse, 1994] Song, Y-Q. and J.P. Thomesse. Classification of Networks. In *Industrial Communications Workshop*, Paris, 1994.

[Song and Simonot, 1996] Song, Y-Q. and F. Simonot. Messages scheduling in FDDI for Real-Time Communication. *Actes des Conférences de RTS'96*, pp. 287-301, 1996.

[Sorenson, 1974] Sorenson, P.G. A methodology for real-time systems development, least laxity algorithm. *Ph.D. Thesis*, University of Toronto-Canada, 1974

[Stallings, 1984] Stallings, W. Local Network Performance. *IEEE Communications Magazine*, 22(2): 27-36, 1984.

[Stankovic, 1988] Stankovic, J.A. Misconceptions about real-time computing: A serious problem for next generation systems. *Computer - IEEE Computer Magazine*, 21(10): 10-19, 1988.

[Stankovic and Ramamritham, 1990] Stankovic, J.A. and K. Ramamritham. What is Predictability for Real-Time Systems? *The J. of Real-Time Systems*, 2, 247-254, 1990.

[Stankovic *et al.*, 1998] Stankovic, J., M. Spuri, K. Ramamritham and G. Buttazzo. Deadline Scheduling for Real-Time Systems, EDF and Related Algorithms. *Kluwer Academic Publishers*, 1998.

[Strosnider *et al.*, 1988] Strosnider, J.K., T.E. Marchok and J. Lehoczky. Advanced real-time scheduling using the IEEE 802.5 token ring. In *Proceedings 9th IEEE Real-Time Systems Symposium*, pp. 42-52, 1988.

[Strosnider and Marchok, 1989] Strosnider, J.K. and T.E. Marchok. Responsive, deterministic IEEE 802.5 token ring scheduling. *The J. of Real-Time Systems*, 1, 133-158, 1989.

[Tanenbaum, 1988] Tanenbaum, A.S. Computer Networks. *Prentice-Hall*, second edition, 1988.

[Thomesse, 1993] Thomesse, J.P. Le réseau de terrain FIP. *Réseaux et Informatique Répartie*, 3(3): 287-321, 1993.

[Thomesse *et al.*, 1995] Thomesse, J.P., Z. Mammeri and L. Vega. Time in distributed systems: cooperation and communication models. In *5th IEEE Workshop on Future Trends of Distributed Computing Systems*, IEEE Computer Society Press, pp. 41-49, 1995.

[Thomesse, 1998] Thomesse, J.P. A Review of the FieldBuses. *Annual Reviews in Control*, 22: 35-45, 1998.

[Thomesse, 1999] Thomesse, J.P. Fieldbuses and Interoperability. *Control Engineering Practice*, 7(1): 81-94, 1999.

[Thomesse and Leon, 1999] Thomesse, J.P. and M. Leon. Main Paradigms as a Basis for Current Fieldbus Concepts. In *Fieldbus Technology – FeT'99*, pp. 2-15, Magdeburg, Germany, 1999.

[Tindell *et al.*, 1995a] Tindell, K., A. Burns and A. Wellings. Analysis of Hard Real-Time Communications. *Real Time Systems*, 9(2): 147-172, 1995.

[Tindell *et al.* 1995b] Tindell,K., A. Burns and A. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3(8): 1163-1169, 1995.

[Topolcic, 1990] Topolcic, C. Experimental Internet Stream Protocol: Version 2 (ST-II). *Internet RFC 1190*, October, 1990.

[Tovar and Vasques,1999] Tovar, E. and F. Vasques. Cycle time properties of the PROFIBUS timed-token protocol. *Computer Communications*, 22(13): 1206-1216, 1999.

[Tovar *et al.*, 1999] Tovar, E., F. Vasques and A. Burns. Supporting real-time distributed computer-controlled systems with multi-hop P-NET networks. *Control Engineering Practice*, 7(8): 1015-1025, 1999.

[Ulusoy, 1995] Ulusoy, O. Network access protocol for hard real-time communication systems. *Computer Communications*, 18(12): 943-948, 1995.

[Vega and Thomesse, 1995] Vega, L. and J.P. Thomesse. Temporal properties in distributed real-time applications. In *13th Workshop on Distributed Computer Control Systems (DCCS'95)*, pp. 91–96, 1995.

[Vega, 1996] Vega, L. Modèles de Coopération et de Communication entre Processus Temps Réel Répartis. *Ph.D. Thèse*, INPL, France, 1996.

[van As, 1994] van As, H.R. Media access technologies: The evolution towards terabit/s LANs and MANs. *Computer Networks and ISDN Systems*, 26(6): 603-656, 1994.

[Voelcker, 1986] Voelcker, J. Helping computers to communicate. *IEEE Spectrum*, march 1986.

