Modélisation et étude de comportement d'une ligne de communication TCP/IP

Melha Bitam

Université Joseph Fourier – GRENOBLE 1

N°. attribué par la bibliothèque

|_|_|_|_|_|_|_|_|_|_|

THESE

Pour obtenir le grade de

DOCTEUR DE L'UJF

Spécialité « **AUTOMATIQUE – PRODUCTIQUE** »

préparée au Laboratoire d'Automatique de Grenoble

dans le cadre de l'École Doctorale « **Électronique, Électrotechnique, Automatique, Télécommunication et Signal** »

Présentée et soutenue publiquement
par

Melha BITAM

Le 15 juin 2005

Titre :

MODÉLISATION ET ÉTUDE DE COMPORTEMENT D'UNE LIGNE DE COMMUNICATION TCP/IP

Directeur de thèse:
M. Hassane Alla (Laboratoire d'Automatique de Grenoble)

JURY :

Rapporteur	Mme. Isabel DEMONGODIN	Professeur à l'Université d'Angers
Rapporteur	M. Kamel BARKAOUI	Professeur au CNAM, Paris
Examinateur	M. René DAVID	Directeur de Recherche Émérite au CNRS
Examinateur	M. Jean-Jacques LESAGE	Professeur à l'ENS Cachan
Examinateur	M. Antonio FAVELA	Professeur à l'ITESM, Monterrey.
Examinateur	M. Hassane ALLA	Professeur à l'Université Joseph Fourier

"Quand je regarde en arrière"

Quand je regarde en arrière, je n'ai nul regret, je n'aurai pas voulu vivre autrement... De toutes façons, un fantasme n'est jamais que cela. Je ne me dis pas : J'aurais voulu être un citoyen d'Athènes au temps de Périclès, ni un citoyen de Grenade sous les Abencérages, ni un bourgeois de la Vienne des valses. Je suis né dans un canton écarté de haute montagne, d'une vieille race qui, depuis des millénaires n'a pas cessé d'être là, avec les uns, avec les autres... qui, sous le soleil ou la neige, à travers les sables garamantes ou les vieilles cités du Tell, a déroulé sa saga, ses épreuves et ses fastes, qui a contribué dans l'histoire, de diverses façons, à rendre plus humaine la vie des hommes.

Les tenants d'un chauvinisme souffreteux peuvent aller déplorant la trop grande ouverture de l'éventail: Hannibal a conçu sa stratégie en punique; c'est en latin qu'Augustin a dit la cité de Dieu, en arabe qu'Ibn Khaldoun a exposé les lois des révolutions des hommes. Personnellement, il me plait de constater dès le début de l'histoire cette ample faculté d'accueil. Car il se peut que les ghettos sécurisent, mais qu'ils stérilisent c'est sûr.

C'est par là que je voudrais finir. Ceux qui, pour quitter la scène, attendent toujours d'avoir récité la dernière réplique à mon avis se trompent: il n'y a jamais de dernière réplique - ou alors chaque réplique est la dernière - on peut arrêter la noria à peu près à n'importe quel godet, le bal à n'importe quelle figure de la danse. Le nombre de jours qu'il me reste à vivre, Dieu seul le sait. Mais quelque soit le point de la course où le terme m'atteindra, je partirai avec la certitude chevillée que quelque soient les obstacles que l'histoire lui apportera, c'est dans le sens de sa libération que mon peuple - et avec lui les autres - ira. L'ignorance, les préjugés, l'inculture peuvent un instant entraver ce libre mouvement, mais il est sûr que le jour inévitablement viendra où l'on distinguera la vérité de ses faux semblants.

Tout le reste est littérature.

Mouloud Mammeri - écrivain algérien.

à mes parents pour leur amour et leur soutien...

Remerciements

Je vais commencer par remercier M. Hassane ALLA qui a travaillé avec moi pendant ces années de thèse et qui m'a aidé par ses connaissances scientifiques et son expérience à mener à bien ce travail de recherche.

Je tiens à remercier M. René David, chercheur émérite au CNRS, qui m'a fait l'honneur d'accepter de présider mon jury de thèse. J'ai tout particulièrement apprécié son esprit d'analyse et ses remarques constructives qui m'ont permis d'apporter un plus aussi bien à la qualité de mon exposé qu'à celui de mon rapport de thèse.

