ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE
M.Ing

PAR
BERNARD TREMBLAY

ALGORITHMES DE GESTION DYNAMIQUE DES RESSOURCES

MONTRÉAL, LE 8 AOÛT 2006

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE

M. Michel Kadoch, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

Mme Maria Bennani, codirectrice
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Pierre Jean Lagacé, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Zbigniew Dziong, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 27 JUILLET 2006

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ALGORITHMES DE GESTION DYNAMIQUE DES RESSOURCES

Bernard Tremblay

SOMMAIRE

La téléphonie IP ainsi que plusieurs autres applications multimédias exigent, des réseaux qui les supportent, des qualités de service nécessaires à leur bon fonctionnement. La meilleure façon de garantir une qualité de service implique la réservation des ressources nécessaires à l'offre de cette qualité de service. Pour rencontrer le critère de mise à l'échelle, il est impossible d'établir une réservation pour chaque flot de données. Ces réservations doivent être réalisées en considérant une agrégation de flots nécessitant la même qualité de service. De plus, même si une réservation de ressources est établie pour garantir une qualité de service à un nombre défini de flots, celle-ci doit demeurer dynamique afin de s'adapter aux besoins des applications ainsi qu'aux exigences économiques des entités utilisant ces applications.

Pour ce faire, ce projet suggère une architecture de réseau MPLS où un serveur SIP, en plus d'aider à l'établissement des sessions, sert également à la gestion dynamique des ressources. Ce serveur établit un contrôle d'accès et se basant sur les requêtes obtenues, évalue les besoins en terme de ressources nécessaires au maintien de la qualité de service. Connaissant la quantité de ressources nécessaires, ce serveur a la capacité d'augmenter ou diminuer les ressources réservées afin d'optimiser la satisfaction des usagers (taux de rejet minimum) ou d'optimiser les coûts (taux d'utilisation maximum).

Quelques algorithmes de contrôle d'accès et d'évaluation des besoins existant ainsi qu'un algorithme proposé, nommé ERL pour Erlang, ont été modélisés et simulés à l'aide du logiciel Opnet. Les paramètres de ces différents algorithmes ont été optimisés afin d'obtenir des taux de rejet et d'utilisation s'approchant le plus possible des taux trouvés par la théorie. Plus un algorithme permet d'obtenir des taux proche de la théorie, plus il est considéré comme performant. Les trois algorithmes ont été comparés afin de déterminer le plus performant.

L'algorithme proposé a permis d'obtenir tous les taux de rejet ou d'utilisation désirés avec la meilleure performance de tous les algorithmes étudiés.

DYNAMIC MANAGEMENT OF RESSOURCES

Bernard Tremblay

ABSTRACT

Telephony IP as several other multimedia applications require, of the networks which support them, of qualities of service necessary to their correct operation. The best way of guaranteeing a quality of service implies the reservation of the resources necessary to the offer of this quality of service. To meet the criterion of scaling, it is impossible to establish a reservation for each data flow. These reservations must be carried out by considering an aggregation of streams requiring the same quality of service. Moreover, even if a reservation of resources is established to guarantee a quality of service to a definite number of streams, this one must remained dynamic in order to adapt to the needs of the application streams aggregate like to the economic requirements of the entities using these applications.

With this intention, this project suggests architecture of network using MPLS where a SIP server, in more of helping with the establishment of the sessions, is also used for dynamic management of the resources. This server, establish an access control and basing itself on the requests obtained, evaluates the requirements in term of resources necessary to the maintenance of the quality of service. Knowing the quantity of resources necessary, this server has capacity to increase or decrease the resources reserved in order to optimize the satisfaction of the users (minimum rejection rate) or to optimize the costs (maximum utilization ratio).

Some algorithms of need assessment and access control existing as well as an algorithm proposed, named ERL, were modeled and simulated using the Opnet software. The parameters of these various algorithms were optimized in order to obtain use and rejection rates approaching as much as possible of the rates permitted by the theory. The more one algorithm makes it possible to obtain rates close to the theory, the more it is regarded as powerful. The three algorithms were compared in order to determine that which was most powerful.

The algorithm proposed, made it possible to obtain all the rates desired rejection or use with the best performance of all the studied algorithms.

REMERCIEMENTS

Ce mémoire s'inscrit dans les activités du laboratoire de gestion de réseaux informatiques et de télécommunications (LAGRIT), de l'École de technologie supérieure (ÉTS).

J'aimerais exprimer ma gratitude à mon directeur de mémoire, M. Michel Kadoch et ma codirectrice Mme. Maria Bennani, qui ont bien voulu accepter et diriger cette recherche et dont les conseils ont été enrichissants. Je tiens également à les remercier pour leur grande patience ainsi que leurs encouragements dans les moments difficiles.

Ma gratitude va aussi aux membres du jury pour avoir pris le temps de lire attentivement et d'évaluer mon mémoire.

Je souhaite remercier l'équipe du laboratoire LAGRIT de l'ÉTS pour m'avoir accueilli et pour m'avoir fourni les ressources nécessaires à la mise en œuvre de ce projet. Un gros merci à M. Essarsari pour son aide au codage des modèles.

Enfin, ma reconnaissance va vers tous ceux qui m'ont aidé dans la réalisation de ce mémoire et en particulier à ma conjointe, Catherine Beaulieu, ma fille Sabrina Tremblay ainsi que mon garçon, Olivier Tremblay.

TABLE DES MATIERES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET DES SIGLES

ABW	Available BandWidth
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
COPS	Common Open Policy Service
CR-LDP	Constrained-based Routed - LDP
CSPF	Constraint Shortest Path First
DNS	Domain Name System
DSCP	Diffserv Service Code Point
EIGRP	Enhanced IGRP
FEC	Forwarding Equivalence Class
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunication Union
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Label Switched Path
E-LSP	EXP-LSP
L-LSP	Label-LSP
FIFO	First In First Out
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
LSR	Label Switch Router
MAC	Medium Access Control
MG	Media Gateway
MGC	MG Controller
MGCP	MG Control Protocol

MPLS	Multi Protocol Label Switching
MSC	Maximum Simultaneous Calls
NGN	Next Generation Network
OC	Optical Carrier
OSPF	Open Shortest Path First
PBX	Private Branch Exchange
PCM	Pulse Code Modulation
PHB	Per Hop Behavior
PSTN	Public Switch Telephone Network
QoS	Quality of Service
RED	Random Early Detection
SCCP	Skinny Client Control Protocol
SIP	Session Initation Protocol
RIP	Routing Information Protocol
RSVP	ReSerVation Protocol
RSVP-TE	RSVP – Traffic Engineering
RTP	Real-time Transport Protocol
SC	Service Controller
TCP	Transmission Control Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VoIP	Voice over IP
VPN	Virtual Private Network
WFQ	Weighted Fait Queuing
WRED	Weighted Random Early Detection

INTRODUCTION

La téléphonie IP (*Internet Protocol*) est en phase de devenir la norme pour le transport de la voix, pas seulement en tant qu'application sur Internet, mais bel et bien pour remplacer les réseaux téléphoniques actuels. Pour ce faire, les réseaux de données qui supporteront cette nouvelle application devront être en mesure de fournir une qualité de service permettant de concurrencer la qualité des réseaux téléphoniques conventionnels.

Pour fournir une bonne qualité de service, la réservation de ressources est le moyen le plus efficace. Des réservations faites sur la base de chaque appel ne pouvant être mises à l'échelle, celles-ci doivent se faire sur une base plus large. Une entreprise peut décider de réserver une certaine quantité de bande passante et la dédier à la téléphonie IP (agrégation de flots). Un mécanisme de contrôle d'accès permettra ensuite aux usagers d'utiliser cette capacité. Cette réservation ayant un prix, il est nécessaire de bien choisir quelle sera la quantité de ressource réservée et de pouvoir modifier cette réservation dynamiquement, en fonction des besoins réels.

Un des objectifs de ce projet est de proposer une architecture de réseau adéquate pour le support de réservations de ressources par agrégation de flots. Dans un contexte de téléphonie IP, un serveur SIP (*Session Initiation Protocol*) implantera les algorithmes d'évaluation dynamique des besoins en ressources. Certains de ces algorithmes possédant plusieurs paramètres de fonctionnement, il fait partie des objectifs de ce projet de déterminer les valeurs permettant d'optimiser les évaluations faites par ces algorithmes. Pour déterminer ces valeurs, les différents algorithmes ont été modélisés dans un serveur SIP à l'aide du logiciel de simulation Opnet. Le comportement des algorithmes est étudié en simulant un profil de communications téléphoniques typique.

La première phase du projet fut de modéliser adéquatement l'architecture de réseau proposée. Un réseau MPLS supportant l'ingénierie de trafic et la qualité de service a été

modélisé. Les différentes options de configuration ont été approfondies afin de déterminer comment le réseau MPLS supporterait l'architecture proposée. Cette étape permet de simuler l'environnement dans lequel évoluera le serveur SIP avec gestion dynamique des ressources réservées.

La seconde étape fut de comprendre l'implantation et le fonctionnement des modèles SIP déjà établis. Pour comprendre ces implantations, le fonctionnement du logiciel de modélisation Opnet se devait d'être étudié en profondeur puisque la modélisation de processus implique une très bonne connaissance de tous les aspects de ce logiciel. Cette étude en profondeur a permis par la suite de modéliser les algorithmes.

La troisième étape fut la modélisation des différents algorithmes de contrôle d'accès et d'évaluation dynamique des ressources à l'intérieur d'un serveur SIP. Ces algorithmes sont :

1- Contrôle d'accès sans évaluation des ressources.
2- Contrôle d'accès et gestion dynamique de la bande passante [Bo].
3- Contrôle d'accès et gestion dynamique de la bande passante, simplifié.
4- Contrôle d'accès et évaluation dynamique des ressources, basée sur le trafic moyen et la théorie d'Erlang.

Le premier chapitre présente les technologies SIP, MPLS. Le second chapitre fait un survol des différentes méthodes permettant d'établir un contrôle d'accès et discute également de quelques architectures utilisant SIP sur MPLS. Le troisième chapitre propose une architecture de réseau de nouvelle génération utilisant SIP et MPLS pour offrir un service de téléphonie IP, dont la qualité de service est garantie par une évaluation dynamique des besoins et une réservation dynamique des ressources nécessaires au support de la qualité de service. Les configurations nécessaires au bon fonctionnement de SIP et MPLS ainsi que la façon de simuler un profil de communication téléphonique y seront également traité. Le chapitre 4 présente la façon

dont l'application voix et le protocole SIP sont implantés dans le logiciel Opnet. Leur compréhension est nécessaire afin de modéliser convenablement les algorithmes étudiés. La proposition d'un nouvel algorithme et la présentation des algorithmes étudiés, ainsi que leur modélisation seront vues en détail au chapitre 5. Le chapitre 6 couvrira l'ensemble des expérimentations, des résultats de simulation et l'analyse de ceux-ci.

CHAPITRE 1

TECHNOLOGIES ABORDÉES

Les protocoles SIP et MPLS sont deux protocoles de réseau voués à un bel avenir. Le premier a pour fonction de mettre en relation deux entités applicatives afin d'établir une session de communication. MPLS a pour objectif de réconcilier la technologie de réseau sans connexion qu'est IP avec les avantages que procurent les technologies de réseau orientées avec connexion. MPLS utilise des circuits virtuels auxquels une qualité de service, nécessaire à certaines applications, peut être attribuée.

MPLS n'interagit aucunement avec les clients, il s'agit d'une technologie de réseau totalement transparente pour l'usager et présente uniquement dans les équipements d'acheminement du réseau. L'usager qui désire établir une session, de voix par exemple, utilisera le protocole SIP qui est un protocole applicatif totalement absent des équipements d'acheminement du réseau. SIP n'étant pas conçu pour dialoguer avec les équipements d'acheminement du réseau, il faut établir une relation, telle que SIP puisse établir des sessions multimédias pour lesquelles une qualité de service est assurée.

Ce premier chapitre présente les technologies de commutation MPLS et de signalisation SIP. L'utilisation de SIP sur un réseau MPLS a fait l'objet de quelques travaux qui seront expliqués. Le contrôle d'accès et la gestion dynamique des ressources ainsi que les algorithmes réalisant ces fonctions et qui seront étudiés dans le cadre de ce mémoire seront présentés.

1.1 Réseaux MPLS

MPLS [RFC 3031] est une technologie dont le but est d'améliorer l'acheminement des paquets IP. Pour cela, les avantages de IP sont combinés à ceux d'une technologie en mode circuit. Un circuit est établi pour l'acheminement de paquets à destination de

l'adresse X. Ce circuit est identifié par une suite d'étiquettes (*Labels*) qui sont ajoutées devant l'en-tête IP. La commutation de niveau 2 sur la base d'étiquettes est plus simple et plus rapide, ce qui permet d'améliorer l'efficacité par rapport à un acheminement conventionnel de niveau 3, plus long et plus complexe.

Dans un réseau MPLS, les technologies IP et MPLS se complètent à merveille. Avec IP viennent les protocoles de routage tels RIP (*Routing Information Protocol*) et OSPF (*Open-Shortest Path First*), qui permettent l'établissement de tables de routage nécessaires à l'acheminement des paquets IP. Ces tables sont utilisées pour déterminer le chemin que suivra un circuit MPLS lors de l'établissement de ce dernier. MPLS n'ajoute aucun protocole de routage, il utilise ceux déjà déployés pour IP. Comme pour le routage, l'adressage utilisé provient de la technologie IP. Pour sa part, MPLS ajoute la possibilité d'acheminer les paquets IP à l'aide de circuits virtuels.

Les premiers développements de MPLS étaient surtout dédiés à augmenter la rapidité d'acheminement en se basant sur une technologie à commutation d'étiquettes plutôt que sur un routage à base d'adresses. Il apparut assez rapidement que MPLS pouvait faire beaucoup plus que de simplement améliorer la rapidité d'acheminement. Avec un réseau orienté connexion, les réservations de ressources, dans une optique de qualité de service, sont beaucoup plus simples. Lors de la création d'un circuit, il est possible de faire une réservation propre à ce dernier. Il est donc aisé d'appliquer une architecture de qualité de service sur un réseau MPLS. La technologie MPLS peut supporter les approches de QoS (*Quality of Service*) que sont *Intserv* [RFC 2215] et *Diffserv* [RFC 2475].

Un autre intérêt majeur de MPLS est le support de l'ingénierie de trafic. Par exemple, sur un réseau MPLS, il est possible de créer deux circuits, suivant des chemins totalement différents, pour acheminer du trafic provenant d'une même source vers une même destination. Cela permet de faire de la répartition de charge, de sélectionner un

chemin en fonction de caractéristiques voulues et d'offrir une sécurité accrue par la redondance de chemin.

Dans une optique de transport de la voix ou de toutes autres applications multimédias, l'utilisation d'un réseau MPLS est tout à fait appropriée de par sa rapidité de commutation, son support de la qualité de service et par son support de l'ingénierie de trafic.

1.1.1 Les bases de la technologie de réseau MPLS.

Un domaine MPLS est composé de deux types de routeurs. En périphérie, les LER (*Label Edge Router*) font la transition entre les domaines MPLS et non MPLS. Les LSR (*Label Switch Router*) sont des routeurs de cœur, ne faisant aucune transition et acheminant les données uniquement par commutation d'étiquettes (Figure 1).

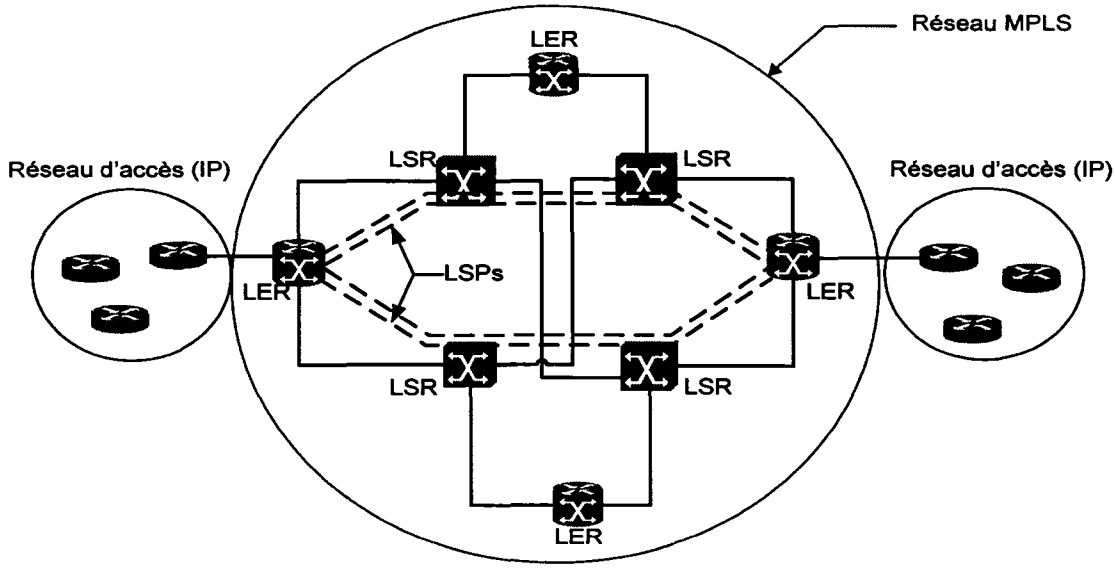


Figure 1 Réseau MPLS

Le rôle d'un LER est de recevoir un paquet IP provenant d'un réseau non MPLS et d'y ajouter une étiquette afin que le reste de l'acheminement puisse se faire à l'aide d'un LSP (*Label Switch Path*), c'est-à-dire un circuit virtuel MPLS. À l'autre bout du LSP, le

LER supprime l'étiquette afin que le paquet puisse continuer son chemin par routage conventionnel. Pour ajouter la bonne valeur d'étiquette, le LER doit faire correspondre le paquet entrant à un des LSP existants. Pour ce faire, le LER utilise des tables de correspondance appelées FEC (*Forwarding Equivalence Class*). Une FEC spécifie les caractéristiques que doit posséder un paquet pour être acheminé sur un LSP en particulier. Une entrée de la table du LER contient les informations suivantes :

1- Caractéristiques devant avoir un paquet entrant pour être acheminé sur le LSP correspondant à cette entrée.
   - Adresse IP de la source
   - Adresse IP de la destination
   - Protocole de transport
   - Port source
   - Port destination
   - Valeur du champ *Type of Service*

2- Caractéristiques propres au LSP.
   - Étiquette a être ajoutée
   - Interface de sortie

On peut donc faire une correspondance de type : si le paquet IP provient de l'interface E0, provient de l'adresse A, à destination de l'adresse B, que le protocole de transport est TCP, en provenance du port C et à destination du port D et que le champ ToS a la valeur X, alors ajouter une étiquette Y et transmettre sur l'interface Z. Bien qu'une entrée puisse être très précise, comme le suggère l'exemple précédent, une entrée de la FEC pourrait également spécifier : tous les paquets à destination du réseau B ajouter l'étiquette Y et transmettre sur l'interface Z.

La valeur de l'étiquette ajoutée par le LER correspond à un LSP. Cependant, la valeur de l'étiquette n'est pas constante tout au long du chemin. La valeur d'une étiquette n'a de signification qu'entre deux commutateurs MPLS. Les *Label Switch Router*, comme leur nom l'indique, font de la commutation d'étiquettes. Leur table de correspondance est beaucoup plus simple et contient les informations suivantes :

- Interface d'entrée
- Étiquette d'entrée
- Interface de sortie
- Étiquette de sortie

La correspondance est : pour les paquets reçus sur l'interface X ayant la valeur d'étiquette 1, transmettre sur l'interface Y avec la valeur d'étiquette 2.

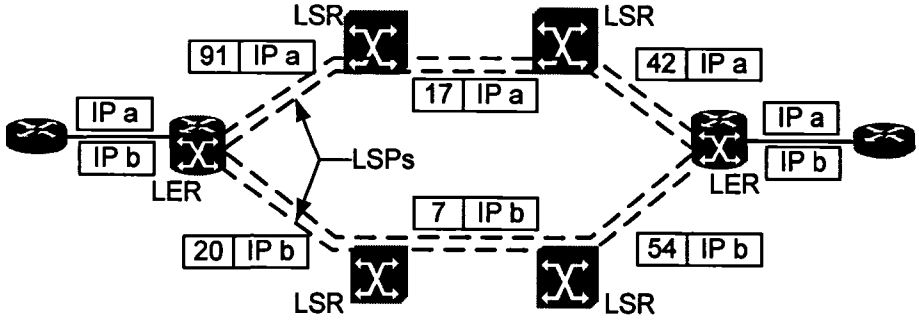


Figure 2 LSP et étiquettes

Un LSP origine d'un LER ajoutant une étiquette aux paquets IP, suit le chemin déterminé par les tables de correspondances des différents LSR et se termine sur le LER qui supprime l'étiquette (Figure 2)

1.1.2 Distribution d'étiquettes

La distribution des valeurs d'étiquettes nécessaire à la formation des tables de correspondances peut prendre plusieurs formes. Elle peut être indépendante ou ordonnée, non sollicitée ou sur demande. Si la distribution est indépendante, chaque routeur est responsable d'associer une étiquette à chacune des entrées de sa table de routage et de transmettre cette information. Par exemple, « pour me transmettre un paquet à destination du réseau X, utilisez la valeur d'étiquette suivante ». Pour ce faire, un protocole de routage quelque peu modifié, tel BGP-4 (*Border Gateway Protocol*) [RFC 3107] qui est en mesure de transporter les valeurs d'étiquettes en plus de l'information de routage, doit être utilisé.

La distribution peut également être ordonnée. Dans ce cas, un LER est responsable d'initier la création d'un LSP, les routeurs MPLS voisins doivent suivre la consigne. La création ordonnée d'un LSP peut être non sollicitée ou sur demande. L'établissement d'un LSP non sollicité est initié par le *Egress* LER (celui qui est situé à la fin d'un LSP) et s'établis en sens inverse du chemin. Par exemple, le *Egress LER* transmet une étiquette au LSR le précédant et lui indique, « pour rejoindre le réseau 192.168.10.0, utilise la valeur d'étiquette suivante : 47». Le LSR recevant cette étiquette passera l'information au LSR le précédant : « pour rejoindre le réseau 192.168.10.0 utilisez l'étiquette suivante : 52 » (Figure 3). Cette méthode permet d'élaborer des FEC associant un réseau de destination à un LSP. Cela évite de multiples décisions de routage. Le LER *ingress* (celui qui est situé à l'entrée d'un LSP) associant le paquet à un LSP, suivra le circuit jusqu'à la sortie sans aucune autre décision de routage.

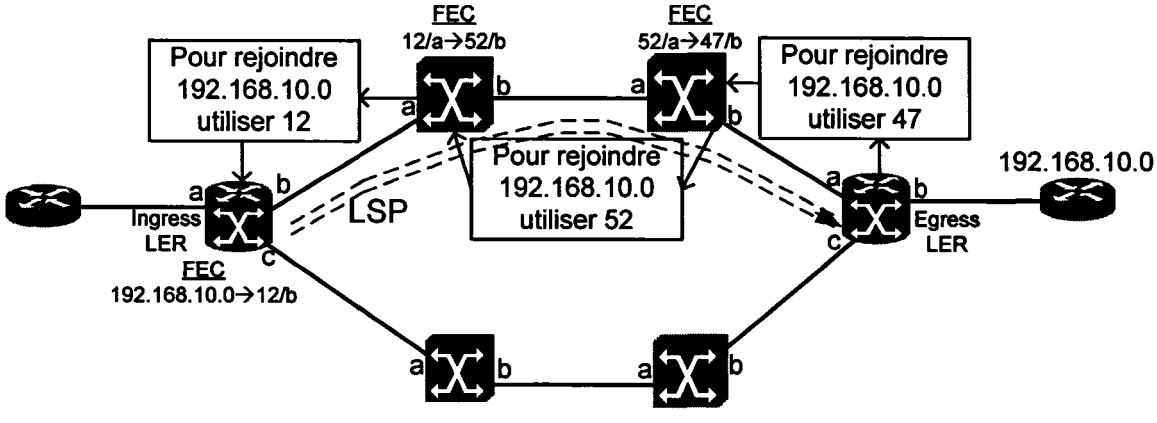


Figure 3 Distribution ordonnée d'étiquette

La création ordonnée et à la demande d'un LSP (Figure 4) est initiée par une demande d'un LER *Ingress* transmise à un LER *Egress* en lui indiquant : « Je désire l'établissement d'un LSP ». La distribution des étiquettes se fera, par la suite, de la même façon que dans le cas du LSP non sollicité.

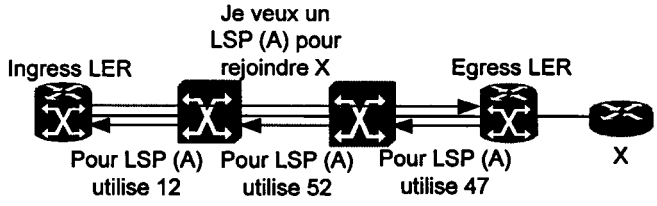


Figure 4 Création à la demande d'un LSP

Pour la distribution des étiquettes, peu importe la méthode utilisée, un protocole spécialisé doit être utilisé pour l'établissement des LSPs. LDP (*Label Distribution Protocol*) [RFC 3036] est un protocole qui a été conçu exclusivement pour la technologie MPLS dont le rôle est la distribution des étiquettes. Il utilise quatre types de message. Le premier type de message permet aux différents routeurs MPLS de se découvrir. Il s'agit d'un message de type *Hello* qui est transmis à l'aide de l'adresse *Multicast* « à tous les routeurs du sous réseau ». Ce message sert à informer les routeurs voisins que l'instigateur du message est en mesure d'utiliser la technologie MPLS et que l'échange d'étiquettes peut être fait à l'aide du protocole LDP. Si un routeur MPLS veut participer à l'élaboration de LSP, il doit établir une session avec ses voisins à l'aide de messages de type *Session*. Une session LDP est faite à l'aide du protocole de transport TCP afin de s'assurer de la livraison des messages. Suite à l'établissement d'une session, les messages d'avertissements servent à créer, changer ou supprimer les associations d'étiquettes définissant les LSPs. Finalement les messages de notification servent à transporter d'autres types d'informations tels les messages d'erreurs.

Au début de MPLS, l'amélioration de l'acheminement des paquets IP étant le but, le mode non sollicité était utilisé puisqu'il permettait de faire correspondre les tables de routage IP à des LSP. Les développements ultérieurs de MPLS, tels le support de la qualité de service et de l'ingénierie de trafic, ont imposé le mode « à la demande ». La création d'un LSP à l'aide d'une requête peut être réalisée par LDP. Le protocole RSVP (*Resource ReSerVation Protocol*) [RFC 2205] peut également être utilisé pour l'établissement d'un LSP. Pour un réseau MPLS dont les LSP sont créés à la demande,

l'utilisation de RSVP, un protocole éprouvé et déjà présent dans la majorité des équipements de réseaux, évite l'ajout d'un protocole supplémentaire.

1.1.3 MPLS et la qualité de service

La qualité de service est aujourd'hui une caractéristique essentielle à tout réseau moderne. MPLS étant surtout utilisé avec des réseaux IP, elle se doit de supporter les deux architectures de qualité de service définies pour les réseaux IP, c'est-à-dire *Intserv et Diffserv*.

L'ajout de RSVP comme protocole de distribution d'étiquettes n'est pas sans relation avec l'ajout du support de la qualité de service par la technologie MPLS. L'architecture de QoS *Intserv* pour les réseaux IP nécessite une réservation de ressources le long d'un chemin. Pour ce faire, le protocole RSVP transmet une requête à la destination. Le routeur de destination fait savoir au routeur précédent qu'il a les ressources suffisantes. Ce dernier répond à son tour au routeur qui le précède qu'il a les ressources et ainsi de suite jusqu'à la source de la requête. Ceci est similaire à l'élaboration d'un LSP à la demande. Pour adapter RSVP au réseau MPLS, l'ajout d'un champ permettant le transport de l'étiquette associée au flot pour lequel la réservation est faite est suffisant. Un LSP est ainsi associé à un flot de données pour lequel une réservation a été faite à l'aide de RSVP.

Le protocole LDP est en mesure de supporter toutes les méthodes de création de LSP discutées précédemment dont l'établissement à la demande. Au début de MPLS, l'établissement d'un LSP à la demande consistait en une requête pour obtenir un circuit dont la destination est un réseau en particulier. Le support de la qualité de service a complexifié la façon de réaliser les LSP à la demande. LDP ne supportant pas les requêtes avec qualité de service, il a été adapté pour permettre la création de LSP selon certaines contraintes. CR-LDP (*Constraint-based Routed - LSP*) [RFC 3213] fournit les mécanismes nécessaires à l'établissement d'un LSP suivant un chemin précis. Le LER

initiant la requête à l'aide de CR-LDP spécifie le chemin que doit suivre le LSP. Pour ce faire, il doit être au courant de l'état du réseau afin de déterminer un chemin qui correspond à certaines contraintes.

Contrairement à RSVP qui permet un routage des LSP par qualité de service et par flot individuel (*Intserv*), CR-LDP ne s'applique qu'à des agrégats de trafic (*Diffserv*). Les contraintes peuvent être la bande passante nécessaire, le nombre de sauts maximum, un délai. Les contraintes peuvent également être de nature politique ou exiger un certain type d'équipement de réseau. Les contraintes permettent de déterminer un chemin qui rencontre certaines exigences qui ne sont pas uniquement de type qualité de service. Il est possible de créer un LSP en spécifiant d'éviter certaines parties de réseau.

Un réseau MPLS supportant l'architecture de qualité de service *Diffserv* attribue ses ressources en fonction des types de trafic supportés. Les ressources sont attribuées à chaque classe de service par le gestionnaire et chaque paquet doit être identifié pour connaître son type. Pour bien traiter les paquets, six bits sont disponibles dans l'en-tête IP pour catégoriser les paquets. Les LSR regardant seulement l'en-tête MPLS, où trois bits peuvent être utilisés pour le marquage (bits EXP), deux méthodes permettent de déterminer à quelle classe de service appartient un paquet. Avec la première méthode, les circuits sont appelés L-LSP (Label - LSP). Dans ce cas, l'étiquette est utilisée par le LSR pour déterminer la catégorie et le comportement (*queuing* et *scheduling*) qu'il doit utiliser. Les 3 bits EXP sont utilisés pour déterminer la priorité de rejet, utile en cas de congestion. Dans ce type de LSP, tous les paquets sont traités de la même façon (même qualité de service), mais il est possible de déterminer différentes priorités de rejet. Avec la seconde méthode, les 3 bits EXP permettent le transport, dans un même LSP, de différentes classes de service. Le LSR se base sur la valeur de ces 3 bits pour déterminer la classe de service. Bien que les paquets soient traités par classe plutôt que par flot, le LER s'assure de la disponibilité des ressources lors de la création du E-LSP (EXP - LSP) à l'aide des contraintes.

Réserver des capacités sur la base de flot semble être la solution la plus efficace et la plus susceptible de garantir une qualité de service, par contre il est impensable que chaque équipement de réseaux ait à gérer plusieurs millions de flots différents, chacun ayant ses caractéristiques. L'architecture de qualité de service *Intserv* ne pouvant être mise à l'échelle, l'intérêt pour celle-ci n'est plus très grand. Il est réaliste de croire que l'architecture *Diffserv* est vouée à un avenir beaucoup plus prospère. L'attribution des ressources étant globale, cette technologie est beaucoup plus simple et permet un meilleur contrôle de la part du gestionnaire de réseau. L'architecture *Diffserv*, utilisée sur un réseau strictement IP, n'est pas la meilleure garantie de qualité de service si l'utilisation qui en est faite n'est pas compatible avec la configuration des ressources. Pour s'assurer que le nombre de paquets traités selon une classe de service donnée n'excède pas les ressources qui ont été assignées à cette classe, le gestionnaire doit sélectionner quels paquets seront traités avec cette classe. Appliqué à un réseau MPLS, bien que le marquage individuel des paquets demeure une nécessité, il est plus simple à réaliser. Lors de l'ajout de l'en-tête MPLS par le LER, le marquage de l'étiquette peut être fait en même temps que la sélection de la valeur de l'étiquette. La gestion des ressources de chaque classe de qualité de service est également simplifiée dans la mesure où le routage d'un LSP par contraintes est en fait un contrôle d'accès ; si les ressources ne sont pas suffisantes, il sera impossible de déterminer un chemin offrant les contraintes de qualité de service requises par le LSP. Dans une certaine mesure, il est possible de considérer un LSP comme étant un agrégat de flots pour lesquels une réservation de ressources existe.

1.1.4 Ingénierie de trafic

L'ingénierie de trafic [RFC 2702] implique l'adaptation de l'acheminement du trafic aux conditions du réseau avec comme objectifs de fournir une bonne performance aux usagers et une utilisation efficace des ressources du réseau. Son rôle est de fournir aux ISP (*Internet Service Provider*) un contrôle efficace des flots de trafic transitant sur leurs réseaux. L'ingénierie de trafic, sur un réseau strictement IP, nécessite une connaissance

des flots de trafics. La nature des réseaux IP fait en sorte que cette connaissance est basée sur des mesures complexes et incomplètes de l'état du réseau. Cette difficulté de réaliser une ingénierie de trafic efficace sur un réseau strictement IP provient de l'absence de circuits. MPLS ajoute aux réseaux IP cette caractéristique cruciale. La technologie MPLS est donc tout désignée pour supporter l'ingénierie de trafic.

Le deuxième élément limitant l'ingénierie de trafic est la configuration manuelle des routeurs. Une autre approche permet de parvenir aux mêmes objectifs, c'est le routage par contraintes. Avec CR-LDP, un LSP est créé en fonction de contraintes. Ces contraintes peuvent être de nature à offrir une qualité de service, elles peuvent également être spécifiques à l'ingénierie de trafic.

L'architecture de qualité de service *Intserv* étant impossible à appliquer aux réseaux de grande taille, l'approche *Diffserv* demeure la seule architecture réaliste. Le routage de LSP par contraintes permettant l'ingénierie de trafic en plus de la qualité de service, le protocole RSVP n'était plus vraiment adapté aux nouveaux rôles de la technologie MPLS. Les promoteurs de RSVP ne lâchant pas prise, RSVP fut adapté au routage par contraintes et à l'ingénierie de trafic dans une nouvelle version appelée RSVP-TE (*RSVP - Traffic Engineering*) [RFC 3209] .

Lorsque MPLS est utilisé pour l'ingénierie de trafic, le concept de « *Traffic Trunk* » est appliqué. Un *Traffic Trunk* est une agrégation de flots de trafic ayant la même classe de service et placée à l'intérieur d'un LSP (Figure 5). Il s'agit d'une représentation abstraite de trafic à laquelle certaines caractéristiques peuvent être associées. Le chemin que suit un *Trafic Trunk* peut être modifié, utiliser un LSP différent. Dans cette optique, le trafic entrant dans un réseau MPLS n'est pas associé à un LSP mais à un *Traffic Trunk* qui est lui-même associé à un ou des LSP.

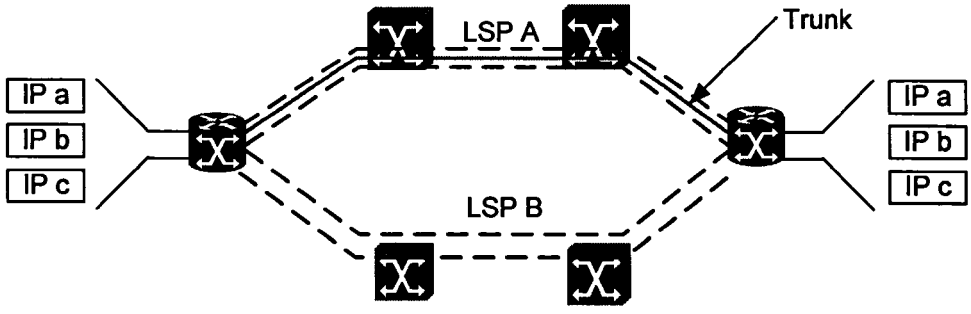


Figure 5 *Traffic Trunk* passant sur un LSP

IP a, b et c (Figure 5) sont des flots de trafics ayant des caractéristiques communes et la même destination. Si pour une raison quelconque, relative à l'ingénierie de trafic, il faut modifier le chemin suivi par ces flots, il est très simple de mettre en œuvre cette modification en associant le *Trunk* avec le LSP B plutôt qu'avec le A.

Les attributs des *Trunk* sont des paramètres définissant leurs caractéristiques comportementales. Ces attributs sont :

- Attributs de paramètres de trafic
- Attributs de sélection du chemin et maintenance
- Attributs de priorité
- Attributs de préemption
- Attributs de résilience
- Attributs de politique

Les paramètres de trafic caractérisent les flots de trafic les constituant ou plus précisément la FEC déterminant le trafic y transitant. Les attributs de sélection du chemin et de maintenance définissent comment le chemin suivi par le *Trunk* est calculé. Il peut être établi manuellement par un administrateur. Si aucune restriction ne s'applique, le chemin peut être calculé à l'aide d'un protocole basé sur la topologie (routage conventionnel). Finalement, le chemin peut être calculé en se basant sur un routage avec contraintes si des exigences en termes de bande passante ou de politique existent. Ces attributs permettent également de définir plusieurs chemins avec une hiérarchie de préférence entre ceux-ci, en cas de panne. Les règles de préférence définissent le chemin alternatif. Un attribut permet également de caractériser

l'adaptabilité du *Trunk* aux ressources du réseau. Par une modification des ressources du réseau, si un meilleur chemin existe, cet attribut permet ou interdit au *Trunk* d'utiliser un nouveau chemin. Également, il est possible de partitionner la charge de trafic du *Trunk* sur des chemins différents ainsi que de caractériser le pourcentage de trafic suivant chaque chemin. L'attribut de priorité définit l'importance relative des *Trunk* pour la détermination des chemins lors de la phase d'établissement de connexion ou en cas de panne. La préemption définit la possibilité pour un *Trunk* d'imposer la modification du chemin suivi par un autre *Trunk*. Un *Trunk* désirant suivre un certain chemin peut imposer aux *Trunks* de niveau préemptif inférieur de modifier le chemin qu'ils utilisent. L'attribut de résilience définit le comportement que doit avoir le *Trunk* en cas de panne. Finalement les attributs de politique déterminent le comportement que doivent avoir les protocoles sous-jacents lorsque le *Trunk* devient non conforme.

Pour établir un routage par contraintes des *Trunks*, des paramètres d'états de la topologie du réseau sont nécessaires. Parmi ces paramètres, les attributs de ressources sont essentiels. Un premier attribut détermine la proportion de ressources disponible pour l'élaboration de *Trafic Trunk*. Cet attribut, appelé *Maximum Allocation Multiplier* s'applique à la bande passante des liens et aux capacités de temporisations. Des attributs de classes de ressources existent également. Ces attributs, qui n'ont rien à voir avec les classes de qualité de service, sont une abstraction permettant de définir des politiques de routage particulières. Ces différentes classes de services, appelées couleurs, permettent par exemple d'établir la classe de ressource rouge qui est constituée de liens de télécommunication OC-48 (*Optical Carrier* - 48). Il est ainsi possible, lors du routage par contraintes, de spécifier que le chemin suivi par le *Trunk* devra être établi sur des liens OC-48.

Le concept de *Trunk*, les attributs le paramétrant, ainsi que le concept d'attributs de ressources, donnent une bonne idée de ce que permet l'ingénierie de trafic sur un réseau de technologie MPLS.

1.1.5 Réseau de nouvelles génération et Voix sur MPLS

Tel le web et le courriel, la voix sur IP est vouée à un bel avenir. Les applications précurseurs de la voix sur IP, permettant un dialogue entre deux PC, sont loin d'être l'aboutissement de cette technologie. À terme, le but est l'élaboration d'une infrastructure de réseau unique permettant la convergence totale des réseaux à commutation de paquets (IP) et de circuits (voix).

Les technologies de réseau par paquets et par circuits se séparent en deux plans nommés commutation (ou transport) et contrôle (ou signalisation). Le plan de commutation est constitué des fonctions permettant le transport de l'information. Dans un réseau IP il s'agit de décisions prises par chaque routeur, pour chaque paquet, à l'aide d'une table de routage. Dans un réseau de voix, il s'agit de fonctions associant les circuits d'entrées aux circuits de sorties. Pour sa part, le plan de contrôle est constitué de l'ensemble des fonctions fournissant les informations nécessaires au plan de commutation. Dans un réseau IP, il s'agit des protocoles de routage élaborant les tables auxquels se fie le plan de commutation. Dans un réseau de voix, il s'agit de l'ensemble des fonctions de signalisation permettant à l'usager de faire une requête d'élaboration d'un circuit, la sélection du chemin que suivra ce circuit et la signalisation aux différents commutateurs des informations nécessaires à l'association entrée-sortie.

Dans un réseau IP, ces deux plans sont intégrés dans un ensemble homogène. Cette fusion des plans débouche sur une technologie de réseau dont le plan de contrôle ne sert qu'à l'élaboration de tables de routage. Ce plan de contrôle est peu flexible et ne permet pas une gestion efficace de la part du gestionnaire. Autrement dit, l'ingénierie de trafic sur ce type de réseau est pratiquement impossible. Cette approche de la réseautique considère que le réseau est stupide, se limitant à l'offre d'un service d'acheminement et que l'intelligence doit se retrouver à l'extérieur de celui-ci à l'aide d'application client serveur. Il en résulte une grande difficulté pour les gestionnaires de réseaux à fournir des services à valeurs ajoutées autres que l'acheminement.

À l'opposé des réseaux IP, l'approche des réseaux téléphoniques conventionnels suggère que l'intelligence doit être centralisée dans l'infrastructure du réseau, les terminaux étant plutôt stupides (Figure 6). Cette approche de la réseautique permet d'offrir une multitude de services à valeur ajoutée, tels l'appel en attente, l'afficheur, la boite téléphonique, la redirection d'appel, etc. Une des raisons permettant à ce réseau de fournir une multitude de services est la séparation physique des plans de commutation et de contrôle.

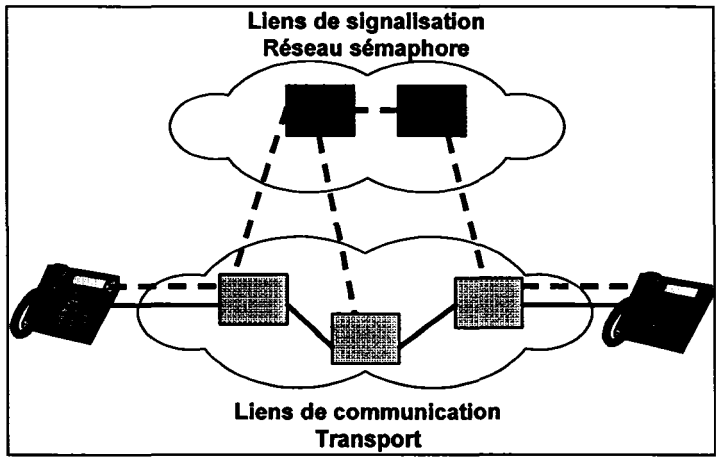


Figure 6 Séparation, plans commutation et contrôle, réseau téléphonique

L'impératif pour les ISP étant maintenant de fournir des services à valeur ajoutée autres que le simple acheminement, la structure des réseaux de nouvelles générations combineront l'omniprésence des réseaux à commutation de paquets IP et l'idée que le réseau doit être plus intelligent à l'aide d'une conception basée sur une structure différenciant acheminement et contrôle. Ces réseaux, en plus de permettre l'offre de nouveaux services, fourniront la quincaillerie nécessaire à la convergence des réseaux à commutation de paquets et des réseaux téléphoniques.

L'architecture des réseaux de nouvelle génération, telle que définie par le *Multiservice Switching Forum* [MSF1, MSF2], l'*International Packet Communications Consortium* [IPCC], l'*International Telecommunication Union* [ITU] ou dans une moindre mesure

par l'*Internet Engineering Task Force* [RFC 3294][RFC 3261][RFC 3465], est constituée de façon à ce que les fonctions d'acheminements et de contrôles puissent être physiquement séparées. Ces architectures ont comme rôle de définir une infrastructure de réseau permettant l'offre de service à valeur ajoutée. Le premier de ces services sera l'offre d'un réseau de téléphonie IP, VoIP (*Voice over IP*), permettant la convergence des réseaux de données et de téléphonie.

Les différents organismes travaillant sur l'architecture des réseaux de nouvelle génération ne définissent que l'architecture globale. Ils n'élaborent aucun nouveau protocole, se limitant à préciser quels protocoles existant pourraient être utilisés pour chaque section de leur architecture.

Dans l'optique des réseaux de nouvelle génération, la technologie MPLS est un excellent candidat au titre de plan de commutation. Comme pour toute technologie de réseau orientée connexion, la séparation des plans d'acheminement et de contrôle est implicite. En effet, les LSP étant créés, la commutation d'étiquettes est la seule fonction nécessaire aux commutateurs MPLS. Le but du plan de contrôle est la sélection du chemin que devra suivre le LSP et la signalisation aux différents commutateurs des informations nécessaires à l'association d'étiquettes entrée-sortie. Pour la sélection du chemin, les protocoles de routage IP peuvent être utilisés. Ce contrôle peut également être physiquement séparé en utilisant le routage par contraintes ou la configuration manuelle. L'ingénierie de trafic permis par la technologie MPLS rend possible un contrôle efficace du plan de commutation. Pour la signalisation des associations entrée-sortie déterminant les chemins, les protocoles RSVP-TE ou CR-LDP peuvent être utilisés.

La technologie IP seule ne fourni pas la quincaillerie nécessaire à l'élaboration d'un service de VoIP à valeur ajoutée. Dans une optique de convergence des réseaux et d'offre d'un service de téléphonie à valeur ajoutée, concurrençant le réseau téléphonique actuel, la téléphonie IP devra être implantée sur un réseau de nouvelle génération.

MPLS étant le principal candidat au titre de plan de commutation, la voix sur MPLS, ou plutôt la voix sur IP sur MPLS est un incontournable.

1.2 Signalisation SIP

L'émergence des applications de téléphonie, et plus largement des communications multimédia sur les réseaux IP, a suscité le besoin de définir un protocole dédié à la signalisation d'appels. De ce besoin est né le protocole SIP [RFC 3261]. Les buts de ce protocole sont la localisation des usagers, le destinataire pouvant ne pas être fixe, ainsi que l'élaboration, la modification et la terminaison de sessions entre usagers. Le protocole SIP se situe dans les hautes couches du modèle OSI et, par conséquent, est indépendant des technologies de réseau.

1.2.1 Description du protocole SIP

Le protocole SIP est basé sur une architecture client serveur : la communication s'établit entre un UAC (*User Agent Client*) situé sur le terminal appelant et un UAS (*User Agent Server*) situé sur le terminal appelé. Chaque partie communicante est identifiée par une adresse de format URL (*Uniform Resource Locator*) qui se présente sous la forme sip://user@domain. L'utilisation d'une adresse de ce type permet à un utilisateur d'être mobile et de changer d'adresse IP au besoin. Si l'appelant connaît l'adresse de l'appelé, il peut le contacter directement. Si l'adresse de l'appelé n'est pas connue, l'adresse URL est utilisée conjointement avec un mandataire SIP (*proxy SIP*) servant d'intermédiaire lorsque l'adresse IP n'est pas connue (Figure 7). Avant que le mandataire SIP puisse servir d'intermédiaire, les utilisateurs doivent s'enregistrer auprès de ce dernier. À l'aide du nom de domaine de l'appelé, l'appelant interroge le DNS (*Domain Name Service*) pour obtenir l'adresse IP du mandataire situé dans le domaine auquel appartient l'appelé. L'appelant transmet ses messages au mandataire SIP de l'appelé qui sert alors de relais en acheminant les messages à l'adresse IP de l'appelé, enregistré au préalable.

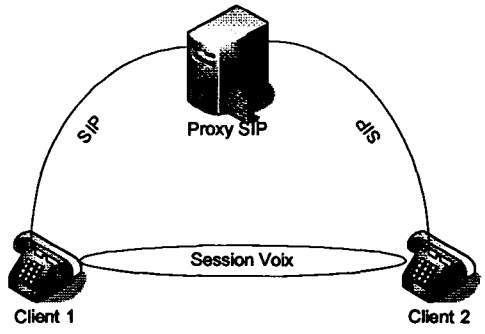


Figure 7 Architecture SIP

Pour implanter la logique décrite précédemment, SIP utilise plusieurs messages de requête, également appelés méthodes et six classes de messages de réponse.

Messages de requête

Les messages de requête requièrent qu'une action particulière soit prise par un autre UA ou par un mandataire SIP. Ces requêtes sont :

- **REGISTER** : Cette requête est utilisée par un client désirant enregistrer son adresse IP auprès du mandataire auquel il est attaché.
- **INVITE** : Cette requête indique que l'utilisateur correspondant à l'URL spécifié est invité à participer à une session. Le corps du message décrit cette session (ex : audio encodé PCM (*Pulse Code Modulation*), suivant loi mu à 8000 échantillons/sec, en utilisant le protocole de transport RTP (*Real Time Protocol*)).
- **ACK** : Cette requête confirme que le terminal appelant a bien reçu une réponse définitive à sa requête INVITE et que la session peut débuter.
- **BYE** : Cette requête est utilisée pour terminer une session préalablement établie.
- **CANCEL** : Cette requête est envoyée par un terminal ou un serveur mandataire afin d'annuler une requête non validée par une réponse finale. Par exemple, si une machine ayant été invitée à participer à une session, et ayant accepté l'invitation ne reçoit pas de requête ACK, alors elle émet une requête CANCEL.
- **OPTIONS** : Cette requête est utilisée pour détecter la disponibilité ou interroger les capacités d'un UAC ou d'un UAS.

Messages de réponse

Un message de réponse est généré par un UAS ou par un serveur mandataire en réponse à une requête transmise par un UAC. Les six classes de réponses sont :

- **1xx Informational** : Cette classe de réponse indique le statut d'un appel avant qu'il ne soit complété.
    - **100 Trying** : Un serveur mandataire peut transmettre cette réponse à l'appelant pour lui faire part qu'il tente de contacter l'appelé.
    - **180 Ringing** : La requête INVITE a été reçue et le UAS est en train de transmettre une alerte de type sonnerie. Si le UAS répond automatiquement, cette réponse n'a pas besoin d'être transmise.
- **2xx Succes :** Cette classe de réponse confirme le succès d'une requête.
    - **200 OK :** Cette réponse est transmise lorsqu'une requête de type INVITE a été acceptée. Ce message est également transmis en réponse aux requêtes REGISTER, BYE, CANCEL,
- **3xx Redirection :** Messages de redirection transmis par le serveur mandataire
    - **300 Multiple choices :** Cette réponse retourne au UAC un message contenant plusieurs localisations possibles pour contacter le terminal demandé.
    - **301 Moved permanently :** Cette réponse contient la nouvelle adresse URL d'un contact. Cette nouvelle adresse est permanente.
    - **302 Moved temporarily :** Cette réponse contient une adresse URL pour le terminal demandé. Cette adresse URL est temporaire.
    - **305 Use Proxy :** Cette réponse retourne l'adresse URL d'un autre serveur mandataire devant être utilisé pour contacter le terminal désiré.
- **4xx Client Error:** Classe retournant un message d'échec de la requête dû à une erreur du client.
    - **400 Bad request :** La requête n'a pas été comprise par le serveur
    - **401 Unauthorized :** La requête pour laquelle cette réponse est reçue nécessite une autorisation du client.

- **403 Forbidden :** La requête a été reçue, formulée correctement et a été comprise par le serveur qui ne rendra pas le service demandé par la requête.

- **5xx Server Error :** Classe retournant un message d'échec de la requête dû à une erreur du serveur.
  - **500 Server Internal Error**
  - **501 Not implemented :** La requête reçue ne peut être exécutée car non supportée par le serveur.
  - **503 Service Unavailable :** La requête demandée ne peut être traitée de façon temporaire.
- **6xx Global Error :** Cette classe de réponses indique que le serveur mandataire sait que la requête sera un échec. La requête ne devrait donc pas être transmise à d'autres localisations. Seul le serveur mandataire responsable de l'usager identifié par le URL peut répondre avec cette classe.
  - **603 Decline :** Indique que l'appelé est occupé ou tout simplement qu'il ne désire pas prendre ses appels.
  - **604 Does not exist Anywhere :** L'usager spécifié dans la requête n'existe pas.
  - **606 Not acceptable :** Cette réponse peut être utilisée pour implanter une négociation. Cette réponse indique que certains aspects de la session ne sont pas acceptables par le UAS appelé.

1.2.2 Serveur mandataire SIP (Proxy SIP)

Le serveur mandataire SIP est utilisé lors de l'établissement d'une session. Pour illustrer ces principales versions et fonctions, quelques exemples seront utilisés. Ces exemples permettront également d'illustrer l'utilisation des différentes requêtes et classes de réponse.