[Wedde *et el.*, 1994] Wedde, H.F., B. Korel and D.M. Huizinga. Formal Timing Analysis for Distributed Real-Time Programs. *Real Time Systems*, 7(1): 57-90, 1994.

[Xiao and Ni, 1999] Xiao, X. and L.M. Ni. Internet QoS: A Big Picture. *IEEE Network*, 13(2): 8-19, 1999.

[Yao and Zhao, 1991] Yao, L.J. and W. Zhao. Performance of an extended IEEE 802.5 protocol in hard real-time systems. In *Proceedings of the IEEE INFOCOM'91*, pp. 469-478, 1991.

[Yao *et al.*, 1995] Yao, L.J., W. Zhao and C.C. Lim. A comparative study of three token ring protocols for real-time communications. *International Journal of Mini and Microcomputers*, 17(2): 84-97, 1995.

[Yavatkar *et al.*, 2000] Yavatkar, R., D. Hoffman, Y. Bernet, F. Baker and M. Speer. SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks. *Internet draft,* draft-ietf-issll-is802-sbm-10, January 2000.

[Zhang and Burns, 1995] Zhang, S. and A. Burns. Guaranteeing synchronous message sets in FDDI networks. In *13th Workshop on Distributed Computer Control Systems*, DCCS'95, IFAC, pp. 107-112, Toulouse – France, 1995.

[Zhang *et al.*, 1993] Zhang, L., S. Deering, D. Estrin, S. Shenker and D. Zappala. RSVP: A new Resource ReSerVation Protocol. *IEEE Network,* 7(5): 8-11, 1993.

[Zhao and Ramamritham, 1987] Zhao,W. and K. Ramamritham. A virtual time CSMA protocol for hard real-time communications. *IEEE Transactions on Software Engineering*, SE-13(8): 938-952, 1987.

[Zhao *et al.*, 1990] Zhao, W., J. Stankovic and K. Ramamritham. A window protocol for transmission of time constrained messages. *IEEE Transactions on Computers*, 39(9): 1186-1203, 1990.

[Znati, 1991] Znati, T. Deadline Driven Protocol for Transmission of Real-Time Traffic. In *Proceedings of the 10th IEEE International Conference on Computers and Communications*, pp. 667-673, 1991.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur FONSECA José A., Professeur, Université d'Aveiro (Portugal),
Monsieur BOURCERIE Marc, Professeur, LISA/Université d'Angers,
Monsieur ROMARY Laurent, Chargé de Recherche HDR, LORIA/UHP Nancy I.

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur LEON-CHAVEZ Miguel Angel

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Qualité de service et ordonnancement dans les systèmes de communication temps réel".

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"INFORMATIQUE"**

Fait à Vandoeuvre le, **15 septembre 2000**

Le Président de l'I.N.P.L.,

J. HARDY

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 5 4 5 0 1
VANDŒUVRE CEDEX

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55

Résumé

Nous nous intéressons ici aux systèmes de communication temps réel à architectures réduites, c'est-à-dire sans couches réseau et transport. La plupart de ces systèmes sont conçus pour satisfaire certains besoins temporels des utilisateurs, essentiellement sur les échanges périodiques et parfois pour les échanges sporadiques. La satisfaction des besoins est en général assurée par une configuration statique ou hors ligne de l'ordonnancement des messages.

Notre thèse propose une solution dynamique à la place des solutions statiques. Cette solution s'inspire de l'architecture de Qualité de Service définie dans les grands réseaux et des techniques de services Intégrés de l'Internet qui sont généralement utilisés au niveau de la couche réseau.

Nous proposons donc un protocole de réservation dynamique de ressources qui peut être implanté sur n'importe quel protocole MAC de type centralisé pour n'importe quelle topologie. Ce protocole peut accepter par ailleurs différents algorithmes d'ordonnancement. Il a été spécifié en termes de systèmes états-transitions et a été validé en utilisant l'outil ObjectGEODE.

Mots-clés : Temps Réel, Systèmes de Communication, Réseaux de Terrain, Qualité de Service, Ordonnancement.

Abstract

In this thesis, we are interested in special real time communication systems called Fieldbus networks, i.e. without network nor transport layers. Most of these systems are designed to satisfy certain user time-related requirements, basically the periodic exchanges and sometimes the sporadic exchanges. Generally, the requirements are met and guaranteed by a static or off-line configuration of the message scheduling.

We propose a dynamic solution in place of the static solutions. This solution takes as starting points the Quality of Service architecture defined in the wide-area networks and the Integrated service model of the Internet community which are generally used at the network layer.

We propose then a dynamic resource reservation protocol which can be used by any MAC protocol of type centralized and for any topology. In addition, this protocol can accept various scheduling algorithms. It was specified in terms of state-transition systems and was validated using the ObjectGEODE tool.

Keywords: Real time, Communication Systems, Fieldbus networks, Quality of Service, Scheduling.