Mes remerciements vont également à M. Kamel BARKAOUI, professeur au CNAM de Paris qui a accepté d'être rapporteur de mon travail de thèse. Ses remarques perspicaces ainsi que ses qualités humaines ont rendu agréable cette dernière phase de lecture et de corrections. Je remercie aussi Mme. Isabel DEMONGODIN, professeure à l'Université d'Angers, qui a accepté d'être rapporteur. Elle m'a fait l'honneur de lire et de juger mon travail.

Je tiens à remercier également M. Jean-Jacques LESAGE, professeur à l'ENS de Cachan et M. Antonio FAVELA, professeur à l'ITESM de Monterrey pour avoir fait partie du jury de soutenance. Tout deux m'ont fait part de remarques pertinentes et constructives.

Mes remerciements s'adressent également aux membres des équipes CS2, PROSED et aux membres du projet NECS. Tous font preuve de sympathie et d'esprit scientifique. Leurs remarques et conseils tout au long des différentes réunions et exposés ont également contribué à la construction progressive de ce travail de recherche.

Je remercie l'ancien et le nouveau directeur M. Luc DUGARD et M. Alain BARRAUD pour m'avoir accueilli au sein du laboratoire. Je remercie également l'équipe technique et administrative pour tous les efforts fournis pour toujours nous faciliter notre travail ainsi que pour la bonne ambiance qu'ils aident à faire régner.

Un mot aussi pour toutes mes amies et pour tous mes amis qui m'ont soutenus par leur présence et leur sympathie. Ce travail s'est fait aussi grâce à elles et à eux. Je les en remercie.

Enfin et surtout, mes remerciements s'adressent à mes parents et à ma famille. Je n'aurais eu ni les moyens ni la force d'accomplir ce travail sans eux. Malgré la distance, nous avons été très proche toutes ces années. Leur soutien et leur amour m'ont permit et me permettront de toujours aller de l'avant et de me surpasser. Encore merci.

Table des matières

Table des figures

Introduction

Les réseaux de communication offrent à ce jour un ensemble de services très diversifié aussi bien pour les industriels, les banques, la recherche, que pour des particuliers et récemment dans la médecine et la commande à distance. En effet, les efforts fournis ces dernières décennies dans ce domaine permettent d'atteindre des objectifs très différents et de plus en plus ambitieux. Cet essor grandissant fait que les réseaux et notamment l'Internet est devenu un outil de travail quotidien à de nombreuses personnes. A cette forte utilisation des réseaux viennent s'ajouter des contraintes de performances (comme le respect des débits en sortie et de la bande passante), de sécurité (comme la confidentialité et l'authenticité) et de qualité de service (comme la garantie des temps d'attente des paquets) que les fournisseurs de service se doivent de respecter. L'étude des performances des réseaux ainsi que leur modélisation exacte est donc un pas important pour la connaissance des influences des différents liens des lignes de transmission sur les données qui les traversent.

Ce travail de thèse s'insère dans le cadre du projet Network Controlled System (NECS[1], projet CNRS) qui traite de problèmes où le contrôle interagit avec la théorie de l'information. Un schéma typique de contrôle par réseaux est donné sur la figure 1.