L'appelant connaît l'adresse de l'appelé

À la base, le protocole SIP sert à l'établissement de sessions entre usagers. Dans la mesure où l'appelant connaît l'adresse IP de l'appelé, il peut échanger des messages SIP directement avec ce dernier (Figure 8)

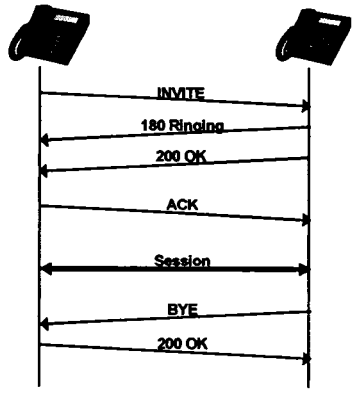


Figure 8 SIP sans serveur mandataire

L'adresse IP étant connue, la requête INVITE est transmise directement au terminal appelé. Le terminal appelé transmet un message de réponse indiquant qu'une sonnerie informe l'appelé de l'appel entrant. Dans un cas ou l'appelé aurait été une application répondant automatiquement, ce message n'aurait pas été transmis. L'appelé répondant à l'appel, le message de succès OK est transmis. Enfin, l'appelant ayant reçu confirmation de l'appelé, il lui transmet une requête ACK pour démarrer la session tel que négocié. La requête BYE demande la terminaison de l'appel, requête confirmée par le message de succès OK.

L'appelant connaît seulement l'URL de l'appelé et le mandataire est de type *stateless*

Dans la majorité des cas, l'appelant ne connaît que l'URL de l'appelé. L'utilisation d'une adresse URL permet de faire abstraction de l'adresse IP qui peut être dynamique. Avant même de pouvoir utiliser un serveur mandataire, il faut que celui-ci connaisse

l'adresse IP des URL dont il a la charge. Pour ce faire, les UA doivent s'enregistrer auprès du mandataire de leur domaine (Figure 9)

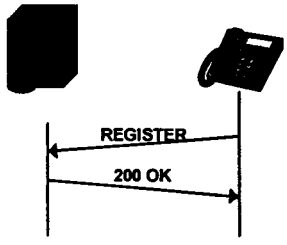


Figure 9 Enregistrement d'un UA au près de son mandataire

Le serveur mandataire étant maintenant au courant de l'adresse IP de l'appelé, lorsqu'un appelant désirera contacter ce client, le mandataire sera en mesure d'acheminer les requêtes à l'adresse IP de l'appelé.

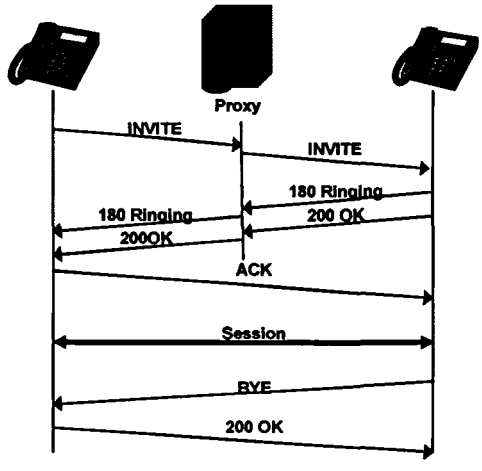


Figure 10 SIP avec mandataire *Stateless*

Dans cet exemple, un serveur mandataire de type *stateless* est utilisé (Figure 10). Un mandataire *stateless* traite chaque requête individuellement. Une fois la requête acheminée ou répondue, aucune information n'est conservée. Un mandataire de ce type n'a aucune mémoire des requêtes ou messages reçus et transmis

Plusieurs adresses IP correspondent à un URL et mandataire *Statefull*

Il est possible pour un usager d'enregistrer plusieurs adresses IP où il est possible de le rejoindre. Par exemple, un usager peut enregistrer l'adresse IP de son téléphone au bureau, de son cellulaire et finalement de son PC. Cet usager peut également assigner une priorité d'appel à ces différentes options.

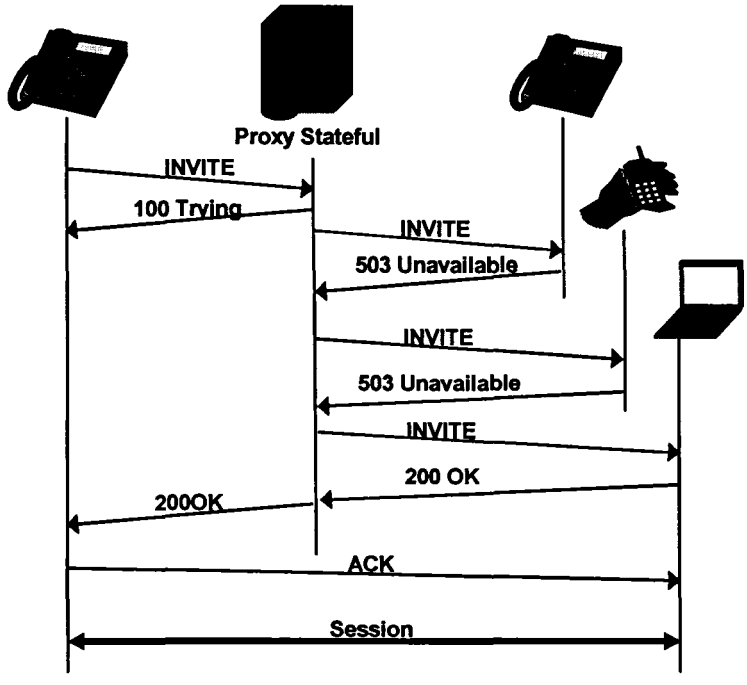


Figure 11 Proxy *statefull* et multiples localisations

Lorsque le serveur mandataire est *Statefull*, il garde trace des requêtes et réponses reçues dans le passé pour traiter les requêtes et réponses à venir. Par exemple, à la Figure 11, il achemine la requête au premier UA inscrit par l'appelé. Recevant la réponse *Unavailable*, il ne transmet par cette réponse à l'appelant mais transmet plutôt une requête à la deuxième adresse. Recevant une réponse positive de la troisième adresse, il achemine cette réponse à l'appelant pour qui le processus a été totalement transparent.

1.2.3 Réseaux VoIP avec gestion et caractéristiques intéressantes de SIP

SIP permet d'élaborer facilement une session entre deux applications en utilisant non seulement le réseau Internet, mais également la philosophie décentralisatrice de ce réseau. Bien que ce protocole ait permis une simplification du processus, si on le compare au protocole H.323, SIP n'est pas nécessairement la solution idéale pour tous les types de téléphonie Internet. Pour une application de téléphonie de relève, à bas prix et soumis aux aléas d'Internet, SIP est excellent. Cependant, pour prétendre au titre de technologie prenant le relais des systèmes téléphoniques conventionnels, la téléphonie Internet doit rencontrer des exigences en termes de qualité de service. La téléphonie IP est d'ailleurs l'application clé qui nécessitera le déploiement de la QoS dans les réseaux IP. C'est ici que la philosophie décentralisatrice de l'Internet blesse. Quel fournisseur de service à intérêt à investir dans le déploiement et la gestion de la QoS sans aucun revenu supplémentaire?

Le déploiement de la téléphonie Internet, en tant que remplaçant de la technologie actuelle passe par un réseau de VoIP avec gestion. La gestion d'un réseau de VoIP implique l'assurance que la qualité de service est fournie, une facturation adéquate des clients, permet d'offrir des services de types PBX (*Private Branch eXchange*) et, très important, une interopérabilité avec le réseau téléphonique PSTN (*Public Switched Telephone Network*). L'utilisation d'une infrastructure avec gestion de services, ici la VoIP, est une nécessité afin que la téléphonie Internet permette la convergence des réseaux voix et données. Cisco, par l'intermédiaire de sa plate-forme *Call Manager* [CiscoCM], ainsi que la plupart des petits et grands fabricants de matériel de télécommunications ont compris cette nécessité et chacun propose sa plateforme de gestion de VoIP.

Pour l'établissement des appels dans une telle architecture, Cisco ainsi que plusieurs manufacturier supportent le protocole SCCP (*Skinny Client Control Protocol*) qui est un protocole de signalisation dédié à la téléphonie IP avec plate-forme de gestion. La complexité se retrouve dans la plate-forme et le coté client du protocole est très simple.

Le protocole SCCP est une propriété de Cisco et aucune référence n'est disponible sur ce protocole. Un autre protocole utilisé pour la signalisation par les plateformes de gestion VoIP est MGCP (*Media Gateway Control Protocol*) [RFC 3661]. MGCP est un protocole utilisé entre un MGC (*Media Gateway Controller*) et des MG (*Media Gateway*) (Figure 12). À l'origine, le but d'un MG était de passer d'un lien téléphonique à un réseau IP et vice-versa.

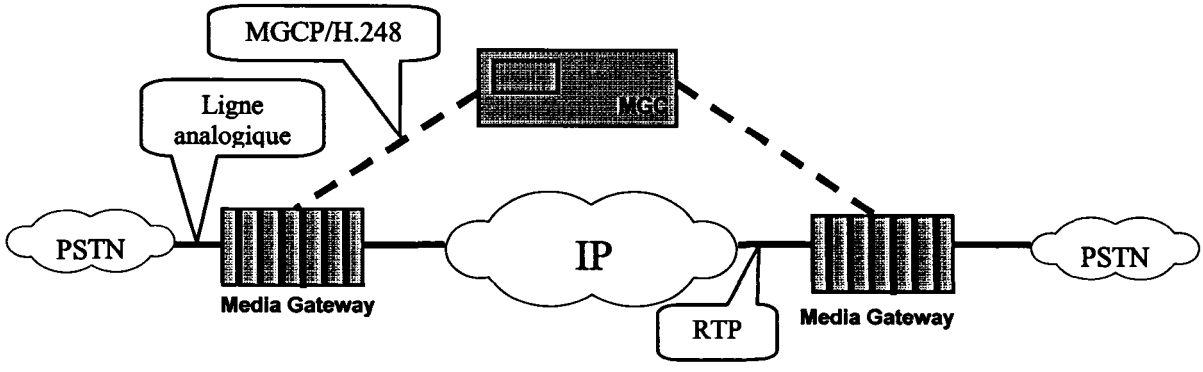


Figure 12 Protocole de signalisation MGCP

MGCP peut être étendu en considérant les téléphones comme des MG, pouvant ainsi servir de protocole de signalisation pour la VoIP, au même titre que SCCP et SIP. En plus de la signalisation, de part son origine, MGCP implante des mécanismes de contrôle d'accès évitant d'acheminer un appel sur la dorsale IP si la capacité de celle-ci n'est pas suffisante ou si le MG de sortie ne possède pas assez de ligne de sortie. MGCP et SCCP sont des protocoles conçus pour qu'un contrôleur central, en l'occurrence un MGC ou un *Call Manager*, contrôle les clients désirant établir un appel téléphonique. Ces deux protocoles ont été conçus spécifiquement pour la téléphonie. Ils sont tout deux de très bon candidat pour un réseau de VoIP avec gestion et la majorité des téléphones IP supportent ces deux protocoles.

Bien que SIP soit conçu dans un esprit opposé, c'est-à-dire décentralisé, il possède des avantages lui permettant d'être considéré comme un candidat pour la signalisation d'appels téléphoniques sur un réseau VoIP avec gestion. Comparativement à SCCP et MGCP, le coté client de SIP est assez complexe. Cette complexité fait en sorte que

MGCP et SCCP sont préférés pour des combinés téléphoniques IP simple. Cependant, nous considérons qu'il s'agit d'une vision à court terme et considérons que la VoIP ne sera pas la seule application profitant de la convergence des réseaux et de l'approche de réseau avec gestion centralisé essentiel à cette application. Les réseaux de nouvelles génération qui sont à mi-chemin entre l'approche décentralisé de l'Internet et l'approche centralisé promettent l'émergence d'une foule d'application nécessitant l'établissement d'une session avec qualité de service. À plus long terme, nous considérons SIP comme étant le protocole permettant la plus grande versatilité. Sans entrer dans les détails nécessaires à l'utilisation de SIP comme protocole de signalisation sur un réseau de VoIP géré, voici tout de même quelques détails démontrant la pertinence d'utiliser SIP sur de telles infrastructures.

À la base, un serveur mandataire SIP ne sert que de relais entre deux entités SIP désirant établir une session, ce relais permettant de déterminer la localisation de l'appelé. Pour utiliser SIP dans un réseau de VoIP avec gestion, le mandataire SIP doit être utilisé comme plate-forme de gestion centralisée établissant le plan de contrôle du réseau VoIP. Le serveur mandataire doit être en mesure d'établir la facturation, d'offrir des services de type PBX, permettre une interconnexion avec le réseau PSTN, assurer la qualité de service et le contrôle d'accès. Les appels doivent donc obligatoirement débuter et se terminer en passant par le/les serveur(s) mandataire(s).

Pour réaliser les conditions nécessaires à l'utilisation de SIP dans une architecture centralisée, SIP possède les caractéristiques suivantes. Premièrement, il est possible d'exiger d'un client qu'il contacte un serveur mandataire en particulier, sans s'occuper de la résolution de nom. Il est également possible d'imposer à un mandataire SIP d'acheminer les requêtes à une certaine adresse pouvant être un autre mandataire SIP. L'option *Record Route* permet d'imposer une route passant par certains mandataires, si désiré. Ces quelques caractéristiques démontrent qu'il est possible d'utiliser le protocole SIP dans une architecture centralisée. La définition d'extension SIP pour un support

adéquat de la VoIP sur un réseau avec gestion est l'objet de plusieurs recherches et pourrait faire l'objet de quantité d'autres projets.

Nous considérons donc SIP comme un excellent candidat au titre de protocole de signalisation pour l'élaboration de sessions applicatives au niveau du plan de contrôle d'un réseau de nouvelle génération.

CHAPITRE 2

ÉTAT DE L'ART

2.1 Architectures SIP sur MPLS

Tel que vu aux sections précédentes, SIP et MPLS sont deux candidats de choix pour une infrastructure de réseau de nouvelle génération supportant un service de VoIP à valeur ajoutée. MPLS supporte la séparation des plans de contrôle et de commutation, l'ingénierie de trafic et la qualité de service, concepts essentiels à une infrastructure de transport d'un réseau de nouvelle génération. SIP permet l'établissement de sessions entre applications utilisant les services de ce réseau de nouvelle génération.

L'utilisation de SIP dans une infrastructure de type centralisé a d'abord été abordée pour le support de la qualité de service pour les applications utilisant SIP. [Salsano] utilise cette approche en utilisant un mandataire SIP appelé *QSIP Server*. Une station contacte un serveur QSIP qui interagit avec d'autres serveurs QSIP à l'aide du protocole QSIP. Ces serveurs contrôlent des points d'accès de qualité de service et les appels passent par ces points d'accès à partir desquels la qualité de service est assurée. La gestion du domaine *Diffserv* est réalisée à l'aide du protocole COPS (*Common Open Policy Service protocol*) [RFC 2748], ce qui permet une configuration dynamique des paramètres de configuration *Diffserv* des routeurs du domaine.

Par la suite, [Zhang] a étendu le modèle précédant en y incluant la technologie MPLS. La différence provient du fait que COPS est utilisé par les serveurs SIP, appelé *TE-SIP server*, afin d'informer les routeurs MPLS de bordure, les LER, qu'un tunnel est nécessaire pour l'établissement d'un appel avec qualité de service. Leur approche modifie les points d'accès de QoS par des routeurs LER, le domaine *Diffserv* par un domaine MPLS, le rôle de COPS et QSIP est conservé.

Cette approche, SIP et MPLS pour le support de la QoS, souffre d'un problème particulier. L'établissement d'un LSP par appel lors de la réception d'un paquet INVITE implique des délais d'établissement d'appel relativement longs en plus de ne pas pouvoir supportée une mise à l'échelle. Pour cette raison, [Bo] propose une architecture, selon laquelle, les LSP sont créés à l'avance. Un serveur mandataire SIP amélioré, appelé *TA-Server* maintient un lien étroit avec les réservations établies et est en mesure d'accepter ou de refuser l'établissement d'un appel en se basant sur la capacité disponible.

L'architecture proposée par [Salsano][Zhang][Bo] est également proposée dans plusieurs travaux portant sur la qualité de service et le contrôle d'admission. Déjà en 1998, [Doshi] proposait d'utiliser des *Virtual Provisionning Server* réalisant un contrôle d'admission et la gestion des ressources sur des circuits préalablement réservés entre des *Packet Circuit Gateway* afin de garantir une qualité de service. Cette architecture est également à la base d'un type de contrôle d'admission appelé *Per Call Reservation with Path-based Bandwidth Allocation*[Lucent]. Selon cette approche, des *Virtual Trunk Group*, chemin ayant une capacité préalablement réservée, définissent la limite du nombre d'appels pouvant être accepté par le CAC.

L'architecture dont il sera question est également à la base des réseaux de nouvelle génération. Ces réseaux sont constitués de nœuds de services, situés en périphérie d'une infrastructure de réseau dédié, offrant des services à valeur ajouté aux clients dont le trafic aura à transiter par le nœud de services et le réseau dédié.

2.2 Contrôle d'admission d'appel

Le contrôle d'admission d'appel (*Call Admission Control CAC*) est une fonction nécessaire à tous réseaux supportant la qualité de service. Supposons un réseau sur lequel 256kbps de bande passante est réservée pour la qualité de service EF associée aux appels téléphoniques. Si quatre appels de 64kbps sont en transit, cette réservation est

adéquate. Que se passe-t-il si un cinquième appel utilisant la même qualité de service est initié. 320kbps devront se partager la bande passante et les tampons calibrés pour recevoir 256kbps maximum, diminuant ainsi la qualité de cette classe de service. Pour s'assurer de ne pas diminuer la qualité d'une classe de service, il faut s'assurer qu'il n'y transite pas plus de trafic que ce qui est autorisé. L'établissement du cinquième appel aurait dû être refusé par un contrôle d'admission.

2.2.1 Type de contrôle d'admission

Les mécanismes de contrôle d'admission peuvent se diviser en trois catégories principales. Le premier type de contrôle d'admission, appelé *Local-based CAC,* fonctionne sur la passerelle de sortie du réseau et est basé sur de l'information locale. Par exemple, il peut s'agir d'une limitation en terme de circuits DS0. Il peut également s'agir d'une configuration statique limitant le nombre de connexion de type VoIP. La plate-forme *Call Manager* de Cisco utilise cette approche, appelée *Location-based CAC.* Par exemple, pour un appel à destination local, n'importe qui peut appeler n'importe quand avec n'importe quel codec. Pour les appels à destination extérieure, la commande *max-conn* permet de limiter le nombre d'appels autorisés en fonction de la destination. Ce type de contrôle d'admission ne tient pas compte de l'état du réseau et est statique.

Afin de tenir compte de l'état du réseau, les autres types de contrôle d'admission doivent utiliser une stratégie leur permettant de connaître l'état du réseau. Le second type de contrôle d'admission, appelé *Ressource-Based CAC*, détermine les ressources nécessaires et vérifie la disponibilité de celles-ci. Ce type de contrôle d'accès calcule les ressources disponibles à l'aide d'une architecture complexe (*Calculation-Based*) et sur le calcul de valeurs limites qui affecteraient la qualité de service. L'appel est accepté si l'utilisation des ressources en résultant ne dépasse pas les valeurs limites qui ont été calculées. Dans la catégorie *Ressource-Based* se trouve également les contrôles d'accès de type *Reservation-Based.* RSVP et ATM sont deux technologies supportant ce type de contrôle d'admission. Bien que ce type de contrôle d'accès soit le seul permettant de

garantir une qualité de service jusqu'à la fin d'un appel, elle peut difficilement être mise à l'échelle due à la multitude d'états générés par chaque appel.

Finalement, le dernier type de contrôle d'admission reconnu est le type *Measurement-Based CAC*. Ces mécanismes de contrôle d'accès obtiennent de l'information relative au réseau en étudiant la transmission de paquets. Des paquets sondes sont transmis à la destination, celle-ci retourne les sondes avec des informations tel que le taux de perte, le délai et la variance. Sur la base des ces informations, un calcul de moyenne, d'écart-type ou un algorithme complexe de prédiction permet d'accepter ou refuser un nouvel appel. Bien que ce type de contrôle d'admission tienne compte de l'état du réseau, il ne permet pas de garantir la qualité de service pour la durée de l'appel.

2.2.2 Contrôle d'admission par gestion de ressources

Considérant un réseau pour lequel la réservation de ressources est impossible, l'approche *Measurement-Based* est la plus appropriée. Pour un réseau supportant une architecture de qualité de service de type *Intserv* ou ATM, l'approche *Per Call Reservation-Based* est implicite. L'approche *Local-Based* est limitée et ne permet pas de tenir compte des ressources du réseau. Pour un réseau supportant une architecture de qualité de service *Diffserv*, des réservations sont établies, mais le concept de flot étant inexistant et le chemin suivi par les paquets pouvant être modifié, seule l'approche *Measurement-Based* est adéquate.

Qu'en est-il si le réseau est de type MPLS supportant l'architecture *Diffserv*? Les réservations, en plus de s'appliquer à une classe de service, peuvent également s'appliquer à un LSP en particulier. Les paquets suivent un chemin dédié et la sélection du trafic pouvant entrer dans un LSP représente un flot de donnée similaire à ceux utilisés par les réseaux *Intserv*. En fait, l'utilisation d'un réseau MPLS permet de combiner les avantages de *Intserv* et de *Diffserv*, en simplifiant la gestion des ressources tout en assurant la qualité de service tout au long de l'appel.

Pour un réseau où il est possible de maintenir un état d'utilisation par lien est de conserver le trafic sur un chemin déterminé, tel MPLS, deux nouvelles approches de contrôle d'admission existent, il s'agit des types *Per Call Reservation with Path-based Bandwidth Allocation* et *Link Bandwidth Reservation with Measurment-based CAC*. Il s'agit de types combinant les types de base que sont les contrôles d'admission *Ressource-based(Reservation-based)* et *Measurment-based.*

Per Call Reservation with Path-based Bandwidth Allocation

Pour réaliser ce type de contrôle d'admission, de la bande passante est réservée pour différents chemins, situés entre deux routeurs de bordure, afin de garantir la qualité de service. Un gestionnaire s'occupe de noter le nombre d'appels, ou la bande passante équivalente, passant par chaque chemin. Ce gestionnaire réalise un contrôle d'accès qui refuse un appel lorsque celui-ci doit passer par un chemin pour lequel il n'y a plus de ressource disponible. Ce type de contrôle d'accès possède plusieurs avantages. Il peut être distribué entre plusieurs gestionnaires, requiert peu de connaissance de l'état du réseau (seulement les réservations établies). Un contrôle d'accès de ce type s'accorde parfaitement avec des technologies de réseau orientées connexion, chaque connexion étant un chemin pour laquelle une bande passante peut-être réservée pour plusieurs appels. Le nombre de chemin progresse en n^2 du nombre de routeurs de bordure mais une architecture hiérarchique peut être définie en utilisant de multiples décisions de contrôle d'accès (ex : chemin d'entrés, chemin de transport, chemin de sortie, chacun avec un contrôle d'accès devant être positif pour que l'appel soit accepté.).

Link Bandwidth Reservation with Measurment-based CAC

Cette méthode établit des chemins, mais ne réserve aucune ressource pour ces chemins. Une classe de service Diffserv est réservée sur chaque lien et tous les chemins passant par un même lien partage la bande passante allouée à la voix. Cela évite d'avoir à établir une réservation pour chaque chemin, cependant, la bande passante utilisée sur chaque lien doit être surveillée Pour un lien donné, si un niveau de congestion est

détecté pour la classe de service voix, les gestionnaires responsables du contrôle d'accès pour les chemins passants par ce lien doivent être informés afin de refuser un nouvel appel qui aurait à passer par ce chemin. Il est également possible que ces gestionnaires refusent x% des nouveaux appels devant passer par ce chemin afin de prévenir l'état de congestion. [Houck] et [Houck2] décrivent ce modèle et en présentent plusieurs résultats.

2.2.3 Per Call Reservation with Dynamic Path-based Bandwidth Allocation

Basé sur une architecture de réseau permettant un contrôle d'accès de type *Per Call Reservation with Path-based Bandwidth Allocation,* [Bo] décrit un algorithme permettant de faire ce contrôle d'accès, basé sur la disponibilité de ressources. En plus, son algorithme permet d'utiliser les connaissances apportées par le contrôle d'accès pour déterminer si la quantité de ressources réservées est suffisante pour le patron de trafic à un moment donné (*Measurement-Based*). En fonction de ce qui a été mesuré, les réservations de ressources sont mises à jour dynamiquement, à la hausse ou à la baisse afin d'optimiser les ressources du réseau (*Reservation-Based*). Ainsi, la combinaison du contrôle d'accès et la gestion dynamique des ressources débouchent sur une nouveau type de contrôle d'admission appelé *Per Call Reservation with Path-based Bandwidth Allocation.*

C'est dans cet optique que nous optimiserons l'utilisation de l'algorithme défini par [Bo] et que nous suggèrerons un nouvel algorithme ayant le même objectif.

CHAPITRE 3

RÉSEAUX DE SERVICES

Le précédant chapitre a introduit différentes technologies de première importance pour les réseaux de nouvelle génération. L'objectif de ces réseaux NGN (*Next Generation Network*) est l'offre de services à valeurs ajoutées. Tel que discuté à la section 1.2.3, le déploiement de téléphonie IP avec qualité de service passe par un réseau de VoIP avec gestion centralisée. L'offre d'un service de téléphonie IP avec qualité de service et gestion centralisée est le prototype même d'un service à valeur ajoutée d'un réseau de nouvelle génération.

Pour commencer, ce troisième chapitre fera la description d'une architecture de réseaux de nouvelle génération dont l'objectif est d'offrir un service de téléphonie IP avec qualité de service et gestion centralisée. Pour garantir la qualité de service, le gestionnaire du service de téléphonie utilisera un algorithme de contrôle d'accès avec évaluation dynamique des besoins, dont l'étude est l'objectif de ce projet. Pour mener à bien les simulations, une modélisation adéquate est élaborée à l'aide du logiciel OPNET. Une dorsale MPLS est configurée en tenant compte des possibilités offertes par le logiciel. Les différentes options de configuration d'un réseau MPLS sont expliquées et la configuration retenue est détaillée. Les options de configuration de la signalisation SIP sont également expliquées et la configuration retenue détaillée. Finalement, les configurations nécessaires à la génération du trafic de voix sont décrites.

3.1 Architecture proposée

Pour que le réseau puisse offrir un service de téléphonie IP avec qualité de service, il faut que l'application téléphonie puisse avoir une certaine relation avec la gestion des ressources du réseau. Pour ce faire, un mandataire SIP, faisant une gestion centralisée des appels, est intégré dans un SC (*Services Controller*). Ce gestionnaire de services est

en interaction avec le réseau MPLS et il est responsable de la gestion des ressources allouées aux différents services. Pour le service de téléphonie IP, un VSC (*Voice Service Controller*) utilise un contrôle d'accès acceptant ou refusant l'établissement d'une session SIP en fonction des ressources disponibles. Ce VSC doit également évaluer dynamiquement les besoins en ressources pour le service de téléphonie. Au besoin, le VSC informe le SC de la nécessité de modifier, à la hausse ou à la baisse, les ressources du réseau de service allouées au support de l'application de voix (Figure 13).

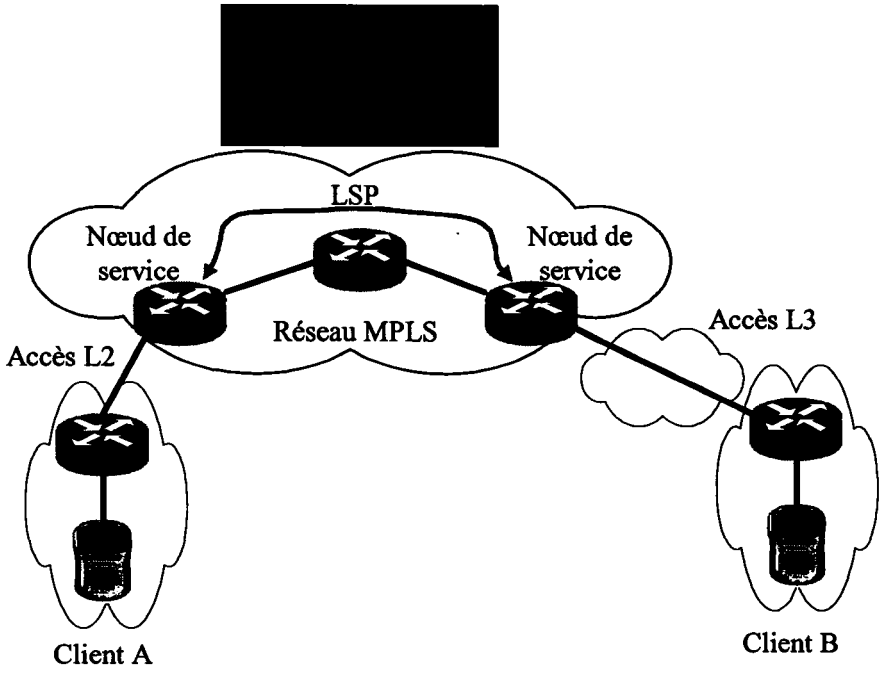


Figure 13 Architecture proposée

Notre architecture se base sur les développements successifs des modèles développés par [Salsano][Zhang][Bo]. Notre équivalent des *QSIP Server*, *TA-Server* ou *TE-SIP Server* ne représente qu'un groupe de fonctions d'un SC. Ce groupe de fonctions, appelé VSC est en fait un serveur SIP *statefull* adapté à une plate-forme de gestion centralisé, un contrôle d'accès basé sur la disponibilité des ressources et un algorithme d'évaluation des besoins permettant de déterminer si la quantité de ressources réservées est adéquate.

L'établissement d'un appel se déroule comme suit. Le client désirant établir un appel transmet une requête au SC local. Le SC local associe la requête au service de téléphonie et remet la requête au VSC. Le VSC s'assure que les ressources sont disponibles entre le client et le nœud de service. Si les ressources sont suffisantes, les fonctions de mandataire SIP déterminent la localisation du destinataire. Le destinataire étant trouvé, le VSC valide que les ressources sont suffisantes entre le destinataire et le nœud de service du client appelé. L'accès d'un client au réseau de services du fournisseur peut être de type L2 ou L3, ce qui implique une gestion de QoS pour différentes technologies. Si les deux clients possèdent un accès adéquat, le VSC s'assure qu'un LSP adéquat (bande passante, QoS) est disponible entre les deux nœuds de service. S'il est possible d'établir un appel avec qualité de service de bout en bout, le serveur SIP achemine la requête au destinataire.

Le SC est responsable de la gestion des ressources des différents équipements du réseau. Cela peut se faire par une interrogation directe des équipements ou par la gestion d'une base d'informations contenant les ressources disponibles sur le réseau. De cette façon, la décision relative à la disponibilité des ressources est rapide. Le SC doit maintenir cette base d'informations. Si le VSC autorise un appel, le SC doit diminuer d'autant la quantité de ressources disponibles. Lorsqu'un appel se termine, le VSC en informe le SC afin que les ressources libérées soient considérées comme disponible. Au besoin le SC peut interroger les équipements pour s'assurer de la conformité de sa base d'informations.

Tableau I

Exemple de base d'informations sur la QoS

Besoins	Accès L2 A	Accès L3 B	LSP A-B (1)	LSP A-B (2)
64Kbps EF	256Kbps CBR	128Kbps Gold	32Kbps EF	128Kbps EF

Dans l'exemple ci-dessus la base d'informations utilisée par le SC permet de déterminer que la bande passante et la qualité de service disponibles sur les accès des clients A et B sont suffisantes et que l'appel peut être acheminé par le LSP (2).

Une autre fonction du SC est la gestion des réservations ou configuration des ressources associées aux différentes classes de service. Si un lien possède 256Kbps de bande passante pour la qualité de service réservée aux appels téléphoniques et que les besoins augmentent, si la bande passante réservée n'est pas augmentée, cela peut mené à un taux de rejet d'appel élevé. Si celle-ci est trop élevée, cela impose une sous optimisation des ressources du réseau. Le SC doit donc être en mesure de configuré dynamiquement ces réservations afin d'optimiser l'utilisation des ressources du réseau. Ces réservations, ou mise à jour de réservations peuvent être réalisé avec tout protocole de réservation, tel CR-LDP, RSVP-TE pour la dorsale MPLS, PNNI s'il s'agit d'un réseau ATM ou RSVP dans le cas d'un réseau d'accès IP (figure 13). Il pourrait également s'agir du protocole COPS comme le suggère [Salsano]. L'évaluation des besoins en ressources est réalisée par une des fonctions du VSC. Cette fonction sera l'objet d'une étude plus approfondie puisqu'elle implante les algorithmes d'évaluation des besoins basés sur le contrôle d'accès qui est l'objet du chapitre 4 et 5.

3.1.1 L'évaluation des ressources par contrôle d'admission

L'algorithme de contrôle d'accès et d'évaluation des besoins, présenté par [Bo], à quelques détails prêt, peut être utilisé directement. Comme le contrôle d'admission se fera pour une classe de service, pour un type de trafic, en l'occurrence des appels téléphoniques, son algorithme doit être adapté pour le contrôle d'admission d'un seul type de trafic. Ensuite, il faut appliquer son algorithme au chemin sur lequel transitera l'appel. Il ne s'agit donc pas de planifier les ressources locales nécessaires et d'accepter ou refuser un appel sur la base des ressources locales mais bien d'appliquer cet algorithme sur toutes les sections du chemin entre la source et la destination avant de prendre une décision.

3.2 Modélisation d'un réseau MPLS

La première étape dans la modélisation d'un réseau est sa schématisation. Pour ce faire, différents nœuds et liens dédiés à la technologie désirée se retrouvent dans une palette d'objets (Figure 14). Pour la schématisation de notre réseau MPLS, la palette contient les nœuds LER et LSR. La palette contient également des liens physiques servant à relier ces nœuds et des liens virtuels LSP qui servent à la définition des chemins que doivent suivre les LSP configurés. Finalement, la palette contient un objet de configuration globale pour la technologie MPLS.

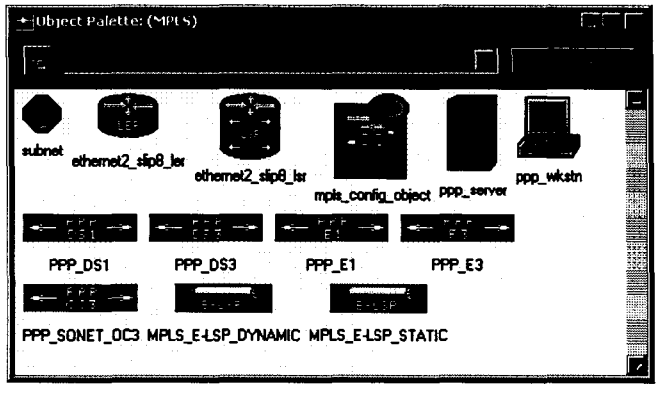


Figure 14 Palette MPLS

Des LER ainsi que des LSR forment une dorsale de réseau MPLS. En périphérie de cette dorsale sont disposés deux sous réseaux qui contiendront des usagers utilisant la signalisation SIP ainsi qu'un sous réseau qui contiendra un SC nommé *Call Manager* puisque qu'il est responsable uniquement d'un service de téléphonie. Les commutateurs MPLS sont interconnectés à l'aide de liens OC-3 et les sous réseaux reliés à l'aide de liens DS3 (Figure 15).

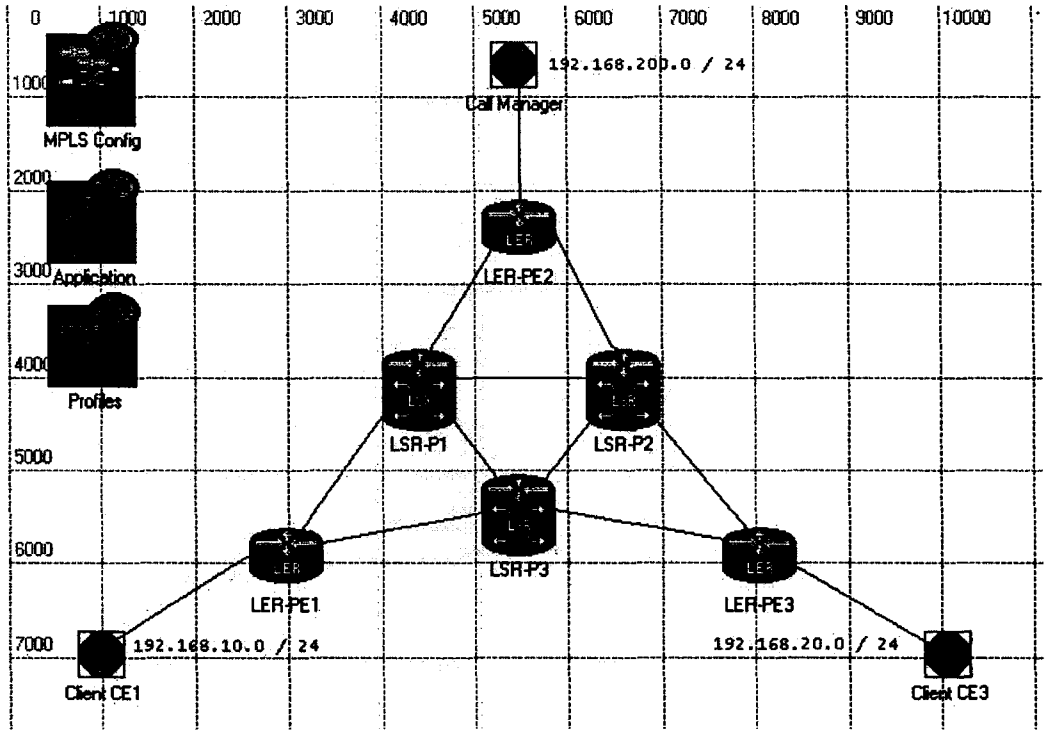


Figure 15 Topologie du réseau MPLS

Une fois le réseau schématisé, un objet *MPLS configuration* est ajouté afin de configurer certaines options générales.

3.3 Configuration d'un réseau MPLS

Une fois le réseau MPLS schématisé, il faut le configurer de telle sorte qu'il soit opérationnel selon des critères sélectionnés. Les étapes de configuration d'un réseau MPLS sont : la définition des FEC, la définition des *Trunks*, la définition des LSP et la configuration des commutateurs MPLS.

3.3.1 Définition des FEC

Une FEC se compose d'une ou plusieurs entrées permettant de spécifier un trafic. Pour chaque entrée, les adresses IP source et destination, les ports de transport source et destination, le type de protocole transporté ainsi que la valeur du champ ToS peuvent

être utilisés pour cette spécification. Un LER recevant un paquet correspondant à la définition d'une des entrées d'une FEC acheminera ce paquet sur le LSP correspondant à cette FEC. Pour configurer les FEC qui seront utilisées dans notre simulation, il faut éditer l'attribut *FEC Specification* de l'objet *MPLS Configuration* (Figure 16).

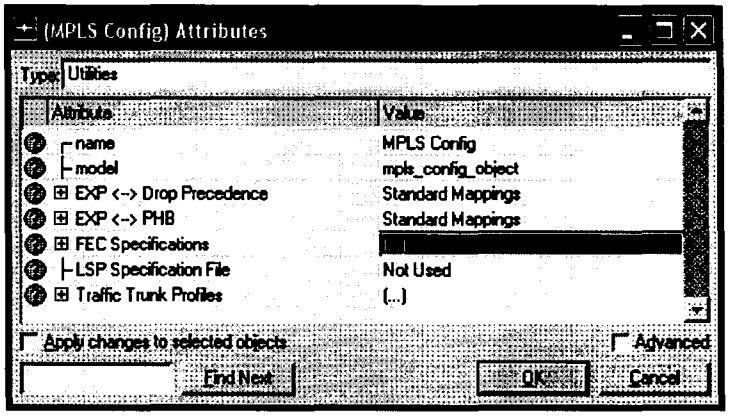


Figure 16 Attributs de *MPLS Configuration*

Dans la fenêtre de spécification des FEC (Figure 17), il faut choisir le nombre de FEC que l'on désire configurer afin que le nombre de lignes approprié soit disponible.

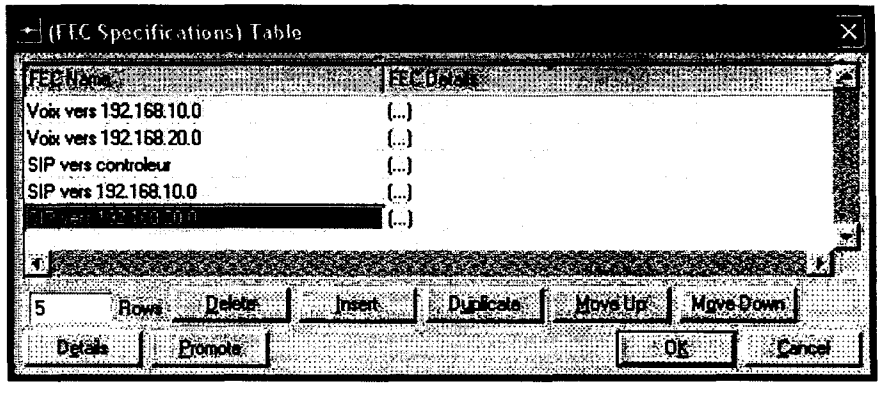


Figure 17 Table définissant les FEC

Pour chaque FEC, la colonne *FEC Name* permet de configurer un nom explicite du trafic qui sera spécifié. L'option *Edit* de la colonne *FEC details* permet de configurer les spécifications du trafic recherché par cette FEC.

La figure suivante présente un exemple des différentes options permettant de caractériser le trafic recherché par la FEC (Figure 18).

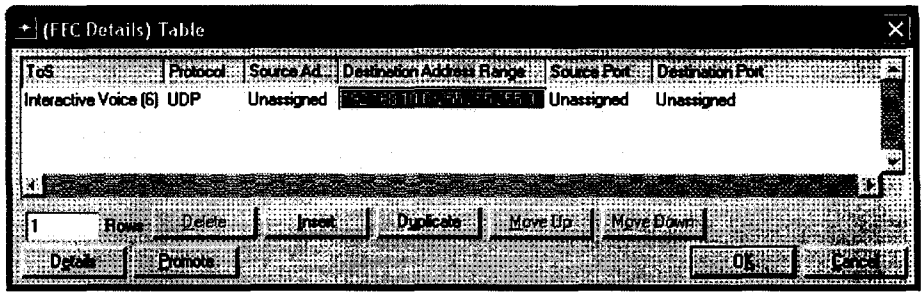


Figure 18 Table de définition d'une FEC

La valeur du champ ToS (Figure 19)

Le champ ToS d'un paquet IP est constitué de 8 bits. La valeur de ce champ peut être configurée selon l'approche, telle que définie dans la spécification IP d'origine en utilisant quatre paramètres [RFC 791]. Ces paramètres sont *Delay*, *Throughput*, *Reliability* et *Precedence*. Le paramètre *Precedence*, définissant l'importance du datagramme, peut prendre les 8 valeurs suivantes : (0) *Best Effort*, (1) *Background*, (2) *Standard*, (3) *Excellent Effort*, (4) *Streaming Multimedia*, (5) *Interactive Multimedia*, (6) *Interactive Voice* et (7) *Reserved.*

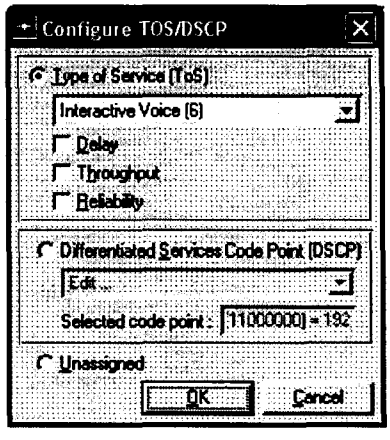


Figure 19 Configuration de la valeur que doit avoir le champ ToS

Le champ ToS des paquets IP peut également être configuré avec l'approche DSCP (*Differenciated Services Code Point*) [RFC 2474] utilisée avec les réseaux IP supportant l'architecture de qualité de service *Diffserv*. Dans ce cas, le champ ToS prend une des valeurs définies pour les différentes classes de service de cette architecture. La valeur de ce champ peut alors être : *Expedited Forwarding* (EF), *Assured Forwarding* (AF1, AF2, AF3, AF4). Si la classe de service du paquet est AF1 à AF4, il s'ajoute une valeur définissant la priorité d'être supprimé en cas de besoin : AFx1 pour les paquets ayant priorité de ne pas être supprimés, AFx2 pour les paquets pouvant être supprimés au besoin, AFx3 pour les paquets de la classe x devant être supprimés en premier.

La valeur du champ Protocole

Il est possible de définir la valeur de champ protocole que doit avoir un paquet pour faire partie de la caractérisation de trafic de la FEC. La valeur de ce champ peut être : TCP, UDP, OSPF, IGRP, EIGRP, ICMP ou toute valeur numérique représentant le type de donnée transporté dans le paquet IP.

Les adresses IP source et destination

Cette contrainte permet de définir l'adresse d'origine et/ou de destination des paquets caractérisés par la FEC. Il est possible d'utiliser une adresse IP unique ou une adresse de réseau assortie d'un masque.

Les ports de transport source et destination

Cette contrainte permet de définir l'application ayant généré le trafic et celle à qui est destiné le trafic à l'aide des ports de transports. La valeur de ce champ peut être : *Custom*, *Database*, *Email*, *Http*, *Ftp*, *Remote Login*, *X Windows*, *Video Conferencing*, *Print*, *Voice* ou toute valeur numérique représentant un port de source ou de destination.

FEC configurées pour notre simulation

Pour notre simulation, nous avons configuré une FEC caractérisant le trafic de type voix à destination du réseau 192.168.10.0 /24 et une autre pour la voix à destination du réseau 192.168.20.0 /24. Pour les données SIP, une FEC détermine le trafic à destination du SC situé à l'adresse 192.168.200.10 et deux autres FEC permettent de séparer le trafic SIP provenant de ce contrôleur à destination des réseaux 192.168.10.0 /24 et 192.168.20.0 /24. Pour configurer ces FEC, les attributs suivants ont été spécifiés : l'adresse de destination (192.168.10.0 /24, 192.168.20.0 /24 ou 192.168.200.0 /24), la valeur du champ ToS (*Interactive Voice* 6) ainsi que le protocole de transport. La voix est transportée par UDP et la signalisation par TCP, c'est la définition du protocole de transport qui permetta de distinguer le trafic voix de la signalisation SIP puisque Opnet utilise la même valeur de ToS, que ce soit des paquets de voix ou des paquets SIP.

3.3.2 Définition des Trafic Trunk

Un *Traffic trunk* ne fait pas partie des fondements de la technologie MPLS. Dans un réseau MPLS sans ingénierie de trafic, les paquets caractérisés par une FEC suivent le LSP correspondant. Le *Trafic Trunk* est un concept relié à l'ingénierie de trafic. Pour déplacer le trafic là où il y a de la bande passante, la FEC n'associe par le trafic à un autre LSP, c'est le *Trafic Trunk* qui est associé à un autre LSP. Lorsque l'ingénierie de trafic est utilisé, la FEC associe un trafic à un *Trafic Trunk* qui est lui-même associé à un ou plusieurs LSP. OPNET impose l'utilisation des *Trafic Trunk*. Pour configurer ceux qui seront utilisés dans notre simulation, il faut éditer l'attribut *Traffic Trunk Profiles* de l'objet *MPLS Configuration*.

Dans la fenêtre de configuration des profils de *Trunk* (Figure 20) il faut choisir le nombre de profils que l'on désire configurer afin que le nombre de lignes approprié soit disponible. La définition d'un profil peut servir à la création de plusieurs *Trunk*.

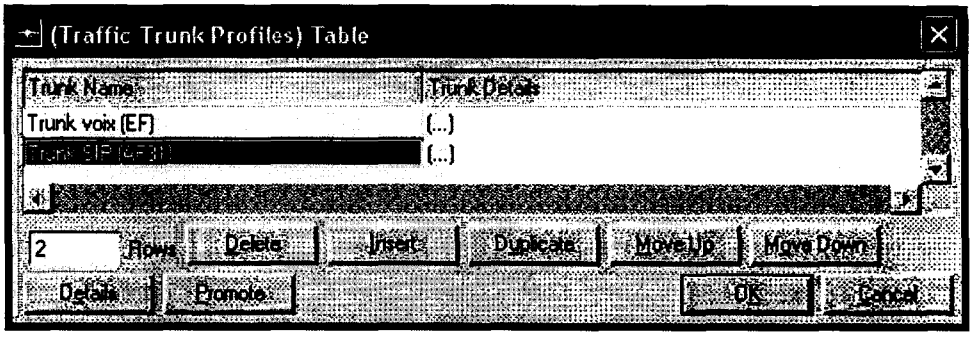


Figure 20 Table des profils de *Trunks*

Pour chaque profil, la colonne *Trunk Name* permet de configurer un nom explicite. L'option *Edit* de la colonne *Trunk details* permet de configurer les caractéristiques qu'auront les *Trunks* utilisant ce profil. La configuration de chaque profil est définie par les paramètres de la fenêtre suivante (Figure 21).

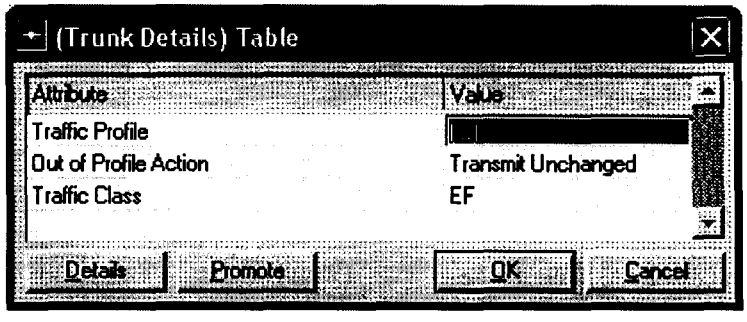


Figure 21 Table de configuration d'un Trunk

Profil du trafic

Le profil de trafic permet de caractériser le flot de bits transitant dans le *Trunk*. Le nombre de bits maximum transmis par seconde, le nombre de bits moyen par seconde et le nombre de bits maximum par salve sont les contraintes caractérisant le flot pouvant emprunter ce *Trunk*.

Dépassement de contraintes

Le dépassement des contraintes survient lorsque le flot de bits entrant dans le *Trunk* dépasse les contraintes spécifiées par le profil de trafic. S'il y a dépassement, cet attribut détermine la politique de *Policing*, c'est à dire l'action devant être posée par le réseau sur le trafic excédentaire. L'action appropriée peut être de ne rien faire, de marquer le trafic avec une priorité moindre ou simplement de supprimer le trafic excédentaire.

Classe de trafic

Un *Trunk* est un agrégat de flots ayant une même classe de service. Cet attribut définit la classe de service *Diffserv* caractérisant ce *Trunk*. Les paquets associés à un *Trunk X* auront une étiquette dont le champ EXP est marqué en fonction de la classe de service de ce *Trunk*. Les routeurs MPLS utilisent la valeur de ce champ pour déterminer la qualité de service à appliquer.

Profils de *trunk* configurés pour notre simulation

Pour notre simulation, nous avons configuré un profil de *Trunk* pour le trafic voix et un autre pour la signalisation SIP. Le profil de *Trunk* associé à la voix est caractérisé par un débit maximum et moyen quelconque. En effet sa capacité devra varier en fonction des besoins déterminés par la prédiction des besoins en bande passante déterminée par la fonction de contrôle d'admission. Le trafic excédentaire ne sera pas supprimé et la classe de service du *Trunk* est *Expedited Forwarding* afin d'assurer la bande passante désirée et de minimiser le délai et la gigue des paquets de voix. Pour la signalisation SIP, un *Trunk* différent sera utilisé afin de marquer ce trafic avec une classe de service différente. Ce *Trunk* aura un débit maximum et moyen de 64000 bps. Le trafic excédentaire ne sera pas supprimé et la classe de service sera *Assured Forwarding* 31. La valeur du champ DSCP spécifiant la classe de service EF associée aux paquets de voix (DSCP 46 : **101** 110) et celle spécifiant la classe de service AF31 associée aux paquets de signalisation (DSCP 26 : **011** 010) sont sélectionnées pour assurer une

compatibilité avec les valeurs de *Precedence* utilisées pour les paquets de voix (*Precedence* = 5, ToS = **101** xxx) et de signalisation (*Precedence* = 3, ToS = **011** xxx).

3.3.3 Autres attributs MPLS généraux

En plus de la définition des FEC et des *Trunk*, l'icône de configuration générale est utilisée pour configurer des paramètres reliés à la qualité de service. Pour le support de *Diffserv*, le champ DSCP de l'en-tête IP possède 6 bits servant à la définition de la classe de service et de la priorité de rejet associé au paquet. Sur une dorsale MPLS, l'en-tête IP n'est pas regardé ; l'examen se limite à l'étiquette MPLS. Pour cette raison, deux types de LSP existent. Le premier type, L-LSP (*Label-LSP*), ne permet qu'une classe de service. Cette classe est déterminée par chaque commutateur à partir de la valeur de l'étiquette. Dans ce cas, le champ EXP de l'en-tête MPLS est utilisé pour spécifier la priorité de rejet. Le deuxième type, E-LSP (*EXP-LSP*), permet d'acheminer jusqu'à 8 classes de service et priorités de rejet. Pour ce faire, la valeur du champ EXP spécifie le PHB (*Per Hop Behavior*), c'est à dire la classe de service.