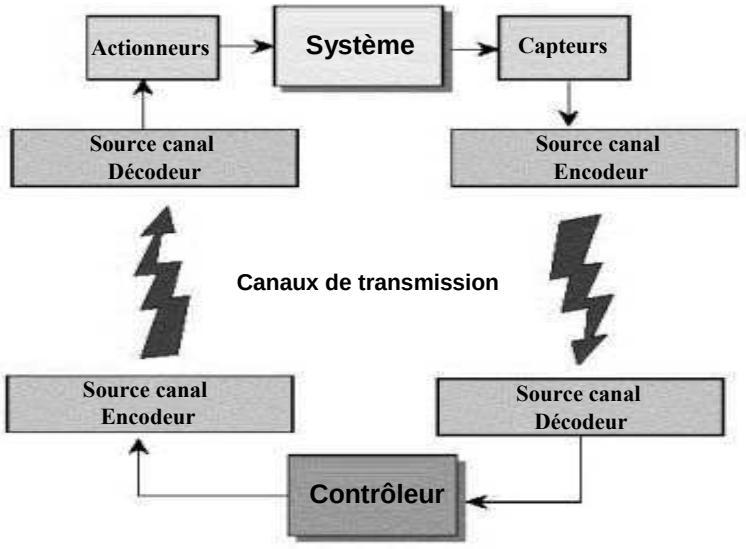


FIG. 1 – *Schéma type d'un contrôle commande par les réseaux.*

1. url: http://www-lag.ensieg.inpg.fr/canudas/necs.htm

Les travaux menés dans le cadre de ce projet visent à évaluer de nouveaux problèmes de contrôle posés entre autres par la considération de nouveaux composants technologiques (tels que les canaux de transmission de données) et de l'augmentation de la complexité des systèmes (mélange dans le même système de dynamiques continues et discrètes). Le projet comprend des travaux qui portent aussi bien sur le codage de données que sur les dynamiques des systèmes complexes ou encore sur la stabilité des réseaux. Pour notre part, nous considérons une ligne en boucle ouverte, i.e., d'un émetteur vers un récepteur et nous désirons étudier cette ligne en termes de délais, débits, pertes d'informations, etc. Pour pouvoir caractériser ces performances, nous passons l'établissement d'un modèle qui fait ressortir les dynamiques principales d'une ligne de communication.

Le plus grand réseau de communication existant est l'Internet. Celui-ci couvre toute la planète et il doit son nom au fait qu'il est le résultat de la connexion de plusieurs réseaux hétérogènes: c'est un réseau de réseaux. L'infrastructure d'Internet est constitué d'épines dorsales (backbone), de réseaux régionaux et nationaux et de réseaux de plus petites tailles appelés les réseaux locaux. C'est cette infrastructure de niveau mondial qui lui vaut sa force.
Un ensemble de protocoles ont été conçus pour gérer le dialogue entre les machines dans un réseau. On peut citer comme exemple de protocoles l'UDP, le HTTP, le FTP, TELNET, l'ARPANET etc. Ces protocoles agissent à des niveau différents et possèdent ainsi des fonctions différentes. Un protocole compatible avec la plupart des applications qui suit et autorise l'inter fonctionnement de n'importe quel serveur avec n'importe quel protocole de niveau inférieur est le TCP/IP ([Pos81]).

Nous nous sommes penchés, durant notre travail de thèse sur les réseaux TCP/IP car se sont les plus utilisés actuellement dans Internet. Ces réseaux sont dits fiables, i.e., ils n'admettent pas de pertes de données. La mise en oeuvre de cette qualité de service se fait par l'acquittement des données bien reçues. Si l'émetteur ne reçoit pas d'acquittement pour un message envoyé, il sait que ce dernier s'est perdu et le réémet. Cependant, avec l'expansion grandissante d'Internet ont commencé des séries de pertes de données qui causaient parfois une baisse soudaine de bande passante. C'est le problème de congestion de données qui est un des problèmes les plus souvent rencontrés dans les réseaux. Différents algorithmes de contrôle de congestion ont été conçus et sont implantés dans le TCP afin de garantir au mieux les performances en termes de délais de transmission et de respect de la bande passante. Ces algorithmes permettent d'estimer l'état du réseau et de réagir face à une congestion. Cependant, le problème des pertes de données dans les réseaux reste entier car les algorithmes sus-cités n'arrivent pas à suivre l'évolution aléatoire voir chaotique des données présentes sur une ligne de transmission et à éviter complètement les pertes.
Plusieurs travaux existent dans la littérature concernant l'amélioration des performances des réseaux et des algorithmes de contrôle de congestion. Nombre d'entre eux se basent sur les connexions *points-à-points* (i.e., les informations utiles au contrôle ne viennent que de l'émetteur et du récepteur) ou sur la modélisation des liens qui constituent une ligne de transmission (notamment les routeurs, méthodes appelées *router-based*). Parmi ces travaux, des modèles, continus et discrets, établis par les communautés automatique et réseaux de communication améliorent le moment de détection des pertes et le temps de réaction à celles-ci. Nous pouvons citer parmi ces travaux, les réseaux de files d'attente (Active Queue Management, AQM), l'al-

gorithme RED (Random Early Detection), ([BCC+98], [FF97], [FJ93], [LPD02]) le TCP Reno, le TCP Tahoe et le TCP SACK ([FF96], [LPD02], [MMFR96]). Des études de performances concernent aussi la stabilité des réseaux ([BB98]) notamment en établissant un modèle analytique des flux ([HC01], [LPD02]).

Alors qu'il est clairement recommandé de proposer des modèles complets qui puissent faire ressortir les différentes dynamiques d'une ligne de communication, nous n'avons pas trouvé de tels modèles dans nos lectures. Ces travaux nous ont d'abord montré que le problème de congestion reste un problème ouvert mais aussi qu'il serait intéressant de considérer, non plus, une connexion point à point émetteur-récepteur ou routeur unique, mais de considérer une *"ligne entière"* émetteur-routeur-récepteur. Les ressources de cette ligne (et notamment le routeur) sont partagées par d'autres émetteurs dont nous ne connaissons pas les débits à priori : c'est ce que nous appelons l' *"environnement extérieur"* ([BA03], [BA05b]).
Nous avons donc étudié les dynamiques et les protocoles d'une ligne TCP/IP soumise à l'environnement extérieur Internet et nous avons établi un modèle à base duquel toutes les dynamiques peuvent être observées à tous les niveaux d'une communication (charge des buffers, baisse des vitesses à l'entrée des routeurs, séparation des flux par les routeurs, priorités, stockage des données, retards, pertes, etc.). Ce modèle détaillé nous a aussi permis de visualiser directement l'effet des changements de paramètres de la ligne et des protocoles sur l'évolution de la transmission sur la ligne : Un large panel de simulations a donc été effectué afin de visualiser ces différentes performances. D'une part, le modèle établi permet de visualiser directement la quantité de pertes produites sur la ligne durant une transmission. D'autre part, les protocoles de communication, à partir des données envoyées et celles reçues, déduisent la quantité de donnée perdue. Nous disposons ainsi d'une comparaison pertes réelles - pertes vues par les protocoles qui peut servir d'évaluation des protocoles mêmes. Plus les deux valeurs des pertes sont proches, plus les protocoles sont performants. Enfin, et grâce aux résultats de simulations de notre modèle, nous avons pu observer les temps de boucle des données sur la ligne et arriver à une modélisation analytique de ces derniers ([BA05b], [BA05a], [BA05c]).

Les systèmes de communications comportent des événements continus aussi bien que discrets. Des flux arrivent au récepteur selon des dynamiques continues. Ces dynamiques sont modifiées ponctuellement par des commandes discrètes provenant des protocoles de contrôle. Ce comportement défini parfaitement celui d'un système hybride. Parmi les outils de modélisation des systèmes hybrides, il y a ceux qui représentent une extension des modèles continus, comme c'est le cas des bonds graph, ceux qui sont une extension des modèles discrets comme les réseaux de Petri hybrides (RdPH), et enfin les modèles mixtes qui combinent les deux parties discrète et continue comme c'est le cas des RdP mixtes et des statecharts mixtes.
Les bond graphs sont généralement utilisés dans les systèmes hydrauliques et électriques. Quand aux modèles mixtes, une de leur caractéristiques est leur généralité puisqu'ils ne font pas d'hypothèses sur le type de phénomènes à utiliser.
D'une part, nous nous intéressons à un domaine très particulier constitué par les réseaux de communication. Il n'est donc pas dans notre intérêt d'utiliser un formalisme trop général. D'autre part, nous désirons établir un modèle lisible et compréhensif qui mette en évidence les différentes parties d'un réseau. Pour cela, notre choix s'est porté sur les réseaux de Petri hybrides comme outils de modélisation des réseaux de communication [Bit04]. Il s'est avéré que cet outil

nous permet de modéliser les réseaux en tenant compte de toute la complexité des différentes dynamiques de tels systèmes (flux de message et protocoles de contrôle).

Ce rapport de thèse est organisé en 4 chapitres comme suit:
Le **premier chapitre** est constitué d'une bibliographie des réseaux de communication en terme d'architecture de réseaux, de définitions et de principes fondamentaux des réseaux. Cette partie permet de familiariser le lecteur autmaticien avec le monde complexe et diverse des réseaux.
Le **second chapitre** concerne les protocoles de contrôle au sien des réseaux. Cette partie, que l'on peut appeler logiciel des réseaux, introduit la problèmatique principale traitée dans notre travail, à savoir les protocoles de congestion et les performances dans les réseaux.
Le **troisième chapitre** concerne la modélisation d'une ligne de communication TCP/IP. L'outil utilisé étant les RdP hybrides, ceux-ci seront présentés avant d'entamer la description détaillée du modèle.
Dans le **quatrième chapitre** sont présentés les résultats de simulation. Ceux-ci permettent en même temps de valider le modèle et d'évaluer les performances du système. Une première idée de contrôle de congestion est aussi donnée en fin de ce chapitre.