Les attributs *EXP* ←→ *Drop precedence* et *EXP* ←→ *PHB* de l'icône de configuration générale spécifient la définition de chaque valeur du champ EXP. Ces valeurs sont configurables selon les besoins de l'administrateur de réseau. La configuration par défaut utilise un *mapping* standard tel que défini au tableau II.

Tableau II

Mapping EXP ←→ PHB Standard

EXP←→PHB		EXP←→Drop
EXP	PHB	Drop
0	AF11	AF11, AF21, AF31
1	AF21	AF11, AF21, AF31
2	AF22	AF11, AF21, AF31
3	AF31	AF12, AF22, AF32
4	AF32	AF12, AF22, AF32
5	AF41	AF12, AF22, AF32
6	EF	AF13, AF23, AF33
7	EF	AF13, AF23, AF33

Les *Trunk* configurés pour la voix et SIP utilisent les classes de service EF et AF31. Pour que ces *Trunk* puissent utiliser un même LSP, ce LSP devra être de type E-LSP. Les paquets transportant de la voix auront un champ EXP égale à 6 et ceux transportant de la signalisation égale à 3.

3.3.4 Définition des LSP

La palette d'objets MPLS contient deux types de LSP. Le premier est dynamique et le second statique. Un LSP statique est un LSP qui suit exactement le chemin schématisé à l'aide de l'objet LSP statique. Cet objet possède un seul attribut de configuration, il s'agit du chemin qu'il doit suivre. Un LSP dynamique est créé à l'aide d'un protocole d'échange d'étiquettes tel CR-LDP ou RSVP-TE. Le schéma élaboré par un objet de type LSP dynamique ne définit que la source et la destination et peut obliger le LSP à passer par un commutateur en particulier. Cet objet possède plusieurs attributs de configuration permettant de définir la façon dont le chemin suivi par le LSP sera déterminé (Figure 22).

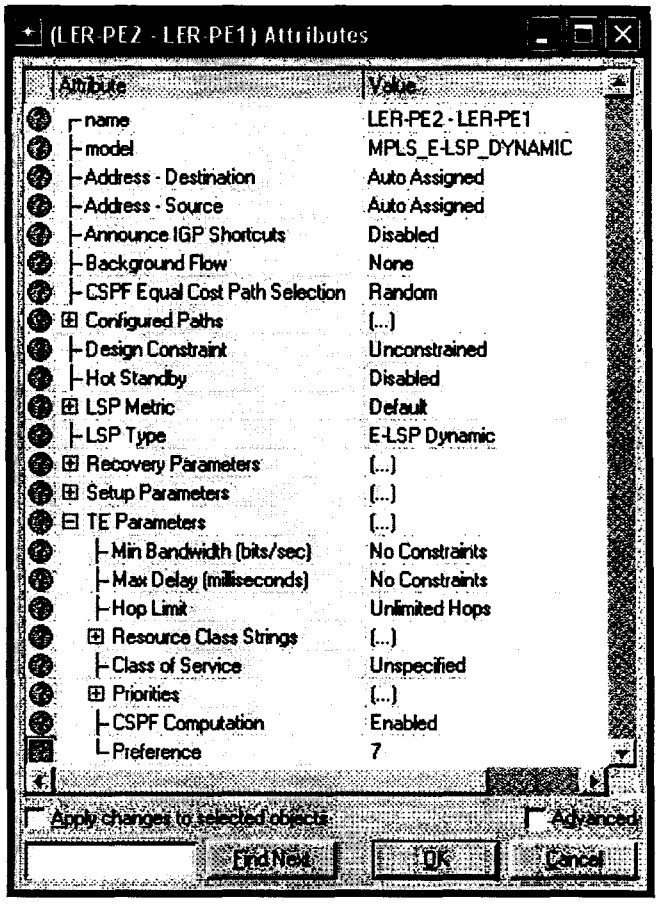


Figure 22 Attributs d'un LSP dynamique

Les attributs *Address Source* et *Destination* se décrivent d'eux mêmes. L'attribut *IGP Shorcuts* (*Interior Gateway Protocol*) est un mécanisme permettant d'associer une étiquette à chaque entrée d'une table de routage. La distribution des étiquettes est associée au protocole de routage de type IGP. Il s'agit d'une des premières utilisations de MPLS, créant un LSP en forme d'arbre dont la racine est le réseau de destination associé à ce LSP. L'attribut *CSPF Equal Cost Path Selection* (*Constraint Shortest Path First*) spécifie quel chemin sera utilisé lorsque deux chemins sont considérés égaux par le protocole de routage avec contraintes CSPF. L'attribut *Design Constraint* permet de spécifier si le LSP peut être modifié ou supprimé pour des considérations d'ingénierie de trafic. *LSP Metric* est la métrique associée au LSP lorsque celui ci est considéré comme un lien (voir *IGP Shortcuts*). Le type de LSP, L-LSP ou E-LSP est spécifié par l'attribut

LSP Type. Les attributs *Recovery Parameters* définissent le comportement à adopter en cas de bris d'un des nœuds constituants le LSP. Le début et la fin de l'existence du LSP sont spécifiés dans les attributs *Setup Parameters*. Finalement, les attributs *TE Parameters* permettent de spécifier les contraintes qui devront être rencontrées par le routage avec contraintes lors de la détermination du chemin que doit suivre le LSP. Ces contraintes peuvent être la bande passante minimale, le délai maximal, un nombre de saut maximum, une ressource quelconque (ex : doit passer par des liens OC-3) et finalement l'attribut *Priorities* permet de spécifier le niveau de préemption.

LSP configurés pour notre simulation

Pour nos simulations, nous avons configuré six LSP dynamiques de type E-LSP. Du réseau 192.168.10.0 au réseau 192.168.20.0 et l'inverse pour le transport de la voix, des réseaux 192.168.10.0 et 192.168.20.0 au réseau 192.168.30.0 et l'inverse pour le transport de la signalisation (Figure 23). Les chemins sont déterminés par contraintes en spécifiant une bande passante minimale de 256Kbps de type EF pour les LSP de voix et une bande passante minimale de 64Kbps de type AF31 pour les LSP de signalisation. Ces réservations pourront être modifiées en cours de simulation pour répondre aux évaluations de besoins établies par les algorithmes étudiés.

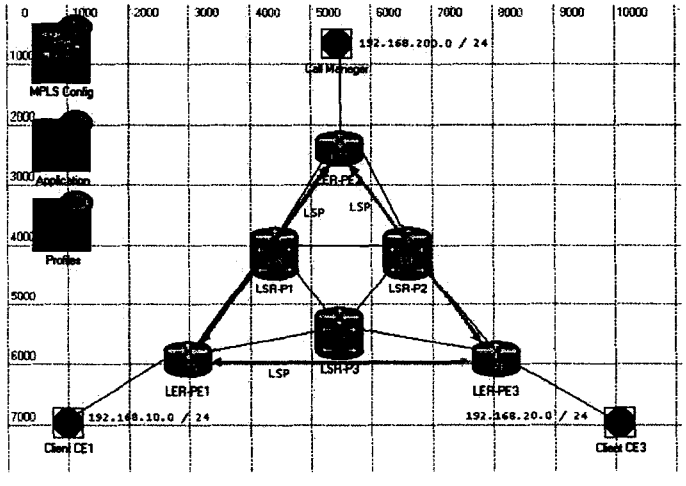


Figure 23 LSPs créés pour notre réseau MPLS

3.3.5 Configuration des LER et LSR

Le schéma du réseau étant complété, les *FEC*, *Trunk* et LSP étant définis, il reste à configurer les éléments du réseau pour qu'ils utilisent ces éléments adéquatement. Les LER et LSR possèdent une multitude de paramètres de configuration pour le support de la technologie MPLS (Figure 24). Le premier de ces attributs, *Status*, doit être actif afin que le routeur supporte la technologie MPLS.

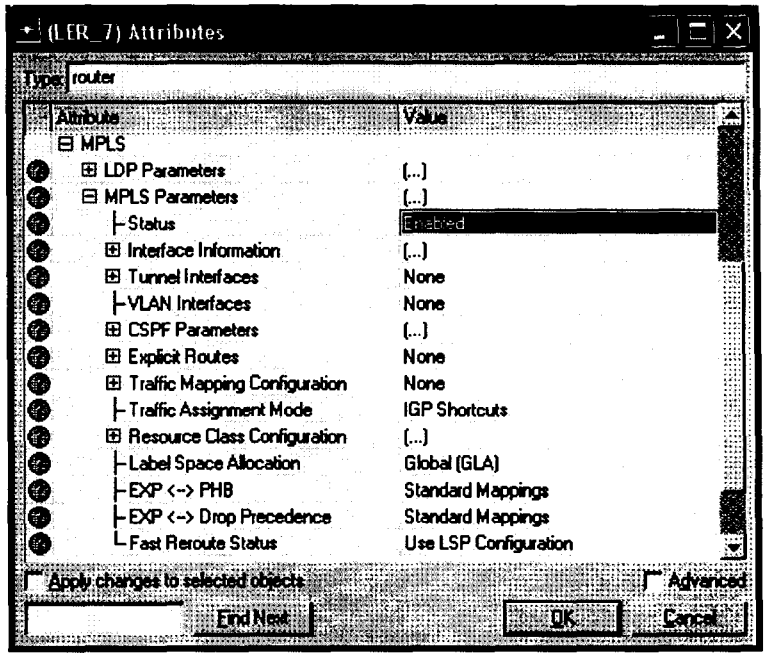


Figure 24 Paramètres de configuration MPLS pour les LER et LSR

Association des FEC, *Trunk* et LSP dans un LER

Le rôle d'un LER est de recevoir les paquets d'un réseau non MPLS et de leur ajouter une étiquette pour qu'ils poursuivent leur chemin sur un LSP déterminé. La détermination du numéro d'étiquette qui sera ajouté, et par conséquent du LSP à suivre, peut se faire de plusieurs façons. Par exemple, à chaque entrée d'une table de routage peut correspondre un LSP menant au réseau de destination. Dans notre cas, nous voulons que la sélection du LSP que doit suivre un paquet soit déterminée par l'utilisation des FEC. L'attribut *Traffic Assignment Mode* doit avoir la valeur *None* afin

que la sélection des LSP soit réalisée à l'aide des FEC. Pour réaliser l'association des FEC, *trunk* et LSP, l'attribut *Traffic Mapping Configuration* doit être édité.

Dans la fenêtre d'association (Figure 25) il faut choisir le nombre d'associations que l'on désire configurer afin que le nombre de lignes approprié soit disponible.

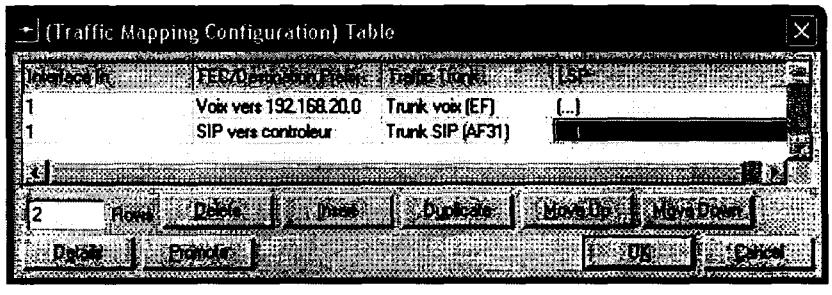


Figure 25 Fenêtre de configuration des associations FEC – LSP

La colonne *Interface In* permet de spécifier les interfaces d'où proviendront les paquets non MPLS que l'on désire étiqueter et acheminer sur la dorsale MPLS. Une interface est un port physique du LER. Ensuite, la colonne *FEC/Destination Prefix* permet de sélectionner la FEC qui sera utilisée pour déterminer l'ensemble des paquets devant suivre le même chemin. Pour séparer le trafic provenant d'une même interface en deux chemins distincts, il faut faire autant d'assignations. Par exemple : « Pour l'interface 1, pour les paquets correspondant aux entrées de la FEC *Voix vers 192.168.20.0*, utiliser la configuration du *Trunk Voix (EF)* et que ce *Trunk* suive le/les LSP déterminé(s) dans la colonne LSP. Pour l'interface 1, pour les paquets correspondant aux entrées de la FEC *SIP vers 192.168.20.0*, utiliser la configuration du *Trunk SIP (AF31)* et que ce *Trunk* suive le/les LSP déterminé(s) dans la colonne LSP ». C'est pourquoi une configuration adéquate des FEC est importante afin qu'un paquet arrivant sur une interface ne puisse être associé à plus d'une FEC.

La colonne LSP permet de sélectionner le/les LSP qui est (sont) utilisé(s) pour acheminer le *Trunk* de trafic déterminé par l'association. Il est possible de déterminer

le/les LSP utilisé(s) de préférence (*Primary*) et celui/ceux à utiliser en cas de problème (*Backup*) (Figure 26).

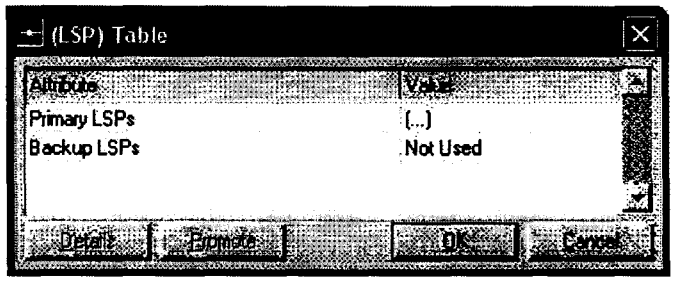


Figure 26 Association des LSP primaires et secondaires d'une association

Le trafic déterminé par la FEC transitera dans un tunnel dont les caractéristiques ont été déterminées par un *Trunk*. Le trafic, ou de façon abstraite le *Trunk*, peut suivre un LSP ou être divisé entre plusieurs LSP primaire ou *Backup*, le poids de chacun déterminant le pourcentage de trafic associé à chaque LSP (Figure 27).

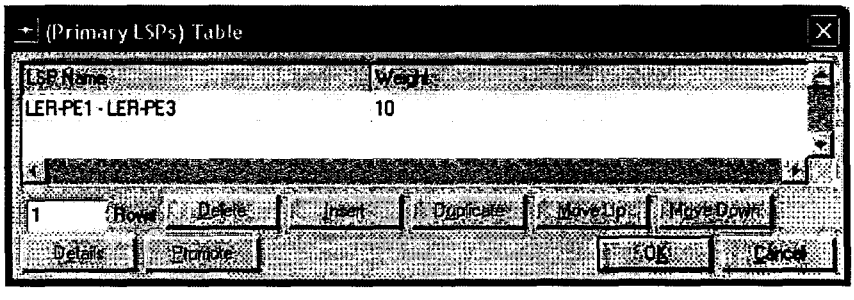


Figure 27 Détermination des LSP à utiliser pour une association

Dans la colonne *LSP Name* apparaissent les LSP configurés au préalable. C'est ici que les LSP configurés sont associés aux FEC et *Trunk*. Ces associations doivent être réalisées pour chaque LER. Pour notre réseau, nous avons configuré uniquement un LSP. Le réseau MPLS est maintenant configuré et prêt à recevoir le trafic de voix et de signalisation.

3.4 Configuration du trafic de voix et de la signalisation SIP

Le réseau est maintenant schématisé et configuré pour être en mesure d'acheminer la voix et la signalisation SIP sur un réseau MPLS supportant la qualité de service. Il reste à configurer le trafic qui transitera sur ce réseau, c'est-à-dire les paquets de voix et de signalisation SIP.

Avant même de configurer l'application de voix et le protocole de signalisation SIP, il faut compléter la schématisation du réseau afin d'y inclure les équipements terminaux qui génèreront le trafic. Ces équipements sont des téléphones IP et un serveur mandataire SIP qui fera office de SC implantant le VSC. Ces objets se retrouvent dans la palette SIP (Figure 28).

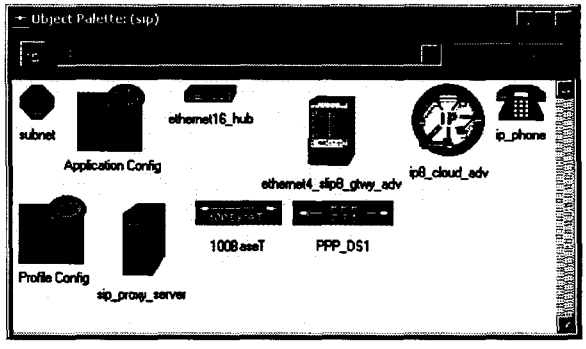


Figure 28 Palette d'objets SIP

Les objets *Application Config* et *Profile Config* doivent également être ajoutés au schéma afin de configurer les aspects généraux reliés au trafic. Les téléphones IP et le mandataire SIP devront être câblés adéquatement dans chaque sous réseau à l'aide de concentrateurs, commutateurs Ethernet, routeurs et de liens physiques appropriés (Figure 29) (Figure 30).

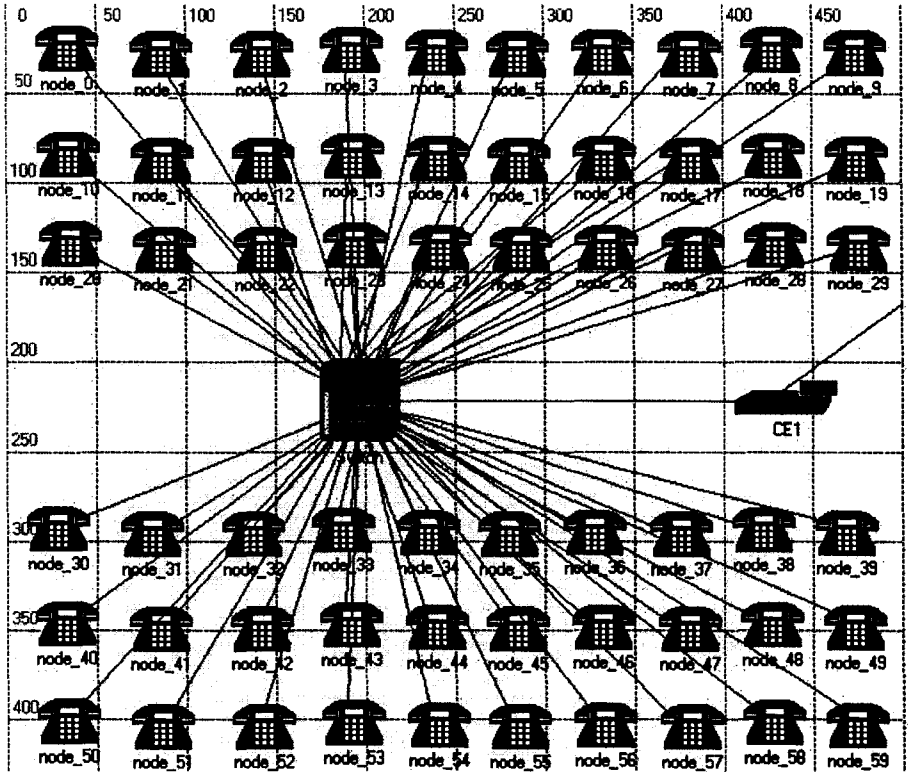


Figure 29 Réseau Client édifice A et B

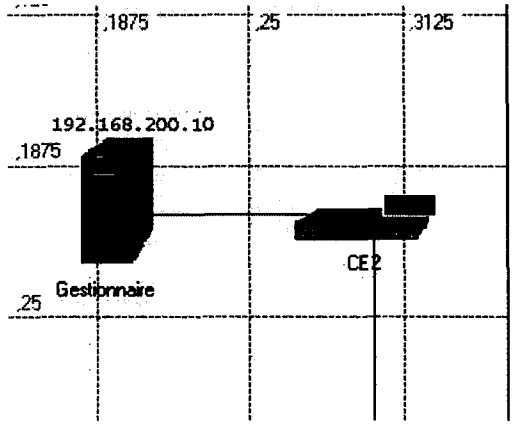


Figure 30 Réseau du Mandataire SIP

3.4.1 Définition de l'application voix

Pour configurer les applications que pourront utiliser les stations reliées au réseau, il faut éditer l'attribut *Application Definitions* de l'icône *Application Config* déposé préalablement (Figure 31).

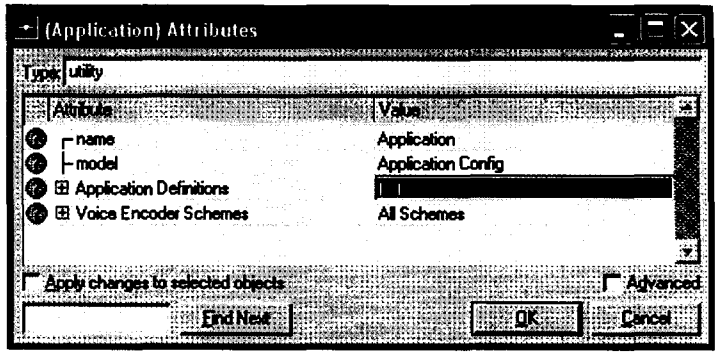


Figure 31 Fenêtre de configuration des applications

Dans la fenêtre de définition des applications (Figure 32), il faut choisir le nombre d'applications que l'on souhaite configurer afin que le nombre de lignes approprié soit disponible.

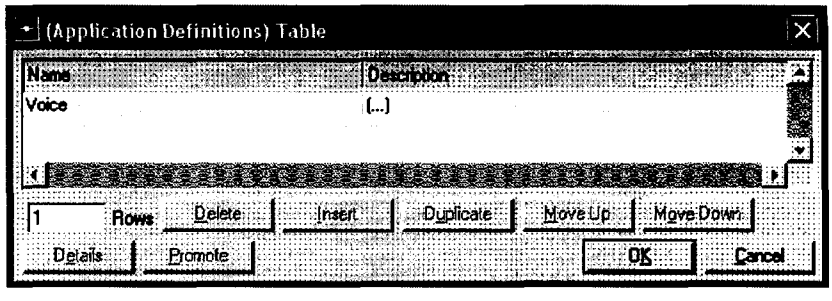


Figure 32 Fenêtre de définitions des applications

Pour nos simulations, une seule application sera utilisée, il s'agit de la voix. La colonne *Name* permet de donner un nom explicite aux applications désirées. Il faut par contre spécifier au logiciel à quoi correspond le nom de l'application inscrit dans la colonne *Name*. Pour ce faire il faut éditer l'application dans la colonne *Description*. La fenêtre suivante apparaît (Figure 33).

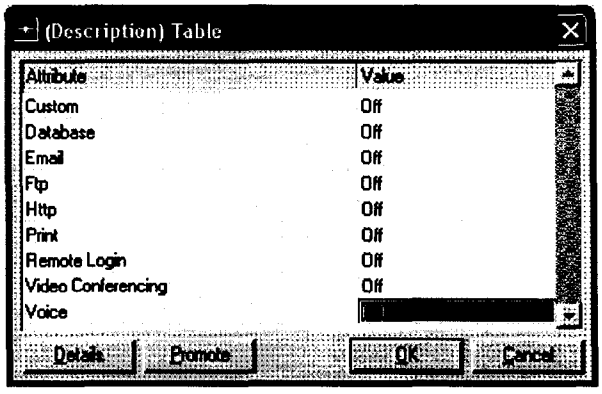


Figure 33 Applications reconnu par le logiciel

Le logiciel Opnet permet de simuler plusieurs applications. Pour ce faire, il doit connaître leur fonctionnement. Par exemple, si une application avait été nommée *MSN Messenger*, il serait impossible de définir cette application car Opnet ne la connaît pas. Pour définir à quoi correspond l'application nommée *Voice,* il faut sélectionner une valeur dans la colonne *Value* de l'attribut *Voice*. Opnet permet l'utilisation de plusieurs schémas d'encodage préalablement configurés. Dans notre cas, pour plus de contrôle sur le type de trafic généré par notre application de voix, il faut éditer la valeur de l'attribut *Voice*.

L'application *Voice* supportée par Opnet possède plusieurs attributs permettant de paramétrer la façon dont les paquets de voix sont générés (Figure 34).

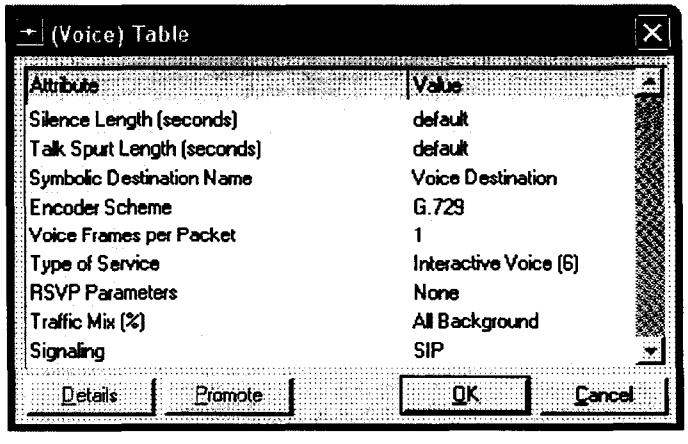


Figure 34 Configuration de l'application voix

Attributs *Silence Length* et *Talk Spurt Length*

Les attributs *Silence Length* et *Talk Spurt Length* servent à définir le pourcentage de temps que les utilisateurs parlent et restent muet. Lorsque les utilisateurs ne parlent pas, il n'est pas nécessaire de générer des paquets et certains encodages tiennent compte de cette particularité de la voix. Cela permet d'économiser de la bande passante. Dans notre cas, nous ne spécifierons rien afin de générer les paquets à un rythme constant.

Attributs *Encoder Scheme* et *Voice Frame per Packet*

Le schéma d'encodage spécifie la façon dont les informations de voix sont compressées et le taux auquel les paquets de voix sont générés. Par exemple, peu importe le taux de compression utilisé, le schéma d'encodage spécifie si les données de voix compressée sont transmises chaque milliseconde ou chaque seconde.

Dans la littérature, un *Voice Frame* est spécifié par le schéma d'encodage et peut avoir plusieurs significations. Par exemple G.711 traite une ms de voix et génère un *Voice Frame* pour cette ms. 20 *Voice Frame* sont transmis par paquet. Pour Cisco, un *Voice Frame* est égal à 10ms de voix et deux sont transmis par paquet. H.323 spécifie qu'un *Voice Frame* contient 8 échantillons de type PCM, donc 1 ms. Ce qui importe vraiment est la quantité de paquets par seconde qui est générée par l'application. En effet, si peu de paquets sont transmis, il y aura un délai trop long entre le moment où un interlocuteur parle et le moment où il sera entendu. À l'opposé, si trop de paquets sont générés, il y aura trop peu de données par paquet et l'essentiel de ce qui transitera sera constitué d'en-tête. Pour G.711, 50 paquets par seconde sont générés.

L'attribut *Voice Frame per Packet* permet de spécifier combien de *Voice Frame* est transmis par paquet. Pour Opnet, un *Voice Frame* est égale à 4ms. Pour générer 50 paquets par seconde, cet attribut est configuré à 5. Le schéma d'encodage G.711 est utilisé afin de générer les données à un taux constant de 64000bps.

Attributs *Type of Service* et *RSVP Parameters*

L'application *Voice*, comme pour toute application disponible avec Opnet, peut utiliser une infrastructure de réseau supportant l'architecture de qualité de service *Intserv* à l'aide du protocole RSVP. Comme nous n'utiliserons pas cette architecture, cet attribut n'est pas configuré. Pour le support de l'architecture de QoS *Diffserv*, le champ ToS des paquets IP transportant la voix peut être marqué à la source. Normalement, dans une infrastructure *Diffserv*, ce n'est pas l'usager qui marque les paquets pour une raison évidente. Dans notre cas, les usagers marqueront les paquets de voix avec la valeur *Interactive Voice (6)*. Ce champ ne sera pas utilisé dans un contexte de réseau IP supportant *Diffserv* mais plutôt pour être utilisé par les FEC devant repérer les paquets de voix et de signalisation.

Pour le trafic qui sera généré par SIP, Opnet ne permet pas de spécifier le champ ToS. Les paquets IP transportant des données SIP sont marqués de la même façon que les paquets de voix, c'est-à-dire *Interactive Voice (6)*.

Attributs *Traffic Mix (%)* et *Signaling*

L'attribut *Signaling* permet de spécifier quel type de signalisation sera utilisé. Il est possible de ne pas utiliser de signalisation, d'utiliser H.323 et dans le cas qui nous intéresse, la signalisation SIP. L'attribut *Traffic Mix* est très important par rapport à la durée que prennent les simulations. Pour cette application, deux types de trafics sont générés. Le trafic de voix et le trafic de signalisation. La génération de trafic de voix est très longue pour une simulation; si ce trafic est nécessaire, il faut configurer cet attribut à *All Discrete*. Cet attribut permet également de ne pas générer de trafic de voix, seulement le trafic de signalisation associé à un appel, ce qui réduit grandement le temps de simulation. Dans notre cas, nous nous intéressons au fonctionnement d'un algorithme lié à la signalisation, le trafic de voix n'est donc pas nécessaire.

3.4.2 Définition de profils

Le profil sert à associer plusieurs applications utilisées par une même catégorie d'utilisateurs. Bien que nous n'utilisions qu'une application, la façon de configurer les stations nous oblige à configurer un profil. En effet, une station supporte un profil définissant les applications utilisées.

Pour définir un profil, il faut éditer l'attribut *Profile Configuration* (Figure 35) de l'icône du même nom.

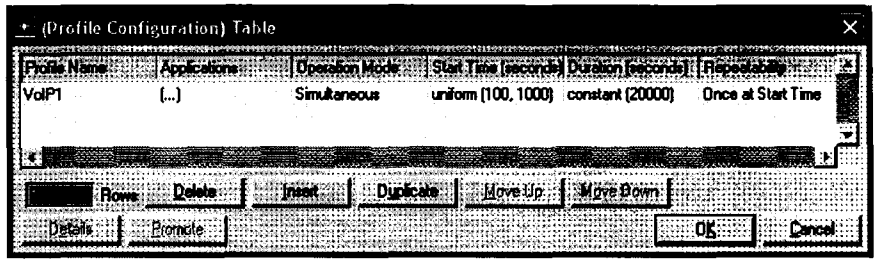


Figure 35 Attributs d'un profil

La première étape est de donner un nom au profil. Pour configurer les applications associées à ce dernier, il faut éditer la colonne *Applications*. Les applications préalablement configurées sont disponibles. La colonne *Operation Mode* spécifie si les applications sélectionnées doivent fonctionner en parallèle (*Simultaneous*) ou l'une après l'autre (*Serial*). Pour nos simulations, comme nous n'utilisons qu'une application, le mode d'opération n'a aucune importance. La configuration des attributs *Start Time* et *Duration* dépendent des simulations. Pour simuler un taux de trafic différent à différentes périodes de la journée, il faudra définir autant de profiles. Par exemple, pour configurer un taux de trafic X pendant 8 heures, l'attribut *Duration* devra être configuré en conséquence. L'attribut *Start Time* permet de démarrer les profils à différents moments. Par exemple, un premier profil démarre à t=0, comme il a une durée de 8 heures, *Start Time* spécifie au second profil de démarrer 8 heures après le début de la simulation. La période spécifiée pour le démarrage, doit être configurée adéquatement.

En effet, lors de la période de démarrage du profil, chaque station établi son premier appel. Afin de ne pas avoir une pointe de trafic occasionnée par cette période de démarrage, celle-ci doit être du même ordre de grandeur que le temps entre deux appels générés par une même station.

Chaque application spécifiée dans un profil possède également quelques attributs de configuration (Figure 36).

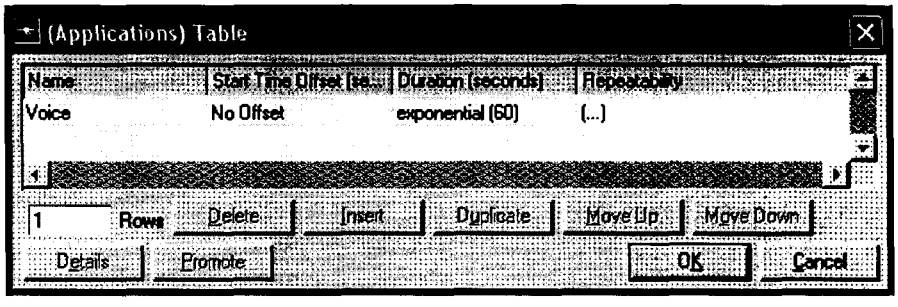


Figure 36 Table de configuration des applications d'un profil

La colonne *Name* est celle permettant de sélectionner une des applications préalablement configurées. La colonne *Start Time* permet de spécifier combien de temps après le début du profil l'application est fonctionnelle. Pour une application de type HTTP ou FTP la colonne *Duration* spécifie que l'application est fonctionnelle jusqu'à la fin du profil et la colonne *Repeatability* spécifie que l'application se répète une fois. Pour les applications de voix et de vidéo, ces colonnes sont beaucoup plus importantes. Elles permettent de spécifier la longueur des appels et le temps entre deux appels. Le taux de trafic est donc configuré à l'aide de ces attributs.

3.4.3 Taux d'appels et de trafic

La colonne *Duration* (Figure 36) spécifie que les appels auront une durée moyenne de 60 secondes, selon une distribution exponentielle. L'attribut *Repeatability* est utilisé pour spécifier comment est réalisé l'intervalle de temps entre deux appels (Figure 37).

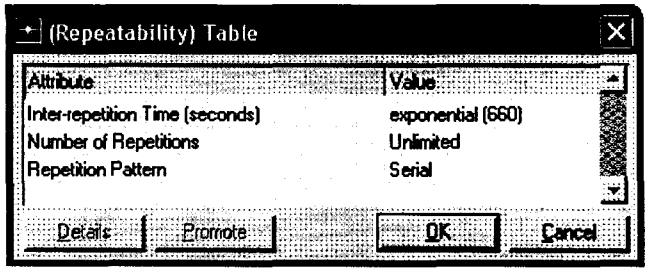


Figure 37 Répétitivité des appels

L'attribut *Inter-repetition Time* spécifie la durée qui doit s'écouler entre deux appels d'une même station. L'attribut suivant spécifie combien de fois un appel sera établi. Finalement, l'attribut *Repetition Pattern* spécifie si la durée d'inter-répétition est calculée entre le début d'un appel et le début du suivant (*Concurrent*) ou si cette période est calculée entre la fin d'un appel et le début du suivant (*Serial*).

3.5 Configuration des équipements de téléphonie

Les icônes *Application* et *Profil* ont permis de définir les applications qui seront utilisées pour générer le trafic. Ces applications et profils seront associés aux différents équipements, clients et serveurs, pour déterminer qui supporte quoi. Dans le cas présent, des stations représentées par des téléphones SIP supporteront la téléphonie et un serveur SIP supportera les fonctionnalités de mandataire.

3.5.1 Configuration du Serveur SIP

Le serveur SIP n'est pas nécessaire à l'application voix. Il est rendu nécessaire par l'utilisation du protocole de signalisation SIP et par l'architecture de réseau proposée. La configuration du serveur mandataire SIP est relativement simple, il suffit de lui donner une adresse (*Server Address*) et de lui dire de supporter SIP. L'adresse sera utilisée plus tard pour spécifier aux stations quel mandataire SIP utiliser. Bien que le mandataire SIP ait peu d'attribut, il sera la composante principale de nos simulations.

Nous y implanterons différents contrôles d'accès qui ajouteront une multitude d'attributs de configuration.

3.5.2 Configuration des terminaux téléphoniques

La dernière étape dans la configuration des éléments constituant notre réseau est la configuration des stations supportant l'application de voix. Il faut configurer les téléphones pour le support de la voix et pour l'utilisation de SIP (Figure 38).

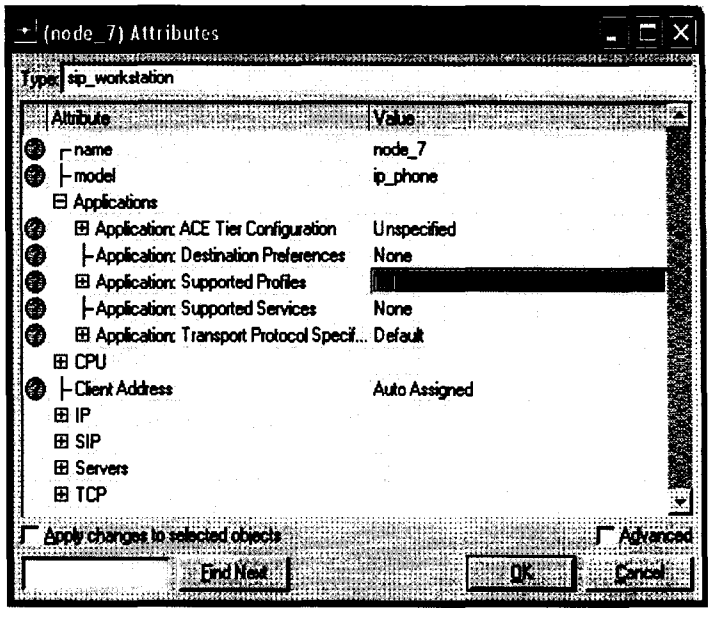


Figure 38 Attributs d'application d'une station voix

Pour une application conventionnelle, les stations supportent un/des profil(s) (*Supported Profiles*) et les serveurs supportent une/des application(s) (*Supported Services*). Pour les applications de voix et de vidéo, chaque station est à la fois client et serveur. Si elle initie un appel, c'est le client qui est utilisé, si elle reçoit un appel, c'est le serveur. Pour que chaque téléphone puisse initier ou recevoir des appels, il faut configurer ses attributs *Supporterd Profiles* et *Supported Services* avec le profil et l'applications, configurés préalablement.

Pour faciliter l'interprétation des résultats, seules les stations du réseau 192.168.10.0 pourront initier des appels. Elles supporteront le *Profil* voix mais pas le service puisqu'elles ne pourront agir comme récepteurs d'appel. À l'opposé, les stations du réseau 192.168.20.0 ne pourront que recevoir des appels. Elles supporteront le service voix, mais aucun profil n'y sera configuré.

En plus du support de la voix, les stations doivent être configurées pour l'utilisation de SIP (Figure 39). Il faut paramétrer leur UAC en spécifiant quel mandataire utiliser.

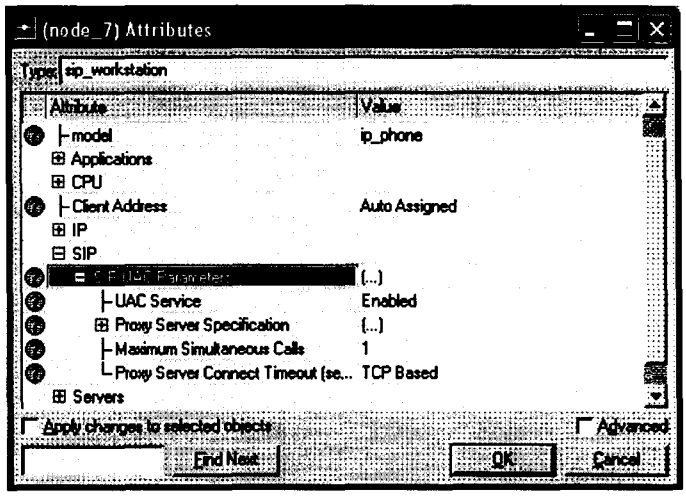


Figure 39 Attributs SIP des stations

Pour qu'une station utilise SIP, il faut que l'attribut *UAC Service* soit à *Enable.* Sinon, même si l'application voix est configurée pour utiliser SIP, le processus nécessaire au fonctionnement de SIP ne sera pas activé. L'attribut *Proxy Server Specification* permet de spécifier quel serveur mandataire doit être contacté. Ainsi, il est possible d'acheminer les paquets de signalisation au mandataire qui servira au contrôle d'accès. Finalement, l'attribut *Maximum Simultaneous Calls* permet de spécifier combien d'appels simultanés sont possibles; dans notre cas, un seul par station.

La configuration du réseau est maintenant complétée. Une dorsale MPLS sera utilisée pour acheminer de la voix ainsi que la signalisation SIP sur des *Trunk* auxquels une qualité de service distincte est attribuée. Ces *Trunk* suivront des LSP dont le chemin est déterminé par des contraintes garantissant les ressources nécessaires aux qualités de service requises par les *Trunk*. Une configuration adéquate du/des profil(s) permettra de simuler un patron de trafic adéquat.

La prochaine étape sera de décrire comment le protocole SIP est implanté dans OPNET afin de connaître l'emplacement où la programmation des algorithmes de contrôle d'accès doit être réalisée.

CHAPITRE 4

IMPLANTATION DE L'APPLICATION DE VOIX ET DE SIP

La modélisation du réseau étant accomplie, il faut maintenant comprendre le fonctionnement des différents nœuds utilisés pour la schématisation. Le fonctionnement des téléphones IP et du mandataire SIP sera détaillé. L'établissement d'un appel fait à l'aide de SIP sera expliqué en détail afin d'avoir une compréhension globale des différents processus déjà implantés dans OPNET. Cette compréhension est essentielle afin de connaître les processus devant être modifiés pour implanter les algorithmes de contrôles d'accès et d'évaluation des besoins qui seront étudiés. Un exemple de modification apporté au code existant sera décrit. Afin de faciliter la lecture du chapitre qui suit, la convention typographique suivante a été utilisée. ***CALL*** : représente un état, ***call*** : représente un processus, *call()* : représente une fonction et **CALL** : représente un message de protocole.

4.1 Description des modèles téléphone IP et mandataire SIP

Les nœuds de type *ip_phone* et *sip_proxy_server* sont dérivés de nœuds parents appelés *ethernet_server_adv* et *ethernet_wkstn_adv*. Ces deux parents sont identiques. Il s'agit de nœuds originaux en terme de fichier, pas en terme de contenu. Ces nœuds sont constitués de modules que l'on peut associer aux différentes couches du modèle OSI (Figure 40).

4.1.1 Modules Receiver et Transmitter

Les modules *Receiver (hub_rx_0_0)* et *Transmitter (hub_tx_0_0)* reçoivent et transmettent les trames sur le médium. Il s'agit des ports auxquels les liens peuvent se connecter.

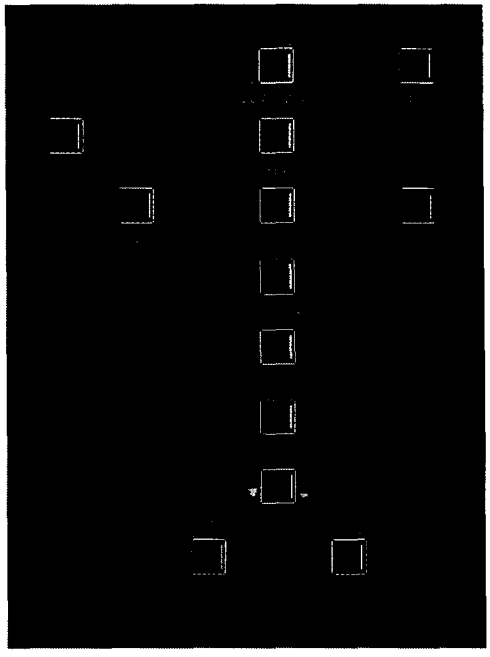


Figure 40 Modélisation d'un téléphone IP et d'un mandataire SIP

4.1.2 Module MAC et processus ethernet_mac_v2

Le module MAC (*Medium Access Control*) contient le processus ***ethernet_mac_v2*** (Figure 41) qui modélise la couche MAC pour les interfaces Ethernet de type 10BaseT, 100BaseT ou 1000BaseX. Son rôle est d'encapsuler les paquets reçus des protocoles de la couche supérieure dans des trames Ethernet et de transmettre ces trames dans un ordre FIFO (*First In First Out*). Ce processus s'assure d'un partage équitable de la bande passante entre les nœuds attachés au même hub.

Figure 41 Modélisation du processus ethernet_MAC_v2

4.1.3 Module ARP et processus ip_arp_v4

Le module ARP (*Address Resolution Protocol*) contient le processus ***ip_arp_v4*** (Figure 42) qui modélise le protocole ARP permettant d'obtenir l'adresse physique, à partir de l'adresse IP. Dans le cas d'un nœud Ethernet, le processus ARP obtient l'adresse MAC. Ce processus est important afin de supporter un adressage physique arbitraire. Il est ainsi possible de faire abstraction des différents types d'adresse physique à l'aide d'un adressage uniforme au niveau IP. Une table ARP globale est créée automatiquement par le processus ***ip_arp_v4*** pour permettre la transition d'adresse en cours de simulation.

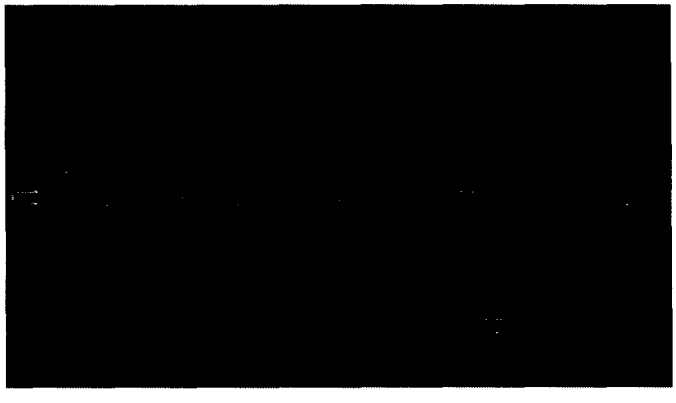


Figure 42 Modélisation du processus ip_arp_v4

4.1.4 Module IP, processus racine et enfants

Le module IP contient le processus ***ip_dispatch*** (Figure 43) qui modélise l'ensemble des fonctions de la couche réseau. Les paquets IP arrivant sur une interface d'entrée sont acheminés sur une interface de sortie appropriée en se basant sur l'adresse IP de destination. Un protocole de routage dynamique peut être utilisé pour configurer automatiquement les tables de routage. Dans le cas d'un nœud de type station ou serveur, la table de routage est relativement simple. Le processus ***ip_dispatch*** utilise une période de temps déterminée pour acheminer chaque paquet, déterminant ainsi un taux auquel les paquets sont traités.

Figure 43 Modélisation du processus ip_dispatch

Le processus ***ip_dispatch*** est en fait un processus racine qui démarre et utilise plusieurs autres processus dits enfants. Ces processus sont utilisés au besoin selon la configuration utilisée. I***p_dispatch*** s'assure de transmettre le paquet au bon processus enfant. Voici une liste de différents processus enfants :

- ***ip_custom_mrp*** : gestion du multicast.
- ***ip_icmp*** : manipulation des messages ICMP.
- ***ip_igmp_host*** : manipulation des messages IGMP.
- ***ip_igmp_rte_intf*** : routage multicast.
- ***ip_rte_central_cpu*** : décision de routage et délai de traitement.
- ***ip_vpn*** : tunnels IP.

- ***ipv6*** : manipulation des paquets IPv6.
- ***ip_output_iface*** : mécanismes de file d'attente (ex : WFQ)
- ***mpls_mgr*** : manipulation des paquets MPLS

4.1.5 Processus enfant MPLS_MGR

Le processus racine ***ip_dispatch*** créé ce processus lorsque MPLS est activé. Ce processus représente le plan d'acheminement de MPLS. Lorsqu'un nœud, plus précisément le processus ***ip_dispatch***, reçoit un paquet ayant une étiquette ou correspondant à une FEC, il ne fait aucun traitement sur ce paquet et le transmet directement au processus ***mpls_mgr*** (Figure 44).

Figure 44 Modélisation du processus mpls_mgr

Le paquet est donc transmis de ***mpls_mgr*** en ***mpls_mgr*** sur les nœuds où MPLS est activé plutôt que router par un autre processus enfant de ***ip_dispatch***. Lors de l'initialisation, ce processus établit les tables permettant d'élaborer les LSP configurés de façon statique. À la sortie de l'état ***INIT_WAIT***, le protocole de signalisation de LSP dynamique est appelé, RSVP ou LDP, afin de créer les LSP dynamiques. En fonction du protocole de signalisation sélectionné, les processus suivants sont utilisés.

- ***mpls_ldp_mgr*** : Il s'agit du plan de contrôle LDP. Il démarre les processus ***mpls_discovery_mgr***, ***mpls_session_mgr*** et ***mpls_lsp_mgr***.

- ***mpls_discovery_mgr*** : Broadcast un message **HELLO** pour découvrir les routeurs MPLS.
- ***mpls_session_mgr*** : S'occupe des sessions TCP entre routeur LDP.
- ***mpls_lsp_mgr*** : S'occupe de l'échange de d'étiquettes entre pair LDP.

4.1.6 Module IP_encap et processus ip_encap_v4

Le module *IP_encap* contient le processus ***ip_encap_v4*** (Figure 45) qui s'occupe de l'encapsulation et de la désencapsulation. Il agit comme une interface entre les hautes couches et le module IP. Plusieurs protocoles de couches supérieures peuvent être connectées sur ce module. Il peut s'agir d'un protocole de transport tels TCP (*Transport Control Protocol*) et UDP (*User Datagram Protocol*), un protocole de routage tel IGRP (*Interior Gateway Routing Protocol*), EIGRP (*Enhanced IGRP*) ou OSPF ou un protocole de réservation tel RSVP.

Figure 45 Modélisation du processus ip_encap_v4

4.1.7 Module UDP et processus udp_v3

Le module UDP contient le processus ***udp_v3*** (Figure 46) qui modélise le protocole de transport UDP. Le principal service fourni par ce processus est l'indexation (*Mapping*) des numéros de port avec les processus applicatifs utilisant UDP. Ce processus assigne également un port local s'il n'est pas spécifié par les requêtes applicative sortante.

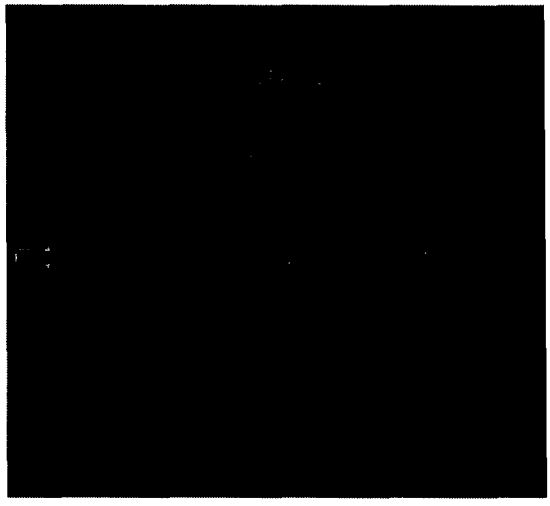


Figure 46 Modélisation du processus udp_v3

4.1.8 Module TCP, processus racine et enfants

Le module TCP contient le processus ***tcp_manager_v3*** (Figure 47) qui modélise un gestionnaire de connexion. Il s'agit d'un processus racine faisant la gestion de connexion transport, chaque connexion étant elle-même un processus ***tcp_conn_v3*** enfant de ce processus racine. ***Tcp_manager_v3*** reçoit toutes les commandes des couches inférieures et supérieures et les dirige vers le bon processus de connexion enfant.

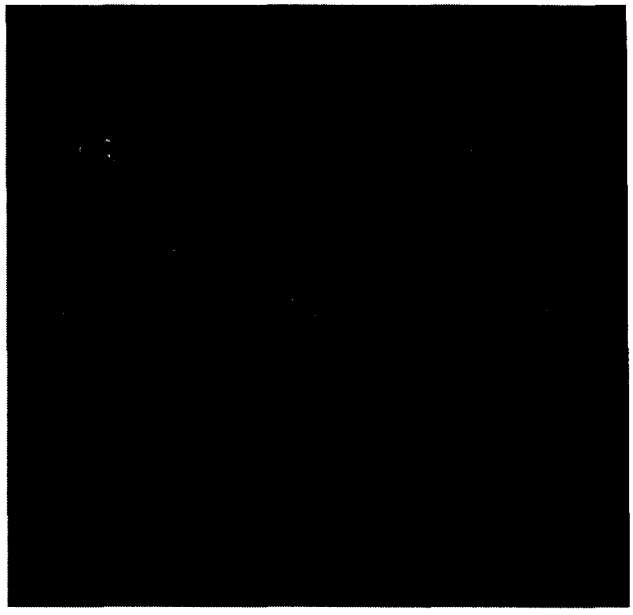


Figure 47 Modélisation du gestionnaire de connexion tcp_manager_v3.

Pour chaque connexion TCP, le gestionnaire ***tcp_manager_v3*** créé un processus ***tcp_conn_v3*** (Figure 48). Il s'agit de l'implantation d'une connexion TCP, c'est à dire le contrôle de la connexion/déconnexion, le contrôle d'erreurs, de flot, de congestion, etc.