Nous terminons ce travail par quelques conclusions qui récapitulent le travail réalisé. Des travaux de perspectives sont aussi proposés. Ils concernent la continuation de certains points qui n'ont pu être mis en oeuvre durant la thèse ainsi que d'autres travaux nouveaux qui peuvent constituer une bonne continuité de notre présent travail de recherche.

Chapitre 1

Les réseaux de communication

Les réseaux de communication constituent l'une des avancées les plus spectaculaires de ces dernières décennies. Rapidement, ces derniers ont touché des domaines très divers et sont entrés dans le "quotidien" de beaucoup de personnes.
Initialement étudiés et conçus par des chercheurs des communautés informatique et réseaux, ce domaine reste ouvert à des extensions toujours nouvelles et des chercheurs de la communauté automatique se sont récemment investis dans les réseaux pour résoudre des problèmes tels que la congestion ([LPD02], [PFTK98], [ARA03], [HC01]), la stabilisation des systèmes commandés par réseaux ([BB98]) ou encore l'évaluation de performances des réseaux par des modèles stochastiques [Roy05], [ARA03]. Pour notre part, nous nous intéressons au problème de la congestion dans Internet et les réseaux étant un domaine nouveau pour les automaticiens, nous pensons utile de reprendre les définitions et principes fondamentaux des réseaux de communication. Ce premier chapitre a donc pour but de présenter au lecteur automaticien le monde des réseaux et de lui faire découvrir toute la complexité de ses protocoles et la diversité de ses performances.

1.1 Objectifs des réseaux

Avant d'entrer dans les détails techniques de conception et d'architecture des réseaux de communication, revenons d'abord aux objectifs premiers de ces derniers.
Beaucoup d'organisations ont un nombre important d'ordinateurs parfois fort distants. Ce qui les amènent à utiliser les réseaux pour communiquer. De façon générale, les objectifs techniques des réseaux sont:

- **le partage des ressources**, c'est à dire rendre accessible à chaque membre du réseau les programmes, les données et les équipements indépendamment de sa localisation physique.

- **Le besoin d'une plus grande fiabilité** en disposant d'alternatives aux ressources employées. Par exemple, si une unité centrale est en panne, d'autres unités peuvent prendre en charge son travail. En effet, dans des applications telles que le trafic aériens ou les applications militaires, il est très important de pouvoir garantir le bon fonctionnement des

systèmes y compris en cas de problèmes de matériels.

– **la réduction des coûts**. A titre indicatif, les gros ordinateurs sont environ dix fois plus rapides que les petits, par contre, ils coûtent des milliers de fois plus chers. ce rapport qualité/prix a amené les concepteurs de systèmes à utiliser un gros ordinateur, appelé *client*, comme base de leur système et à partager ensuite les données sur de plus petits ordinateurs que l'on appelle *serveurs de fichiers*, d'où les modèles connus sous le nom de *clients-serveurs*.

– **média de communication** Un autre objectif moins technique cette fois mais tout aussi important est que les réseaux d'ordinateurs constituent un puissant **média de communication**. En effet, à partir des années 90, les réseaux ont commencé à offrir des services destinés aux particuliers. Ce sont l'accès à l'information, la communication entre individus (comme le courier électronique ou la vidéoconférence) et les jeux interactifs.

La diversité des services rendus par les réseaux a pour conséquence leur utilisation dans une gamme très diversifiée d'application. Celles-ci vont des applications militaires, contrôle de trafic aérien, commande de centrales nucléaires, aux applications en médecine et en téléopération, en passant par les applications bancaires, les jeux interactifs et la vidéoconférence.

1.2 Caractéristiques physiques des réseaux

On peut avoir recours aux réseaux pour contrôler un robot qui se trouve dans la même pièce que nous ou pour consulter la banque de données d'une bibliothèque se trouvant de l'autre côté de l'atlantique. De ce fait, la taille des réseaux peut varier de quelques mètres à quelques dizaines de milliers de kilomètres. Les techniques de transmission qui leur sont appliquées changent elles aussi complètement. De façon générale, la taille des réseaux et la technique de transmission utilisée représentent les deux caractéristiques principales des réseaux.