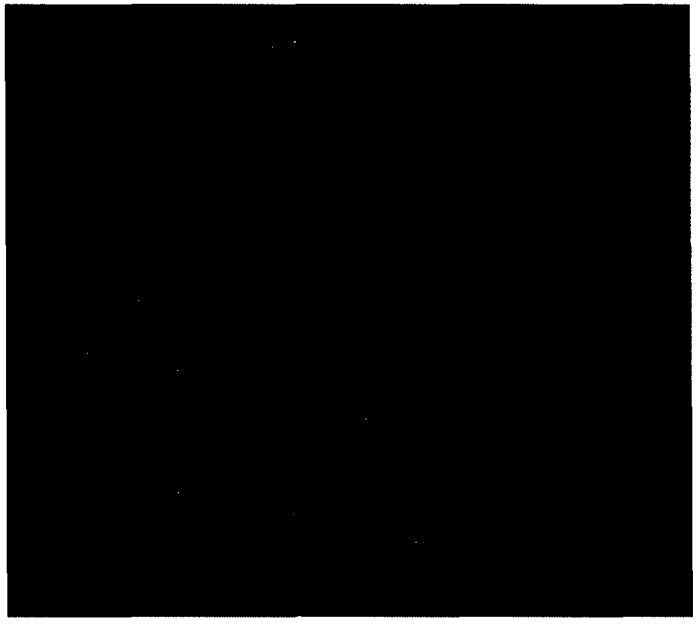


Figure 48 Modélisation d'une connexion TCP

4.1.9 Module TPAL, processus racine et enfants

Le module TPAL contient le processus ***tpal_v3*** (Figure 49) qui fourni une interface uniforme aux applications et aux différents processus de couche transport possible. Il s'agit d'un processus racine qui crée un processus enfant pour chaque couche transport possible. Toutes les interactions entre applications distantes sont organisées en session, une session étant une conversation entre deux applications à travers un protocole de transport. Le processus ***tpal_v3*** fourni un registre global des serveurs. Chaque serveur s'inscrit au début d'une simulation en fournissant le nom du/des services, le/les ports et la popularité (répartition entre différents serveurs offrant le même service). Les applications clientes peuvent choisir un serveur à partir de ce registre.

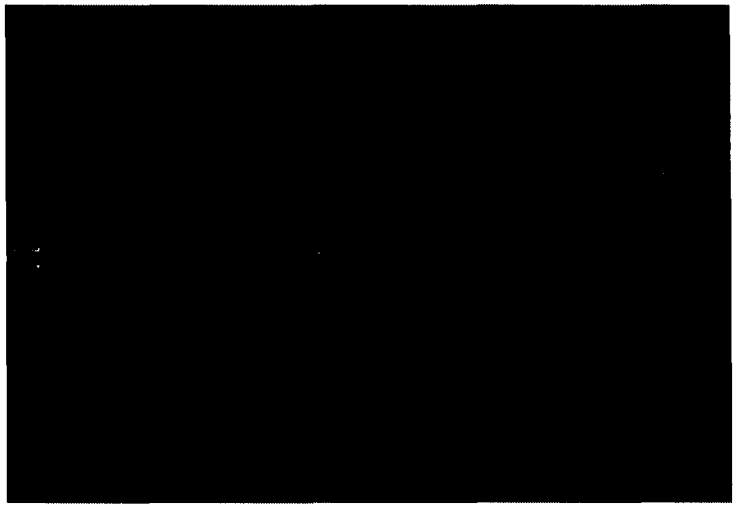


Figure 49 Modélisation du processus d'interfaces de transport tpal_v3

Lorsque TCP est utilisé, le processus ***tpal_v3*** génère un processus enfant ***tpal_intf_tcp_v3*** (Figure 50) qui agit comme interface entre les applications et la couche transport TCP. Les requêtes de l'application reçue par le processus racine ***tpal_v3*** sont dirigées vers ***tpal_intf_tcp_v3***. Celui-ci convertit l'interface générique en une interface spécifique au modèle TCP et vice versa.

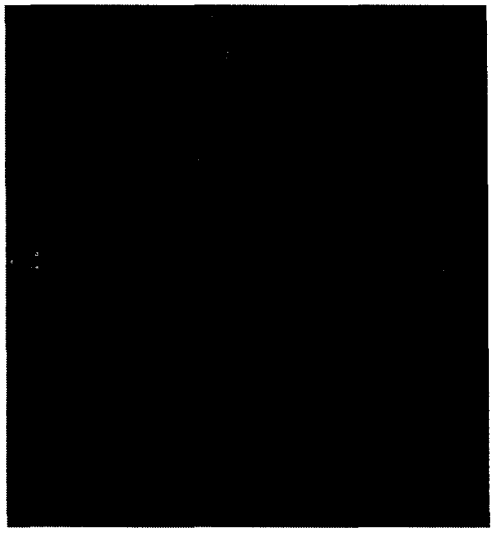


Figure 50 Modélisation de l'interface tpal_intf_tcp_v3

4.1.10 Module Application, processus racine et enfants

Le module *Application* contient le processus ***gna_clsvr_mgr*** (Figure 51) qui modélise un usagé. Il s'agit d'un processus racine créant une multitude de processus enfants modélisant les différents aspects de la couche application. Lors de l'initialisation, ce processus créé des processus enfants serveurs pour chaque service supporté, un processus ***gna_profile_mgr*** pour chacun des profils configurés sur le nœud et démarre les processus SIP client ***sip_uac_mgr*** et/ou SIP serveur ***sip_UAS_mgr*** au besoin. Ces différents services sont enregistrés dans TPAL.

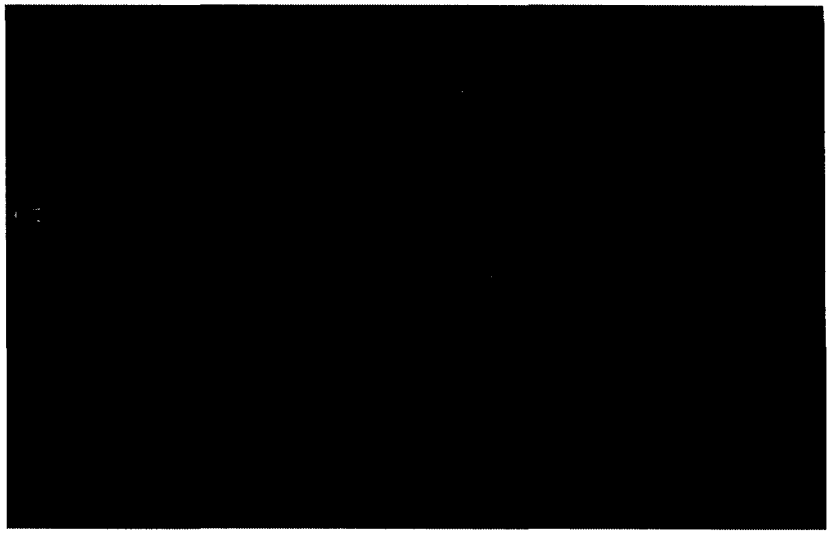


Figure 51 Modélisation du processus racine gna_clsvr_mgr

Le processus enfant ***gna_profile_mgr*** (Figure 52) créé pour chaque profil les processus applications clientes pour la durée du profil. ***Gna_profile_mgr*** utilise les paramètres configurés dans une ligne de l'icône *Profil configuration* et il s'occupe de faire fonctionner ces services en *Serial* ou en *Simultaneous*.

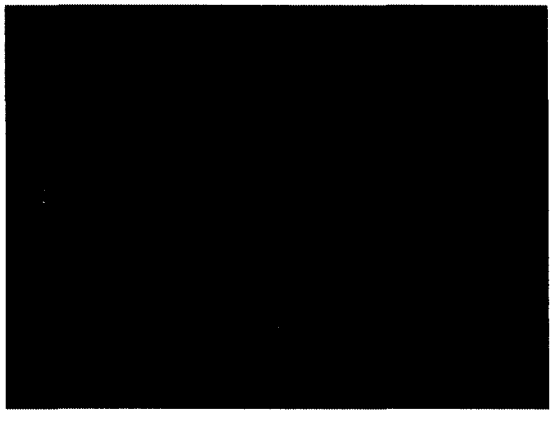


Figure 52 Modélisation d'un processus gestionnaire de profile *gna_profile_mgr*

Les processus « *applications manager* » sont des enfants des processus ***gna_profile_mgr*** qui est lui-même enfant du processus racine ***gna_clsvr_mgr***. Les différents processus « *applications manager* » sont les suivants.

- ***gna_ftp_mgr*** : Il s'agit d'une session FTP entre un client et un serveur. Dans une session FTP il peut y avoir plusieurs connexions. Chaque connexion est représentée par un processus ***gna_ftp_cli***.

- ***gna_email_mgr*** : Il s'agit d'une session de courriel entre un client et un serveur. Dans une session courriel il peut y avoir plusieurs connexions. Chaque connexion est représentée par un processus ***gna_email_cli***.

- Les services *Database*, *Rlogin*, *Print*, *Http* suivent le même principe.

- Les applications *Voice* et *Video* sont quelques peu différentes. Deux processus interviennent, le *calling_mgr* (***gna_voice_calling_mgr*** , ***gna_video_calling_mgr***) et le *called_mgr* (***gna_voice_called_mgr*** , ***gna_video_called_mgr***) Le service *Voice* sera décrit en détail dans la prochaine section. Le service *Video* fonctionne de la même façon que le service *Voice*.

4.2 Processus applicatifs voix et SIP

Le but de ce mémoire étant d'implanter et d'étudier des algorithmes de contrôle d'accès et d'évaluation des besoins en ressources lors de l'établissement d'appels voix à l'aide de SIP, les processus intervenant dans l'établissement d'une session de voix seront vus en détail. La description de ces processus sera détaillée dans une forme permettant de comprendre la façon dont est établis une session de voix.

4.2.1 Résumé de connaissances reliées à l'établissement d'un appel

Afin de mieux comprendre l'ensemble des processus qui seront expliqués dans les prochains points, un résumé du fonctionnement global de ces processus dans l'établissement d'un appel sera réalisé. Pour suivre la description du fonctionnement global, veuillez vous référez à la Figure 53. Vous pourrez également revenir à cette figure lors de la description des processus intervenant dans un appel.

1. Le processus ***gna_clsvr_mgr*** est responsable de la gestion de la couche application. Il créé les processus serveurs d'application (ex : ***gna_ftp_mgr***). Si l'application voix est supportée, un processus ***gna_voice_called_mgr*** (serveur) est démarré pour recevoir les requêtes provenant du réseau. Les différents serveurs établissent une

connexion passive avec le processus ***tpal***, c'est-à-dire un socket qui est à l'écoute des requêtes entrantes. Un processus ***gna_profile_mgr*** est démarré pour chaque profile configuré. G***na_profile_mgr*** est responsable de créé des processus client pour les application configuré dans le profile (ex : ***gna_ftp_cli, gna_voice_calling_mgr***). Lorsqu'un processus ***gna_voice_calling_mgr*** est créé, c'est qu'une session de voix est initialisée par la station.

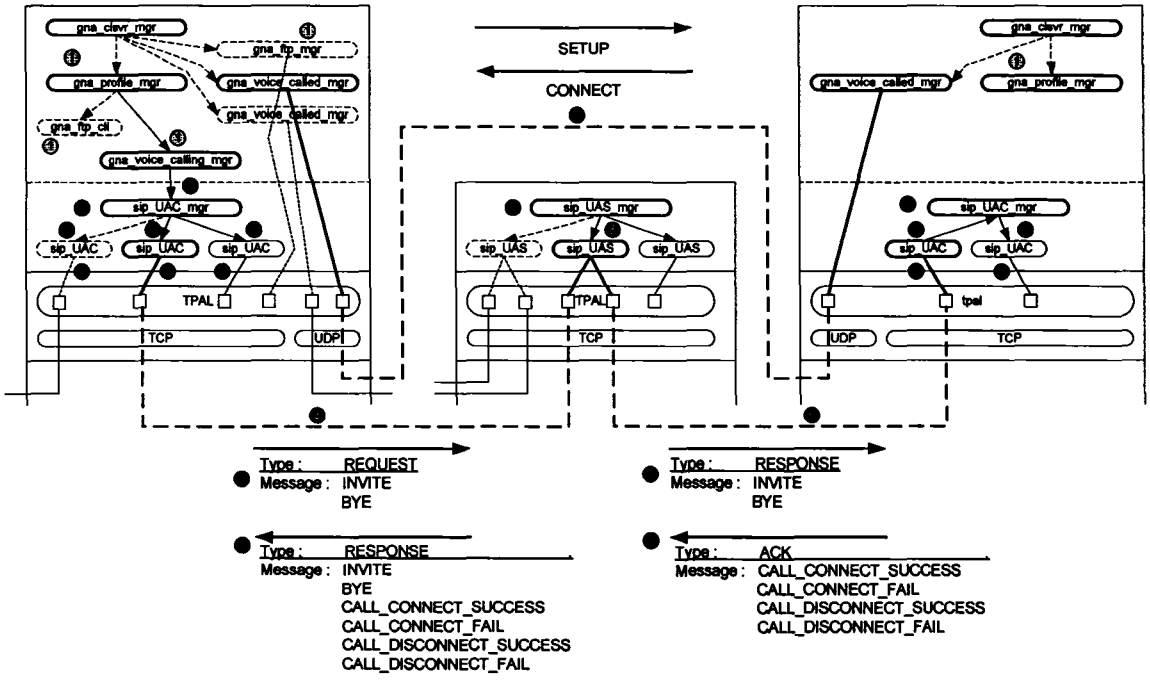


Figure 53 Résumé de l'établissement d'un appel

2. Si le protocole de signalisation n'est pas utilisé, ***gna_voice_calling_mgr*** peut établir une connexion active avec un ***gna_voice_called_mgr***, établir la session de voix à l'aide des messages **SETUP** et **CONNECT** et commencer l'échange de paquets de voix.
3. Si le nœud est configuré pour que l'application *Voice* utilise la signalisation SIP, un processus ***sip_uac_mgr*** est démarré. S'il s'agit d'un mandataire SIP, le processus ***sip_uas_mgr*** est démarré.

4. S***ip_uac_mgr*** gère l'ensemble des sessions SIP. Pour chaque session SIP, ***sip_uac_mgr*** créé un processus ***sip_uac***. ***Sip_uac_mgr*** peut recevoir une requête du processus ***gna_voice_calling_mgr*** afin d'établir une session SIP préalable à l'établissement de la session de voix. Il passe cette requête à un ***sip_UAC***. Le processus ***sip_uac*** reçoit une requête du ***sip_uac_mgr***, il peut également recevoir un paquet SIP provenant du mandataire.
5. Lors de la création d'un ***sip_uac***, celui-ci s'enregistre dans ***tpal*** pour établir une connexion passive afin d'être à l'écoute de requête entrante. Lorsqu'un ***sip_uac*** reçoit un une requête de ***sip_uac_mgr***, il doit établir une connexion active avec le mandataire avant de lui passer des messages SIP. Avant de recevoir un paquet SIP, le mandataire contacte le ***sip_uac*** de destination qui est en écoute passive afin de passer en mode actif. Dans tous les cas, lorsqu'un ***sip_uac*** passe en mode actif, un autre ***sip_uac*** est créé et enregistré afin d'être en écoute passive.
6. Lorsqu'une connexion active est établie entre un ***sip_uac*** et un ***sip_uas*** des messages SIP peuvent être échangés. Les processus ***sip_uas*** fonctionnent de la même façon que les ***sip_uac***. Chacun est responsable d'une session SIP, un ***sip_uas*** étant responsable des fonctions mandataires d'une session SIP. Afin d'acheminer les messages au ***sip_uac*** destinataire, le ***sip_uas*** doit établir une session active avec le ***sip_uac*** de destination. Le ***sip_uas_mgr*** est responsable de la gestion de l'ensemble des ***sip_uas***.
7. Considérant que des connexions transports sont actives (entre le ***sip_uac*** initiant l'appel et le ***sip_uas*** du mandataire ; entre le ***sip_uas*** du mandataire et le ***sip_uac*** de l'appelé), les messages SIP de type **REQUEST, RESPONSE** et **ACK** peuvent être transmis. Les messages de type **RESPONSE** sont transmis du ***sip_uas*** vers les ***sip_uac***. Les types **REQUEST** et **ACK** sont transmis des ***sip_uac*** vers le ***sip_uas*** en fonction des besoins. Si la session SIP est établie, la session de voix peut également être établie. La session de voix s'établie comme s'il n'y avait pas eu de signalisation, voir 2.

4.2.2 Fonctionnement du processus gna_voice_calling_mgr

Une station est configurée pour recevoir des appels de voix en utilisant la signalisation SIP. Son processus ***gna_clsvr_mgr*** a démarré les processus enfants ***sip_uac_mgr*** et ***gna_voice_called_mgr***. Une station est configurée pour initier des appels de voix en utilisant SIP. Son processus ***gna_profile_mgr*** a démarré les processus enfants ***sip_uac_mgr*** et ***gna_voice_calling_mgr***. Un processus ***sip_uas_mgr*** est configuré sur un mandataire SIP.

Le processus ***gna_voice_calling_mgr*** (Figure 54) est responsable de démarrer une session de voix, de générer des paquets de voix et de compléter l'appel.

Figure 54 Modélisation du processus d'appel sortant

Dans l'état d'initialisation ***INIT***, le temps que devra durer l'appel est déterminé, le port de destination et le protocole de transport sont sélectionnés, un serveur est choisi parmi la liste établit par le module TPAL. Dans le cas d'une application *Voice*, le serveur sélectionné est en fait la station de voix qui sera appelée. L'adresse du serveur, ou plutôt de la destination de l'appel, est l'adresse TPAL. Si la destination n'existe pas, le processus passe à l'état ***END***. Si la destination existe et qu'il n'y aucun mécanisme de signalisation, le processus passe à l'état ***OPEN***. Dans notre cas, la signalisation SIP est utilisée, le processus passe donc à l'état ***SETUP***.

En entrant dans l'état ***SETUP***, une session doit être élaborée à l'aide du protocole de signalisation SIP. Si la session SIP est établie, le processus passe à l'état ***OPEN***, sinon il passe à l'état ***END***.

Si la session SIP est établie ou si aucune signalisation n'est utilisée, le processus entre dans l'état ***OPEN***. En entrant dans l'état ***OPEN***, différentes variables sont initialisées et l'application demande au processus ***tpal*** d'établir une connexion active. Pour établir une connexion active, ***tpal*** est responsable de créer une connexion transport avec l'application serveur. **Tpal** est donc responsable d'établir une connexion TCP. Si la connexion TCP est établie, **tpal** a établis une connexion active et le ***gna_voice_calling_mgr*** passe à l'état ***CONNECT*** ; sinon il passe à l'état ***RELEASE.*** L'état ***RELEASE*** est responsable de terminer la session SIP pour ensuite passé à l'état ***END***.

Si la connexion transport est établie, la sortie de l'état ***OPEN*** est responsable de transmettre un message **SETUP** à la destination et de passer à l'état ***CONNECT.*** Sur réception du message **SETUP**, l'application appelée doit transmettre un message **CONNECT**. Si le message **CONNECT** est reçu, le processus passe dans l'état ***IDLE*** où il transmet et reçoit des paquets de voix. Si le message **CONNECT** n'est pas reçu, le processus passe à l'état ***RELEASE***.

Dans l'état ***IDLE***, il est possible de transmettre ou de recevoir un paquet de voix. Il est possible de terminer la session de voix en passant l'information au processus **tpal** pour qu'il termine la connexion transport. Il est également possible de recevoir une interruption de **tpal** confirmant la fin de la session voix. Lorsque la session de voix est terminée, le processus passe dans l'état ***RELEASE*** afin de terminer la session SIP pour finalement terminer le processus représentant une session de voix.

4.2.3 Fonctionnement du processus gna_voice_called_mgr

Pour une station supportant le service *Voice*, un ***gna_voice_called_mgr*** (Figure 55) est enregistré dans ***tpal***. La signalisation SIP ayant fonctionné et la couche transport ayant établis une connexion active grâce à l'enregistrement préalable du processus ***gna_voice_called_mgr*** en écoute passive, ce processus est responsable de la session voix du côté appelé.

Figure 55 Modélisation du processus d'appel entrant

Lors de l'initialisation de ce processus, l'état ***OPEN*** est responsable de l'initialisation de différentes variables et d'établir une connexion passive avec ***tpal***. Une connexion passive permet de dire au processus de la couche transport, ***tpal***, d'accepter les connexions entrantes au niveau de la couche transport pour ce processus. Si ***tpal*** envoi une interruption de type *Abort* ou *Close*, le processus passe dans l'état ***CLOSED*** responsable de terminer ce processus. Si ***tpal*** établis une connexion passive, le processus passe dans l'état ***CONNECT***. Dans cet état le processus attend de recevoir un message de type **SETUP**. S'il reçoit ce message, il passe dans l'état ***IDLE***. Lorsque le message **SETUP** est reçu, en sortant de l'état ***CONNECT*** les actions suivantes sont exécutées. Transmission d'un message **CONNECT** et configuration du type de trafic qui sera généré à l'aide des informations reçues dans le message **SETUP**.

Dans l'état ***IDLE***, il est possible de transmettre ou recevoir des paquets de voix. Il est également possible de terminer l'application en passant l'information à **tpal** afin que ce dernier termine la connexion transport. Il n'y a pas de message de fermeture au niveau de l'application. L'état ***CLOSED*** est uniquement responsable de détruire le processus qui vient de terminer son existence. Le processus ***gna_voice_called_mgr*** n'a aucun lien avec SIP, seul le coté client pouvant demandé l'établissement ou la terminaison d'une session SIP.

4.2.4 Fonctionnement du processus sip_UAC_mgr

Lorsque SIP est utilisé, le processus ***gna_clsvr_mgr*** vérifie s'il s'agit d'un mandataire ou d'un hôte. S'il s'agit d'un hôte, un processus ***sip_uac_mgr*** (Figure 56) est démarré et enregistré comme service dans le processus ***tpal***. Les noms de processus utilisé par OPNET génèrent un malentendu. Normalement, un hôte est à la fois UAC (Client) et UAS (Serveur). Dans OPNET, un UAC est à la fois client et serveur, il s'agit du processus situé dans un hôte. Un UAS est un processus situé dans un serveur mandataire SIP.

Figure 56 Modélisation du processus gestionnaire de session SIP

Lors de l'initialisation de ***sip_uac_mgr***, plusieurs variables sont initialisées. La présence d'au moins un serveur mandataire est vérifié. Une structure d'espace mémoire partagée est créée pour les données ou statistiques globales à l'ensemble des sessions SIP. Finalement, un processus ***sip_uac*** est créé afin d'accepter une requête SIP entrante. Si l'initialisation ne fonctionne pas, le processus passe dans l'état ***INIT-FAIL***. Si l'initialisation fonctionne, le processus passe dans l'état ***WAIT***. Lorsque le processus est dans l'état ***WAIT***, deux conditions sont possibles. La première condition est qu'une requête peut être reçue de l'application voix locale pour établir un appel. Dans ce cas, le ***sip_uac_mgr*** vérifie si un processus ***sip_uac*** est disponible, lui transmet la requête et démarre un autre processus ***sip_uac*** en écoute passive. Dans l'état ***WAIT***, la deuxième condition possible est que le ***sip_uac*** qui est à l'écoute des requêtes provenant du réseau informe le ***sip_uac_mgr*** qu'il est passé à l'état actif et donc qu'il ne peut plus être à l'écoute des requêtes entrantes. Dans ce cas, le ***sip_uac_mgr*** créer un autre ***sip_uac*** qui sera à l'écoute pour une nouvelle session.

4.2.5 Fonctionnement du processus sip_uac

Le processus ***sip_uac*** (Figure 57) est responsable d'une session SIP, de la signalisation de l'appel préalable à une session de voix jusqu'à la fin de la session de voix qui impose la terminaison de la session SIP associé. Ce processus est créé pour chaque appel. Le premier processus ***sip_uac*** est créé lors de l'initialisation du ***sip_uac_mgr*** pour être à l'écoute des requêtes entrantes. Un processus ***sip_uac*** est responsable de la signalisation SIP d'une session de voix. Lorsqu'un ***sip_uac*** passe à l'état actif, c'est à dire lorsqu'il s'occupe d'une session, le ***sip_uac_mgr*** en est informé afin qu'un autre ***sip_uac*** soit créé.

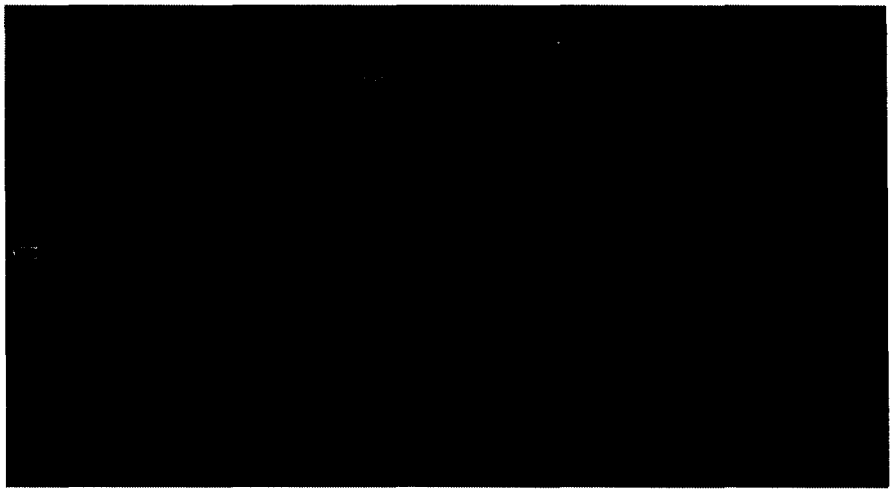


Figure 57 Modélisation du processus sip_uac, une session SIP au niveau client.

Lors de l'initialisation de ce processus, plusieurs variables sont initialisées. Le UAS (mandataire) qui sera contacté est sélectionné, une connexion passive est ouverte avec ***tpal*** et le ***sip_uac_mgr*** est informé que le ***sip_uac*** est prêt prendre un appel. Dans l'état ***LISTEN***, le ***sip_uac*** peut passer à l'état ***FORK*** sous deux conditions. Premièrement, une application a fait une demande d'établissement de session au ***sip_uac_mgr*** qui passe cette requête au ***sip_uac*** passif. Il peut également s'agir d'un paquet de type **SIP_RESPONSE** reçu d'un mandataire. Dans les deux cas, la requête est mise dans une file afin d'être traité dans un autre état.

En sortant de l'état ***FORK***, le processus vérifie si le paquet mis en fil d'attente dans l'état précédent est de type **RESPONSE** (paquet provenant d'un UAS) ou une requête a traitée (provenant d'une application locale). La transition est basée sur le type de message SIP devant être traité. Pour un message de type requête, le processus passe dans l'état ***REQ_PROC***, s'il s'agit d'un paquet de type **RESPONSE**, le processus passe dans l'état ***RESP_PROC***. Si la session SIP doit être terminée, le processus passe dans l'état ***LULL*** afin de terminer la session proprement.

Lorsque le processus doit traiter une requête provenant d'une application locale, via le ***sip_uas_mgr***, il passe à l'état ***REQ_PROC***. Dans cet état, si une connexion transport

active existe avec le mandataire, toutes les requêtes mises en file sont traitées. Si la connexion avec le mandataire n'est pas active, le ***sip_uac*** ferme la connexion passive, informe le ***sip_uac_mgr*** qu'il n'est plus à l'écoute, donc qu'un nouveau ***sip_uac*** doit être créé, et tente d'ouvrir une connexion active avec le mandataire. Le processus passe dans l'état ***WAIT*** en attendant que la connexion active soit réalisée. Si la connexion active réussi, le processus repasse à l'état ***REQ_PROC***, sinon le processus passe directement à l'état ***END***. La connexion transport avec le mandataire étant active, les requêtes misent en fil peut être transmise. S'il s'agit d'une requête **INVITE**, le nombre de lignes possible est vérifié, s'il y en a suffisamment, une requête est transmise au mandataire ; le nombre de réponses attendues est augmenté de 1 et le nombre de lignes disponibles est diminué de 1. S'il n'y a pas de ligne disponible, l'application est informée que la requête ne peut être exécutée. S'il s'agit d'une requête **BYE**, un paquet **BYE** est transmis au mandataire et le nombre de réponses attendues est augmenté de 1. Après la transmission d'une requête **INVITE** ou **BYE**, le processus repasse à l'état ***FORK*** et comme le service n'est pas complété, il repasse à l'état ***LISTEN***.

Le processus recevant une nouvelle réponse d'un mandataire ou requête de l'application, celle-ci est mise en file et le processus passe à l'état ***FORK*** qui déterminera s'il s'agit d'une requête ou d'une réponse. S'il s'agit d'un paquet **RESPONSE** provenant d'un mandataire, le processus passe dans l'état ***RESP_PROC***. Dans cet état, le type de **RESPONSE** reçu du mandataire est déterminé **(INVITE, CALL_CONNECT_ SUCCES, CALL_CONNECT_FAIL, CALL_DISCONNECT_FAIL, CALL_ DISCONNECT_ SUCCES ou BYE)** et une fonction correspondante traite adéquatement cette réponse.

Lorsqu'une réponse **INVITE** est traitée, lorsque le nombre de lignes disponibles est suffisant, l'appel est accepté en transmettant un message **CALL_CONNECT_SUCCES** au mandataire qui s'occupera de l'acheminer à l'appelant. Si aucune ligne n'est disponible, l'appel est refusé en transmettant un message **CALL_CONNECT_FAIL** au

mandataire. Lorsqu'une réponse **BYE** est traitée, un message **CALL_ DISCONNECT_ SUCCES** est transmis au mandataire et le nombre de ligne disponible est augmenté de 1. Lorsque qu'une réponse **CALL_CONNECT_SUCCES** est traitée, l'application est informée de la réussite de l'appel et le nombre de réponse attendue est décrémenté. Si une réponse **CALL_CONNECT_FAIL** est traitée, l'application est informée, le nombre de ligne disponible augmenté de 1 et le nombre de réponses attendues décrémenté. Si une réponse **CALL_DISCONNECT_SUCCES** est traitée, l'application est informée, le nombre de lignes disponibles augmenté de 1 et le nombre de réponse attendue est décrémenté. Si une réponse **CALL_DISCONNECT_FAIL** est traitée, le message est simplement détruit. Suite au traitement des messages de type **RESPONSE**, le processus repasse dans l'état ***FORK***.

Arrivant dans l'état ***FORK***, si le service est complété, c'est à dire que si l'état de l'appel est déconnecté et qu'aucune réponse n'est attendue, le processus passe dans l'état ***LULL*** qui passera à l'état ***CLOSE***. **Tpal** est informée que la connexion transport doit être terminée. Lorsque **tpal** informera le processus que la connexion transport est fermée, le processus passera dans l'état ***END*** qui s'occupera de libérer la mémoire et de détruire le processus.

4.2.6 Fonctionnement du processus sip_uas_mgr

Lors de l'initialisation d'un nœud, si celui-ci est configuré pour être serveur mandataire, le module application démarre le processus enfant ***sip_uas_mgr*** (Figure 58) modélisant le processus racine du mandataire SIP.

Figure 58 Modélisation du processus gestionnaire de session SIP au niveau du serveur

Lors de son initialisation, le processus ***sip_uas_mgr*** exécute quelques tâches. Il initialise le nombre d'appel actif à 0, initialise ses attributs de mandataire à partir des paramètres configurés sur le nœud. Un seul paramètre est nécessaire, il s'agit du nombre d'appels simultanés possibles. Comme la structure du mandataire sera modifiée pour ajouter les algorithmes de contrôle d'accès, cette fonction sera modifiée pour ajouter les attributs de mandataire relatif à ces algorithmes. Une structure de mémoire qui sera partagée avec les différents processus ***sip_uas*** est créée et un processus ***sip_uas*** est démarré afin qu'il soit en écoute passive pour la réception d'une requête. Le processus passe ensuite dans l'état ***WAIT***.

Le processus ***sip_uas_mgr*** sort de l'état ***WAIT*** uniquement lorsqu'un ***sip_uas*** devient actif. Le ***sip_uas_mgr*** est alors responsable de créer un autre processus ***sip_uas*** en écoute passive. ***Sip_uas_mgr*** est responsable de l'ensemble des paramètres et statistiques globales à l'ensemble des sessions SIP. Chaque session SIP utilisant le mandataire est établis par un ***sip_uas*** différent. Les différentes interactions entre les ***sip_uas*** et le ***sip_uas_mgr*** se font à l'aide de la structure de mémoire partagée. La compréhension de cette structure de mémoire est très importante car elle sera modifiée pour refléter les modifications du mandataire.

Structure de mémoire partagée

Lors de l'initialisation du processus ***sip_uas_mgr***, une structure de mémoire partagée est initialisée. La mémoire est allouée pour cette structure, un pointeur est utilisé pour accéder à cette structure. Lors de la création d'un nouveau processus ***sip_uas***, le pointeur sur cette structure est passé en paramètre. Cette structure, nommée SIPT_UAS_PtC_Mem (Parent-to-Child memory) se définie comme suit :

Quelques variables de fonctionnement

- int* strm_to_tpal_ptr; un pointeur sur le *stream* de sortie
- int* open_lines_count_ptr; un pointeur sur le nombre de ligne
- int* active_calls_count_ptr; un pointeur sur le nombre d'appel

Quelques manipulateurs de statistique

- Stathandle call_setup_requests_stathandle;
- Stathandle calls_connected_stathandle;
- Stathandle calls_rejected_stathandle;
- Stathandle active_calls_stathandle;
- Stathandle call_duration_stathandle;

4.2.7 Fonctionnement du processus sip_UAS

Le processus ***sip_uas*** (Figure 59) est responsable d'une session SIP au niveau du mandataire. Ce processus est créé pour chaque appel. Le premier processus ***sip_uas*** est créé lors de l'initialisation du ***sip_uas_mgr***. ***Sip_uas*** étant responsable de la signalisation d'un appel, il doit y en avoir au moins un à l'écoute pour les appels entrant. Lorsqu'un ***sip_uas*** passe à l'état actif, c'est à dire lorsqu'il s'occupe d'un appel, le ***sip_uas_mgr*** en est informé afin qu'un autre ***sip_uas*** soit créé.

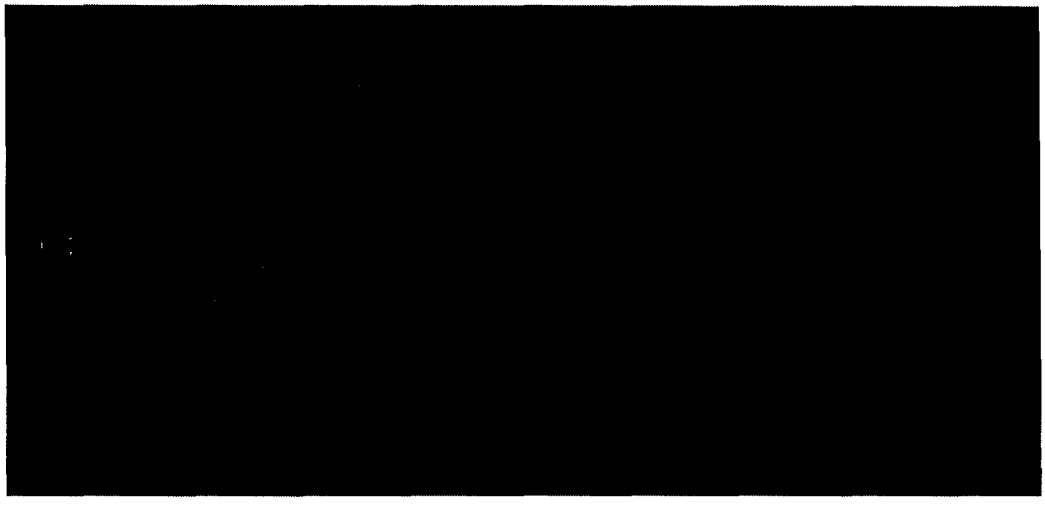


Figure 59 Modélisation du processus sip_UAS, une session SIP au niveau serveur

Lors de l'initialisation, le pointeur sur la structure mémoire *Parent-to_Child* est obtenu, quelques variables sont initialisées et une connexion passive est réalisée afin qu'un ***sip_uac*** puisse se connecter.

Dans l'état ***OPEN***, si le processus reçoit une indication qu'une connexion active est en train d'être établie au niveau transport, le ***sip_uas_mgr*** est informé afin de créer un nouveau ***sip_uas***. Si la connexion est établie au niveau transport, le nombre de session active de la structure de mémoire partagée est incrémenté et le processus passe dans l'état ***LISTEN***. Cinq conditions sont possibles pour sortir de l'état ***LISTEN***.

- **REQUEST :** Paquet SIP de type **REQUEST** est reçue
- **ACK :** Paquet SIP de type **ACK** est reçue
- CONN_CLOSED : **Tpal** informe le processus que la connexion transport est terminée.
- FIN_RCVD : **Tpal** informe le processus qu'un paquet transport de type FIN est arrivé.
- ESTAB : **Tpal** informe le processus qu'une connexion transport a été établie avec le ***sip_uac*** de l'appelé, le message SIP peut donc être acheminé.

Un message de type **REQUEST** a été reçu par le ***sip_uas***, il passe donc dans l'état ***REQ_PROC***. S'il s'agit d'un paquet **INVITE**, le nombre de ligne disponible est vérifié. Le type de message est changé pour **RESPONSE**, c'est à dire le type de message allant d'un ***sip_uas*** vers un ***sip_uac***. Le manipulateur d'appel est sauvegardé, le nombre de ligne disponible décrémenté et la requête est relayé au ***sip_uac*** de destination. Si le nombre de ligne n'était pas suffisant, un paquet SIP de type **CALL_CONNECT_FAIL** est transmis ***sip_uac*** source. S'il s'agit d'un **REQUEST BYE**, le type de message est changé pour **RESPONSE** et est transmis au ***sip_uac*** de destination.

Un message de type **ACK** a été reçu par le ***sip_uas***, il passe donc dans l'état ***ACK_PROC***. Quatre messages de type **ACK** sont possibles, il s'agit de **CALL_CONNECT_SUCCESS, CALL_CONNECT_FAIL, CALL_DISCONNECT_SUCCESS** et **CALL_DISCONNECT_FAIL**, la fonction correspondante au message reçu sera exécutée.

- **CALL_CONNECT_SUCCESS :** changer le type de message pour **RESPONSE**, obtenir le temps auquel l'appel a été établi, incrémenté le nombre d'appel actif, mettre à jour les statistiques et transmettre le paquet au destinataire.

- **CALL_CONNECT_FAIL** : changer le type de message pour **RESPONSE**, incrémenté le nombre de ligne disponible, mettre à jour les statistiques et transmettre le paquet au destinataire.
- **CALL_DISCONNECT_SUCCESS** : changer le type de message pour **RESPONSE**, calculer le temps qu'a durée l'appel, incrémenté le nombre de ligne disponible, mettre à jour les statistiques et transmettre le paquet au destinataire.
- **CALL_DISCONNECT_FAIL** : changer le type de message pour **RESPONSE** et transmettre le paquet au destinataire.

Si ***tpal*** indique que la connexion transport a été fermée, le processus passe dans l'état ***FREE***. Le processus ajuste les variables, libère la mémoire et passe dans l'état ***END*** qui détruira le processus. Si la ***tpal*** indique qu'un paquet transport de type FIN a été reçu, c'est qu'un ***sip_uac*** veut fermer sa connexion transport active, le processus demande donc à ***tpal*** de terminer la connexion. Finalement, si la couche transport vient d'établir une connexion, les paquets peuvent donc être transmis sur celle-ci.

4.3 Modification de processus

Le but de ce travail étant la modélisation et l'analyse d'algorithmes de contrôle d'accès lors de l'établissement d'un appel à l'aide de SIP, certains processus devront être modifiés ou ajoutés afin d'implanter ces algorithmes. La modification de processus comprend l'ajout de paramètre de configuration, de statistiques et de code implantant les dits algorithmes.

4.3.1 Ajout d'un paramètre de configuration

La description de la procédure à suivre pour ajouter un paramètre de configuration est réalisée pour le processus ***sip_uas_mgr***. Le but est simple, il s'agit d'ajouter un paramètre permettant de configurer le mandataire SIP pour que l'acceptation de l'appel ne soit pas basée sur le nombre de lignes disponibles mais sur le contrôle d'accès qui

sera implanté. La première étape est d'éditer le processus auquel on veut ajouter un paramètre de configuration à l'aide de l'éditeur de processus (Figure 60).

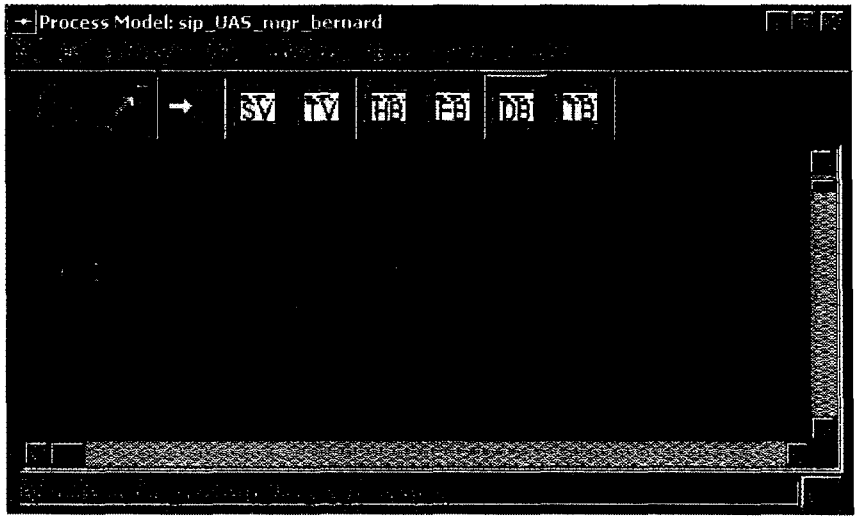


Figure 60 Éditeur de processus

Les différents paramètres de configuration qui sont paramétrables par l'usager sont disponibles dans le menu *Interface → Model Attributes.*

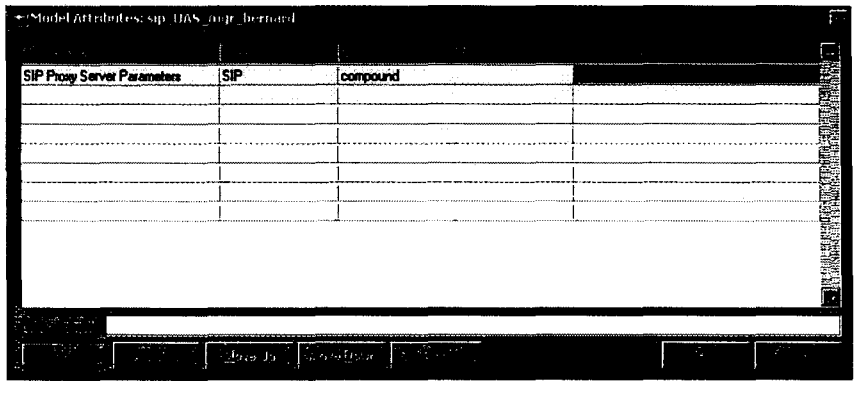


Figure 61 Paramètres de configuration du processus sip_UAS_mgr

Dans la fenêtre d'attributs (Figure 61), on peut voir qu'il y a un seul paramètre de configuration appelé *SIP Proxy Server Parameters.* Ce paramètre fait partie du groupe SIP et est de type *compound.* Le type de paramètre pourrait être *integer* ou *char* ;

compound est un type particulier qui est en fait une structure de paramètres. En sélectionnant *SIP Proxy Server Parameters* et en cliquant sur le bouton *Edit Properties* nous obtenons la structure de ce *compound* qui est composé de deux paramètres (Figure 62).

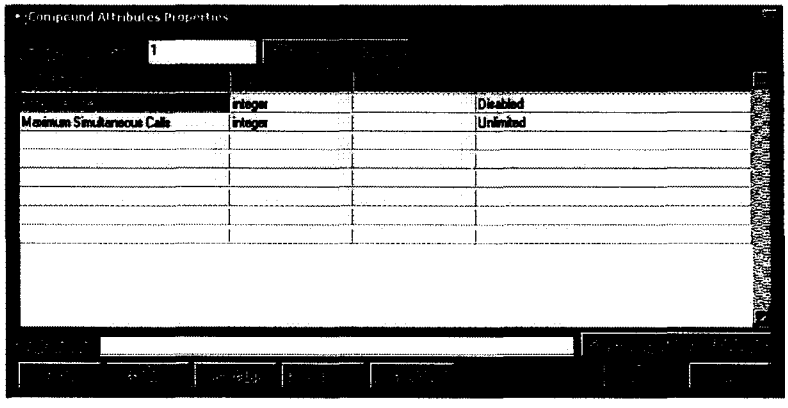


Figure 62 Configuration d'un paramètre de type compound

Le *compound SIP Proxy Server Parameters* est constitué de deux paramètres. *Proxy Service* permet de configurer si le processus UAS sera actif et *Maximum Simultaneous Calls* permet de configurer le nombre d'appels simultanés que pourra gérer le mandataire SIP. Chacun de ces paramètres peut prendre certaines valeurs préalablement configurer. Pour configurer des valeurs préalables, il faut sélectionner le paramètre et éditer ces attributs (Figure 63).

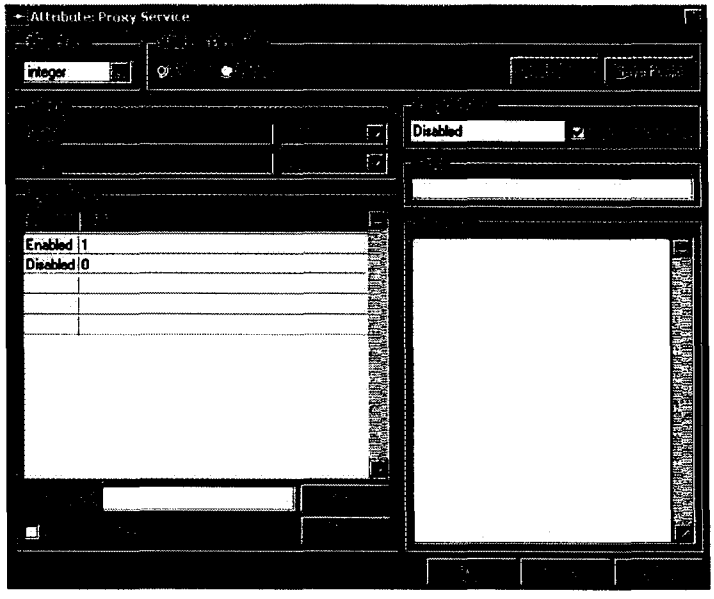


Figure 63 Configuration d'un paramètre de type autre que compound

Le paramètre *Proxy Service* est de type Int, il peut donc prendre des valeurs entières. Deux valeurs sont configurées, 1 et 0. À chaque valeur est associé un *Symbol* afin de rendre la configuration plus simple par l'usager. Dans la fenêtre de configuration, il sera possible de sélectionner une des deux valeurs, *Enabled* ou *Disabled.* Dans cette fenêtre, il est également possible de configurer la valeur à prendre par défaut, d'ajouter un commentaire décrivant le rôle de ce paramètre et une petite case appelée *Allow other values* permet ou non d'assigner une autre valeur que celle préalablement définies. Dans le cas de la variable Proxy Service, permettre à l'usager de sélectionner autre chose que *Enabled* ou *Disabled* ne ferait aucun sens.

Pour nos besoins nous ajouterons un paramètre à ce processus. Ce paramètre sera ajouté au *Compound SIP Proxy Server Parameters* et s'appellera ***SIP Access Control*** (Figure 64). Ce paramètre permettra de configurer le mandataire SIP pour qu'il accepte ou refuse d'établir un appel sur la base du contrôle d'admission sélectionné. Le contrôle d'accès par défaut est de type MSC (*Maximum Simultaneous Calls*). Un second type de contrôle d'accès, nommé ABW (*Available BandWidth*) est ajouté. Le paramètre ***SIP Access Control*** permet de choisir le type de contrôle d'accès. Un paramètre Maximum

Simultaneous Calls existe déjà si le contrôle d'accès sélectionné est de ce type. Si le contrôle d'accès sélectionné est ABW, un paramètre du même nom est ajouté afin de configurer la quantité de bande passante disponible (non montré dans la Figure 64).

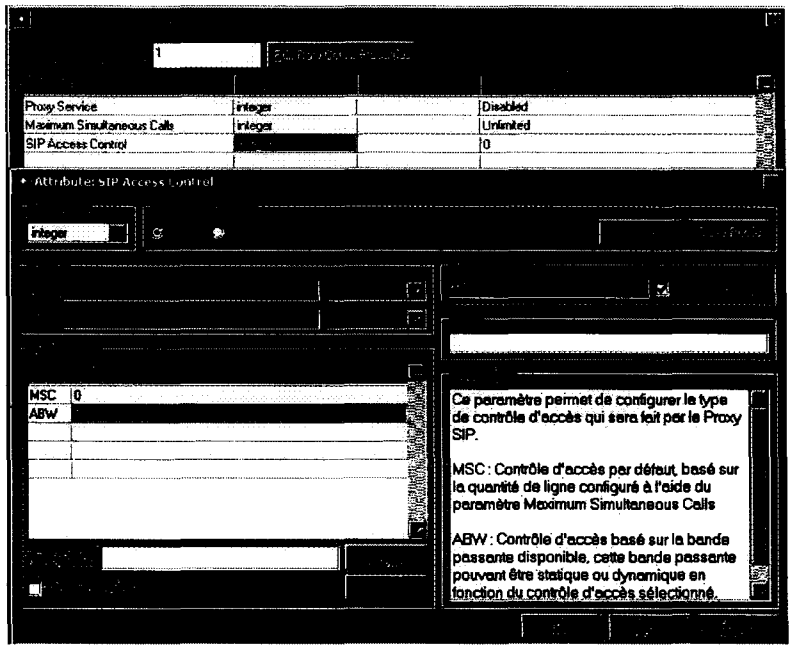


Figure 64 Ajout du paramètre SIP Access Control

Les paramètres de configuration du processus *sip_uas_mgr* étant modifiés, le travail ne s'arrête pas à ce niveau. Les paramètres de configuration seront disponibles via l'interface d'un nœud (*Right-Click, Edit Attributes*), il faut donc modifier l'interface de ce nœud. Dans tous les nœuds existants, plusieurs sont en fait des nœuds dérivés. Par exemple, le *SIP Proxy Server* est un nœud dérivé de *ethernet_server_adv*. L'interface du nœud dérivé ne peut être modifiée sans que celle du nœud parent ne le soit. Le nœud *ethernet_server_adv* est donc édité à l'aide de l'éditeur de nœud (Figure 65).

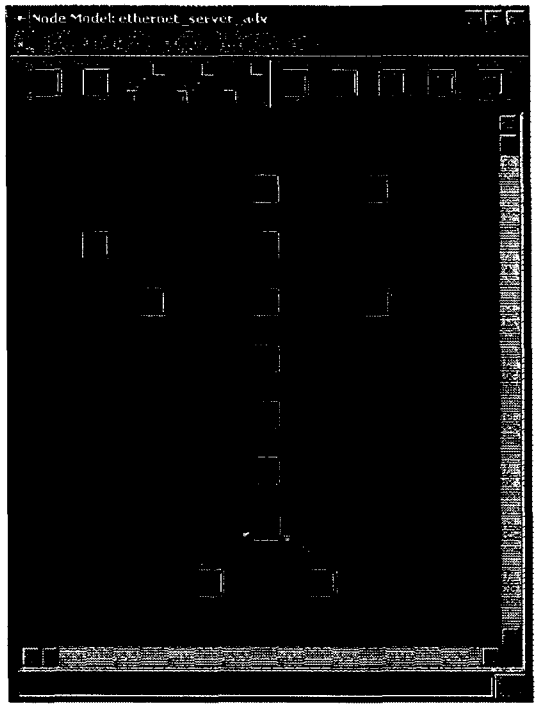


Figure 65 Noeud ethernet_server_adv

À l'Aide du menu *Interfaces* → *Node Interfaces*, il est possible de configurer l'ensemble des paramètres de configuration disponibles pour ce nœud. Avant de pouvoir configurer les paramètres qui seront disponibles à l'usager, il faut comprendre comment ceux-ci sont mis à la disponibilité de ce dernier.

Un nœud est constitué de modules, à chaque module correspond un processus racine. Par exemple, le processus racine du module application est ***gna_clsvr_mgr***. Pour un processus racine, il est possible de déclarer quels sont les processus enfants de cette racine à l'aide du menu *File* → *Declare Child Process Models*. Le processus *sip_uas_mgr* que nous venons de modifier est enfant du processus racine ***gna_clsvr_mgr***. La configuration du processus racine et de ces enfants étant faites, cela permet de regrouper les différents paramètres de ces processus afin qu'il soient disponibles au niveau du module **application** (Figure 66).

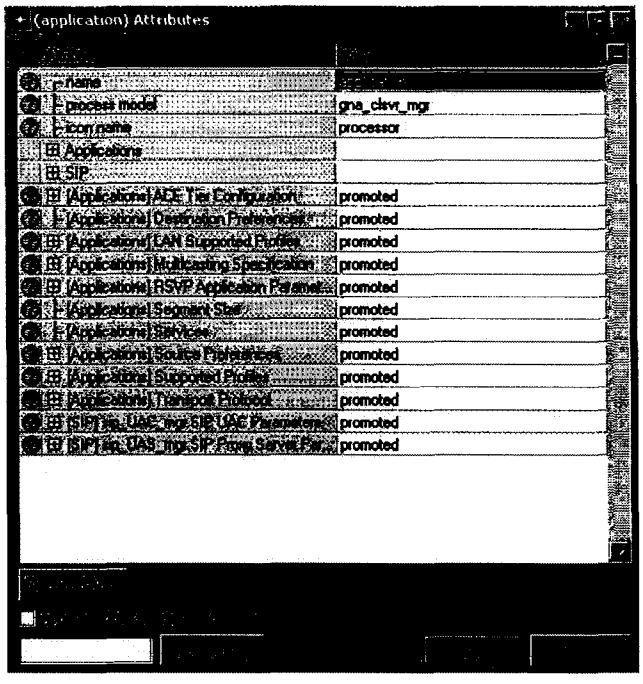


Figure 66 Paramètres de configuration du module applications

Ces différents paramètres proviennent du processus ***gna_clsvr_mgr*** et de tous ses enfants déclarés. ***Sip_uas_mgr*** étant déclaré enfant du processus racine du module application, on y retrouve ses paramètres. La dernière ligne de l'image précédente est *[SIP] sip_UAS_mgr.SIP Proxy Server Parameters*. La valeur de la majorité des attributs est ***Promoted***. Cela est nécessaire afin que ces attributs soient disponibles au niveau du nœud. Si *[SIP] sip_UAS_mgr.SIP Proxy Server Parameters* n'était pas ***Promoted***, il ne serait pas disponible au niveau du nœud. Cette façon de faire permet de faire un processus ***gna_clsvr_mgr*** et des processus enfants généraux, qui pourront être adaptés à plusieurs situations en fonction des paramètres mis en disponibilité. Si certains paramètres ne sont pas désirés au niveau du nœud, il est possible de leur donner une valeur par défaut plutôt que de les ***Promoted***.

Maintenant que nous sommes certains que les attributs du *sip_uas_mgr* seront disponibles au niveau du nœud, il faut configurer l'interface du nœud à l'aide du menu *Interfaces* → *Node Interfaces* de l'éditeur de nœud (Figure 65).

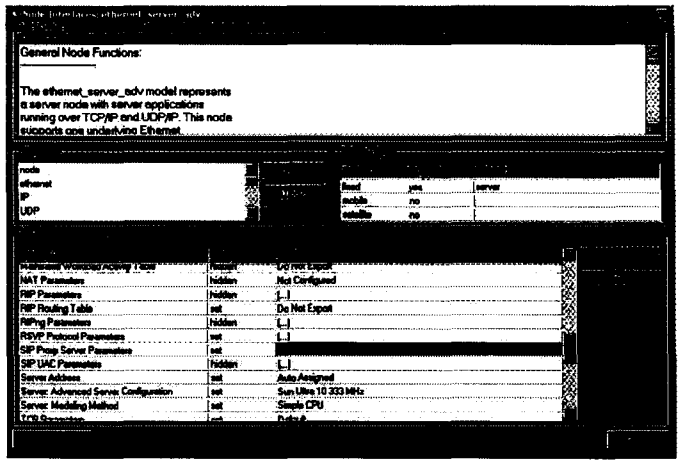


Figure 67 Configuration de l'interface du nœud

Dans la fenêtre permettant de configurer l'interface du nœud (Figure 67), tous les attributs qui ont été *Promoted* de tous les modules sont disponibles. Il est possible de changer le nom de ces paramètres pour un nom plus représentatif ou simplifié, il est également possible de faire un réglage par défaut en choisissant le *Status Set* et en configurant les valeurs initiales. Le nœud parent étant modifié pour tenir compte du nouveau paramètre de configuration, tous les noeuds qui sont dérivés de celui-ci, dont le *SIP Proxy Server*, ont maintenant accès à ce paramètre (Figure 68).

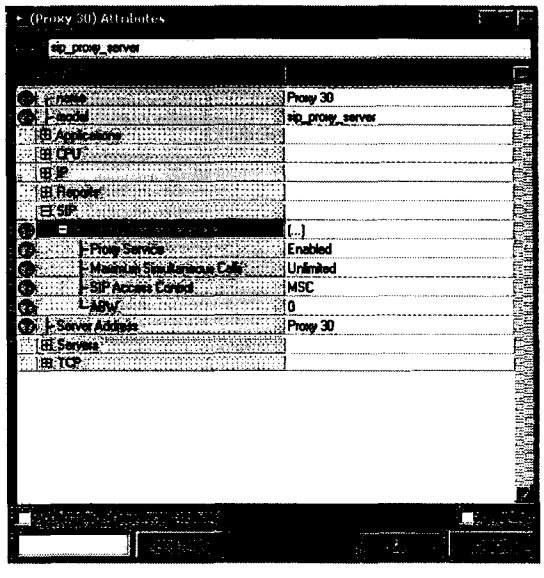


Figure 68 Attributs du processus sip_uas_mgr

4.3.2 Ajout d'une statistique à un processus existant

La description de la procédure à suivre pour ajouter une statistique est réalisée pour le processus ***sip_uas_mgr***. Le but est simple, il s'agit d'ajouter une statistique permettant de déterminer la quantité d'appel rejeté par le UAS par manque de bande passante. Dans le processus original, la statistique *Rejected Calls* ne fait pas de distinction entre les appels rejetés par le UAS et les appels rejetés par les UAC. La statistique ajoutée permettra donc d'analyser le taux de rejet des différents algorithmes de contrôle d'accès, seul les appels étant rejetés par le UAS étant significatifs pour cette nouvelle statistique.

La première étape est d'éditer le processus qui génèrera la statistique, dans notre cas, le ***sip_uas_mgr*** (Figure 60). À l'aide du menu *Interfaces* → *Local statistics* une fenêtre permettra de configurer les statistiques disponibles pour le processus (Figure 69).

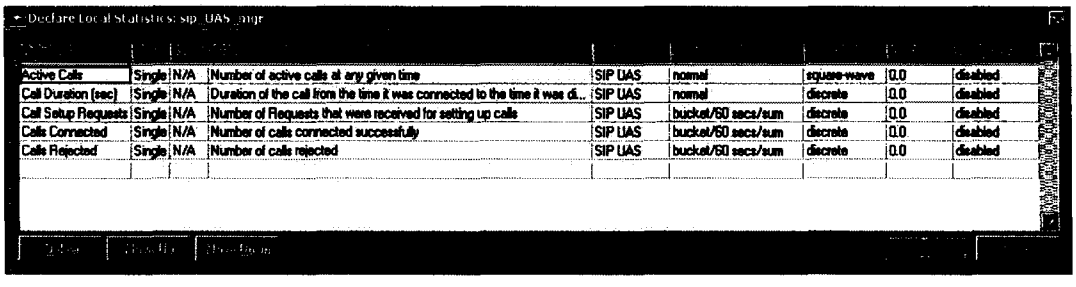


Figure 69 Statistiques relatives au processus sip_UAS_mgr

Ce tableau comporte le nom des statistiques qui seront disponibles pour la simulation, une description de chaque statistique, le groupe auquel appartient la statistique, son mode de capture ainsi que la façon dont la statistique sera représentée dans le graphique suite à la simulation. Dans notre cas, nous ajoutons la statistique *Calls Rejected by UAS* au groupe SIP UAS. La description est : « Nombre d'appel rejeté par le mandataire par manque de bande passante ». Il reste à configurer le mode de capture ainsi que le type de graphique généré. En cliquant sur le mode de capture, une fenêtre permet de configurer ce mode (Figure 70).

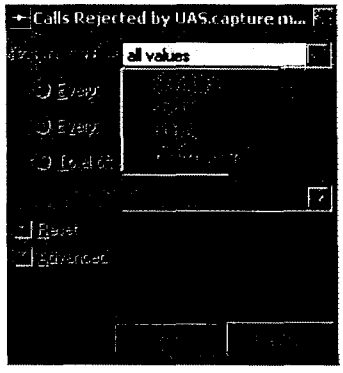


Figure 70 Mode de capture d'une statistique

Quatre modes de capture sont possibles :

- *All values* : Toutes les mises à jour de la statistique sont enregistrées
- *Sample* : Enregistre seulement certaines mises à jour. À toutes les $n^{ième}$ mise à jour ou à tous les T secondes.
- *Bucket* : Considère un ensemble de points appelés *bucket* et enregistre une valeur représentative de ce *bucket*. Un *bucket* peut être spécifié en tant qu'intervalle de temps ou par le nombre de mise à jour qu'il contient. La valeur retenue pour ce *bucket* peut être la somme/temps, la valeur minimale ou maximale, un échantillon moyen, la moyenne du temps ou une somme.
- *Glitch removal* : Lorsque plusieurs mises à jour arrivent en même temps, seul la dernière est retenue.

Pour notre statistique, le mode de capture sélectionné est *bucket*, chaque *bucket* représentant 60 sec et pour cette période de 60 secondes, la valeur retenue sera la somme des mise à jour, donc le nombre d'appel rejeté par le UAS par période de 60 secondes. La représentation de la statistique dans le graphique sera de type *Discret* (Figure 71).

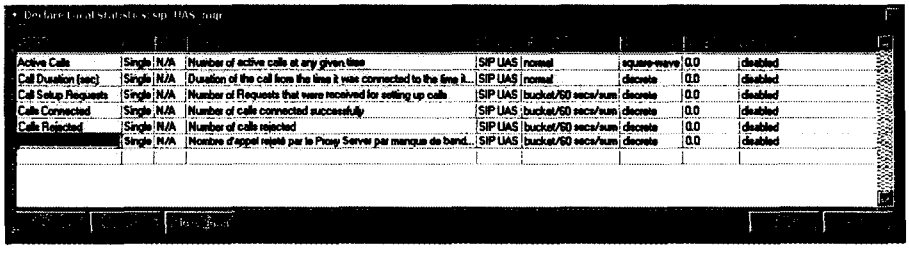


Figure 71 Ajout de la statistque Calls Rejected by UAS

Il faut maintenant rendre cette statistique disponible au niveau du nœud. Comme pour l'ajout d'un paramètre de configuration, les statistiques sont reliées à un processus. Tous les processus intervenant dans un nœud peuvent générer des statistiques, cependant, il n'est pas nécessaire que toutes ces statistiques soient disponibles au niveau du noeud.

En éditant le nœud parent, soit *ethernet_server_adv* (Figure 65), à l'aide de l'éditeur de nœud et en sélectionnant le menu *Interfaces* → *Node Statistics* une fenêtre permet de sélectionner les statistiques qui seront disponibles au niveau du nœud (Figure 72). Par défaut, une statistique n'est pas disponible, il faut donc ajouter celle nouvellement créer à ceux disponibles au niveau du nœud.

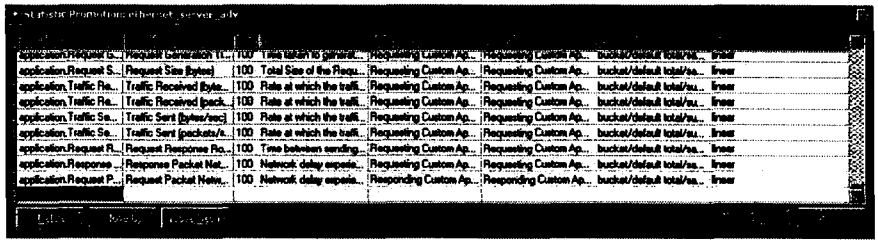


Figure 72 Statistiques disponibles au niveau du nœud

En cliquant sur une ligne, une fenêtre apparaît avec l'ensemble des statistiques qui existent mais qui ne sont pas disponibles au niveau du nœud (Figure 73). Il s'agit de *Promoted* la statistique nouvellement créer.

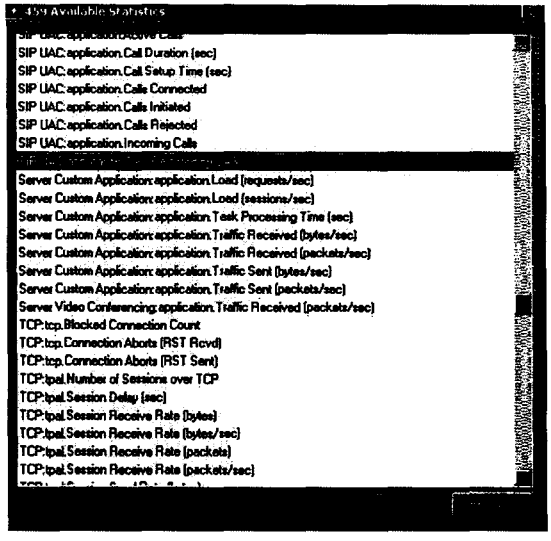


Figure 73 Ensemble des statistiques non disponibles au niveau du nœud

La nouvelle statistique peut maintenant être sélectionnée dans le nœud parent ou dans tous les nœuds dérivés de ce dernier (*Right-Click, Select Statistics*).

4.3.3 Configuration du premier cas de simulation

La première étape de la configuration d'un nouveau contrôle d'accès ou de toute autres fonctions est d'obtenir la valeur des paramètres configurés par l'usagé. Ces valeurs sont misent dans des variables locales. Il faut également associer les statistiques sélectionnées par l'usager à des variables locales.

Variables d'états et associations aux attributs configurés en 3.3.2

La statistique sélectionnée est associée à une variable qui sera utilisée dans le code pour la manipulation de la statistique. Il s'agit d'une variable de type ***Stathandle***. Celle-ci est définie en cliquant sur l'icône **SV** (*states variables*) (Figure 60). C'est ici que tous les variables d'états qui seront utilisées tout au long de la vie du processus sont définies. Une variable de type ***Stathandle*** s'appelant *UAS_calls_rejected_by_uas_stathandle* est définie (Figure 74). Une variable *int bande_passante* est également définie pour contenir la quantité de bande passante configurée par l'usager.

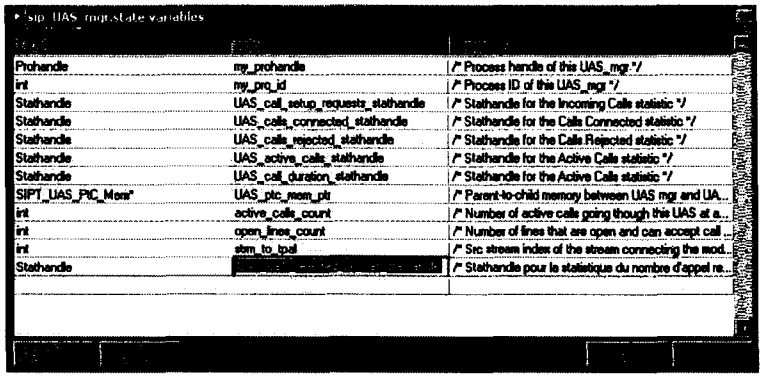


Figure 74 Ajout d'une variable d'état de type Stathandle

Lors de l'initialisation du processus il faut enregistrer cette nouvelle statistique. En fait, il s'agit d'associer la variable de type *Stathandle* *UAS_calls_rejected_by_uas_stanthandle* avec la statistique *Calls Rejected by UAS* définie à l'étape précédente. La fonction ***UAS_calls_rejected_by_uas_stathandle = op_stat_reg ("SIP UAS.Calls Rejected by UAS", OPC_STAT_INDEX_NONE, OPC_STAT_LOCAL)*** permet de réaliser cette association. La fonction *op_stat_reg()* enregistre la statistique *Calls rejected by UAS* du groupe SIP UAS et retourne un manipulateur de statistique à la variable *UAS_calls_rejected_by_uas_stathandle*. Les paramètres de la fonction *op_stat_reg()* sont la statistique enregistrée ainsi que deux autres paramètres. OPC_STAT_INDEX_NONE signifie que la statistique n'a pas de dimension, il s'agit de la seule valeur possible pour ce paramètre. OPC_STAT_LOCAL signifie qu'il s'agit d'une variable locale, c'est à dire qu'elle mise à jour uniquement par le processus local. Ce paramètre pourrait prendre la valeur OPC_STAT_GLOBAL, dans ce cas, il n'y aurait qu'une statistique à mettre à jour pour tous les processus du même type existant dans un réseau.

De façon similaire, un ensemble de fonctions permettent de lire la valeur d'un attribut configuré par l'usager (ex : *Available Bandwidth* = 100000) et de copier cette valeur dans une variable d'état qui peut être utilisée par le code.
Les paramètres suivants ont été ajoutés :

- ***SIP Access Control*** : Est-ce que le contrôle d'accès sera basé sur le nombre de ligne disponible ou sur la bande passante disponible.
- **ABW** : Il s'agit de la bande passante disponible pour les appels.

Les statistiques suivantes ont été ajoutées :

- ***Calls rejected by UAS*** : Il s'agit du nombre d'appel rejeté par le Proxy, par manque de ligne disponible ou par manque de bande passante disponible.
- ***Available Bandwidth*** : Il s'agit de la bande passante disponible.

Lors de l'initialisation du ***sip_uas_mgr***, les fonctions suivantes sont appelées. Elles devront être modifiées pour considérer les statistiques ou les paramètres de configuration ajoutés, tel que décrit au début de cette section.

- *sip_UAS_mgr_attrs_parse_()* : permet de copier les paramètres configurés par l'usager dans des variables accessibles par le processus.
- *sip_UAS_mgr_stats_register_()* : permet d'enregistrer les différentes statistiques sélectionnées par l'usager et accessibles par le processus.
- *sip_UAS_mgr_ptc_mem_create_()* : créer la structure de mémoire partagée. Les nouveaux paramètres et statistiques seront utilisés par les enfants, la structure de mémoire partagée devra donc être modifiée pour permettre un pointeur vers ces nouvelles variables partagées.

Pour que l'ensemble des processus ***sip_uas*** enfants puisse accéder aux variables d'état et aux statistiques configurées dans le processus ***sip_uas_mgr***, la structure de mémoire partagée a été modifiée.

Il ne reste qu'à implanter le code implantant le contrôle d'accès dans les processus *sip_uas* enfants.

4.3.4 Modification du processus sip_UAS

Le fonctionnement normal du ***sip_uas*** est le suivant. Un paquet **INVITE** est reçu, par un ***sip_uas***, celui-ci informe le ***sip_uas_mgr***. Si la quantité de bande passante disponible est suffisante, alors la requête est transmise au ***sip_uac*** de destination. Le mandataire ne fait pas de différence entre les appels qui demeureront locaux et ceux qui devront transiter sur le réseau. Le scénario utilisé impose à chaque téléphone de ne pouvoir qu'appeler les usagers de l'autre réseau. De cette façon chaque appel que le mandataire devra réaliser devra passer par la dorsale MPLS.

Les seuls états ou le contrôle d'admission est réalisé sont les états ***REQ_PROC*** et ***ACK_PROC*** (Figure 59). Ces deux états appellent les fonctions *sip_UAS_req_process (pk_ptr)* et *sip_UAS_ack_process (pk_ptr)* qui devront être modifiées pour supporter le nouveau contrôle d'accès et les nouvelles statistiques.

La fonction *sip_UAS_**req**_process (pk_ptr)* peut traiter deux types de paquets, INVITE et BYE. Dans ces deux cas, le contrôle d'accès et les statistiques ajoutées seront affectées. Dans le cas d'un paquet INVITE, la fonction *sip_UAS_**invite**_req_process (pk_ptr)* est appelée, dans le cas d'un paquet BYE, la fonction *sip_UAS_**bye**_req_process (pk_ptr)* est appelée.

La fonction *sip_UAS_**ack**_process (pk_ptr)* peut traiter quatre types de paquets, CALL_CONNECT_SUCCES, CALL_CONNECT_FAIL, CALL_DISCONNECT_ SUCCES, CALL_DISCONNECT_FAIL. Dans le cas d'un DISCONNECT_FAIL, aucun traitement n'est nécessaire. Les fonctions devant être modifiées sont : *sip_UAS_**connect_success** (pk_ptr)*, *sip_UAS_**connect_fail** (pk_ptr)*, *sip_UAS_ **disconnect_success** (pk_ptr)*. Voici comment chacune des fonctions du ***sip_uas*** est modifiée en fonction du type de paquet afin de supporter le contrôle d'accès de type bande passante disponible.

- Un paquet **INVITE** est reçu :
    - Vérifier si la bande passante disponible est suffisante.
    - Si la bande passante disponible est suffisante, la diminuée de la capacité nécessaire et transmettre le paquet **INVITE** à la destination.
    - Sinon, transmettre **CALL_CONNECT_FAIL** à la source et mettre à jour la statistique du nombre d'appel rejeté par manque de bande passante.
- Un paquet **BYE** est reçu :
    - Transmettre au destinataire
- Un paquet **CALL_CONNECT_SUCCES** est reçu :
    - Diminuer la **statistique** de bande passante disponible. Cette statistique n'est modifiée que si l'appel aboutie.
    - Transmettre à la destination.
- Un paquet **CALL_CONNECT_FAIL** est reçu :
    - Lors de la réception d'un paquet **INVITE**, la bande passante disponible est décrémentée afin de réserver la bande passante à cet appel. Si l'appel est **FAIL**, la bande passante est de nouveau disponible.
    - Transmettre à la destination.
- Un paquet **CALL_DISCONNECT_SUCCES** est reçu :
    - La bande passante disponible est augmentée d'autant
    - La **statistique** de bande passante disponible est mise à jour.

Ce chapitre a décrit comment modifier les processus impliqués dans le contrôle d'accès réalisé par le mandataire lors de l'établissement d'une session SIP. Pour ce faire, un contrôle d'accès basé sur une bande passante disponible a été ajouté. Le prochain chapitre fera une description exhaustive des algorithmes de contrôle d'accès et d'évaluation des besoins qui seront étudiés. Ces algorithmes ont été implantés afin d'être simulés et analysés dans le dernier chapitre.

CHAPITRE 5

ÉTUDES DES ALGORITHMES

Le précédent chapitre a décrit comment implanter le code nécessaire à la modélisation des algorithmes. Il reste maintenant à décrire et implanter ces algorithmes. Ce chapitre fera une description complète des quatre algorithmes modélisés afin d'en avoir une compréhension qui en permettra un étude adéquate. Le premier est un algorithme ne faisant qu'un contrôle d'accès basé sur la bande passante disponible. Le second est l'algorithme de contrôle d'accès et de prédiction des besoins décrits dans [Bo]. Le troisième algorithme est une simplification de l'algorithme précédant. Finalement, un algorithme de contrôle d'accès, dont l'évaluation des besoins est basée sur une mesure du taux de trafic combiné à la théorie d'Erlang est décrit.

5.1 Présentation des algorithmes

5.1.1 Algorithme ABW (Available BandWidth)

Cet algorithme possède un seul paramètre configurable, il s'agit de la bande passante réservée. Cette bande passante est constante tout au long de la simulation et n'est pas modifiée. Chaque appel SIP nécessite 64000 bps. Si les ressources sont suffisantes, l'algorithme ABW accepte l'appel, sinon il le refuse.

Comme cet algorithme ne fait aucune modification de la bande passante, deux scénarios sont possibles. Si beaucoup de bande passante est disponible, le taux de rejet sera faible et l'efficacité d'utilisation également. À l'inverse, si la bande passante est petite, l'efficacité sera meilleure mais le taux de rejet élevé. Dans les deux cas, comme les besoins changent avec le temps, les taux obtenus varieront également et leur moyenne dépendra du profil de trafic.

5.1.2 Algorithme DBP (Dynamic Bandwidth Prediction)

Cet algorithme utilise le contrôle d'accès pour établir une prédiction des besoins en bande passante. S'il refuse un certain nombre d'appels, il considère qu'il est temps d'augmenter la bande passante. S'il accepte un certain nombre d'appels et que la bande passante n'est pas assez utilisée, il considère qu'il faut la diminuer. L'algorithme utilise plusieurs paramètres. Deux paramètres déterminent le nombre d'appels acceptés ou refusés avant de négocier la bande passante à la hausse ou à la baisse. Deux autres paramètres déterminent les niveaux de bande passante considérés comme étant une sous utilisation ou une sur utilisation. Plusieurs variables sont utilisées mais leur rôle n'est pas de modifier le comportement de l'algorithme.

Variables utilisées par l'algorithme

- B_{CON} : Bande passante réservée ;
- B_{CON-S} : Pas de modification de B_{CON} ;
- B_{REQ} : Bande passante requise pour un appel ;
- B_{OCC} : Bande passante utilisée ;
- Inc: Compteur déterminant le moment où augmenter la bande passante ;
- Dec: Compteur déterminant le moment où diminuer la bande passante.

Paramètres de l'algorithme

- N_{INC} : Quantité d'appels refusés avant d'augmenter la bande passante ;
- N_{DEC} : Quantité d'appels acceptés avant de diminuer la bande passante ;
- B_{ACC} : En % de B_{CON}. Paramètre utilisé pour évaluation à la hausse de B_{CON} ;
- B_{DEC} : En % de B_{CON}. Paramètre utilisé pour évaluation à la baisse de B_{CON} ;
- Rnd : Nombre déterminant la probabilité qu'un appel soit rejeté.

Algorithme

La description complète de l'algorithme se retrouve à la Figure 75. Il peut également se résumer en une suite d'actions à poser, ces actions étant fonction de la bande passante nécessaire lorsqu'une nouvelle requête arrive (Figure 76). La bande passante nécessaire est égale à la bande passante occupée par les sessions déjà autorisées + la bande passante requise pour accepter un nouvel appel. Si la bande passante nécessaire est supérieure à

la bande passante réservée (B_{CON}), alors l'appel est refusé, le compteur est mis à 0, le compteur Inc est incrémenté et si sa valeur est supérieure à N_{INC}, B_{CON} est augmenté. Lorsque la bande passante nécessaire est inférieure à B_{DEC}, l'appel est accepté, le compteur Inc est mis à 0, le compteur Dec est incrémenté et si sa valeur est supérieure à N_{DEC}, B_{CON} est diminué. Si la bande passante nécessaire se situe entre B_{ACC} et B_{DEC}, l'appel est accepté et le compteur Inc est mis à 0. Finalement, si la bande passante nécessaire se situe entre B_{ACC} et B_{CON}, un coefficient P_{REJ} est calculé. Si P_{REJ} est supérieur ou égale à Rnd, l'appel est refusé, le compteur Dec mis à 0, le compteur Inc est incrémenté et si sa valeur est supérieure à N_{INC}, B_{CON} est augmenté. Si P_{REJ} est inférieur à Rnd, l'appel est accepté et les deux compteurs mis à 0.

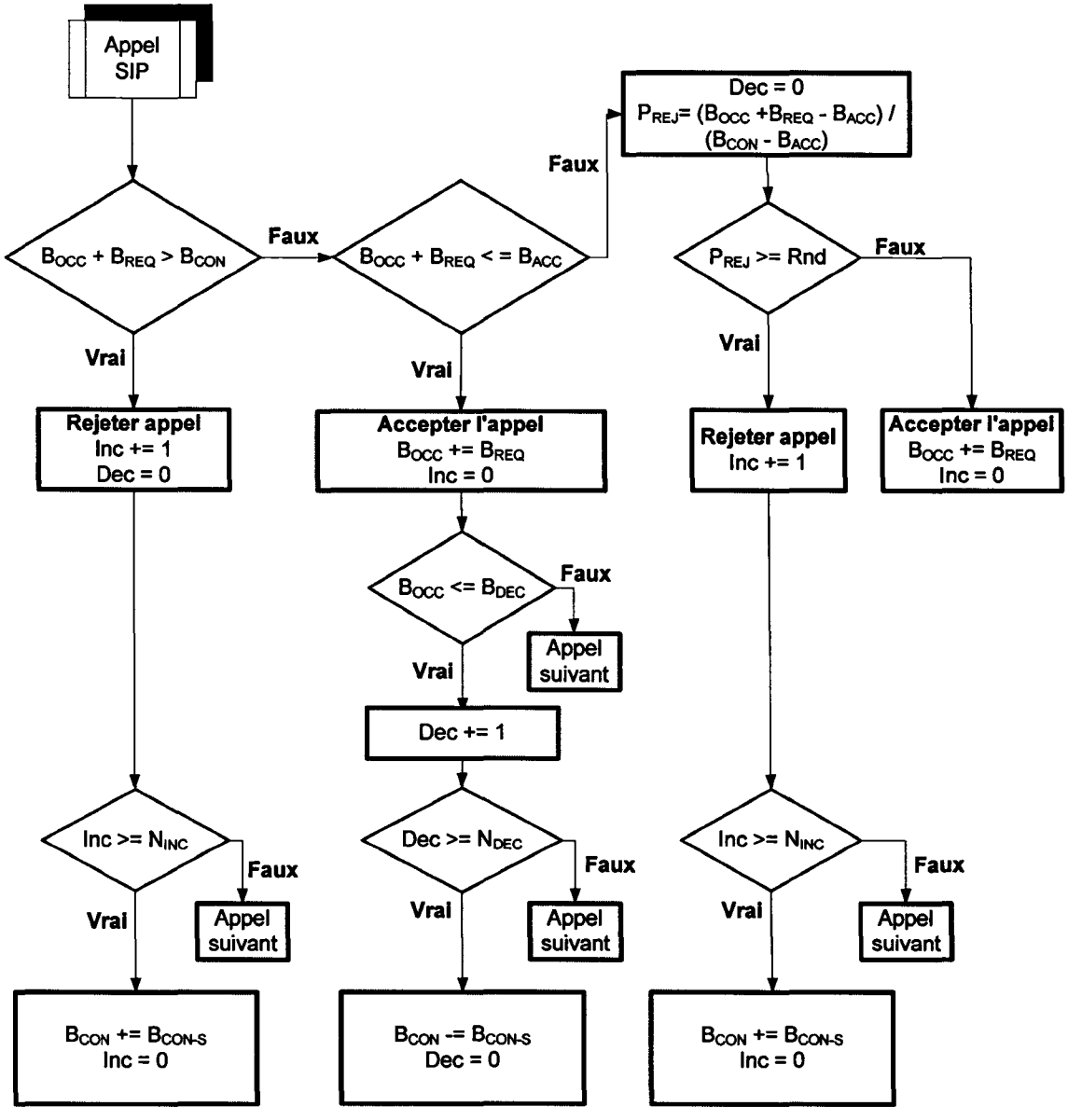


Figure 75 Algorithme DBP, description complète

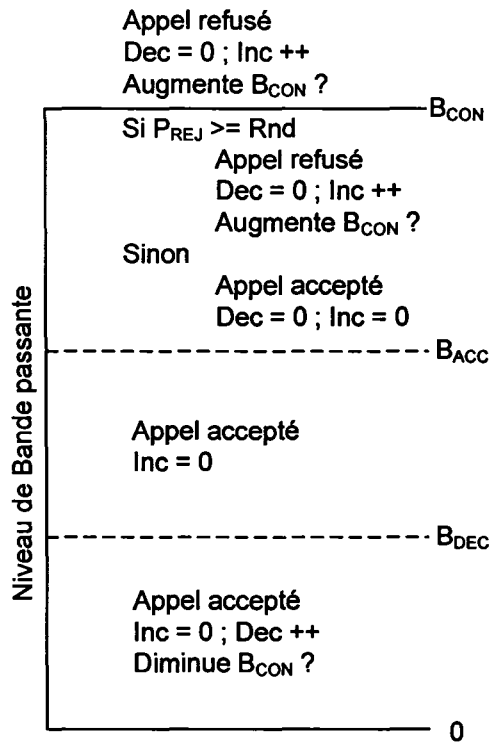


Figure 76 Algorithme DBP, actions en fonction de BW nécessaire

Lorsque Inc > N_{INC} ou Dec > N_{DEC}, la bande passante est respectivement augmentée ou diminuée. Dans ces cas, les valeurs de B_{ACC} et B_{DEC} représente une bande passante différente car leur valeur est en % de B_{CON}.

La figure suivante présente les différents paramètres de cet algorithme, tel que présenté dans la fenêtre de configuration du logiciel Opnet.

- *Access Control* permet de spécifier 3 paramètres globaux :
  - *SIP Access Control* : Algorithme utilisé lors de la simulation ;
  - *Bande passante* : Bande passante réservée au début de la simulation ;
  - *Contract-step* : Pour tous les algorithmes, spécifie la quantité de bande passante pouvant être ajoutée ou supprimée de la réservation. Pour l'algorithme DBP, il s'agit de la variable $B_{CON\text{-}S}$;

- *DBP* permet de spécifier les paramètres relatifs à l'algorithme *Dynamic Bandwidth Prediction* :
    - N_{INC} *et* N_{DEC} : tel que spécifié ci haut ;
    - Bacc-step * B_{CON} = B_{ACC} ;
    - Bdec-step * B_{CON} = B_{DEC} ;
    - Nombre aléatoire = Rnd tel que spécifié ci haut.

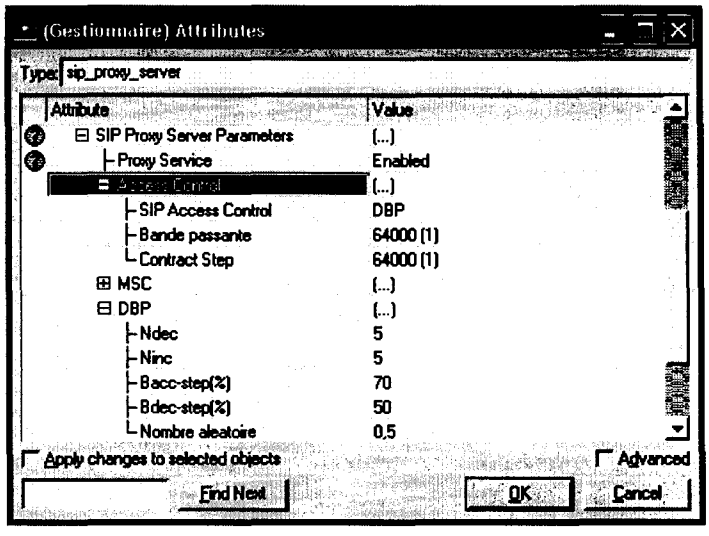


Figure 77 Paramètres de l'algorithme DBP, Opnet

5.1.3 Algorithme DBP simplifié

Une version simplifiée de l'algorithme DBP est ici proposée. Premièrement, la probabilité de rejeter un appel, si le taux d'utilisation de la bande passante est supérieur à B_{ACC}, est supprimée. Pourquoi jeter un appel si la bande passante est disponible ? Le niveau de bande passante B_{ACC} est conservé afin que le compteur Inc soit incrémenté avant qu'un appel soit refusé. Le niveau de bande passante B_{DEC} est également conservé afin que le compteur Dec soit incrémenté afin de diminuer la bande passante réservé lorsque le taux d'appels diminue.

Outre le paramètre Rnd, tous les variables et paramètres sont conservés. L'algorithme est tout de même simplifié, tel que décrit dans la Figure 79. Il peut également se résumer en une suite d'actions à poser en fonction de la bande passante nécessaire (Figure 79).

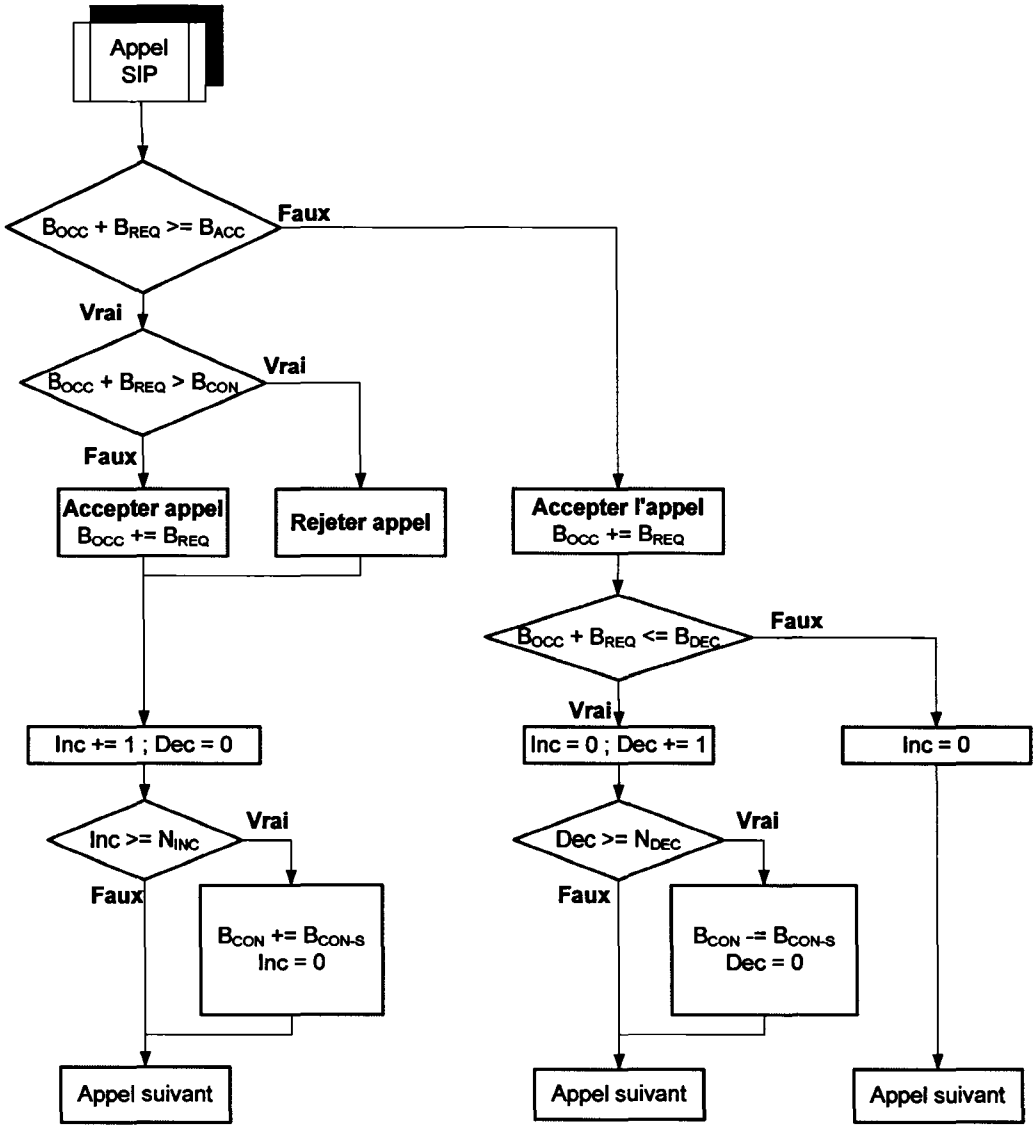


Figure 78 Algorithme DBP simplifié, description complète

Un appel est rejeté seulement si la bande passante n'est pas suffisante. N_{INC} appels consécutifs doivent être établis avec un niveau de bande passante supérieur à B_{ACC} avant

d'augmenter la bande passante. N_{DEC} appels consécutifs doivent être établis avec un niveau de bande passante inférieur à B_{DEC} avant de diminuer la bande passante.

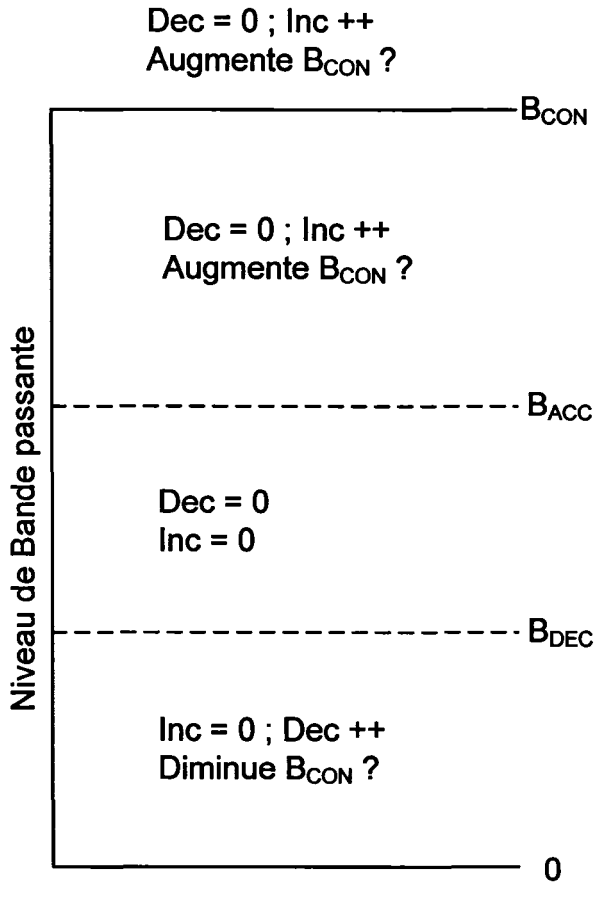


Figure 79 Algorithme DBP simplifié, actions en fonction de BW nécessaire

Outre *Nombre aléatoire*, les différents paramètres de cet algorithme sont les mêmes que pour l'algorithme précédant.

5.1.4 Algorithme ERL (basé sur Erlang)

Cet algorithme, contrairement aux deux précédents, ajoute une dimension temporelle à l'évaluation (prédiction) des besoins. En effet, avec les algorithmes précédents, s'il devait y avoir une fin brusque des appels, comme il n'y aurait aucun appel s'établissant

avec un niveau de bande passante inférieur à B_{DEC}, la bande passante ne serait jamais diminuée. Cet algorithme, comme l'algorithme DBP simplifié, ne refusera aucun appel si la bande passante disponible est suffisante. L'algorithme tentera plutôt de prévoir le moment ou la bande passante doit être augmentée ou diminuée en fonction du taux de trafic moyen. Pour ce faire, le trafic est mesuré en continue. À intervalle régulier, le taux de trafic moyen est calculé afin de déterminer si la bande passante doit être modifiée. À partir du trafic moyen, une prédiction est établie sur la quantité de bande passante nécessaire. Cette prédiction est basée sur la théorie d'Erlang. Pour établir cette prédiction, un paramètre P_R, définissant le taux de rejet désiré, est nécessaire. En fait, la prédiction est basée sur un intervalle de taux de rejet désiré. Pour une bande passante donnée, si le trafic moyen est tel que le taux de rejet P_B est supérieur au taux maximal désiré $P_{SUP} = P_R + (P_R * P_E)$, la bande passante doit être augmentée. Si le trafic moyen est tel que le taux de rejet est inférieur au taux minimal désiré $P_{INF} = P_R - (P_R * P_E)$, la bande passante doit être diminuée.

Variables utilisées par l'algorithme

- B_{REQ} : Bande passante requise pour un appel ;
- B_{CON} : Bande passante réservée ;
- B_{OCC} : Bande passante utilisée ;
- B_{CALC} : Bande passante nécessaire, calculé lorsqu'une modification s'impose ;
- B_{DIM} : Bande passante a libérée avant de diminuer une réservation.
- E : Trafic moyen en Erlang ;
- S : Nombre de circuits (B_{CON} / 64000) ;
- P_B : Probabilité de blocage calculé ;
- Trafic: La quantité total de trafic depuis la dernière évaluation ;
- P_{INF} : Probabilité minimal de blocage désirée = $P_R - (P_R * P_E)$;
- P_{SUP} : Probabilité maximal de blocage désiré = $P_R + (P_R * P_E)$.
- DIM : Vrai, si la bande passante utilisée doit diminuée avant une modification

Paramètres de l'algorithme

- P_R : Taux de rejet désiré ;
- P_E : Écart en % du taux de rejet désiré ;
- T : Période de temps pour le calcul du trafic moyen.

Algorithme

La description complète de cet algorithme se retrouve dans les figures 80 à 83. Le trafic est calculé en continue. À chaque fois qu'un appel débute ou se termine, le nombre d'appels actifs change. De ce fait, la quantité de trafic écoulé (temps * nombre d'appels actifs) est ajoutée au trafic total à chaque fois que le nombre d'appels actifs change. À chaque T secondes (Figure 80), l'algorithme fait ce qui suit.

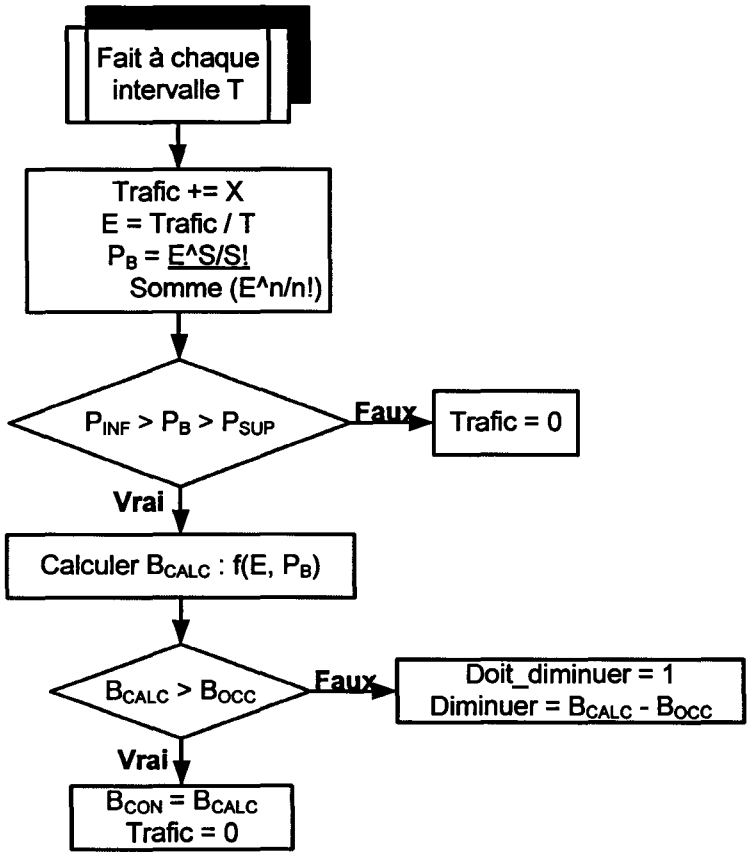


Figure 80 Algorithme ERL à chaque intervalle T

Premièrement, la quantité de trafic écoulé depuis la dernière modification du nombre d'appel actif est ajoutée au trafic total. Le trafic moyen E est ensuite calculé en divisant le trafic total par T. Le taux de rejet P_B est calculé avec formule d'Erlang-B, en fonction du trafic moyen (E) et du nombre de circuits disponibles (S = B_{CON} / 64000).

$$P_B = \frac{\frac{E^S}{S!}}{\sum_{i=0}^{C} \frac{E^i}{i!}}$$

Si la probabilité de blocage est située entre P_{INF} et P_{SUP}, aucune modification de B_{CON} n'est nécessaire. Les bornes P_{INF} et P_{SUP} sont nécessaires car le calcul de P_B ne pourrait qu'être inférieur ou supérieur à un taux de rejet discret. Il en résulterait une modification de la bande passante, à la hausse ou à la baisse, à chaque fois que P_B serait calculé. Afin de minimiser le nombre de modification, un intervalle de taux de rejet désiré est utilisé. Si le taux de rejet calculé est situé dans cet intervalle aucune modification de la bande passante n'est nécessaire Dans ce cas, le trafic total est mis à 0 et l'algorithme prend fin.

Si la probabilité de blocage est supérieure à P_{SUP} ou inférieur à P_{INF}, il faut déterminer la bande passante nécessaire B_{CALC} pour obtenir un taux de rejet situé entre P_{INF} et P_{SUP}. Le calcul avec la formule d'Erlang inversé étant très gourmand en ressource, une table est utilisée pour déterminer B_{CALC}. Si P_B est inférieur à P_{INF}, c'est que trop de bande passante est disponible. À l'inverse, si P_B est supérieur à la P_{SUP}, c'est qu'il n'y a pas assez de bande passante.

B_{CALC} étant déterminé, cela ne signifie pas que la bande passante réservée B_{CON} peut être modifiée sur le champ. Pour modifier B_{CON}, il faut que B_{CALC} soit supérieur à B_{OCC}. S'il fallait diminuer la bande passante réservée sans tenir compte du nombre d'appels actifs au moment de l'évaluation, il pourrait en résulter une bande passante réservée inférieure à celle utilisée. Si B_{CALC} est supérieur à B_{OCC}, B_{CON} peut être modifiée. Si B_{CALC} est inférieur à B_{OCC}, il faudra attendre qu'un nombre suffisant d'appels se termine avant de diminuer la bande passante réservée.

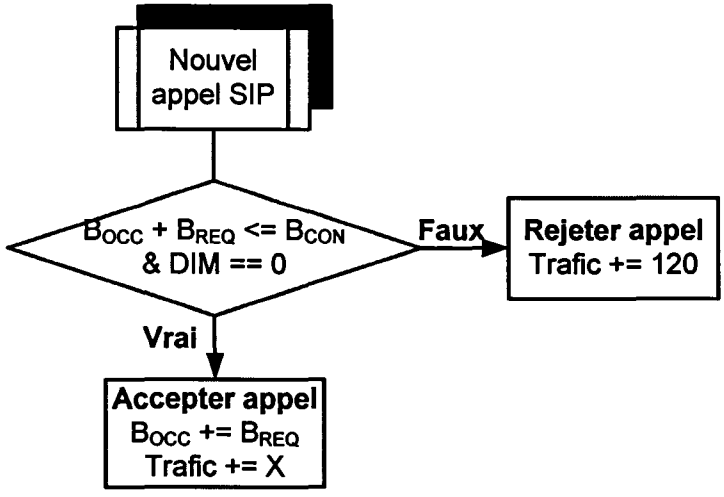


Figure 81 Algorithme ERL, lors d'une requête pour établir un nouvel appel

Lors de la réception d'une requête pour un nouvel appel (Figure 81), deux conditions doivent être rencontrées pour accepter l'appel. Premièrement, il doit y avoir suffisamment de bande passante. Ensuite, il ne faut pas que l'algorithme ait à diminuer la bande passante pour cause de B_{CALC} supérieur à B_{OCC}. Si l'appel est accepté, c'est que le nombre d'appels actifs est modifié et le trafic écoulé depuis la dernière modification est ajouté au trafic total. Si l'appel est refusé, il est nécessaire d'ajouter le trafic qu'aurait généré cet appel au trafic total. Ainsi, 120 secondes sont ajoutées car il s'agit de la durée moyenne de chaque appel. Ne pas inclure ce temps fausserait l'évaluation du trafic moyen car le calcul d'un trafic moyen considère le trafic entrant, qu'il soit accepté ou non.

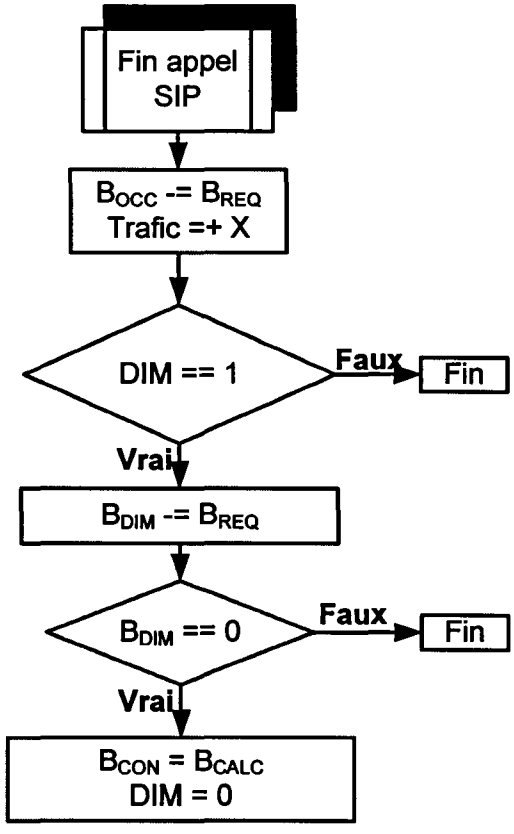


Figure 82 Algorithme ERL, lors de la fin d'un appel

Lors de la réception d'un message de fin d'appel (Figure 82), c'est que le nombre d'appels actifs est modifié et le trafic écoulé depuis la dernière modification est ajouté au trafic total. Si l'algorithme n'a pas besoin de diminuer la bande passante, l'algorithme se termine. Dans un cas ou B_{CALC} aurait été supérieur à B_{OCC}, il doit y avoir $B_{DIM} = (B_{CALC}-B_{OCC}) / B_{REQ}$ appels de terminés avant de diminuer B_{CON}. Donc, si l'algorithme est en cours de réduction du nombre d'appels actifs avant de diminuer la bande passante (DIM == 1), comme un appel vient de se terminer, la variable B_{DIM} doit être diminuée de B_{REQ}. Après la réduction, si DIM == 0, l'algorithme peut diminuer la bande passante et $B_{CON} = B_{CALC}$.

La figure suivante présente les différents paramètres de cet algorithme, tel que présenté dans la fenêtre de configuration du logiciel Opnet.

- *ERL* permet de spécifier les paramètres relatifs à cet algorithme :
  - o *Periode :* *L'intervalle entre deux évaluations;*
  - o Bloc_inf = P_{INF} : Probabilité de rejet minimale;
  - o Bloc_sup = P_{SUP} : Probabilité de rejet maximale.

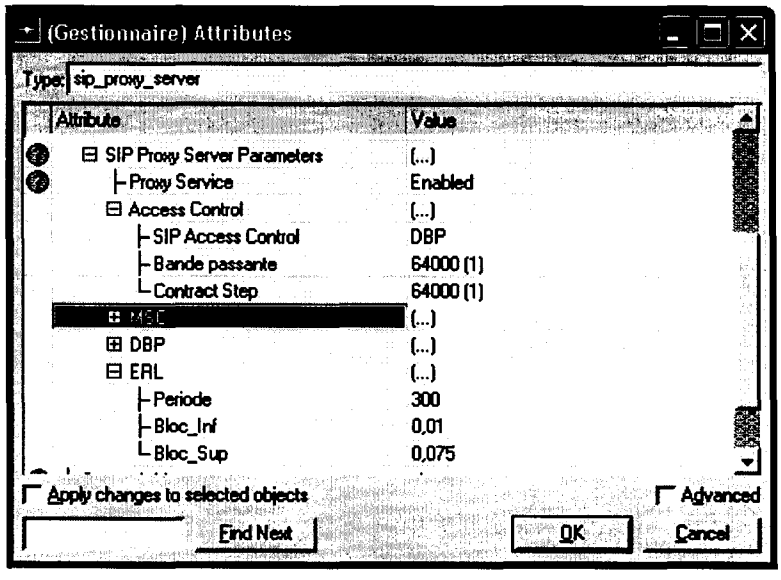


Figure 83 Paramètres de l'algorithme ERL, OPNET

CHAPITRE 6

SCÉNARIOS DE TEST ET SIMULATIONS

Le chapitre précédant s'est attardé à la description des différents algorithmes de contrôle d'accès et de prédiction des besoins. Dans ce dernier chapitre, les différents algorithmes seront simulés en faisant varier la valeur de leurs différents paramètres. Les résultats obtenus permettront de mieux comprendre l'impact des différents paramètres sur le comportement de chacun des algorithmes. Pour chaque algorithme, la valeur de chaque paramètre sera optimisée afin d'obtenir un comportement optimal des algorithmes étudiés. Finalement, lorsque chaque algorithme sera optimisé, ceux-ci seront comparés afin d'en déterminer le plus efficace.

Le chapitre débute par quelques détails sur les distributions exponentielles. Le trafic utilisé lors des expérimentations ayant une distribution exponentielle, le comportement et l'efficacité des différents algorithmes seront liés à cette distribution de trafic. Ensuite, le modèle de trafic utilisé pour réaliser nos expérimentations sera détaillé et validé. Pour démontrer l'utilité des algorithmes, il faut que le trafic moyen change en fonction de l'heure de la journée. C'est l'objectif de ces algorithmes que de modifier la bande passante réservée en fonction de besoins variant avec le temps. Finalement, chaque algorithme sera étudié en détail.

6.1 Théorie sur les distributions exponentielles

Lorsque le contrôle d'accès s'applique sur un trafic ayant une distribution exponentielle, le contrôle d'accès s'apparente à une file de type M/M/k/k où k est le nombre d'appels pouvant être acceptés pour une bande passante donnée. Le taux d'entrée des requêtes d'établissement d'appel (λ) et la durée des appels (μ) suivent une distribution exponentielle dont la résultante est le trafic moyen en erlang ($\rho = \lambda / \mu$). La théorie des fil d'attente impose des limites théoriques au système. Pour une bande passante

permettant l'établissement de k appels et un trafic moyen de ρ erlang, le taux de rejet minimal et le taux d'utilisation maximal sont déterminés par les équations suivantes.

- Taux de rejet = B = $P_n = \frac{\frac{\rho^n}{n!}}{\sum_{i=0}^{k} \frac{\rho^i}{i!}}$, n = k (1)

- Taux d'utilisation = P_n moyen = $\frac{\sum_{n=0}^{k} n * Pn}{k} = \frac{\sum_{n=0}^{k} n * \frac{\frac{\rho^n}{n!}}{\sum_{i=0}^{k} \frac{\rho^i}{i!}}}{k}$ (2)

  Le taux d'utilisation peut également se calculer en utilisant :

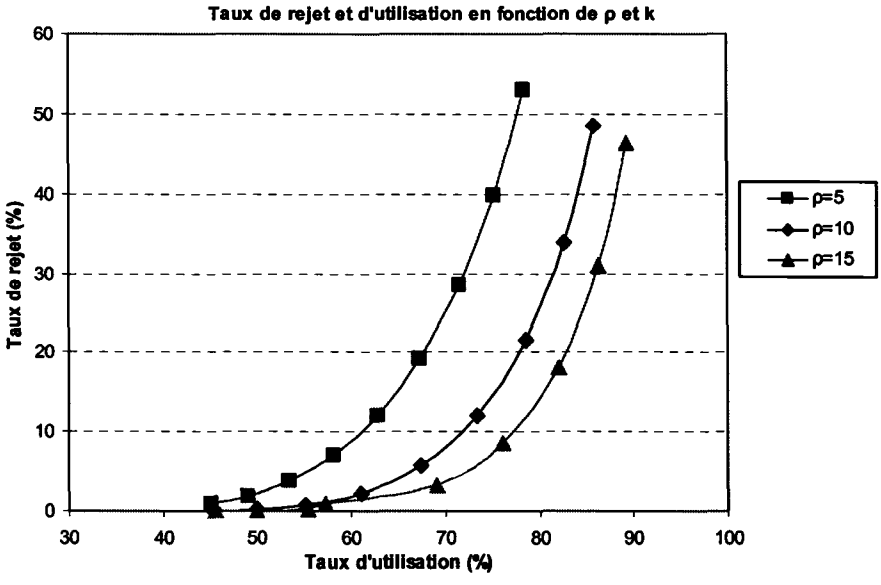
$$\frac{\rho_a}{k}, \rho_a = \rho(1 - P_k) \quad (3)$$

Considérant un trafic moyen de ρ erlang et k circuits, voici ce que la théorie impose aux taux de rejet et d'utilisation :

Tableau III

Taux de rejet et d'utilisation théoriques

ρ = 5			ρ = 10			ρ = 15		
k	Rejet (%)	Utilisation (%)	k	Rejet (%)	Utilisation (%)	k	Rejet (%)	Utilisation (%)
3	52,97	78,38	6	48,45	85,92	9	46,39	89,35
4	39,83	75,21	8	33,83	82,71	12	30,96	86,30
5	28,49	71,51	10	21,46	78,54	15	18,03	81,97
6	19,18	67,35	12	11,97	73,36	18	8,62	76,15
7	12,05	62,82	14	5,68	67,37	21	3,15	69,18
8	7,05	58,09	16	2,23	61,11	24	0,839	57,26
9	3,75	53,47	18	0,71	55,16	27	0,160	55,47
10	1,84	49,08	20	0,19	49,91	30	0,020	49,99
11	0,83	45,08	22	0,04	45,44	33	0,001	45,45



Graphique 1 Taux de rejet et d'utilisation en fonction de ρ et de k

Dans le Graphique 1, les taux de rejet et d'utilisation sont déterminés théoriquement à partir des formules (1) et (2). Pour un taux de trafic donné (ρ), augmenter la bande passante disponible (nombre de circuits k), impose une diminution des taux de rejet et d'utilisation. Ce graphique nous démontre plusieurs aspects de comportement qu'aura tout type de système M/M/k/k. Premièrement, un taux d'utilisation de 100% ne peut être atteint à moins d'avoir un taux de rejet de 100%. Ensuite, ce graphique nous démontre qu'un taux de rejet nul est possible moyennant un grand nombre de circuits relativement au trafic moyen, ce qui impose de diminuer le taux d'utilisation. Finalement, il nous démontre que pour un taux de rejet donné, plus le trafic est élevé, meilleur est le taux d'utilisation. L'inverse est également vrai, pour un taux d'utilisation donné, plus le trafic est élevé, meilleur est le taux de rejet.

6.2 Modèle de trafic

La voix sur IP est une application pour laquelle le contrôle d'accès est essentiel afin de garantir une qualité de service. De plus, le trafic généré par ce type d'application suit

souvent une distribution exponentielle, ce qui renforce l'intérêt porté à ce type de trafic. Les algorithmes de contrôle d'accès seront donc expérimentés avec ce type de trafic.

6.2.1 Profil de trafic d'une entreprise

Un algorithme de contrôle d'accès et de prédiction des besoins en bande passante est utile lorsque le nombre moyen de requêtes pour l'utilisation des ressources change en fonction du temps. Pour que les algorithmes puissent être étudiés adéquatement, cette réalité doit être modélisée. Pour ce faire, un profil de trafic de voix correspondant aux besoins d'une entreprise a été configuré. Le trafic moyen, pendant le « quart » de nuit (00 : 00 à 8 : 00) est d'environ 5 erlang, pendant le « quart » de jour (8 : 00 à 16 : 00), il est de 15 erlang, pendant le « quart » de soir (16 : 00 à 00 : 00), il est de 10 erlang. Entre ces « quarts », il y a des creux de trafic représentant la transition du personnel. Il s'agit de trafics moyens suivant une distribution exponentielle.

6.2.2 Validation de la configuration des taux de trafic

Le taux d'arrivée des appels λ et le taux de service μ définissent le trafic moyen ρ, en erlang, à l'entrée du système ($\rho = \lambda / \mu$).

- Soit Y : la durée des appels (*Attribut Opnet : Call Duration*) ;
- Soit X : la période entre la fin d'un appel et le début d'un second (*Attribut Opnet : Inter-repetition Time*).

Le taux de service μ est inversement proportionnel à la durée des appels.

- $\mu = 1 / Y$ (4)

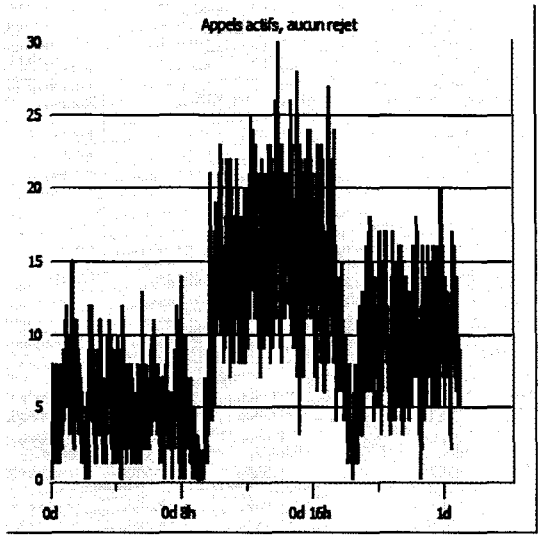
Le taux d'arrivée λ est déterminé par la période entre deux requêtes d'appel. S'il y avait un seul appareil téléphonique, cette période serait déterminée par la durée d'un appel (Y) et le temps entre la fin de cet appel et l'établissement du suivant (X) : $\lambda = 1/(X+Y)$. Considérant N appareils téléphoniques, le taux d'appel entrant dans le système y est proportionnel.

- $\lambda = N/(X+Y)$. (5)

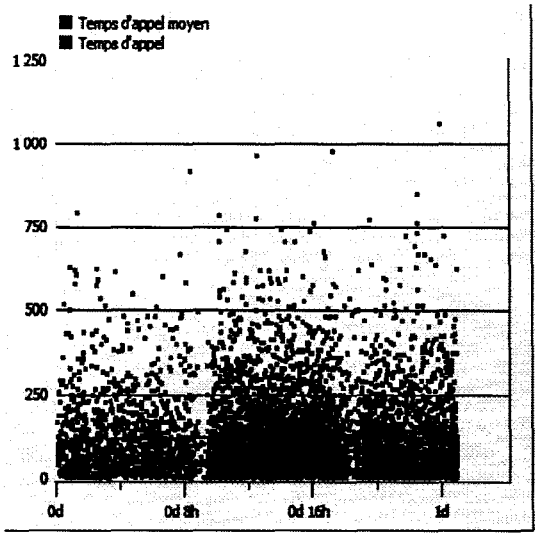
À l'aide des équations 4 et 5, il est possible de déterminer la valeur des attributs X et Y pour générer le trafic en erlang désiré.

- Soit le nombre de stations : N = 120
- Soit la durée moyenne des appels : Y= 120 secondes
- Soit $\rho = \lambda / \mu = (N/(X+Y)) / (1/Y) = (N * Y) / (X + Y) = 5$, 10 et 15 erlang,
  - $\rho = 5 \rightarrow 5 = (120 * 120) / (120 + X) \rightarrow X = 2760$sec ;
  - $\rho = 10 \rightarrow 10 = (120 * 120) / (120 + X) \rightarrow X = 1320$sec ;
  - $\rho = 15 \rightarrow 15 = (120 * 120) / (120 + X) \rightarrow X = 840$sec.

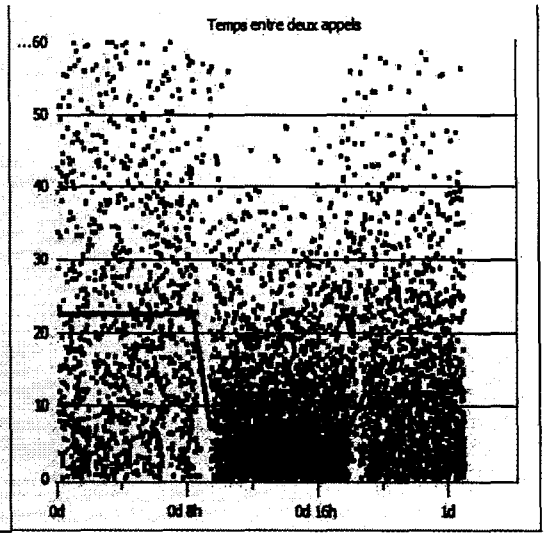
Les graphiques suivants représentent les résultats obtenus à l'aide de ces configurations.



Graphique 2 Profil de trafic entrant (avant contrôle d'accès)



Graphique 3 Durée moyenne d'appel



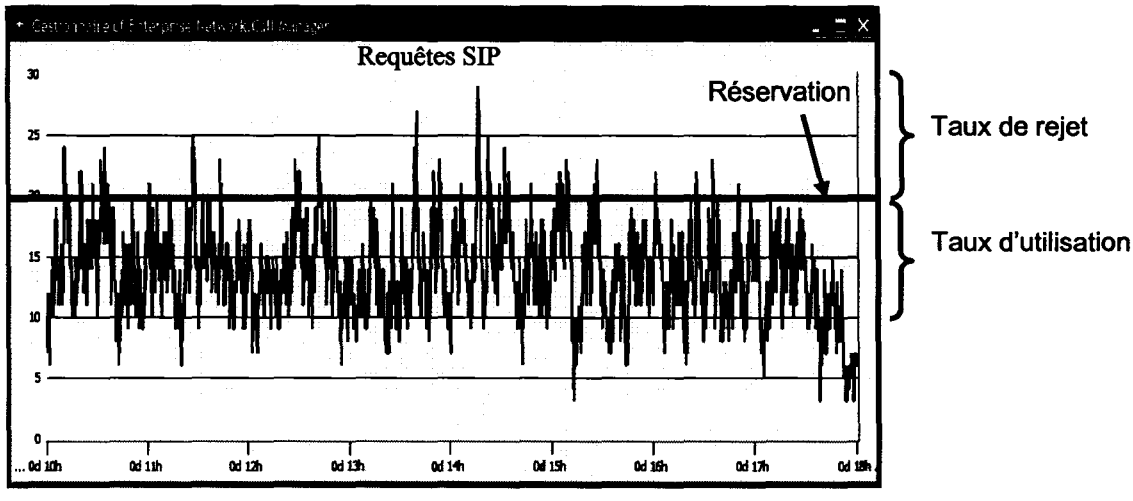
Graphique 4 Temps entre deux appels

Considérant que tous les appels sont acceptés (aucun contrôle d'admission), le profil de trafic configuré génère le trafic représenté dans le Graphique 2; ce qui correspond bien un trafic désiré de 5, 15 et 10 erlang en fonction des moments de la journée. Le Graphique 3 démontre bien une durée moyenne d'appel de 120 secondes selon une distribution exponentielle et le Graphique 4 représente le temps entre deux appels entrant dans le système, c'est-à-dire 2760 secondes / 120 stations = 23 secondes, 840 / 120 = 7 secondes et 1320 / 120 = 11 secondes.

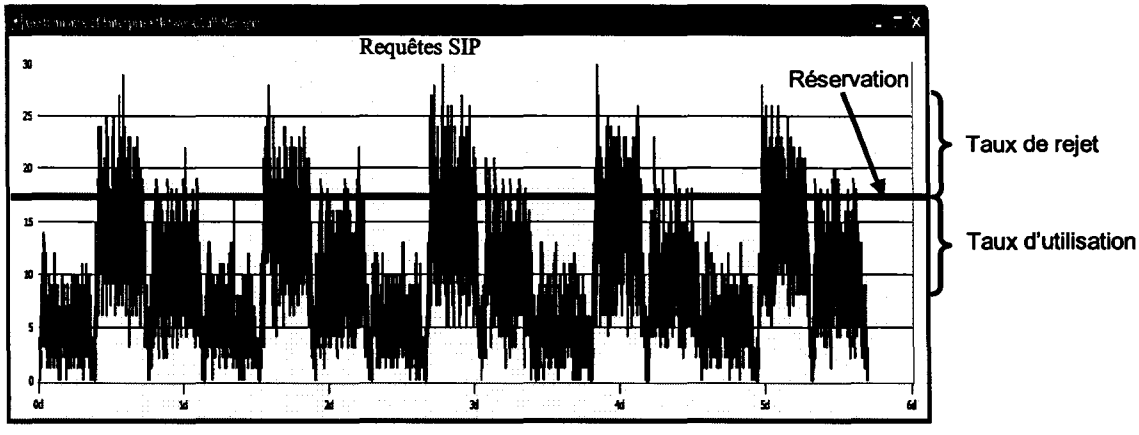
6.3 Objectif des algorithmes

Peu importe l'algorithme de contrôle d'accès et de prédiction, le système s'apparente à une file M/M/k/k. Soit le taux de rejet est minimisé au détriment d'un taux d'utilisation moindre, soit le taux d'utilisation est maximisé au détriment d'un taux de rejet supérieur. Il est impossible d'élaborer un mécanisme qui diminuerait le taux de rejet tout en maximisant le taux d'utilisation au-delà des valeurs théoriques déterminées à la section 6.1. Si tel devait être le cas, la nature "sans mémoire" de la distribution exponentielle devrait être remise en question.

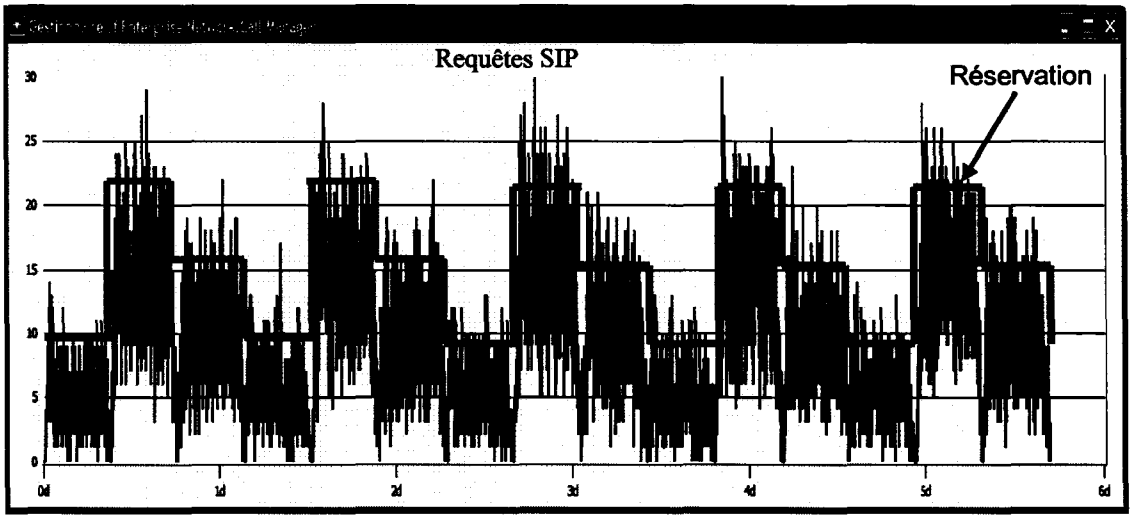
Lorsque le trafic (ρ) est constant (graphique 5), la quantité de bande passante réservée, permettant l'établissement de k appels, peut être déterminée en fonction du taux de rejet ou d'utilisation désiré. Cependant, lorsque le trafic (ρ) varie avec le temps, une réservation fixe de bande passante impose aux taux de rejet et d'utilisation de varier avec le taux de trafic (Graphique 6). L'objectif des différents algorithmes est de modifier un minimum de fois la quantité de bande passante réservée afin que les taux de rejet et d'utilisation désirer demeure fixe malgré la variation du taux de trafic (Graphique 7). Avec un algorithme modifiant adéquatement la bande passante réservée, les taux de rejet et d'utilisation obtenus sur une longue période devrait correspondre à ceux qui seraient obtenus avec une réservation constante établie en fonction du trafic moyen de cette longue période.



Graphique 5 Trafic moyen constant, réservation constante



Graphique 6 Trafic moyen variable, réservation constante



Graphique 7 Trafic moyen variable, réservation variable

6.4 Étude de l'algorithme ABW (Available BandWidth)

Le premier algorithme étudié, ABW, est le plus simple. Le seul paramètre configurable est la bande passante réservée. Cet algorithme fait uniquement un contrôle d'accès, il ne peut estimer la bande passante nécessaire et ne peut modifier la réservation de bande passante au besoin. Les taux de rejet et d'utilisation obtenus avec cet algorithme serviront de base de comparaison pour évaluer l'efficacité des prochains algorithmes qui pourront modifier la bande passante au besoin.

6.4.1 Étude comportementale de l'algorithme ABW

Pour étudier le comportement de cet algorithme, des simulations seront réalisées en modifiant le paramètre *bande passante réservée*. Le nombre maximum d'appels simultanés étant de 30 (Graphique 2), le paramètre *bande passante réservée* prendra des valeurs permettant l'établissement de 8 à 30 appels. Les résultats obtenus seront comparés avec les valeurs théoriques permises par la théorie.

Provenance des résultats

Les valeurs théoriques seront calculées à l'aide des équations (1) et (2) en utilisant le nombre de circuits réservés et le taux de trafic obtenu pour chaque simulation. Les taux pratiques ont été calculés à partir de résultats de simulation. Le taux de rejet pratique est égal à la quantité d'appel refusé divisé par le nombre de requête reçus.

- Rejet pratique = Nb appels refusés / Nb total de requêtes reçues

Le taux d'utilisation pratique est déterminé par le nombre de secondes d'utilisation divisé par le nombre de secondes réservées. Le nombre de secondes d'utilisation est l'addition de la durée de chaque appel établi. Le nombre de secondes réservées est la multiplication du nombre de circuits réservés par la durée totale de la simulation.

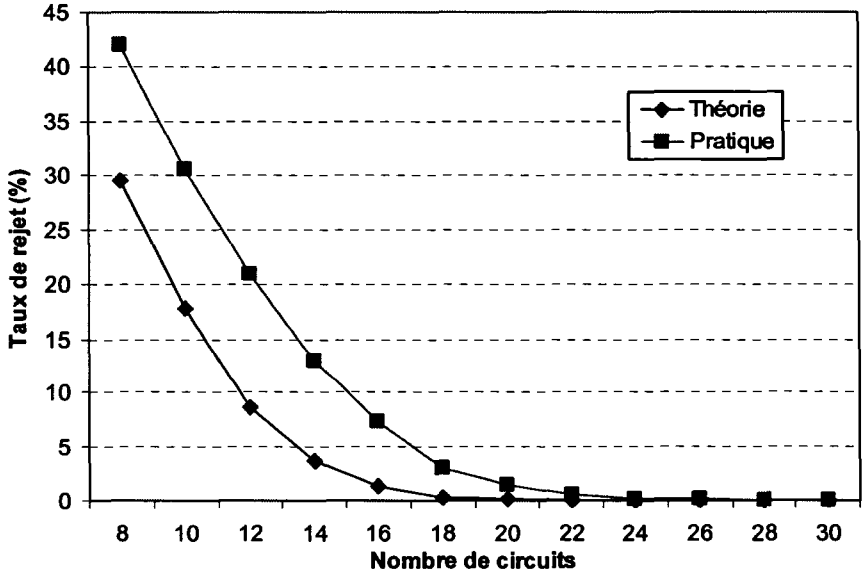
- Utilisation pratique = Nb de secondes utilisées / Nb. de secondes réservées.

RÉSULTATS algorithme ABW

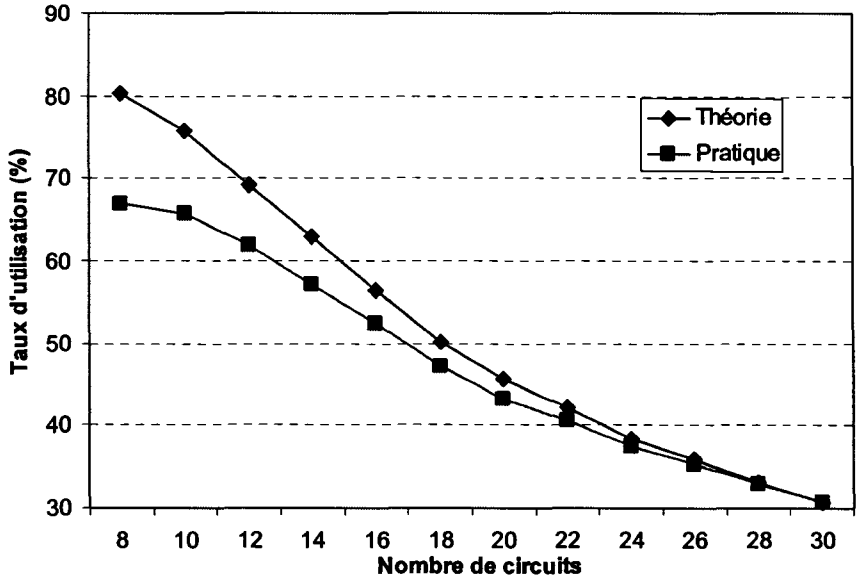
Tableau IV

Résultats, algorithme ABW

Bande passante Bps	Circuits de voix	Taux de trafic réel Erlang	Rejet théorique %	Rejet pratique %	Utilisation théorique %	Utilisation pratique %
512000	8	9,119	29,525	42,009	80,330	66,910
640000	10	9,220	17,828	30,542	75,763	65,730
768000	12	9,071	8,556	20,932	69,122	61,940
896000	14	9,147	3,678	13,003	62,930	57,033
1024000	16	9,144	1,236	7,305	56,444	52,392
1152000	18	9,041	0,302	3,053	50,078	47,220
1280000	20	9,124	0,072	1,432	45,587	43,121
1408000	22	9,260	0,016	0,518	42,084	40,639
1536000	24	9,187	0,002	0,145	38,277	37,513
1664000	26	9,331	0,000	0,071	35,887	35,460
1792000	28	9,253	0,000	0,043	33,048	32,947
1920000	30	9,197	0,000	0,000	30,658	30,635



Graphique 8 Taux de rejet en fonction du nombre de circuits réservés



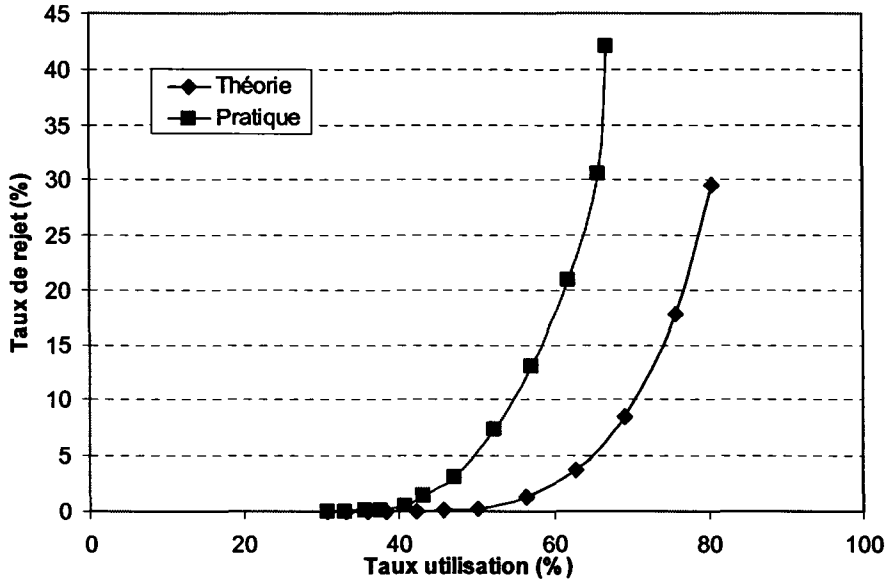
Graphique 9 Taux d'utilisation en fonction du nombre de circuits réservés

ANALYSE algorithme ABW

Les graphiques 8 et 9 démontrent que plus il y a de bande passante réservée (nombre de circuits) plus les taux de rejet et d'utilisation sont bas. Ces taux, autant théoriques que pratiques, sont déterminés par le trafic moyen et par la quantité de circuits réservés. Plus il y a de bande passante disponible, moins il y a de rejet. Cependant, cette diminution du taux de rejet se fait au détriment du taux d'utilisation. Ce comportement était facilement prévisible puisqu'il s'agit du comportement théorique qu'aura tout algorithme de contrôle d'admission.

Le taux de trafic varie en fonction de la période de la journée (Graphique 2) et les algorithmes étudiés ont pour objectif d'en tenir compte. Le trafic moyen des différentes simulations étant d'environ 9,2 erlang (voir tableau 4), si l'algorithme était performant, les taux de rejet et d'utilisation pratiques devraient s'approcher des taux théoriques déterminés par cette valeur et le nombre de circuits disponibles. Les taux de rejet et d'utilisation sont interdépendants, ils sont déterminés par les mêmes paramètres. Le Graphique 10 superpose les courbes théorique et pratique de la relation taux de rejet vs

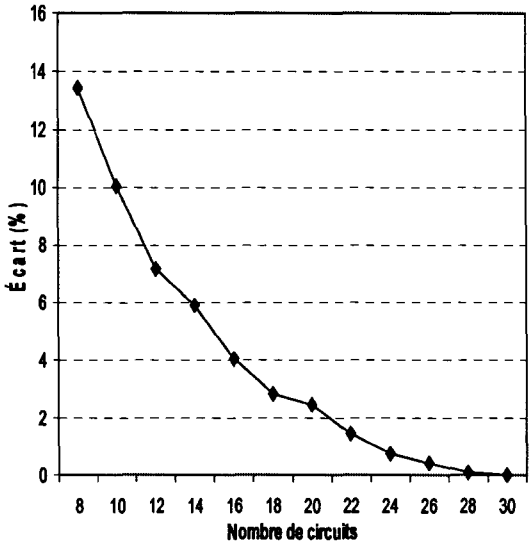
taux d'utilisation. Il ressort clairement que ce simple algorithme ne permet pas d'atteindre les taux de rejet et d'utilisation optimale théorique lorsque le trafic moyen varie.



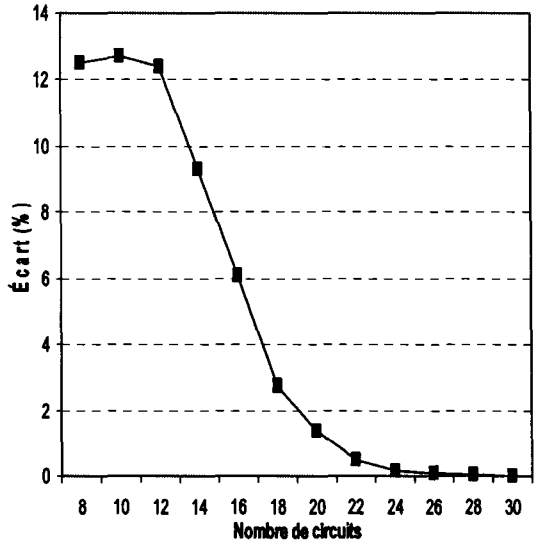
Graphique 10 Taux d'utilisation en fonction du taux de rejet

L'écart entre les taux de rejet théorique et pratique (graphique 11) et l'écart entre les taux d'utilisation théorique et pratique (graphique 12) nous démontre que l'algorithme permet d'atteindre les valeurs optimales théoriques uniquement lorsque le nombre de circuits réservés est grand. Typiquement, le nombre de circuits réservés (> 20) doit être supérieur au plus grand trafic moyen de la journée (15 erlang), ce qui impose un taux d'utilisation minimale et un taux de rejet nul.

Si le taux de trafic n'avait pas varié au cours de la simulation, les taux de rejet et d'utilisation pratiques auraient été similaires aux taux théoriques. Comme le trafic moyen varie au cours de la journée et que la bande passante réservée demeure constante, les taux obtenus n'ont d'autres choix que de s'éloigner de la théorie (graphique 11 et 12). L'objectif des algorithmes suivants sera d'optimisé les taux de rejet et d'utilisation afin qu'ils s'approchent des taux théoriques.



Graphique 11 Écart, taux de rejet



Graphique 12 Écart, taux d'utilisation

6.5 Étude de l'algorithme DBP (Dynamic Bandwidth Prediction)

L'algorithme ABW, étudié à la section précédente, ne permettait pas de modifier la bande passante en fonction du trafic. Le trafic moyen variant avec le temps, les taux de rejet et d'utilisation obtenus ne pouvait s'approcher de ceux permis par la théorie qu'en ayant un nombre de circuits beaucoup plus grand que le trafic moyen. Aussi bien dire que l'algorithme ABW est efficace lorsque aucun algorithme de contrôle d'admission n'est nécessaire.

L'algorithme étudié à la présente section tente d'être efficace, dans un contexte de trafic moyen variable, en augmentant ou en diminuant la bande passante réservée en fonction du trafic moyen. Pour ce faire, l'algorithme estime le trafic moyen et modifie les ressources réservées lorsque certains seuils sont dépassés. Pour ce faire, DBP utilise plusieurs paramètres définissant le comportement de l'algorithme face aux variations du trafic moyen (ex : modifier plus ou moins rapidement la bande passante réservée). Plus l'algorithme réagit rapidement, plus il y aura de modifications des ressources.

Pour étudier le présent algorithme, les différents paramètres seront étudiés l'un après l'autres. Pour chaque paramètre, plusieurs simulations seront effectuées en faisant varier la valeur du paramètre étudié. Il sera ainsi possible de déterminer l'effet de ce paramètre sur les taux de rejet et d'utilisation obtenus par l'algorithme. Ces taux seront toujours comparés aux valeurs théoriques afin de déterminer la valeur optimale du paramètre étudié qui permet à l'algorithme d'obtenir des taux pratiques s'approchant le plus possible des taux théoriques.

Provenance des résultats

Les valeurs théoriques seront calculées à l'aide des équations (1) et (2) en utilisant le nombre de circuits réservés et le taux de trafic obtenu pour chaque simulation. Étant donné que la bande passante est modifiée au besoin, le nombre de circuits réservés utilisé pour le calcul théorique sera le nombre de circuits moyen obtenu à la fin de la simulation.

Les taux pratiques ont été calculés à partir de résultats de simulation. Le taux de rejet pratique est égal à la quantité d'appel refusé diviser par le nombre de requête reçus.

- Rejet pratique = Nb appels refusés / Nb total de requêtes reçues

Le taux d'utilisation pratique est déterminé par le nombre de secondes d'utilisation divisé par le nombre de secondes réservées. Le nombre de secondes d'utilisation est l'addition de la durée de chaque appel établi. Le nombre de secondes réservées est la multiplication du nombre de circuits réservés par la durée totale de la simulation.

- Utilisation pratique = Nb de secondes utilisées / Nb. de secondes réservées.

Pour les premières études, les valeurs obtenues seront présentées dans des tableaux ainsi que sous forme de graphiques. Par la suite, seul les graphiques seront conservés, afin de ne pas dupliquer inutilement les résultats obtenus.

6.5.1 Étude comportementale DBP, paramètres N_{INC} et N_{DEC}

N_{INC} et N_{DEC} seront les deux premiers paramètres étudiés. Ils définissent le nombre d'appels devant être établis, dépassé un certain seuil, avant d'augmenter ou diminuer la réservation de bande passante. Pour les simulations qui permettront de déterminer l'effet de N_{INC} et N_{DEC}, les autres paramètres ont été configurés avec une valeur par défaut. Ces valeurs ont été sélectionnées afin d'éviter les comportements extrêmes que pourraient occasionner des valeurs limites. B_{ACC} = 70 % de B_{CON}, B_{DEC} = 50% de B_{CON} et P_{REJ} = 0,5. Pour bien comprendre l'effet des paramètres N_{INC} et N_{DEC}, voici les différents scénarios qui ont été simulés :

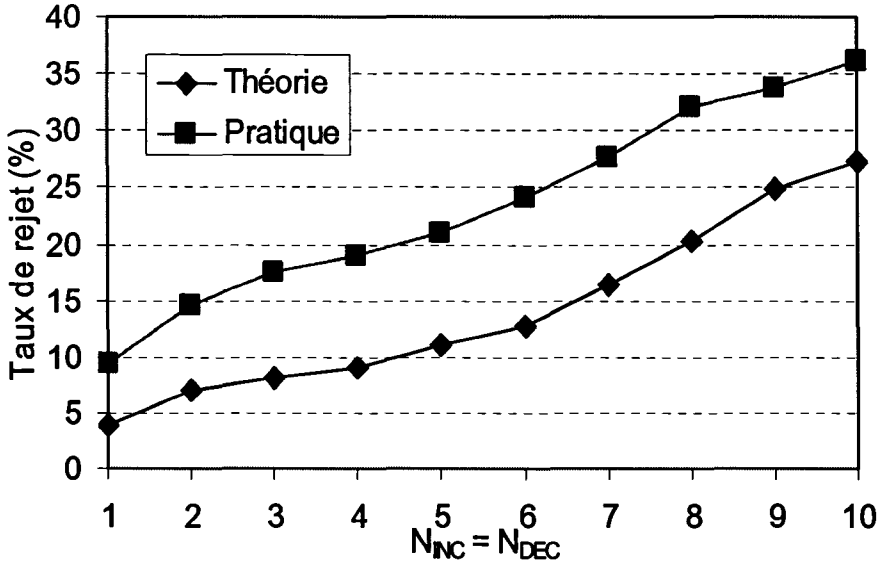
- Scénario 1 : $N_{INC} = N_{DEC}$ = 1 à 10 : effet qu'a l'augmentation de leur valeur ;
- Scénario 2 : N_{INC} = 5 et N_{DEC} = 1 à 10 : pour mieux cerner le rôle de N_{DEC} ;
- Scénario 3 : N_{DEC} = 5 et N_{INC} = 1 à 10 : pour mieux cerner le rôle de N_{INC}.

RÉSULTAT DBP, Scénario 1 : $N_{INC} = N_{DEC}$

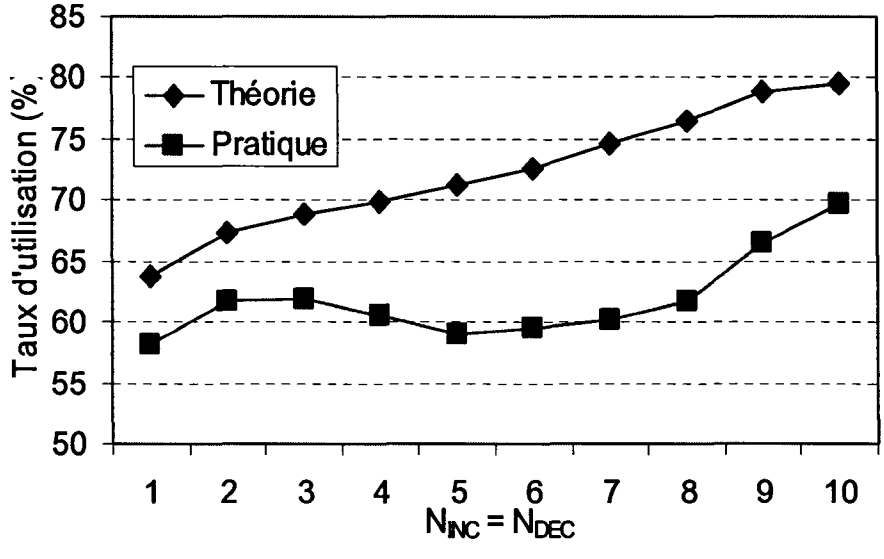
Tableau V

Résultats DBP, $N_{INC} = N_{DEC}$

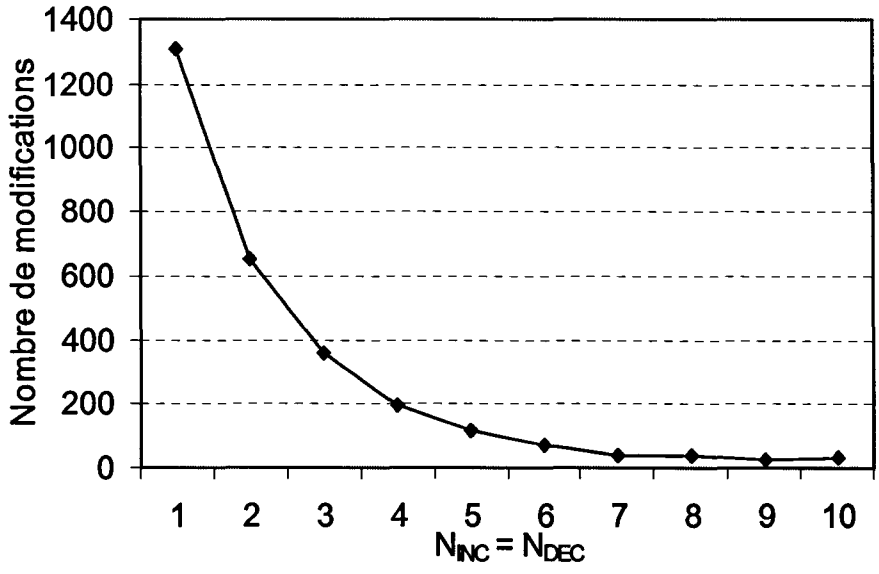
$N_{INC} = N_{DEC}$	Nb de circuits Moyenne	Taux de trafic réel Erlang	Rejet théorique %	Rejet pratique %	Utilisation théorique %	Utilisation pratique %	Modif
1	14,178	9,329	3,783	9,46	63,310	57,770	1312
2	12,514	9,057	7,011	14,57	67,305	61,742	650
3	12,428	9,297	8,036	17,58	68,797	61,960	356
4	12,671	9,252	9,091	18,98	69,891	60,479	196
5	10,576	9,212	10,975	21,09	71,257	59,044	117
6	11,060	9,196	12,666	24,13	72,617	59,579	73
7	11,011	9,135	16,317	27,72	74,705	60,185	40
8	9,341	8,960	20,313	32,10	76,437	61,650	37
9	8,806	9,240	24,893	33,80	78,809	66,402	26
10	9,041	9,153	27,372	36,18	79,541	69,694	31



Graphique 13 Taux de rejet en fonction de $N_{INC} = N_{DEC}$



Graphique 14 Taux d'utilisation en fonction de $N_{INC} = N_{DEC}$



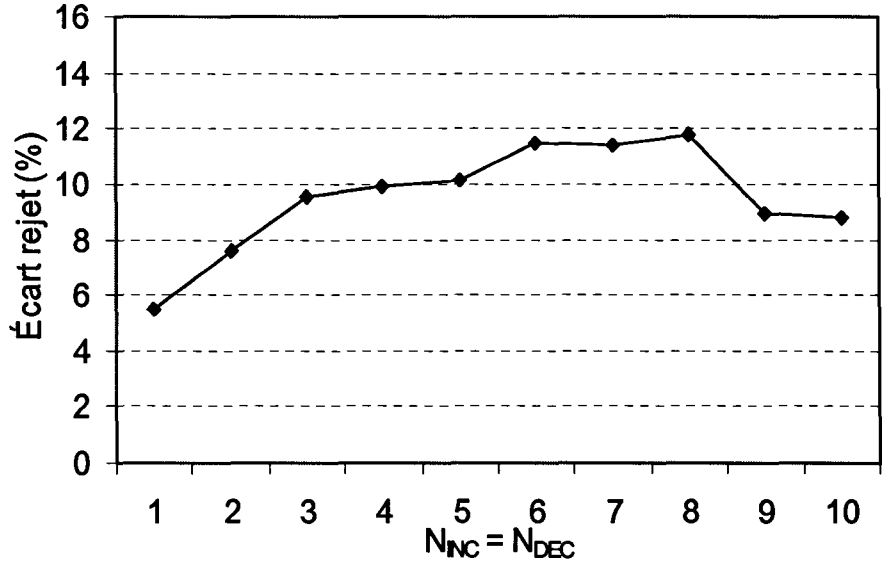
Graphique 15 Nombre de modifications en fonction de $N_{INC} = N_{DEC}$

ANALYSE DBP, Scénario 1 : $N_{INC} = N_{DEC}$

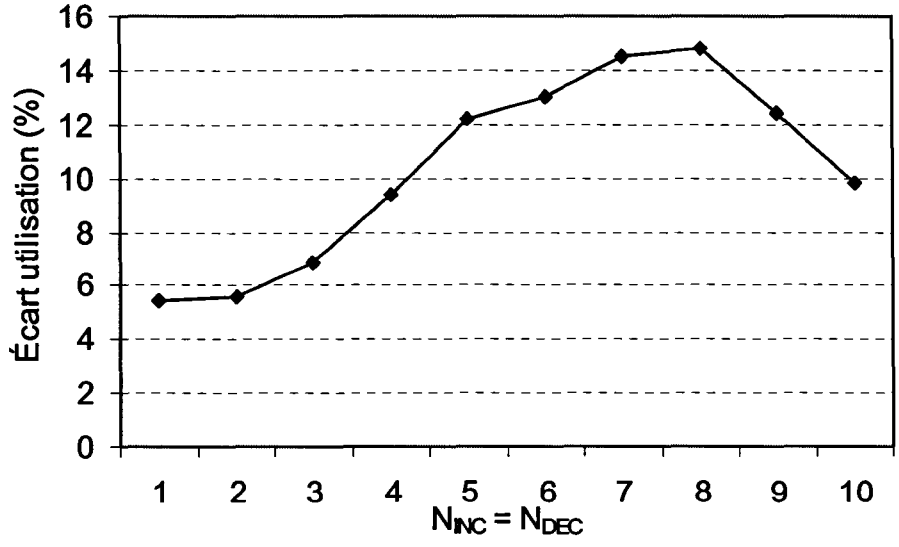
Lorsque les valeurs $N_{INC} = N_{DEC}$ augmentent, les taux d'utilisation (graphique 13) et de rejet (graphique 14) augmentent. L'augmentation du taux de rejet s'explique probablement par la valeur N_{INC} qui spécifie le nombre d'appels devant être rejetés avant d'augmenter la bande passante. N_{DEC}, qui spécifie une quantité d'appels acceptés avant de diminuer la bande passante, contribue sans doute à améliorer le taux de rejet. Pour ce qui est du nombre de modifications (graphique 15), plus la valeur de ces paramètres est élevée, plus il doit y avoir d'appels refusés ou acceptés avant de modifier la réservation. Une valeur élevée pour ces paramètres diminue le nombre de modifications de bande passante en ralentissant la vitesse de réaction, de l'algorithme, aux modifications du taux de trafic.

L'étude des graphiques 16 et 17 démontre que peut importe la valeur de N_{INC} et N_{DEC}, les taux de rejet et d'utilisation pratique sont, en moyenne, 10% en déca de ce que permet la théorie. Les résultats pratiques s'approchent de la théorie uniquement lorsque

N_{INC} et N_{DEC} ont une valeur minimale ou maximale, ce qui impose un taux de rejet maximum ou un taux d'utilisation minimum.



Graphique 16 Écart entre les taux de rejet théorique et pratique



Graphique 17 Écart entre les taux d'utilisation théorique et pratique

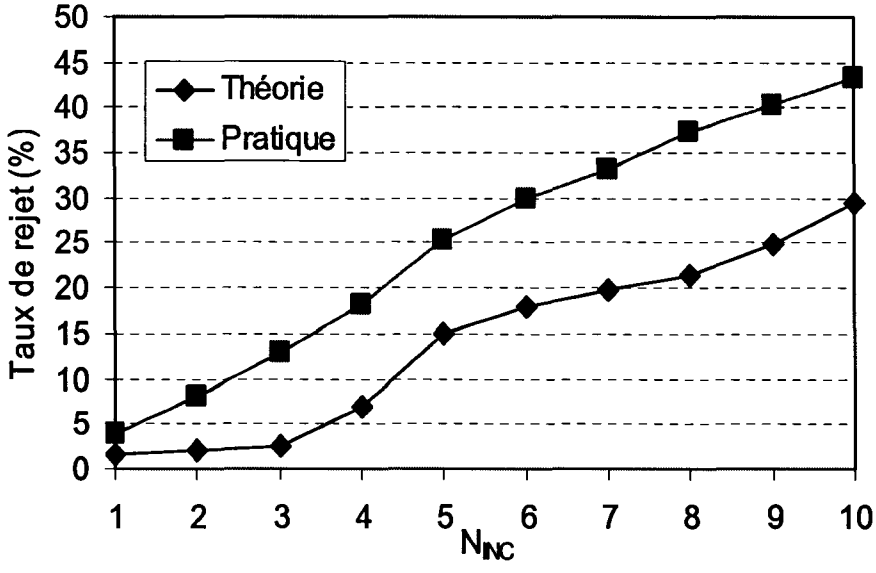
RÉSULTAT DBP, Scénario 2 : $N_{DEC} = 5$, $N_{INC} = 1$ à 10

Maintenant que l'on connaît le comportement de l'algorithme lorsque l'on augmente N_{INC} et N_{DEC} avec une même valeur, voyons ce qui se passe lorsque N_{DEC} est constant et que N_{INC} varie.

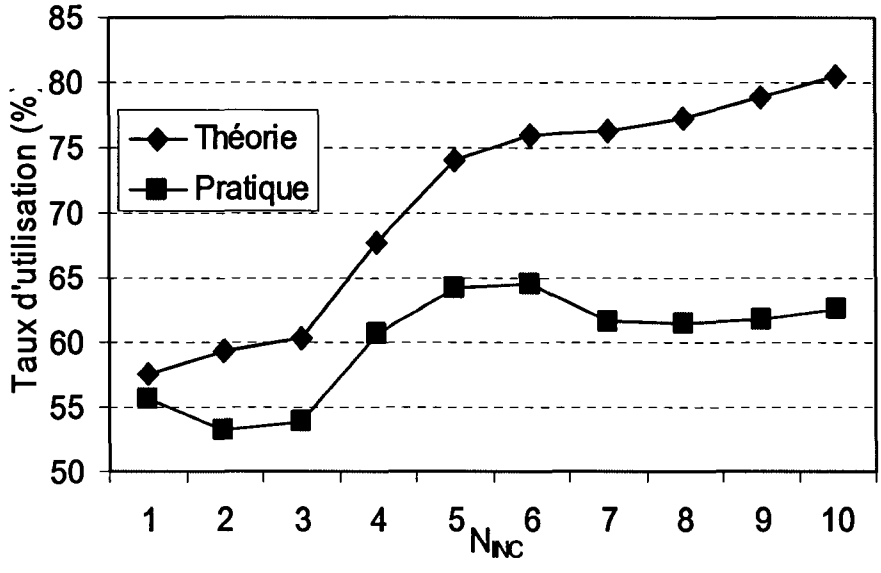
Tableau VI

Résultats DBP, $N_{INC} = 1$ à 10

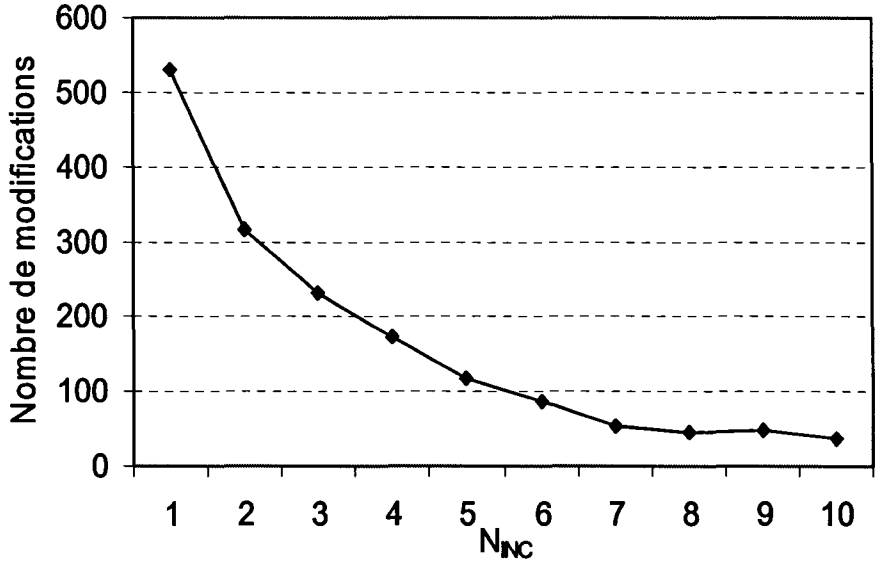
N_{INC}	Nb de circuits Moyenne	Taux de trafic réel Erlang	Rejet théorique %	Rejet pratique %	Utilisation théorique %	Utilisation pratique %	Modif
1	15,812	9,249	1,526	3,91	57,601	55,689	531
2	15,015	9,101	2,114	7,97	59,335	53,282	316
3	14,644	9,081	2,612	12,88	60,392	53,859	233
4	12,848	9,348	6,894	18,19	67,741	60,570	172
5	10,576	9,212	15,014	25,26	74,028	64,255	117
6	10,131	9,368	17,878	29,86	75,935	64,519	86
7	9,487	9,040	19,879	33,07	76,343	61,655	52
8	9,379	9,216	21,350	37,24	77,283	61,431	45
9	8,830	9,283	24,951	40,23	78,897	61,831	48
10	8,155	9,311	29,485	43,39	80,507	62,656	36



Graphique 18 Taux de rejet en fonction de N_{INC}



Graphique 19 Taux d'utilisation en fonction de N_{INC}



Graphique 20 Nombre de modifications en fonction de N_{INC}

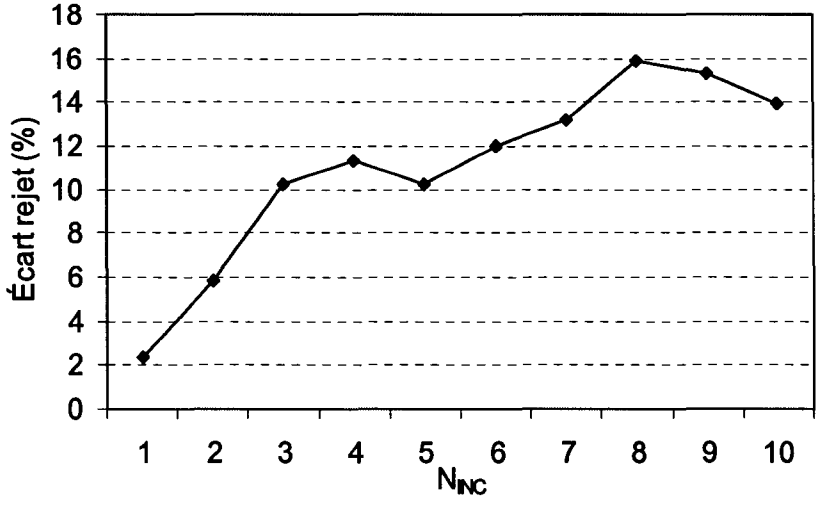
ANALYSE DBP, Scénario 2 : $N_{DEC} = 5$, $N_{INC} = 1$ à 10

Comme le laissait suggérer les résultats de l'analyse précédente, le paramètre N_{INC} à une très grande influence sur le taux de rejet (graphique 18). En fait, le taux de rejet est directement proportionnel à ce paramètre. Pour N_{INC} passant de 1 à 10, le taux de rejet théorique passe de 1,5% à 30% et le taux de rejet pratique de 4% à 40%. Plus N_{INC} est élevé, plus l'algorithme doit rejeter d'appels avant d'augmenter la bande passante.

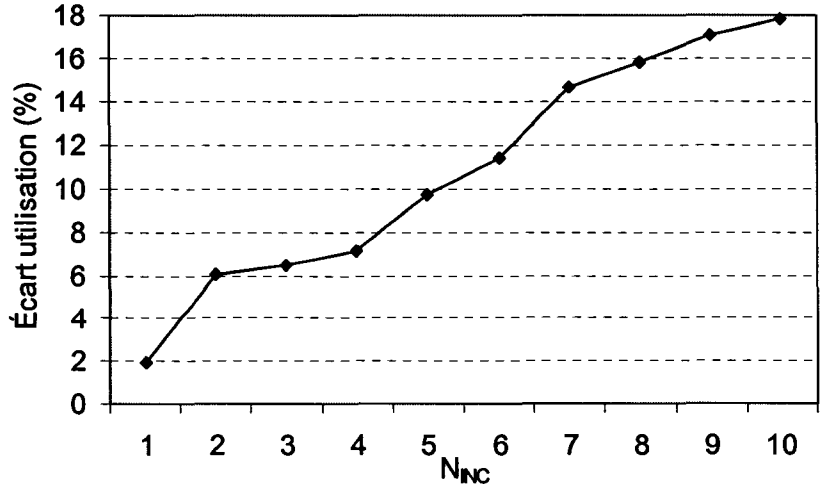
Comme le taux de rejet augmente, il est naturel que le taux d'utilisation augmente également. Le taux d'utilisation passant de 55% à 62% (graphique 19) plutôt que de 57 à 70% (graphique 14), le paramètre N_{DEC} aura également une influence sur le taux d'utilisation. Le nombre de modifications de la bande passante diminue au même rythme que l'on augmente la valeur de N_{INC} (graphique 20), ce qui est conforme à l'hypothèse posée lors de l'analyse précédente.

Les graphiques 21 et 22 nous confirment que pour obtenir des taux s'approchant de ceux permis par la théorie, la valeur de N_{INC} doit être minimale, une valeur de 0 permettant

sûrement d'égaler la théorie. Cependant, cela signifierait que la bande passante peut être modifiée pour chaque appel entrant. Autant oublier le modèle *Per Call Reservation with Path-based Allocation Bandwidth* et utiliser une réservation pour chaque appel. Ce résultat est également comparable à celui obtenu pour l'algorithme ABW pour lequel les taux pratiques s'approchaient de la théorie lorsque la bande passante réservée était suffisamment grande pour accepter tous les appels.



Graphique 21 N_{INC}, écart entre les taux de rejet théorique et pratique



Graphique 22 N_{INC}, écart entre les taux d'utilisation théorique et pratique

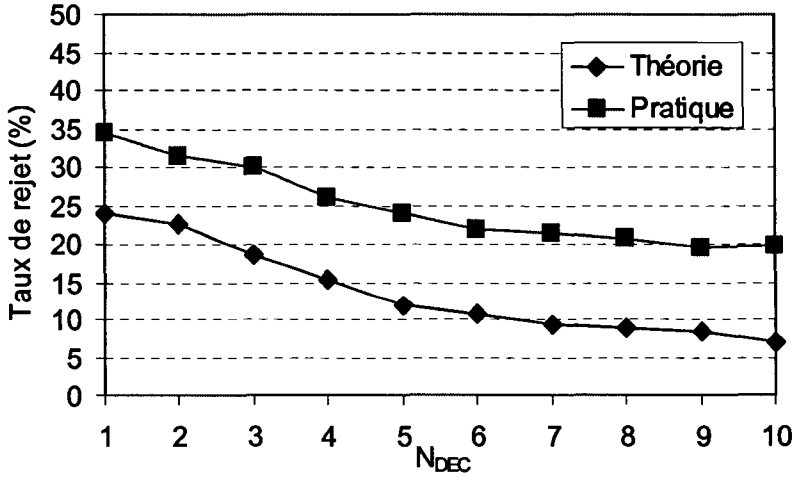
RÉSULTAT DBP, Scénario 3 : $N_{INC} = 5$, $N_{DEC} = 1$ à 10

Voyons maintenant ce qui se passe lorsque le paramètre N_{INC} est constant et que N_{DEC} varie.

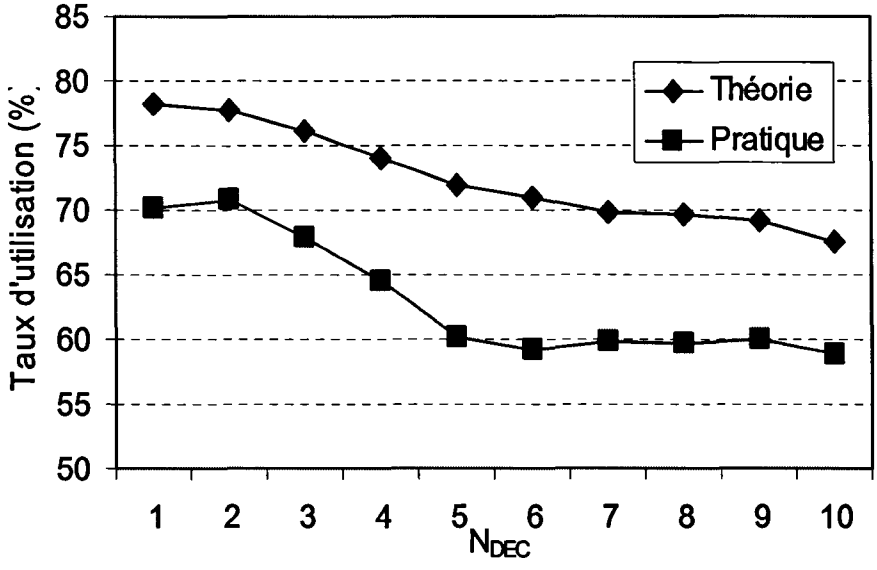
Tableau VII

Résultats DBP, $N_{DEC} = 1$ à 10

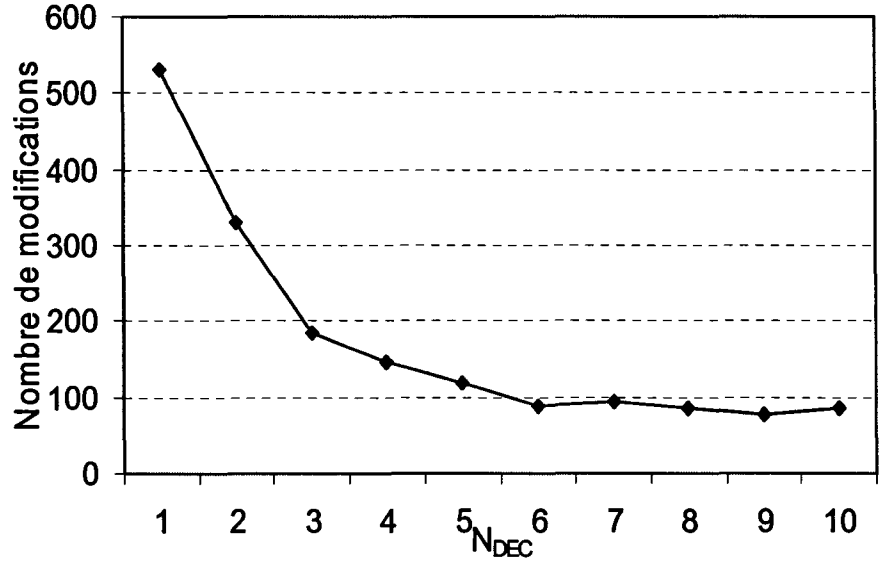
N_{DEC}	Nb de circuits Moyenne	Taux de trafic réel Erlang	Rejet théorique %	Rejet pratique %	Utilisation théorique %	Utilisation pratique %	Modif
1	8,820	9,083	24,014	34,50	78,246	70,167	531
2	9,072	9,095	22,501	31,46	77,696	70,851	330
3	9,901	9,249	18,531	30,03	76,109	67,897	184
4	10,426	9,105	15,256	26,12	74,008	64,528	147
5	11,297	9,212	11,805	23,91	71,918	60,133	117
6	11,575	9,199	10,667	21,80	70,994	59,216	87
7	11,822	9,121	9,416	21,39	69,889	59,886	94
8	12,027	9,193	8,905	20,81	69,634	59,632	86
9	12,307	9,296	8,407	19,48	69,185	59,981	78
10	12,805	9,296	6,870	19,72	67,608	58,861	84



Graphique 23 Taux de rejet en fonction de N_{DEC}



Graphique 24 Taux d'utilisation en fonction de N_{DEC}



Graphique 25 Nombre de modifications en fonction de N_{DEC}

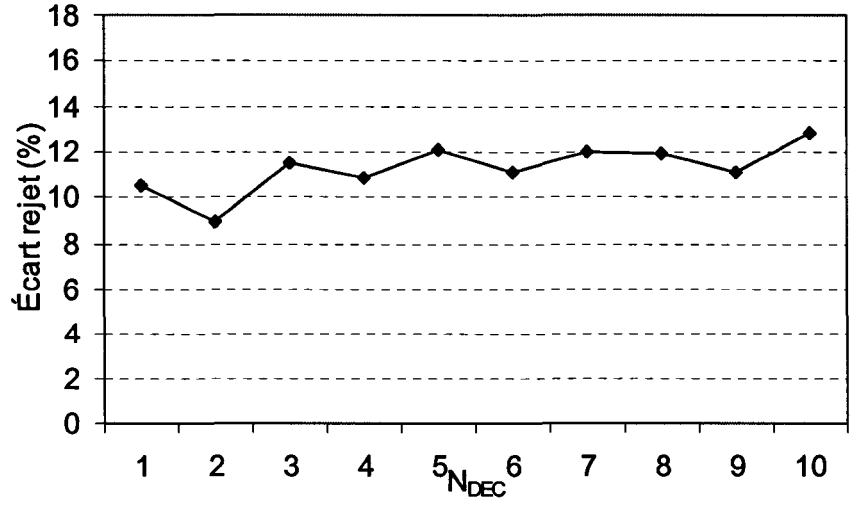
ANALYSE DBP, Scénario 3 : $N_{INC} = 5$, $N_{DEC} = 1$ à 10

L'analyse du scénario 1 laissait suggérer que le taux de rejet était fonction de la valeur de N_{INC}. En fait, la valeur de N_{DEC} à également une influence sur le taux de rejet

(graphique 23). L'augmentation de la valeur du paramètre N_{DEC} permet de diminuer le taux de rejet. Cette diminution du taux de rejet (-15% pour N_{DEC} allant de 1 à 10) est par contre moins rapide que l'augmentation due au paramètre N_{INC} (graphique 18) (+30% pour N_{INC} allant de 1 à 10). Ce comportement s'explique par le fait qu'un N_{DEC} élevé signifie qu'il faut plus de temps pour diminuer la bande passante, évitant ainsi plusieurs refus d'appel. Comme le taux de rejet diminue, il est naturel que le taux d'utilisation diminue également (graphique 24). Le taux d'utilisation diminuant théoriquement de 77% à 67% ou pratiquement de 70% à 60%.

Comme pour le paramètre N_{INC}, le nombre de modifications de la bande passante diminue au même rythme que l'on augmente la valeur de N_{DEC} (graphique 25), ce qui est conforme aux analyses des scénarios 1 et 2.

Contrairement au paramètre N_{INC}, N_{DEC} n'a pratiquement aucune influence sur les écarts entre les taux théoriques et pratiques. Ceux-ci demeure relativement constant aux alentours de 10% (graphique 26 et 27). Ces écarts correspondent à ceux obtenus pour $N_{INC} = 5$ dans l'analyse précédente (graphique 21 et 22).



Graphique 26 N_{DEC}, écart entre les taux théorique et pratique

Graphique 27 N_{DEC}, écart entre les taux d'utilisation théorique et pratique

6.5.2 Étude comportementale DBP, paramètres B_{ACC} et B_{DEC}

B_{ACC} et B_{DEC} seront maintenant étudiés. Ils définissent les seuils utilisés par N_{INC} et N_{DEC}, avant d'augmenter ou diminuer la réservation de bande passante. Pour les simulations qui permettront de déterminer l'effet de B_{ACC} et B_{DEC}, les autres paramètres ont été configurés avec une valeur par défaut. Ces valeurs ont été sélectionnées afin d'éviter les comportements extrêmes que pourraient occasionner des valeurs limites. $N_{INC} = N_{DEC} = 5$ et $P_{REJ} = 0,5$. Pour bien comprendre l'effet des paramètres B_{ACC} et B_{DEC}, voici les différents scénarios qui ont été simulés :

- Scénario 4 : Valeur B_{ACC} et B_{DEC} augmentée en conservant écart constant
- Scénario 5 : L'écart entre les valeurs de B_{ACC} et B_{DEC} est augmentée.

Les valeurs qui seront utilisées pour B_{ACC} et B_{DEC} ne peuvent être quelconques. Pour que leur valeur demeure logique avec l'algorithme, elles doivent rencontrer les conditions suivantes. :

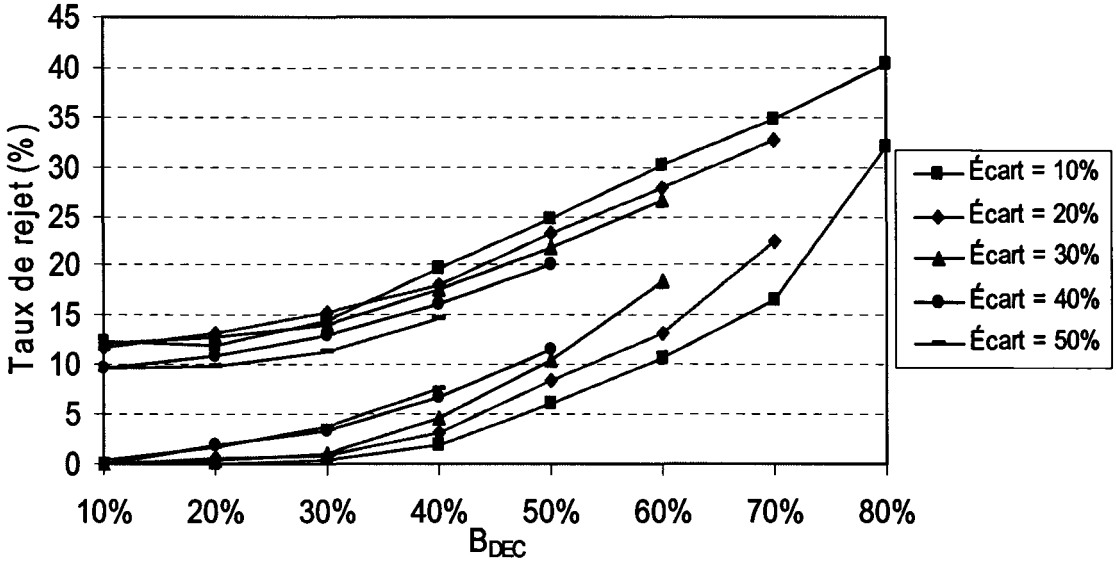
- B_{ACC} ne peut avoir une valeur inférieure ou égale à B_{DEC} ;
- B_{ACC} ne peut prendre la valeur 100% ;
- B_{DEC} ne peut prendre la valeur 0% ;
- L'écart entre B_{ACC} et B_{DEC} ne peut être nul.

RÉSULTAT DBP, Scénario 4 et 5 : B_{ACC} et B_{DEC} avec écart constant et variable

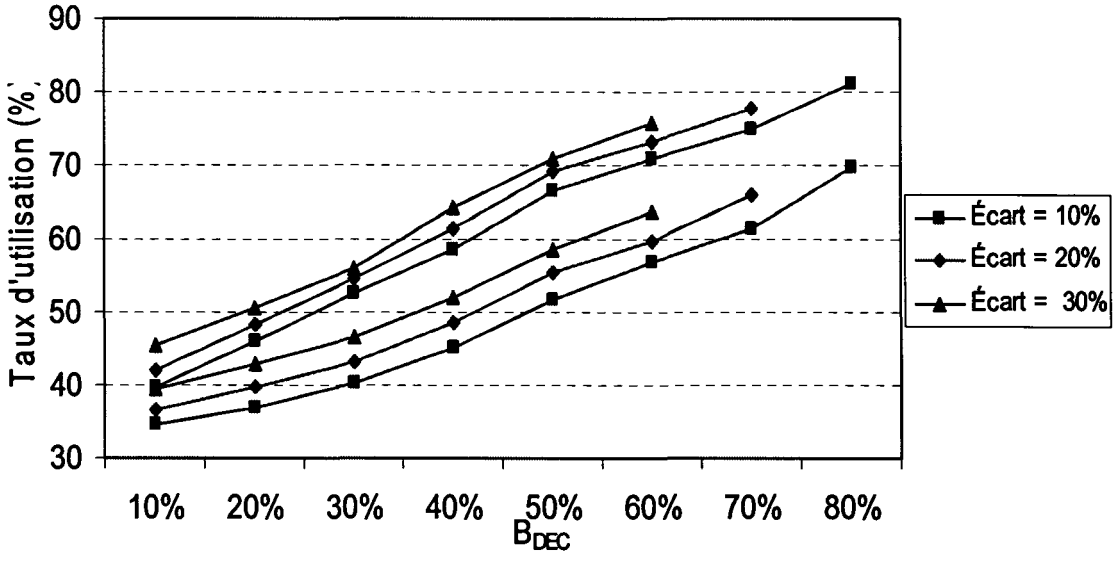
Voyons maintenant ce qui se passe lorsque les paramètres B_{ACC} et B_{DEC} varient

Informations relatives à la lecture des graphiques de résultats

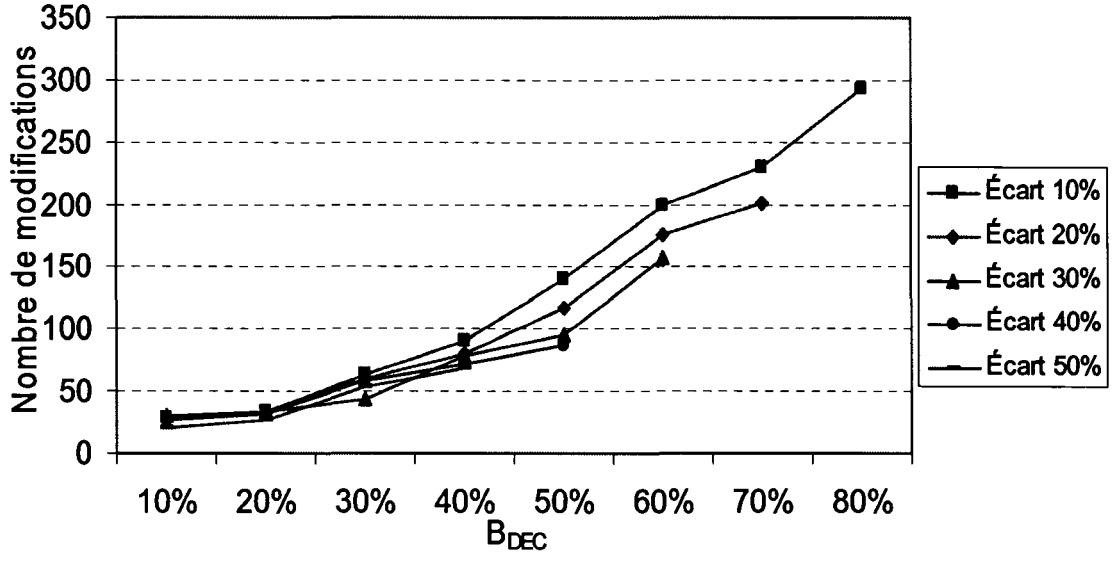
Pour les scénarios 4 et 5, les résultats seront présentés dans les mêmes graphiques. Une courbe correspond à une modification des paramètres B_{ACC} et B_{DEC} conservant un écart constant entre leur valeur. Dans chaque graphique il y aura plusieurs courbes, une pour chaque écart. Dans ces graphiques, seule la valeur de B_{DEC} sera affichée. La valeur de B_{ACC} peut être déterminée en ajoutant l'écart qu'il y a entre B_{ACC} et B_{DEC} (voir légende). Dans chaque graphique, les résultats pratiques et les taux théoriques utilisent la même légende. Pour différencier les courbes théoriques et pratiques, il faut considérer que les résultats pratiques ne peuvent être meilleur que ce que permet la théorie.



Graphique 28 Taux de rejet en fonction de B_{ACC} et B_{DEC}



Graphique 29 Taux d'utilisation en fonction de B_{ACC} et B_{DEC}



Graphique 30 Nombre de modifications en fonction de B_{ACC} et B_{DEC}

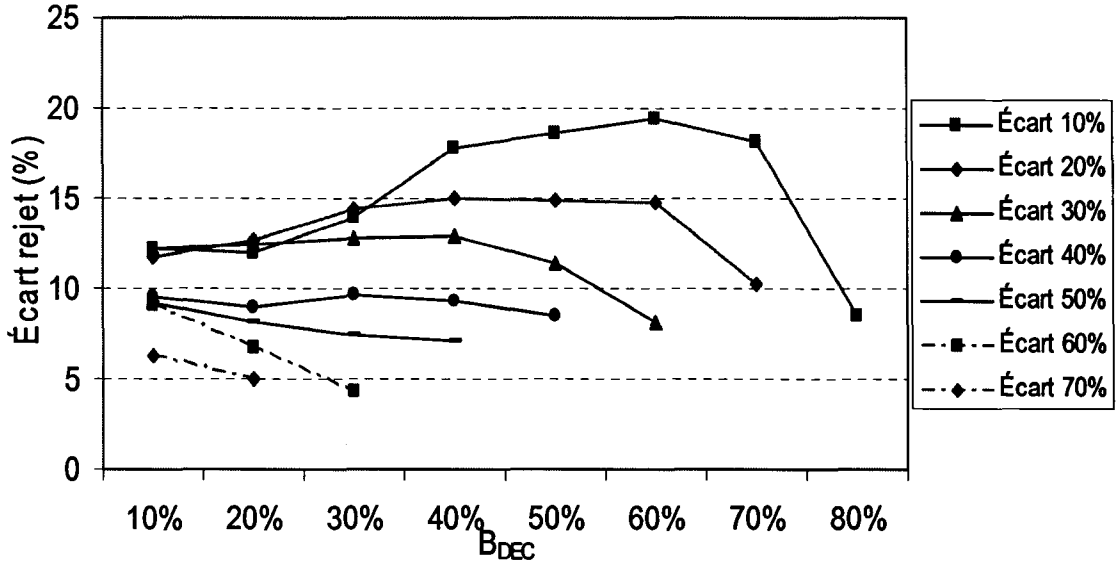
ANALYSE DBP, Scénario 4 et 5 : B_{ACC} et B_{DEC} avec écart constant et variable

L'effet des paramètres B_{ACC} et B_{DEC} sur le taux de rejet peut être étudié à l'aide du graphique 28. Les résultats obtenus démontrent clairement que le taux de rejet augmente lorsque B_{DEC} augmente. Cependant, il faut tenir compte de l'effet qu'a l'écart.

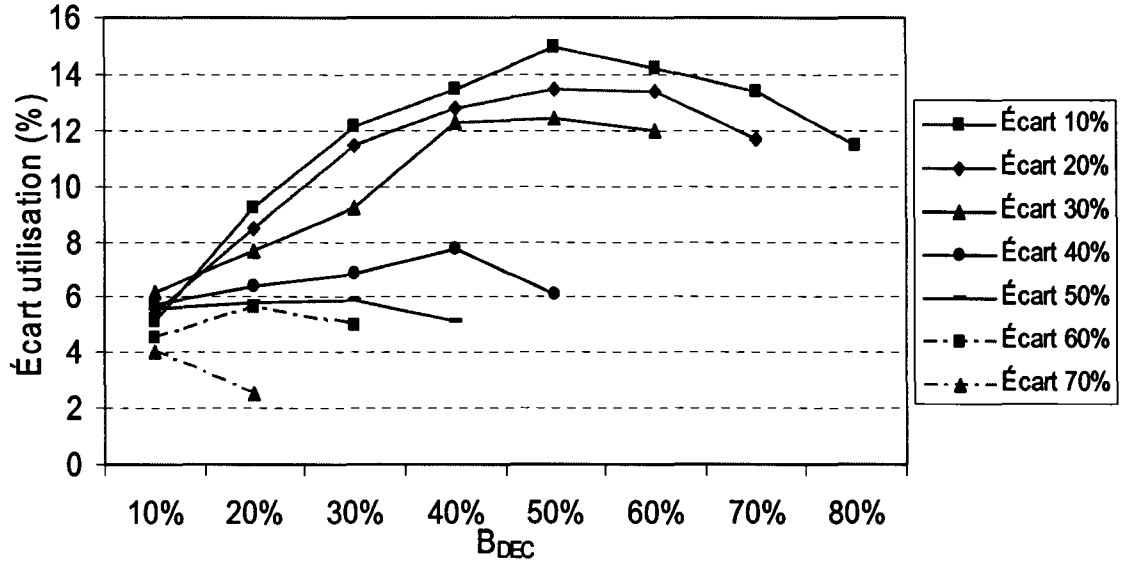
Pour une valeur de B_{DEC} donnée, le taux de rejet pratique diminue lorsque l'on augmente l'écart, c'est-à-dire lorsque l'on augmente B_{ACC}. Un grand écart permet d'obtenir le meilleur taux de rejet, peut importe la valeur de B_{DEC}. Comme le démontre le graphique 31, plus grand est l'écart, plus le taux de rejet pratique s'approche du taux permis par la théorie.

Une valeur minimale de B_{DEC} et un écart maximal, donc une valeur maximale de B_{ACC} semblent donc la configuration idéale. Cela s'explique par le rôle qu'ont ces paramètres dans l'algorithme. Plus B_{DEC} est élevé, plus il est probable que le compteur Dec augmente et que la bande passante soit diminuée. Lorsque la bande passante a diminuée, il faut atteindre un certain nombre de refus avant de l'augmenter à nouveau. Plus B_{ACC} est élevée, moins l'intervalle refusant un appel avec une probabilité P_{REJ} est grand et par conséquent, moins d'appels sont refusés.

L'analyse du taux d'utilisation (graphique 29) nous démontre également qu'un grand écart semble la meilleure configuration pour les paramètres B_{ACC} et B_{DEC}. Peut importe la valeur de B_{DEC}, le taux d'utilisation est meilleur lorsque l'écart est grand. En plus d'obtenir un meilleur taux de rejet et d'utilisation, un grand écart permet aux taux de rejet (graphique 31) et d'utilisation (graphique 32) pratiques d'être le plus près possible de la théorie. Le nombre de modifications (graphique 30) réagit de la même façon. Plus grand est l'écart, moindre est le nombre de modifications.



Graphique 31 Écart entre les taux de rejet théorique et pratique



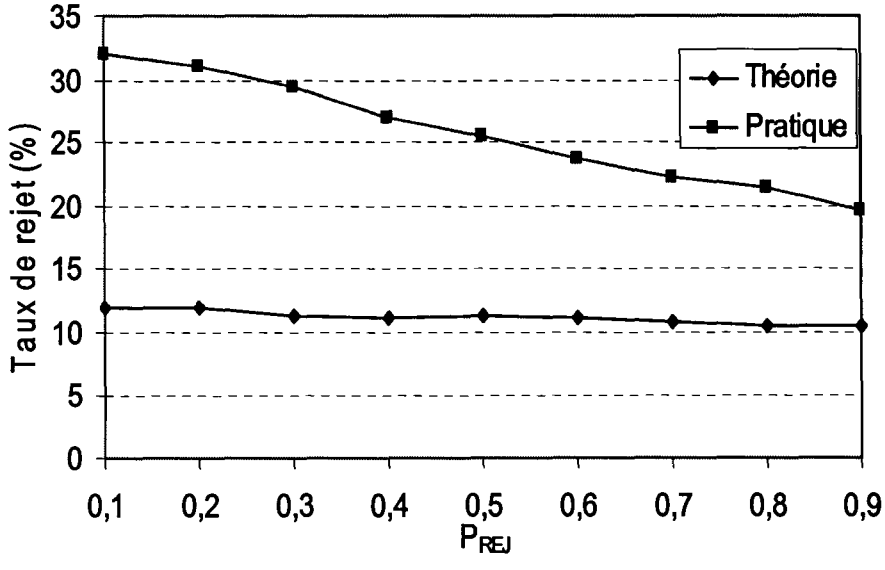
Graphique 32 Écart entre les taux d'utilisation théorique et pratique

6.5.3 Étude comportementale DBP, paramètre P_{REJ}

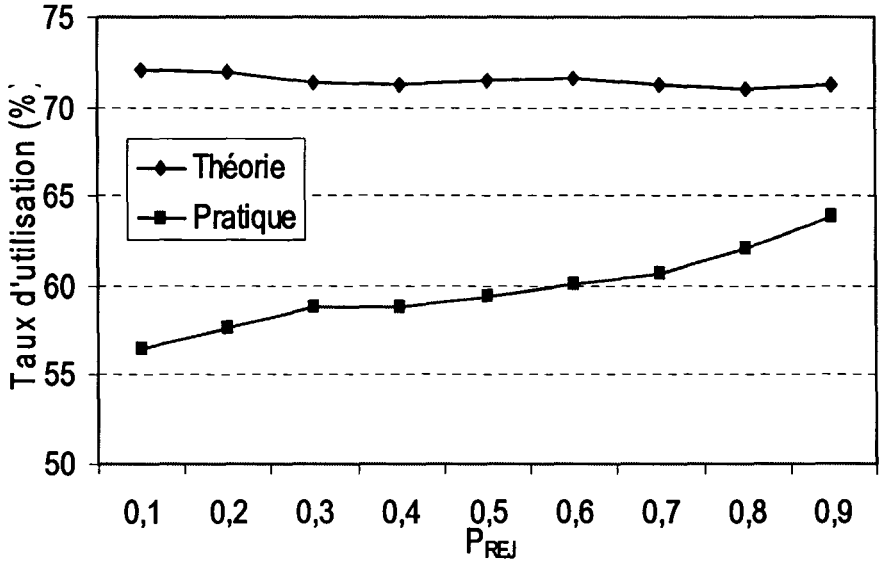
Lorsqu'un appel arrive et que la bande passante utilisée est supérieure à B_{ACC}, un coefficient est calculé. Ce coefficient représente un degré d'utilisation de la bande passante. Si la bande passante utilisée plus celle requise par le nouvel appel est égale à la bande passante réservée, ce coefficient est égale à 1 et l'appel est automatiquement refusé. Si la bande passante utilisée plus celle requise par le nouvel appel est égale à B_{ACC}, ce coefficient est égale à 0 et l'appel est automatiquement accepté. Entre ces deux limites, un appel est refusé si le coefficient d'utilisation est supérieur à P_{REJ}. Il en résulte que la bande passante utilisée ne pourra jamais être égale à celle réservée.

Les prochaines simulations démontrent le comportement de l'algorithme lorsque le paramètre P_{REJ} varie de 0,1 à 0,9. Pour P_{REJ} = 0, le coefficient d'utilisation serait toujours supérieur et l'appel toujours refusé. Pour P_{REJ} = 1, le coefficient serait toujours inférieure et l'appel toujours accepté, au risque de surcharger la bande passante réservée.

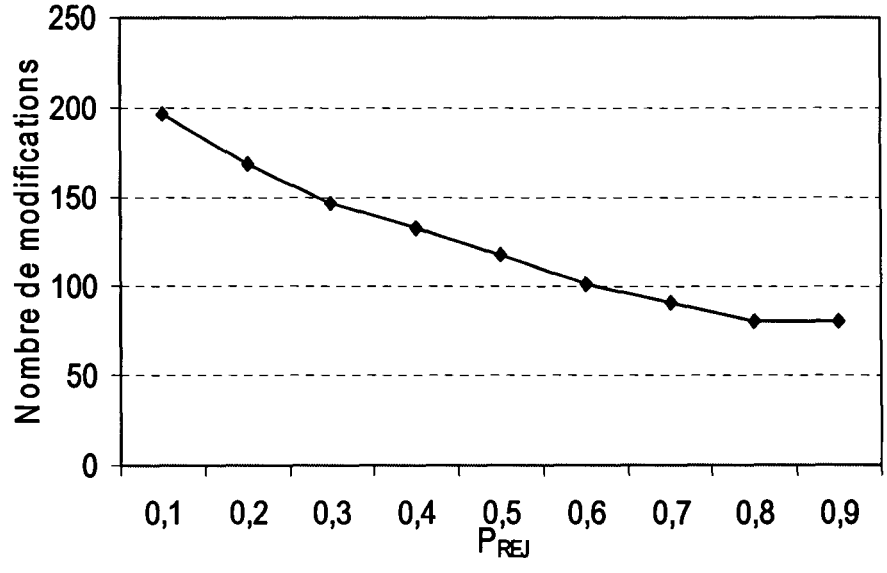
RÉSULTAT DBP, Scénario 6 : Paramètre P_{REJ}



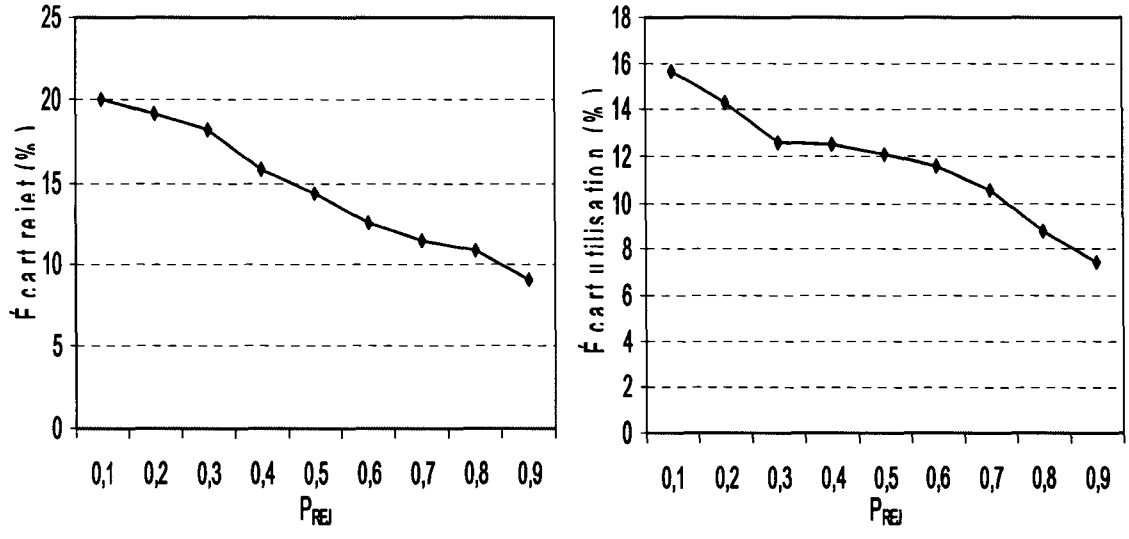
Graphique 33 Taux de rejet en fonction de P_{REJ}



Graphique 34 Taux d'utilisation en fonction de P_{REJ}



Graphique 35 Nombre de modifications en fonction de P_{REJ}



Graphique 36 Écart entre les taux théoriques et pratiques

ANALYSE DBP, Scénario 6 : Paramètre P_{REJ}

Comme on peut le constater sur les graphiques 33 à 36, tous les comportements sont optimaux lorsque le paramètre P_{REJ} est maximal. Un P_{REJ} maximal signifie que les appels ne doivent jamais être refusés lorsqu'il y a suffisamment de bande passante.

6.5.4 Conclusion sur le rôle des paramètres de l'algorithme DBP

L'étude des premiers paramètres, N_{INC} et N_{DEC} a permis de constater que le nombre de modifications est proportionnel à la somme de ces paramètres. Ceux-ci sont responsables de la vitesse de réaction de l'algorithme. Une petite valeur de N_{INC} permet à l'algorithme de réagir rapidement lorsque le taux de trafic augmente. Une petite valeur de N_{DEC} permet à l'algorithme de réagir rapidement lorsque le taux de trafic diminue. Plus l'algorithme est rapide, plus il nécessite de modifications.

Le paramètre N_{INC} influence grandement le taux de rejet puisqu'il spécifie la quantité d'appels rejetés avant d'augmenter la bande passante. En plus, augmenter N_{INC} éloigne énormément les taux pratiques de ceux permis par la théorie. Pour sa part, N_{DEC} peut

prendre la valeur désirée sans affecter l'écart théorique - pratique. Son augmentation diminue le nombre de modifications et le taux de rejet. Une valeur minimale pour N_{INC} et maximale pour N_{DEC} sont les valeurs idéales.

Pour leur part, les paramètres B_{ACC} et B_{DEC} permettent des comportements idéaux lorsque leur valeur est respectivement maximale et minimale. Bien qu'une valeur minimale soit optimale pour B_{DEC}, une valeur nulle aurait pour effet que la bande passante ne soit jamais diminuée. Bien qu'une valeur maximale de B_{ACC} soit optimale, une valeur égale à 100% de la bande passante réservée aurait pour effet de rendre inopérant le paramètre P_{REJ}.

Constatation intéressante, pour que l'algorithme obtienne des résultats s'approchant de ceux permis par la théorie, il faut qu'il y ait un minimum de modifications. Cela peut paraître intéressant, mais à la limite, cela revient à dire que l'algorithme effectue une réservation constante basée sur un estimé à long terme du trafic moyen.

6.6 Étude de l'algorithme DBP simplifié

Il a été démontré dans la section précédente qu'une valeur maximale pour B_{ACC} permet d'optimiser les trois comportements de l'algorithme. De la même façon, il a été démontré que refuser un appel, si un coefficient d'utilisation est supérieur au paramètre P_{REJ}, est inutile car tous les comportements sont optimaux lorsque aucun appel n'est rejeté pour cette raison. Ces constations sont à l'origine de l'algorithme étudié dans la présente section.

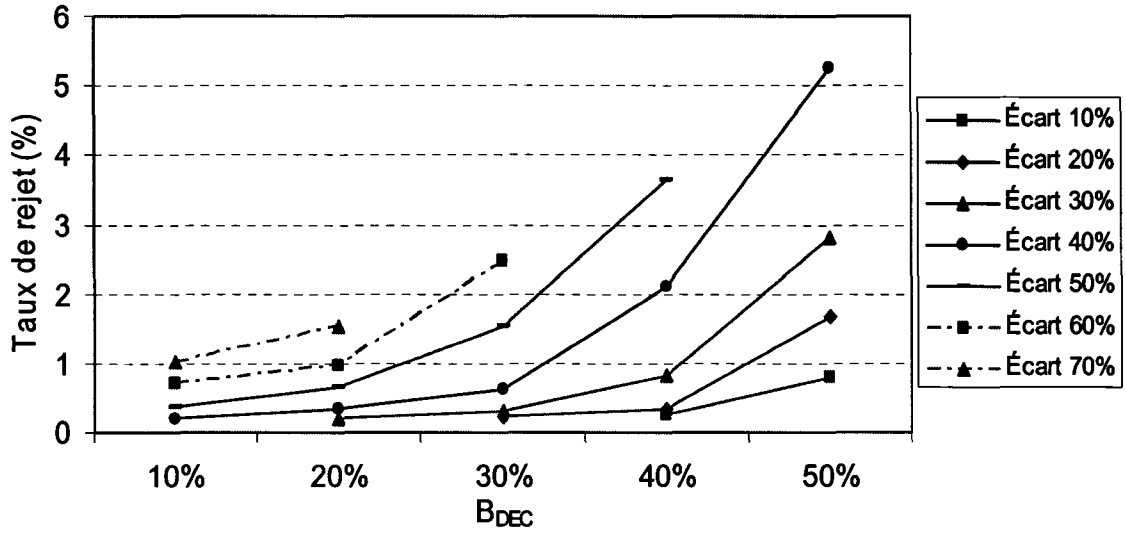
L'algorithme DBP simplifié supprime le paramètre P_{REJ}. Un appel est rejeté seulement si la bande passante n'est pas suffisante. N_{INC} appels consécutifs doivent être établis ou refusé avec un niveau de bande passante supérieur à B_{ACC} avant d'augmenter la bande passante. N_{DEC} appels consécutifs doivent être établis avec un niveau de bande passante inférieur à B_{DEC} avant de diminuer la bande passante.

6.6.1 Étude comportementale DBP simplifié

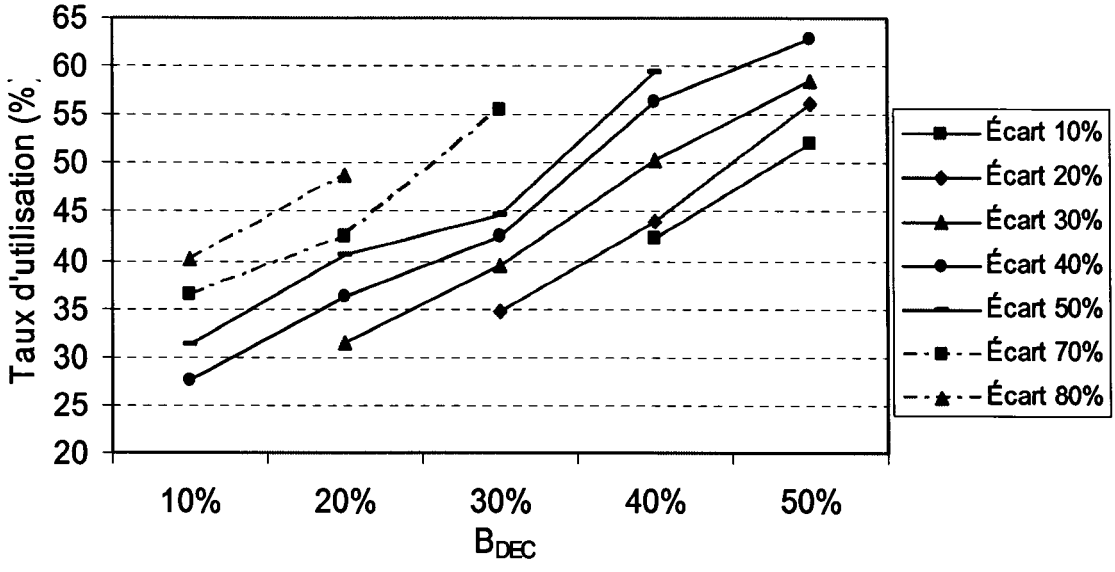
L'élimination du paramètre P_{REJ} n'affectant pas le comportement des paramètres N_{INC} et N_{DEC}, ceux-ci ne seront pas étudiés à nouveau. Les paramètres N_{INC} et N_{DEC} sont associés à la rapidité de réaction de l'algorithme. Pour modifier rapidement la bande passante à la hausse, il faut un petit N_{INC}, pour modifier lentement la bande passante à la baisse, il faut un N_{DEC} élevé. Le nombre total de modifications est proportionnel à la somme de ces deux paramètres.

Pour les simulations qui permettront de déterminer l'effet des paramètres B_{ACC} et B_{DEC}, les paramètres N_{INC} et N_{DEC} utiliseront la même valeur que celles utilisées pour l'étude des paramètres B_{ACC} et B_{DEC} de l'algorithme original. De cette façon, toutes modifications des comportements associés aux paramètres B_{ACC} et B_{DEC} proviendront de la simplification de l'algorithme étudié dans les prochaines lignes.

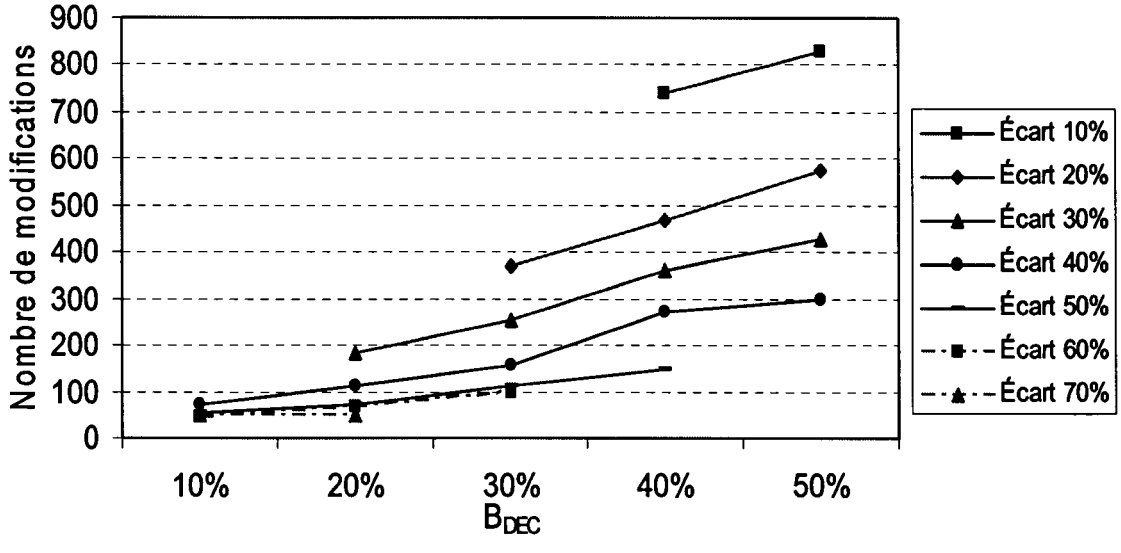
RÉSULTATS DBP simplifié, Paramètres B_{ACC} et B_{DEC}



Graphique 37 Taux de rejet en fonction de B_{ACC} et B_{DEC}



Graphique 38 Taux d'utilisation en fonction de B_{ACC} et B_{DEC}

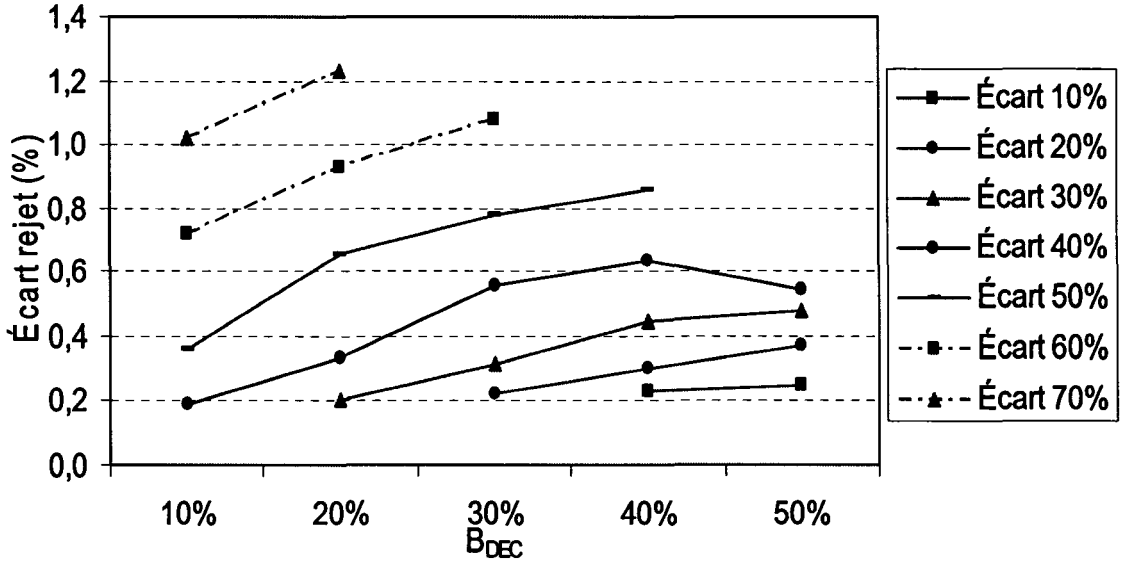


Graphique 39 Nombre de modifications en fonction de B_{ACC} et B_{DEC}

ANALYSE DBP simplifié, Paramètres B_{ACC} et B_{DEC}

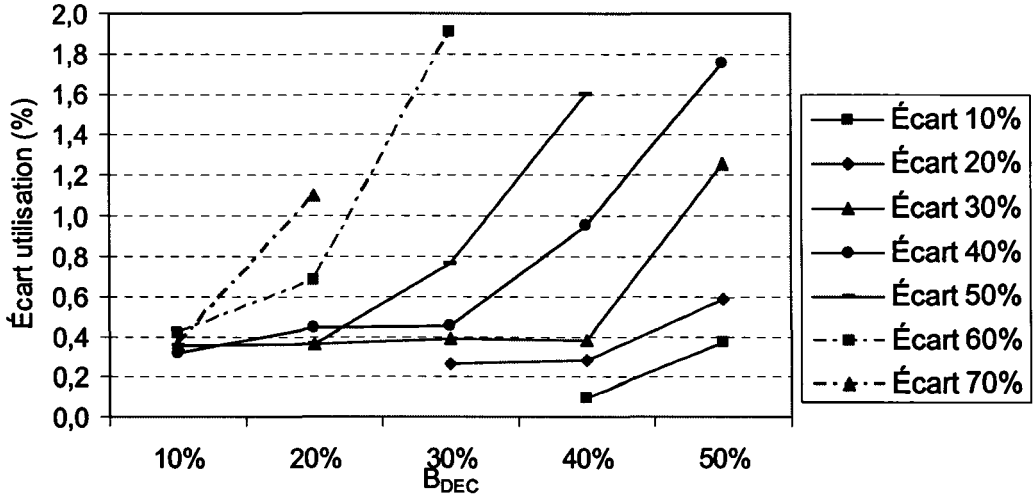
A première vue, l'algorithme simplifié permet d'obtenir de bien meilleur taux de rejet (graphique 37). Une comparaison des graphiques 28 et 37 permet de constater un gain

de 10% à 15%. En fait, la simplification de l'algorithme a permis de diminuer énormément l'écart entre les taux pratique et théorique. Dans l'algorithme original, l'écart entre les taux théorique et pratique était de 5 à 20% (graphique 31). Cet écart est maintenant de 0,2 à 1,2% (graphique 40). Ce gain provient de l'élimination du paramètre P_{REJ}. Celui-ci rejetait des appels avant que la bande passante réservée soit complètement utilisée.



Graphique 40 Écart entre les taux de rejet théorique et pratique

L'algorithme simplifié semble par contre diminuer le taux d'utilisation. Le graphique 29 présente un taux d'utilisation de 40 à 80% comparativement au présent algorithme qui affiche des taux d'utilisation de 25 à 60% (graphique 38). En fait, avec l'algorithme original, le taux de rejet étant beaucoup plus élevé, 10 à 40% (graphique 28), il est normal que le taux d'utilisation soit beaucoup plus élevé. Afin de comparer les deux algorithmes, il serait plus juste d'étudier les écarts théorie – pratique. Alors que l'algorithme original permettait un taux d'utilisation qui s'éloignait de 5 à 15% de la théorie (graphique 32), le présent algorithme affiche des taux d'utilisation se situant de 0,1 à 2% des taux théoriques (graphique 41).



Graphique 41 Écart entre les taux d'utilisation théorique et pratique

Pour ce qui est du nombre de modifications, l'algorithme original provoquait de 25 à 300 modifications (graphique 30), le nombre diminuant pour un écart élevé. Avec cet algorithme simplifié, le nombre de modifications diminue lorsque l'écart augmente (graphique 39), cependant, l'effet de cet écart est beaucoup plus grand. Pour un écart de 10%, 3 fois plus de modifications sont nécessaire avec l'algorithme simplifié. Pour un grand écart, bien que le nombre de modifications soit très respectable, environ 100, il demeure supérieur à ce qu'occasionnait l'algorithme original. C'est que dans l'algorithme original, si un appel était accepté avec $B_{OCC} > B_{ACC}$, le compteur Inc était remis à 0. Comme le niveau d'utilisation était très élevé, cela contribuait à diminuer le nombre de modifications au détriment d'un taux de rejet plus élevé. Le présent algorithme conserve l'état du compteur Inc lorsque $B_{OCC} > B_{ACC}$. Cela permet au taux de rejet pratique de s'approcher du taux théorique au détriment d'un nombre plus élevé de modifications.

Dans certains cas, un taux de rejet plus élevé pourrait être désiré afin d'améliorer le taux d'utilisation. Pour ce faire, il suffit de maximiser la valeur de B_{ACC} afin de réduire l'intervalle B_{ACC} - B_{CON}. Cela se fait par contre au détriment du taux d'utilisation qui s'éloigne des taux permis par la théorie (graphique 41).

6.6.2 Résumé et comparaison des algorithmes DBP

L'algorithme DBP original ne permet que des taux de rejet supérieur à 5% et l'algorithme simplifié ne permet que des taux de rejet inférieur à 5%. Malgré que ces taux soient différents, l'efficacité des algorithmes ne se mesure pas en taux de rejet ou d'utilisation, mais par l'erreur absolue entre les taux pratiques et théoriques.

Le taux de rejet étant toujours supérieur à 5% pour l'algorithme original et inférieur à 5% pour l'algorithme simplifié, est-ce que la diminution des écarts théorie – pratique est réellement une amélioration? Est-ce qu'un écart de 0,58% entre des taux proche de 1% est meilleur qu'un écart de 11,08% entre des taux proche de 10%? Étant donné les différences de taux de rejet impliqué entre DBP et DBP simplifié, la comparaison directe des erreurs absolues ne permet pas une comparaison juste de ces deux algorithmes. Il convient donc de comparer les erreurs relatives.

- $\text{Erreur relative} = \dfrac{\left|\text{Résultat Pratique} - \text{Résultat Théorique}\right|}{\text{Résultat Théorique}}$

Les taux théoriques et pratiques qui sont comparés sont en fait des taux moyens obtenus à partir des taux théoriques et pratiques de l'ensemble des simulations réalisés pour chacun des algorithmes.

Tableau VIII

Comparaison des algorithmes DBP

	Théorique	Pratique	Erreur relative
DBP, taux de rejet	5,64%	16,72%	1,964
DBP-S, taux de rejet	0,67%	1,25%	0,856
DBP utilisation	60,52%	51,87%	0,143
DBP-S utilisation	45,49%	44,80%	0,015

Le calcul des écarts relatifs permet de constater qu'avec l'algorithme original, les taux de rejet pratiques sont 1,96 fois supérieur aux taux théoriques tandis qu'avec l'algorithme simplifié, les taux de rejet pratiques ne sont que de 0,856 fois supérieur aux taux théoriques. Ainsi, l'algorithme simplifié permet un taux de rejet pratique 2,3 fois plus proche de la théorie que l'algorithme original ((1,964) / (0,856)). Pour le taux d'utilisation, le gain est encore meilleur. D'une erreur relative de 0,143 pour l'algorithme original, cette erreur n'est que de 0,015 avec l'algorithme simplifié, soit un gain de ((0,143) / (0,015)) = 9,5.

Bien que l'algorithme simplifié permette d'améliorer les taux de rejet et d'utilisation relatif à la théorie, il demeure qu'il est complexe de configurer ces deux algorithmes pour un taux de rejet ou d'utilisation désiré. De plus, ces deux algorithmes sont complémentaires dans la mesure où le premier permet de bons taux d'utilisation avec taux de rejet élevé et le second permet de bon taux de rejet avec taux d'utilisation peu élevé. Le prochain algorithme aura pour objectif de permette le taux de rejet désiré à l'aide d'un seul algorithme. Il aura également comme objectifs d'améliorer l'écart entre les taux théoriques et pratique et d'ajouter la composante temporelle manquante aux algorithmes DBP.

6.7 Étude de l'algorithme ERL

Un algorithme de contrôle d'accès qui gère la bande passante ne peut avoir qu'un seul objectif, optimiser les taux de rejet et d'utilisation obtenus pour les ressources disponibles. Ceci étant dit, encore faut-il qu'un gestionnaire puisse spécifier s'il préfère un bas taux de rejet combiné à un bas taux d'utilisation ou un haut taux d'utilisation avec un haut taux de rejet. Les algorithmes précédents ne permettaient pas de configurer facilement les taux désirés (utilisation, rejet). L'algorithme proposé, ERL, permettra une configuration simplifiée en ayant un paramètre permettant de configurer le taux de rejet (d'utilisation) désiré. L'algorithme ERL aura également pour objectif d'obtenir des taux pratiques plus proche de la théorie que ce que permettaient les algorithmes DBP, c'est-à-

dire une efficacité accrue. Pour ce faire, ERL intègre une composante temporelle qui était absente des algorithmes DBP.

L'algorithme ERL possède trois paramètres. Le premier spécifie le taux de rejet désiré (P_R). Le second détermine l'intervalle de temps (T) entre deux évaluations. Une évaluation détermine le taux de rejet P_B obtenu dans la période T précédant l'évaluation. P_B est calculé avec la formule d'Erlang, en utilisant le trafic moyen de cette période et le nombre de circuits réservés durant cette même période. Le dernier paramètre (P_E) permet un écart entre P_R et P_B. Si P_B est supérieur de P_E% du taux désiré P_R, le nombre de circuits est augmenté. Si P_B est inférieur de P_E% du taux désiré PR, le nombre de circuits est diminué.

Pour étudier le présent algorithme, les différents paramètres seront étudiés l'un après l'autres. Pour chaque paramètre, plusieurs simulations seront effectuées en faisant varié la valeur du paramètre étudié. Il sera ainsi possible de déterminer l'effet de ce paramètre sur les taux de rejet et d'utilisation obtenus par l'algorithme. Ces taux seront toujours comparés aux valeurs théoriques afin de déterminer la valeur optimale du paramètre étudié qui permet à l'algorithme d'obtenir des taux pratiques s'approchant le plus possible des taux théoriques.

Provenance des résultats

Les valeurs théoriques seront calculées à l'aide des équations (1) et (2) en utilisant le nombre de circuits réservés et le taux de trafic obtenu pour chaque simulation. Étant donné que la bande passante est modifiée au besoin, le nombre de circuits réservés utilisé pour le calcul théorique sera le nombre de circuits moyen obtenu à la fin de la simulation. Les taux pratiques ont été calculés à partir de résultats de simulation. Le taux de rejet pratique est égal à la quantité d'appel refusé divisé par le nombre de requête reçus.

- Rejet pratique = Nb appels refusés / Nb total de requêtes reçues

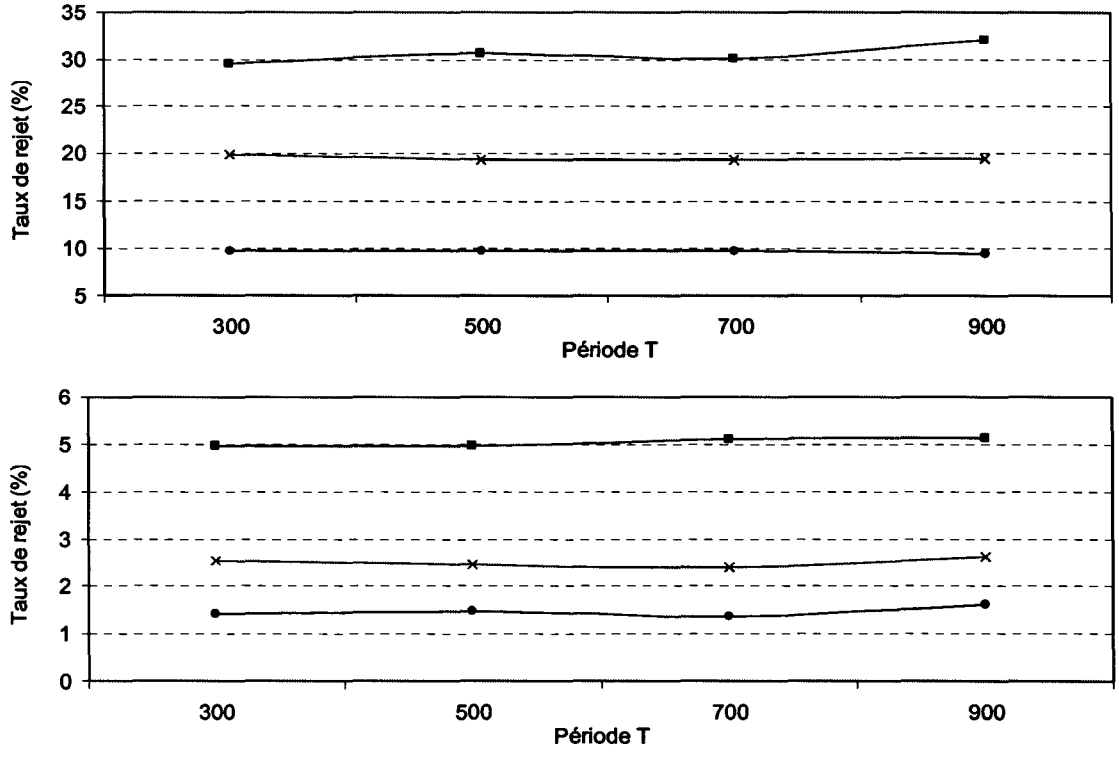
Le taux d'utilisation pratique est déterminé par le nombre de secondes d'utilisation divisé par le nombre de secondes réservées. Le nombre de secondes d'utilisation est l'addition de la durée de chaque appel établi. Le nombre de secondes réservées est la multiplication du nombre de circuits réservés par la durée totale de la simulation.

- Utilisation pratique = Nb de secondes utilisées / Nb. de secondes réservées.

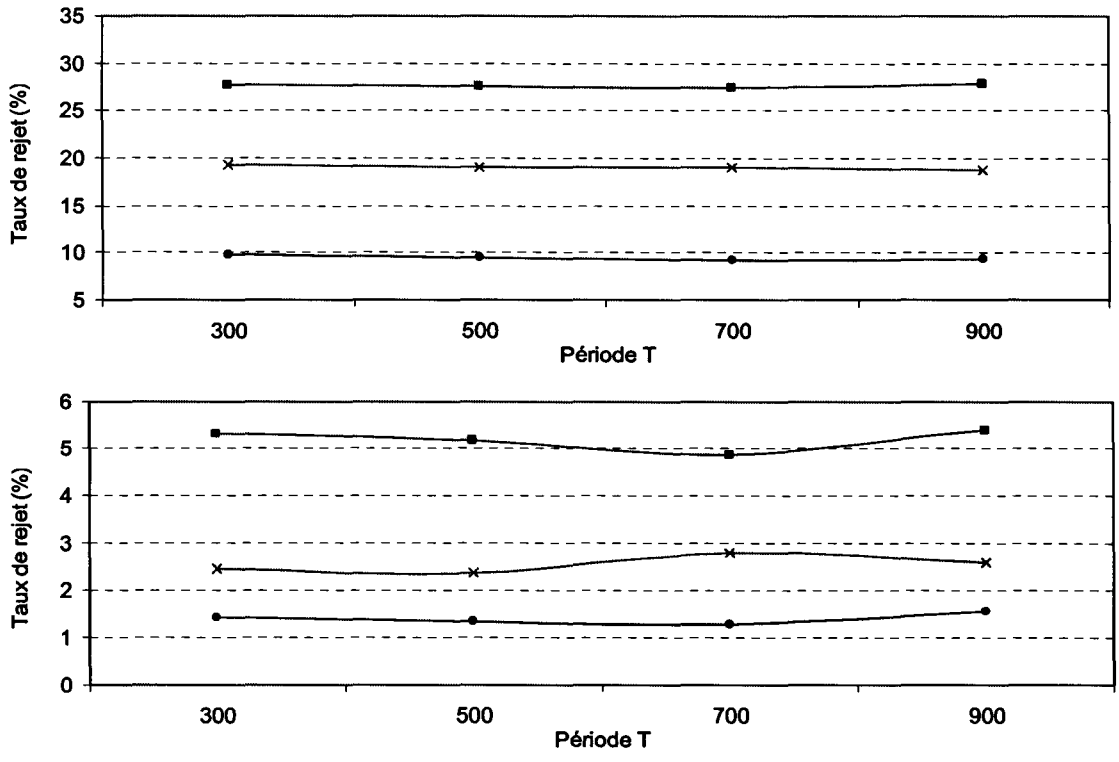
6.7.1 Étude comportementale ERL, paramètre P_R

Le paramètre P_R spécifie le taux de rejet désiré. Le paramètre PR n'affecte pas le fonctionnement de l'algorithme, ni son efficacité, cependant, il faut s'assurer que l'algorithme permet bien d'obtenir ce taux en pratique. Ce sont les paramètres T et P_E qui déterminent le comportement de l'algorithme. Le but de cette étude est de démontrer que l'algorithme permet d'obtenir, peu importe la configuration, un taux de rejet qui s'approche du taux de rejet désiré. Pour cette première étude, le paramètre T prendra des valeurs de 300, 500, 700 et 900 ; le paramètre P_E prendra une valeur de 5% et 50%. Le paramètre P_R est étudié pour des valeurs de 30%, 20%, 10%, 5%, 2% et 1%.

RÉSULTAT ERL : Paramètre P_R



Graphique 42 Taux de rejet obtenus pour P_R = 1, 2, 5, 10, 20 et 30%, T varie, P_E = 50%

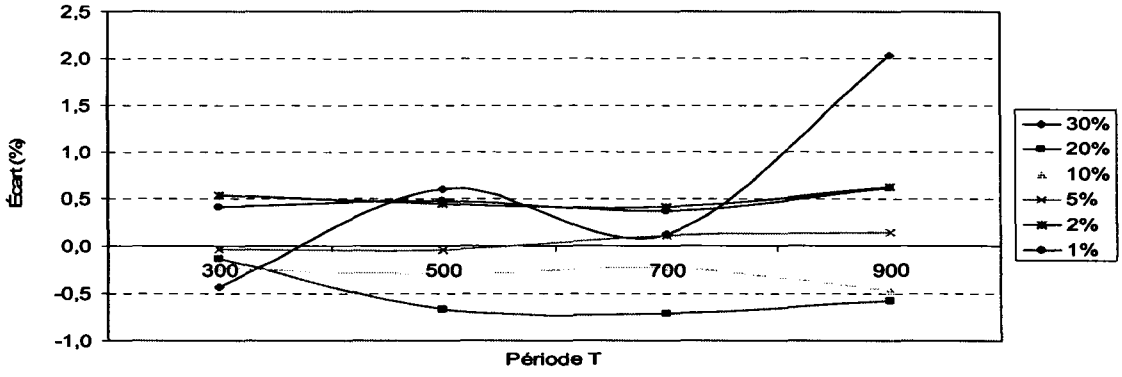


Graphique 43 Taux de rejet obtenus pour P_R = 1, 2, 5, 10, 20 et 30%, T varie, P_E = 5%

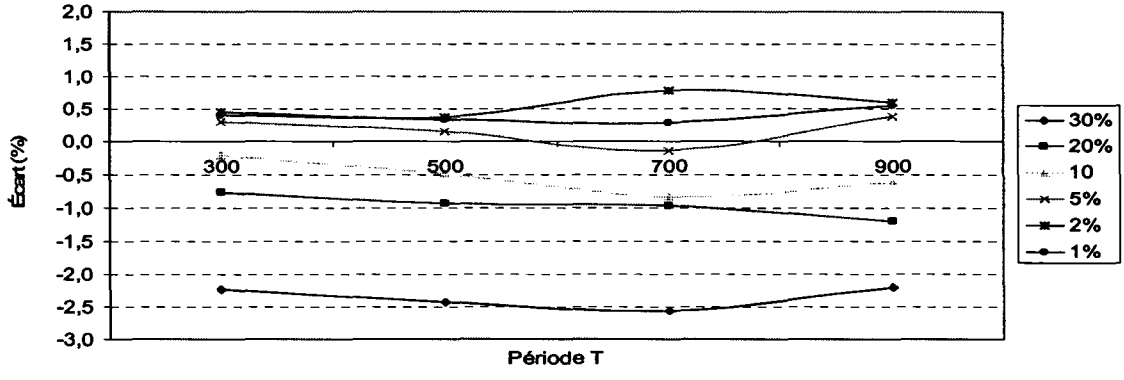
ANALYSE ERL : Paramètre P_R

Les taux de rejet obtenus correspondent relativement bien à ce qui a été configuré. Cependant, ils ne sont pas exacts (graphique 42-43). Les taux obtenus varient un peu avec T et P_E.

Les graphiques 44 et 45 affichent l'écart qu'il y a entre les taux de rejet désiré (P_R) et les taux moyens obtenus. Comme ces graphiques le démontrent, l'algorithme permet d'obtenir un taux de rejet qui varie au maximum de 1% du taux de rejet configuré. Comme l'algorithme ne peut atteindre le taux de rejet désiré à la perfection, P_R ne pourra être utilisé comme taux théorique lors des comparaisons qui seront réalisées dans les analyses qui suivent. Les taux de rejet et d'utilisation théoriques seront donc calculés de la même façon que pour les algorithmes précédents.



Graphique 44 Écart entre les taux désirés et obtenus, $P_E = 50\%$

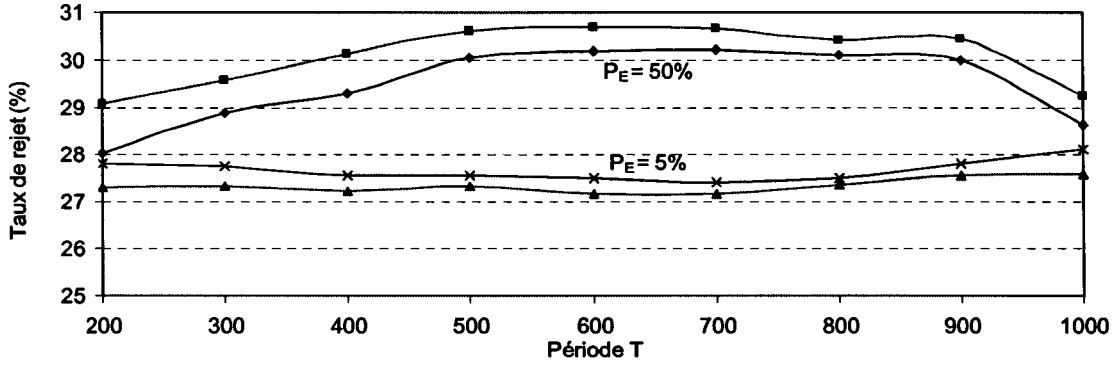


Graphique 45 Écart entre les taux désirés et obtenus, $P_E = 5\%$

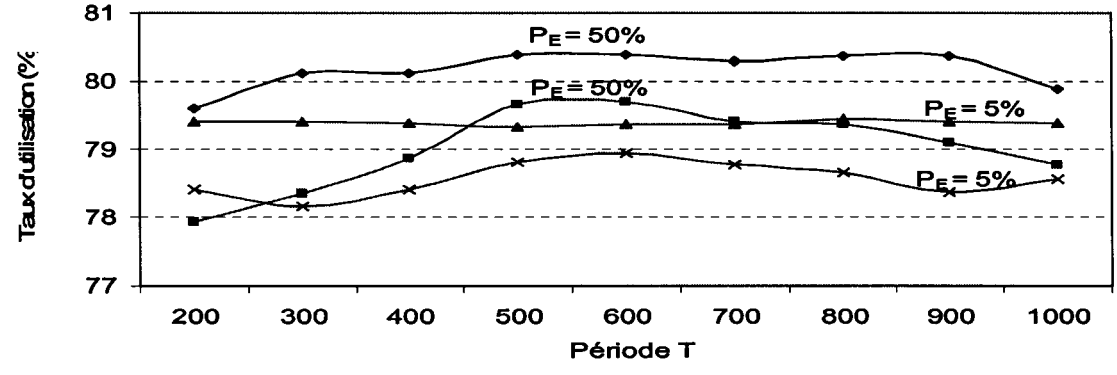
6.7.2 Étude comportementale, paramètre T

Pour réaliser les simulations permettant de déterminer l'effet du paramètre T, le taux de rejet P_R a été configuré à 30%. L'écart de taux de rejet P_E a été configuré à 5% et 50%. Les résultats présenteront donc l'effet du paramètre T lorsque la probabilité de rejet accepté se situe entre 15% et 45% ($P_E = 50\%$) et entre 28,5% et 31,5% ($P_E = 5\%$). Les graphiques présenteront les taux pratiques obtenus et les taux théoriques qu'il aurait été possible d'atteindre. Pour différencier les courbes, il ne faut pas oublier que les taux pratiques ne peuvent être supérieur aux taux théoriques.

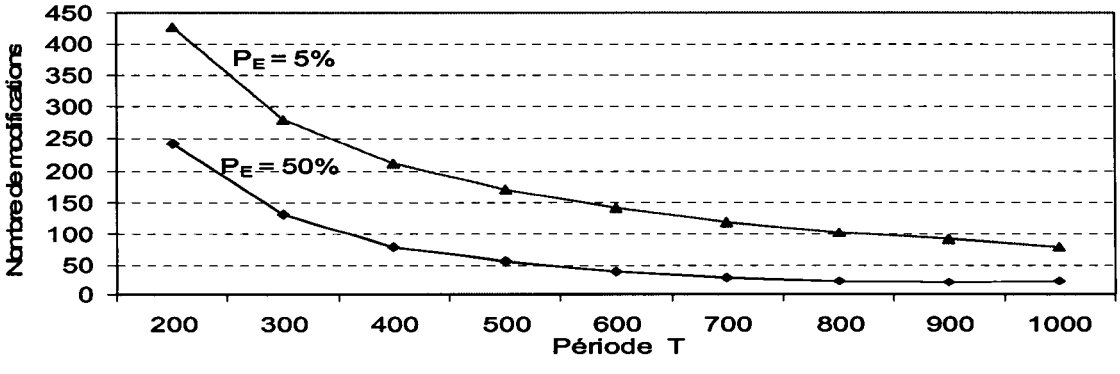
RÉSULTAT ERL : Paramètre T



Graphique 46 Taux de rejet pratique et théorique, en fonction de T



Graphique 47 Taux d'utilisation pratique et théorique en fonction de T



Graphique 48 Nombre de modifications en fonction de T

ANALYSE REL : Paramètre T

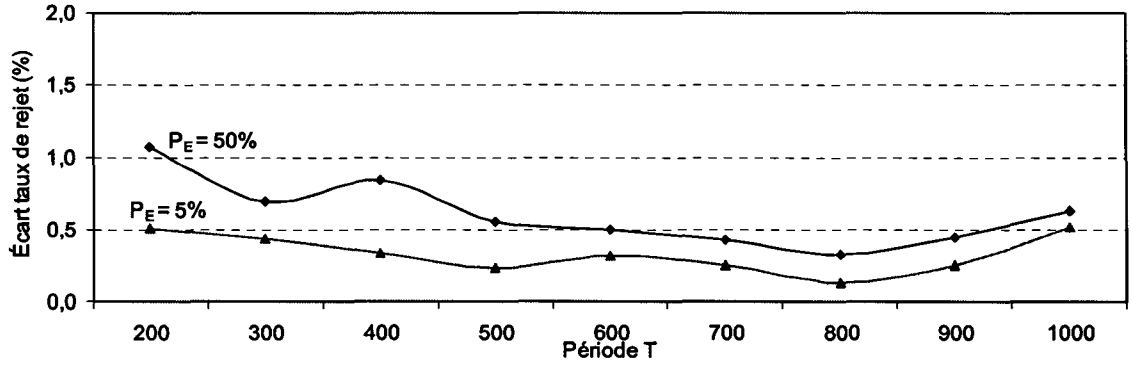
L'analyse du graphique 46 nous démontre que pour approcher le taux de rejet désiré, le paramètre T doit avoir une valeur ni trop grande, ni trop petite. Le taux de rejet théorique s'approche du taux désiré lorsque T se situe entre 500 et 900, même chose pour le taux pratique. Les courbes théoriques et pratiques se distinguent en considérant que les taux de rejet pratiques ne peuvent être inférieur aux taux théoriques. Ce graphique nous démontre également que peu importe la valeur de T, lorsque P_E est petit, il est difficile de s'approcher du taux de rejet désiré. La valeur de T n'affecte pas l'écart entre le taux de rejet théorique et pratique qui demeure presque constant (graphique 49).

Pour le taux d'utilisation, il est normal que celui-ci soit moindre lorsque P_E = 5% (graphique 47) car le taux de rejet obtenu pour P_E = 5% est inférieur à celui obtenu pour P_E = 50% (graphique 46). Le taux d'utilisation maximal, autant pour P_E = 5% que P_E = 50%, se situe aux alentours de T = 500 à 700. L'écart entre les taux d'utilisation théorique et pratique est minimal aux alentours de T = 500 à 700 (graphique 50).

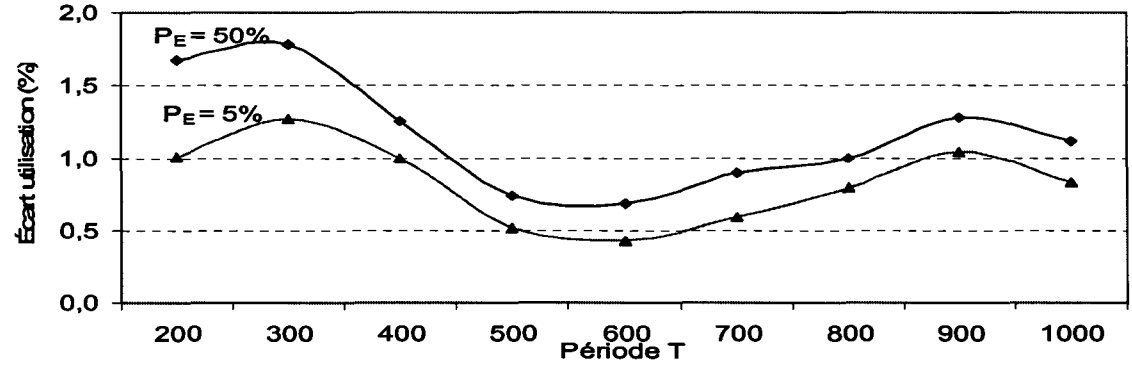
Finalement, comme il était possible de s'y attendre, plus T est grand, moins il y a d'évaluations et moins il y a de modifications (graphique 48). Pour T plus grand que 500, le nombre de modifications est très raisonnable. Ce graphique démontre également que plus l'écart P_E est petit, plus il y aura de modifications. Ce comportement est normal. Si l'intervalle de probabilité de blocage accepté est petit, la probabilité de blocage calculée à plus de chance d'être inférieur ou supérieur à cet intervalle.

L'étude du paramètre T démontre que le délai entre deux évaluations ne doit pas être trop petit ou trop grand. En effet, si cet intervalle est trop petit, le trafic moyen calculé pour cet intervalle risque fortement d'être trop petit ou trop grand dû à la distribution exponentielle du trafic. Si le trafic moyen calculé à chaque intervalle n'est pas assez précis, il en résultera des modifications fréquentes de la bande passante réservée et ces modifications ne pourront faire autrement que d'exagérer, à la hausse ou à la baisse, la

quantité de bande passante nécessaire. À l'inverse, si la période T est trop grande, l'algorithme ne réagira pas assez rapidement à la modification du trafic moyen.



Graphique 49 Écart entre les taux de rejet théorique et pratique

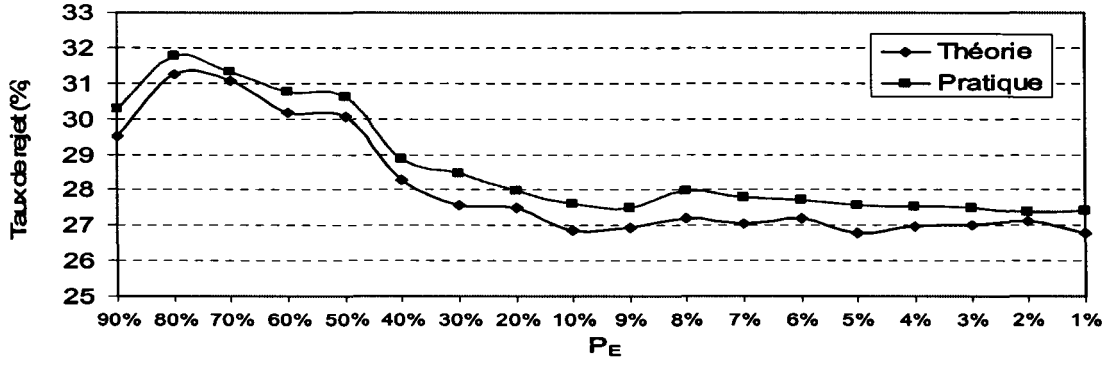


Graphique 50 Écart entre les taux d'utilisation théorique et pratique

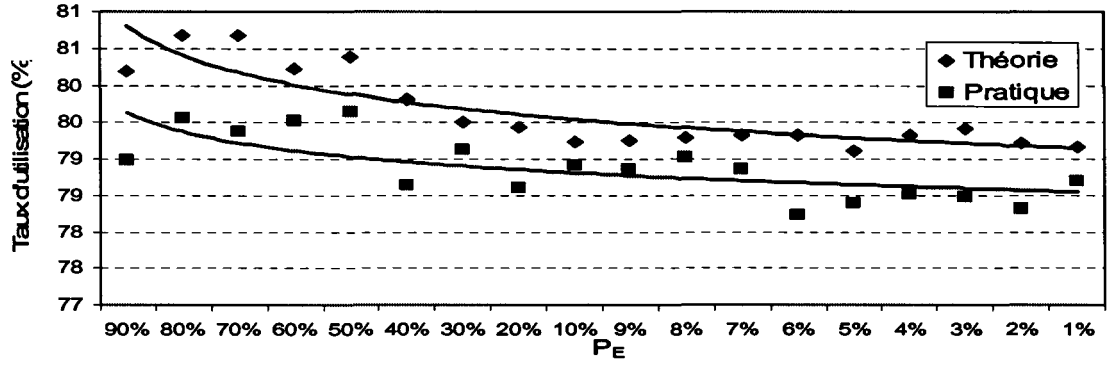
6.7.3 Étude comportementale ERL, paramètre P_E

Pour réaliser les simulations permettant de déterminer l'effet du paramètre P_E, le taux de rejet P_R a été configuré à 30%. La période T a été fixée à 500 secondes, soit une période comprise dans l'intervalle optimale déterminé dans l'étude précédente.

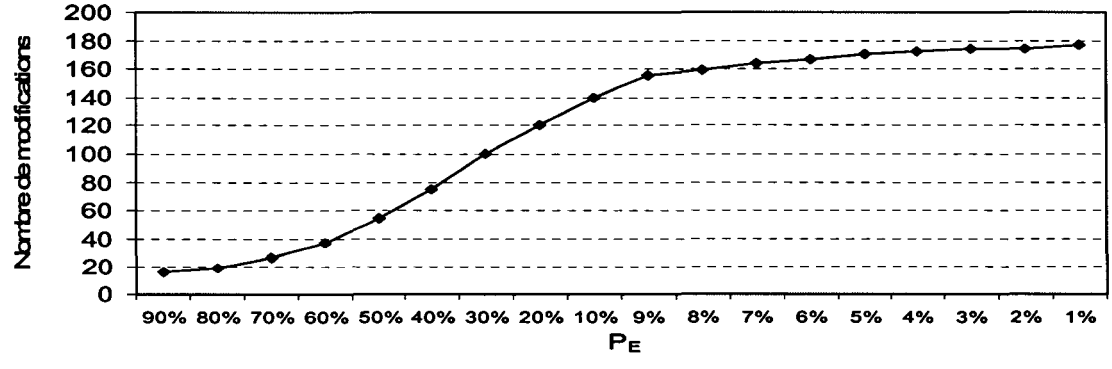
RÉSULTATS : Paramètre P_E



Graphique 51 Taux de rejet obtenu en fonction de P_E



Graphique 52 Taux d'utilisation en fonction de P_E



Graphique 53 Nombre de modifications en fonction de P_E

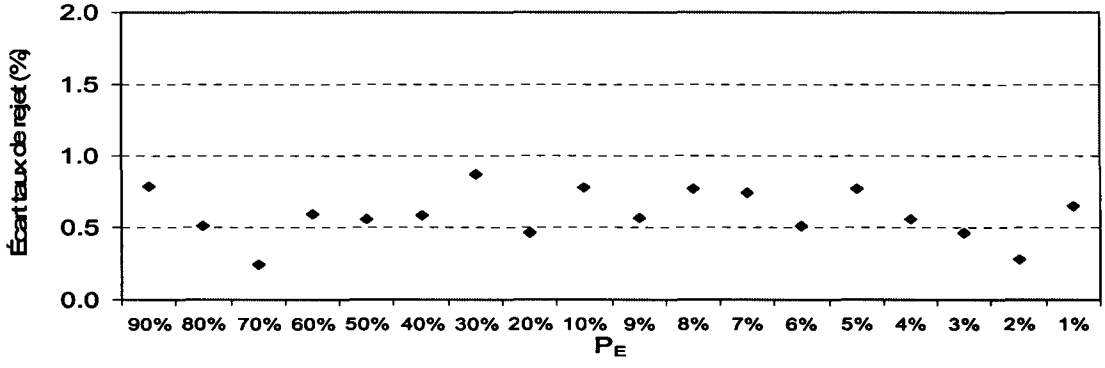
ANALYSE ERL : Paramètre P_E

Le graphique 51 démontre clairement que les taux de rejet obtenus, autant théorique que pratique, ne s'approchent du taux désiré que pour un P_E élevé. En fait, pour un P_R de 30%, le P_E optimal semble se situer aux alentours de 50%.

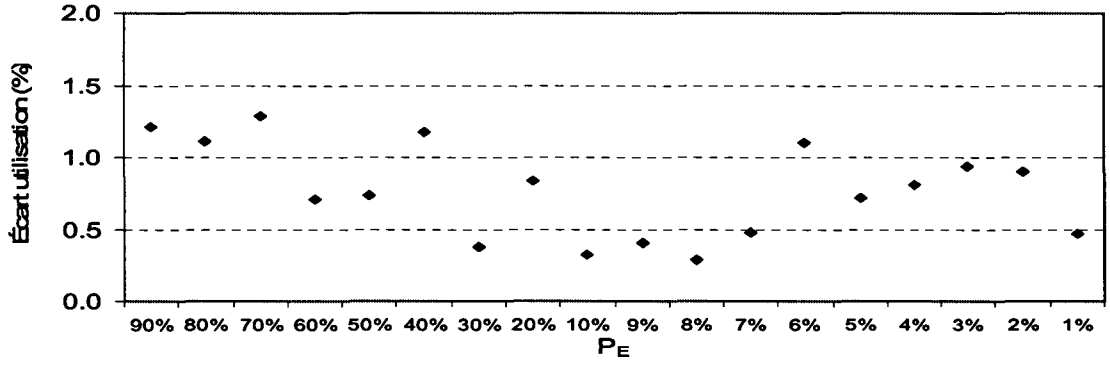
Lorsque P_E est grand, il est normal que le taux de rejet moyen obtenu puisse varier. En effet, lors de chaque évaluation, si l'algorithme ne modifie pas la réservation lorsque le taux de rejet calculé se situe entre 15% et 45% (30% ± 50%), obtenir un taux moyen final situé entre 27,5% à 32,5% est correct. Cependant, lors de chaque évaluation, si l'algorithme ne modifie pas la réservation uniquement lorsque le taux de rejet calculé se situe entre 28,5% et 31,5% (30% ± 5%), il serait normal d'obtenir un taux de rejet moyen final plus près du taux désiré. Considérant l'allure de la courbe (graphique 51), l'effet que devrait avoir le paramètre P_E sur le comportement de l'algorithme, taux moyen final plus près du taux désiré, serait correct si P_R était de 27%.

Pour le nombre de modifications (graphique 53), lorsque P_E est petit, le nombre de modifications est plus grand. Comme remarqué lors de l'analyse du paramètre T, plus l'intervalle de taux de rejet accepté lors de chaque évaluation est petit, plus il y a de chance que le taux de rejet calculé soit inférieur ou supérieur à cet intervalle, ce qui occasionne une modification de la bande passante. Le nombre de modifications est pratiquement proportionnel à l'écart P_E.

Le paramètre P_E ne semble pas avoir d'effet sur l'écart entre le taux théorique et pratique. L'écart entre les taux de rejet est relativement constant à 0,6% (graphique 54). L'écart entre les taux d'utilisation est relativement constant à 0,6% (graphique 55).



Graphique 54 Écart entre les taux de rejet théorique et pratique

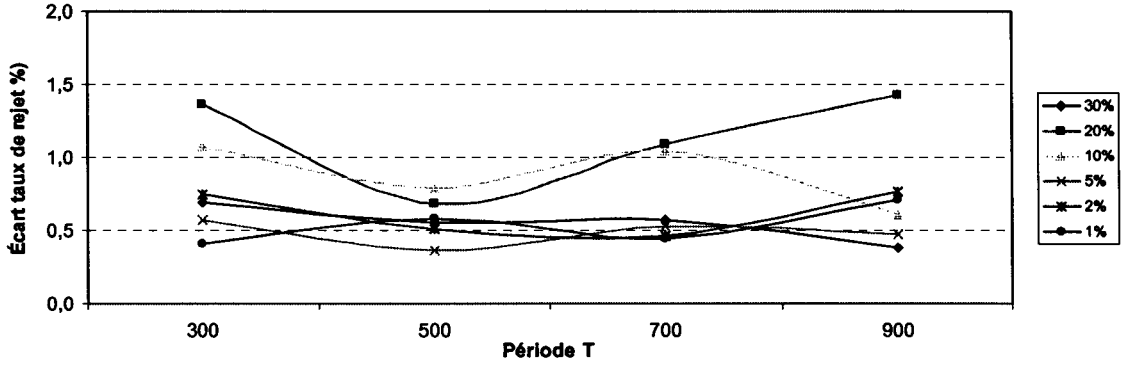


Graphique 55 Écart entre les taux d'utilisation théorique et pratique

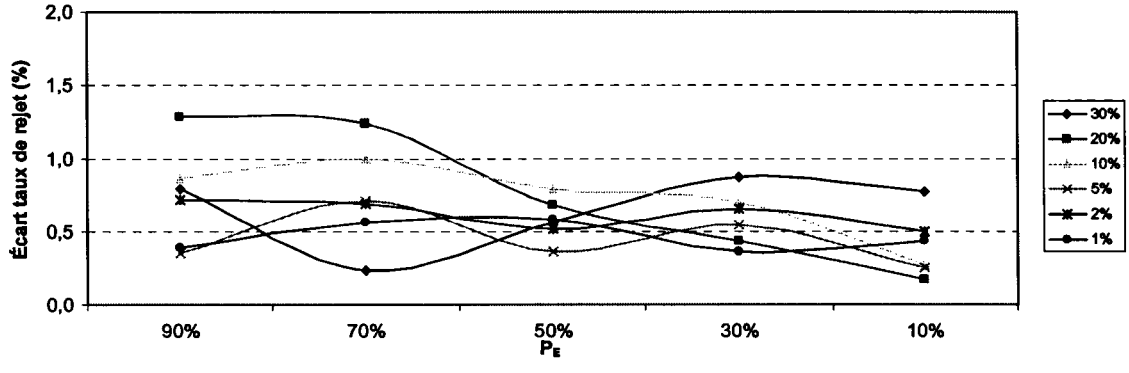
6.7.4 Conclusion sur le rôle des paramètres de ERL

Les différentes analyses réalisées sur les paramètres T et P_E nous démontrent les résultats suivants. Le paramètre T est optimal lorsqu'il vaut entre 500 et 700 secondes. Dans ce cas, le taux de rejet obtenu s'approche du taux désiré et les écarts entre taux théoriques et pratiques sont minimums. Pour le paramètre P_E, il faut faire attention à la conclusion. Lorsque P_E est petit, il y a un grand nombre de modifications, ce qui est un comportement normal. Comme le démontre le graphique 51, la diminution de P_E fait en sorte que le taux de rejet obtenu converge vers un taux de rejet donné. Par contre, ce taux s'éloigne un peu de celui configuré. Ce comportement a été observé dès la première analyse qui nous a permis de constater que les taux de rejet obtenus diffèrent d'environ 1% du taux désiré.

Qu'en est-il de la performance de cet algorithme? Comme pour les autres algorithmes, la façon d'évaluer la performance d'un algorithme est de comparer les taux pratiques aux taux théoriques. Plus les taux pratiques s'approchent des taux théoriques, plus l'algorithme est performant. Les graphiques 56 et 57 nous démontre que peu importe la valeur de T, P_E et P_R, l'écart entre les taux de rejet théorique et pratique est toujours inférieur à 1,5%. La section suivante permettra de comparer l'ensemble des algorithmes afin d'en déterminer le plus efficace.



Graphique 56 Écart entre les taux de rejet théorique et pratique



Graphique 57 Écart entre les taux de rejet théorique et pratique

6.8 Comparaison des algorithmes

Le tableau IX reprend les résultats qui ont été utilisés à la section 6.6.2. Ce tableau intègre également les résultats obtenus à l'aide des algorithmes ABW et ERL. Dans ce

tableau, les taux théorique et pratique sont les moyennes des taux obtenus par l'ensemble des simulations associées à un algorithme. Le calcul de l'erreur relative, détaillée à la section 6.6.2, apporte une base de comparaison commune.

Tableau IX

Comparaison des algorithmes

	Théorique	Pratique	Erreur relative
ABW, taux de rejet	5,10%	9,92%	**0,945**
DBP, taux de rejet	5,64%	16,72%	**1,964**
DBP-S, taux de rejet	0,67%	1,25%	**0,856**
ERL, taux de rejet	12,64%	13,24%	**0,047**
ABW, utilisation	51,68%	47,63%	**0,079**
DBP, utilisation	60,52%	51,87%	**0,143**
DBP-S, utilisation	45,49%	44,80%	**0,015**
ERL, utilisation	67,66%	66,88%	**0,0115**

Le premier algorithme étudié, ABW, utilisait une réservation constante malgré la variabilité du taux de trafic durant la journée. Cette variabilité du taux de trafic a empêchée l'algorithme ABW d'obtenir des taux pratiques proches des taux théoriques. Cet algorithme simpliste permet d'obtenir un taux de rejet 0,945 fois supérieur au taux de rejet théorique et un taux d'utilisation 0,079 fois supérieur au taux d'utilisation théorique.

L'algorithme DBP, fut le premier algorithme à modifier la bande passante au besoin afin d'améliorer l'efficacité. Cependant, l'algorithme ne permit pas d'atteindre cet objectif. Les taux de rejet obtenus étaient en moyenne 1,964 fois supérieurs aux taux théoriques

permis et les taux d'utilisation pratiques étaient 0,143 fois supérieurs aux taux théoriques. C'est dire que cet algorithme était pire que le précédant car les taux pratiques obtenus étaient plus éloignés des taux théoriques que les taux pratiques obtenus en utilisant une bande passante constante. Il fut démontré par l'algorithme suivant, DBP simplifié, que le problème de l'algorithme DBP était l'utilisation d'un paramètre refusant de manière probabiliste un appel même lorsque la bande passante était suffisante.

L'algorithme DBP simplifié, qui fut proposé en se basant sur le constat établi suite à l'étude de l'algorithme DBP, supprime le paramètre de rejet aléatoire. Les résultats obtenus confirment que ce paramètre était superflu. Les résultats obtenus confirme également qu'en modifiant la réservation pour suivre l'évolution du taux de trafic moyen, il est possible d'obtenir des taux pratiques plus près des taux permis par la théorie. L'algorithme DBP simplifié obtient des taux de rejet 0,856 fois supérieurs aux taux théoriques permis. Ce qui est ((1,964) / (0,856)) = 2,3 fois mieux que l'algorithme DBP et ((0,945) / (0,856)) = 1,1 fois mieux qu'une réservation constante. L'algorithme DBP simplifié obtient des taux de d'utilisation 0,015 fois supérieurs aux taux d'utilisation théoriques permis. Ce qui est ((0,143) / (0,015)) = 9,5 fois mieux que l'algorithme DBP et ((0,079) / (0,015)) = 5,2 fois mieux qu'une réservation constante.

L'algorithme ERL avait pour objectifs de simplifier la configuration en utilisant un paramètre spécifiant le taux de rejet et d'améliorer l'efficacité en ajoutant une composante temporelle à l'algorithme. La simplification de configuration est implicite. Pour ce qui est de l'amélioration de l'efficacité, les résultats du tableau IX le prouve hors de tout doute. L'algorithme ERL obtient des taux de rejet 0,047 fois supérieurs aux taux théoriques permis. Ce qui est ((1,964) / (0,047)) = 20,5 fois mieux que l'algorithme DBP, ((0,856) / (0,047)) = 18,2 fois mieux que l'algorithme DBP simplifié et ((0,945) / (0,047)) = 20,1 fois mieux qu'une réservation constante. L'algorithme ERL obtient des taux d'utilisation 0,0115 fois supérieurs aux taux théoriques permis. Ce qui est ((0,143) /

(0,0115)) = 12,4 fois mieux que l'algorithme DBP, ((0,015) / (0,0115)) = 1,3 fois mieux que l'algorithme DBP simplifié et ((0,079) / (0,0115)) = 6,9 fois mieux qu'une réservation constante.

Finalement, pour ce qui est du nombre de modifications, celui-ci est très variable. Il dépend de l'algorithme, de la valeur des paramètres, du taux de rejet ou d'utilisation obtenu. Nous n'avons pas étudié les algorithmes de façon à optimiser ce taux, car les taux de rejet et d'utilisations nous semblaient plus importants. Les chiffres suivant doivent donc être considérés avec parcimonie car ils ne sont pas déduits à partir d'optimisation comme l'ont été les taux de rejet et d'utilisation. Les nombres de modifications servant à la comparaison ont été obtenus en faisant la moyenne de toutes les simulations qui ont été réalisées pour chaque algorithme. DBP : 121, DBP simplifié : 188, ERL : 150. Peu importe l'algorithme, le nombre de modifications est similaire. Pour en déduire quoi que ce soit, il aurait fallu optimiser les algorithmes en fonction du nombre de modification plutôt qu'optimiser en fonction d'obtenir des taux de rejet et d'utilisation pratiques s'approchant de la théorie.

CONCLUSION

Dans un contexte où la réservation de ressources ne peut être ni statique (*Diffserv*), ni dynamique (*Intserv*), ce projet de recherche a présenté une architecture utilisant un contrôle d'accès et une réservation de ressources semi-dynamique évoluant selon l'évolution du trafic moyen. Le but de ce projet de recherche était d'étudier différents algorithmes implantant la logique du contrôle d'accès et d'évaluation dynamique des ressources nécessaires. Après avoir optimisé les paramètres de chaque algorithme, ceux-ci sont comparés en se basant sur les taux de rejet et d'utilisation qu'ils obtiennent. Plus les taux qu'ils obtiennent s'approchent des taux permis par la théorie, plus ils sont considérés comme efficace.

Le premier algorithme étudié, ABW, ne faisait aucune évaluation des besoins. Il s'agissait d'une réservation qui demeurait statique malgré l'évolution du trafic moyen. Les résultats obtenus démontrèrent que la variabilité de ce trafic empêchait l'obtention de taux pratiques proche de ce qu'aurait permis la théorie en considérant la réservation effectuée. Ces résultats démontrèrent donc qu'un algorithme variant la bande passante selon l'évolution des besoins permettrait sûrement d'obtenir des taux pratiques plus près de la théorie.

Le premier algorithme modifiant la bande passante au besoin, DBP, utilisait plusieurs paramètres dont le comportement furent étudiés afin d'en optimiser la valeur. Il fut démontré que cet algorithme était moins efficace qu'une réservation statique. En effet, l'effet d'un paramètre sur l'algorithme faisait en sorte que les taux de rejet et d'utilisation obtenus étaient plus éloignés de la théorie que ce que permettait une réservation statique. Un nouvel algorithme, appelé DBP simplifié, supprimant ce paramètre superflu fut étudié. L'étude de l'algorithme DBP simplifié démontra que le rejet aléatoire d'un appel, lorsqu'il y a suffisamment de bande passante, affecte négativement l'efficacité d'un algorithme. L'étude de cet algorithme démontra

également qu'un algorithme utilisant des réservations variables est plus efficace qu'une réservation statique.

Finalement, un nouvel algorithme fut proposé. Celui-ci avait pour avantages d'utiliser un paramètre de configuration spécifiant le taux de rejet désiré et incluait une dimension temporelle qui n'existait pas dans les algorithmes DBP. Cet algorithme, ERL, a permis d'atteindre des taux pratiques beaucoup plus près des taux théoriques, ce qui démontrait encore plus efficacement qu'un algorithme utilisant une réservation variable en fonction des besoins, permet d'obtenir des taux pratiques presque qu'équivalent à ceux permis théoriquement.

BIBLIOGRAPHIE

RFC

- [RFC 791] DARPA, University of Southern California, "*Internet Protocol*", IETF, Septembre 1981.
- [RFC 2205] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "*Resource Reservation Protocol*", IETF, Septembre 1997
- [RFC 2215] J. Wroclawski, "*General Characterization Parameters for Integrated Service Network Element*", IETF, Septembre 1997.
- [RFC 2474] K. Nichols, S. Blake, F. Baker, D. Black, "*Definition of the Differentiated Service Field*", IETF, Décembre 1998.
- [RFC 2475] S. Blake, D. Black, M. Carlson, E.Davies, Z. Wang, W. Weiss, "*An architecture for Differentiated Service*", IETF, Décembre 1998.
- [RFC 2702] D. Awduche, J. Malcolm, J. Agogbua, J. McManus, "*Requirements for Traffic Engineering over MPLS*", IETF, Septembre 1999.
- [RFC 2748] J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, "*The COPS (Common Open Policy Service) Protocol*", IETF, Janvier 2000.
- [RFC 3031] E. Rosen, A. Viswanathan, R.Callon, "*Multiprotocol Label Switching Architecture*", IETF, Janvier 2001.
- [RFC 3036] L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, "*Label Distribution Protocol Specification*", IETF, Janvier 2001.
- [RFC 3107] Y. Rekhter, E. Rosen, "*Carrying Label Information in BGP-4*", IETF, Mai 2001.
- [RFC 3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "*RSVP-TE Extensions to RSVP for LSP Tunnels*", IETF, Décembre 2001.
- [RFC 3213] Auteurs multiples, "*Constraint-Based LSP Setup using LDP*", IETF, Janvier 2002
- [RFC 3261] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "*SIP : Session Initiation Protocol*", IETF, Juin 2002.

- [RFC 3294] K. Sundell, "*General Switch Management Protocol*", IETF, Juin 2002.
- [RFC 3661] B. Foster, "*Media Gateway Control Protocol*", IETF, Décembre 2003.

Réseaux de service

- [MSF1] Multiservice Switching Forum, "*System Architecture Implementation Agreement*", Mai 2000
- [MSF2] Multiservice Switching Forum, "System Architecture Implementation Agreement, release 2", Mai 2004
- [IPCC] International Packet Communication Consortium, "*Reference Architecture v1.2*", juin 2002
- [ITU] International Telecommunication Union, "*Intelligent Network*", série de recommandations Q.12xx, 1997-2002

Contrôle d'accès et architecture de VoIP avec QoS

- [Bo] R. Bo, B. Tremblay, M. Bennani, et M. Kadoch, "*Integrating Traffic Aggregation into SIP-based IP telephony over MPLS Network*", 2005
- [Salsano] S. Salsano, L. Veltri, "*QoS Control by means of COPS to support SIP based applications*", IEEE Networks, Avril 2002
- [DOSHI] B. Doshi, E. Hernandez-Valencia, K. Sriram, Y.T.Wang and O.C.Yue, "Protocols, Performance, and Controls for Voice over Wide Area Packet Networks," *Bell Labs Technical Journal*, Volume 3, Issue 4, 1998: 297-337
- [Houck] D. Houck and H. Uzunalioglu, "*VoIP: Link-Based Management in MPLS Networks*," MPLS World Congress (Paris, France, February, 2005).
- [Houck2] D. Houck and H. Uzunalioglu, "*An Architecture and Admission Control Algorithm for Multi-Priority Voice over IP (VoIP) Calls*" Proceedings of the 9th International Conference on Intelligence in Service Delivery Networks (Bordeaux, France October, 2004).

- [Lucent] Lucent Technologies, "*Call Admission Control for Quality of Service in VoIP Networks*", 2005
- [Zhang] C. Zhang, C.G. Guy, "*COPS Usage for QoS management between TE-SIP server and GWLSR over a SIP over MPLS network*", ITC-CSCC, 2004

Hyperlien

- [CiscoCM] http://www.cisco.com/warp/public/cc/pd/nemnsw/callmn/index.shtml