1.2.1 Les techniques de transmission

Les techniques globales de transmission sont au nombre de deux et sont: la diffusion et le point à point.
Dans les **réseaux à diffusion**, les ordinateurs se partagent un même support de transmission. Les données envoyées par chaque machine sont présentes sur tout le canal et sont donc reçues par tous les ordinateurs. Un champ d'adresse au niveau de la donnée permet à la machine destinataire de réellement prendre en compte les messages qui lui sont destinés. Des algorithmes d'allocation de canal permettent d'éviter la collision des différents flux envoyés sur un même canal. Les systèmes à diffusion permettent l'envoi de données vers un groupe de destinataires grâce à des adresses spéciales; c'est ce qui est appelé la **diffusion générale** (*broadcasting*). Cette technique permet également la diffusion de messages à un sous ensemble de destinataires. C'est ce qui est appelé la **diffusion restreinte** ou **multipoint** (*mulicast*). Dans de telles configurations, la rupture du support provoque l'arrêt du réseau, par contre la panne d'un élément ne provoque pas (en général) la panne globale du réseau.

Par opposition, **les réseaux point à point** sont formés d'une multitude de connexions car les machines sont reliées deux à deux. Lorsque 2 éléments non directement reliés entre eux veulent communiquer, ils le font par l'intermédiaire des autres noeuds du réseaux. Ainsi, pour qu'un message aille d'une source vers un destinataire, il peut transiter par plusieurs ordinateurs intermédiaires. Plusieurs routes ayant des longueurs différentes sont alors possibles . Les algorithmes de routages jouent ici un rôle important pour déterminer le chemin emprunté par les données.

1.2.2 Taille et topologies des réseaux

La deuxième caractéristique d'un réseau est sa taille. Celle-ci peut aller de quelques mètres à quelques dizaines de milliers de kilomètres. Un réseau de petite ou moyenne taille qui couvre un immeuble ou une entreprise est dit *réseau local* ou LAN (Local area Network). Les LAN utilisent souvent pour leur transmission de simples câbles auxquels sont reliées toutes les machines. Pour cela, un seul terminal est autorisé à émettre en même temps afin d'éviter que différentes données puissent être présentes dans le canal et entrer en collision. Des mécanismes d'arbitrage sont donc nécessaires. Ils ont pout but d'éviter les collisions sur les canaux à accès multiples. A titre d'exemple, sur les réseaux IEEE 802.3[1] appelés communément Ethernet, les ordinateurs peuvent transmettre lorsqu'ils le désirent, si deux paquets entrent en collision, chaque ordinateur attend un temps aléatoire avant de réemettre.

Les réseaux de grandes tailles sont constitués d'ordinateurs connectés à un ensemble de lignes de transmission et de commutateurs. Ces derniers forment ce qu'on appelle un sous-réseau. Les lignes de transmission sont les canaux qui véhiculent l'information (bits) d'une machine à une autre. Les commutateurs, souvent appelés **routeurs**, sont des ordinateurs spécialisés qui reçoivent des données sur une ligne d'entrée et servent à leur trouver une ligne de sortie.
Dans ce type de réseau, tous les ordinateurs peuvent émettre simultanément, il peut arriver alors que les ressources partagées du réseau soient saturées. C'est ce qu'on appelle *la congestion de données*. Pour pallier à cet inconvénient, plusieurs algorithmes d'évitement de congestion existent et sont implantés au niveau des émetteurs et des récepteurs.

Le plus grand réseau existant est l'Internet[2] qui couvre toute la planète. Internet doit son nom au fait qu'il est le résultat de la connexion de plusieurs réseaux hétérogènes. C'est un réseau de réseaux.
Il existe une hiérarchie dans l'infrastructure d'Internet. Au niveau le plus élevé se trouvent *les épines dorsales* (ou backbone). Les backbones sont constitués d'artères de communication à très haut débits et de routeurs très rapides. Au niveau hiérarchique plus bas, se trouvent les réseaux régionaux et nationaux. Ceux-ci peuvent être de grands opérateurs nationaux (comme Transpac ou Numéris par exemple). Enfin, au niveau le plus bas se trouvent les réseaux LAN (figure 1.1).

1. Norme de IEEE pour un type de réseau local

2. Il est souvent adopté d'écrire la première lettre du réseau mondial Internet en majuscule pour la différencier de la couche réseau du modèle TCP/IP (voir plus loin en section 1.3.3) également appelée internet. Cette dernière sera donc écrite en lettre minuscules.

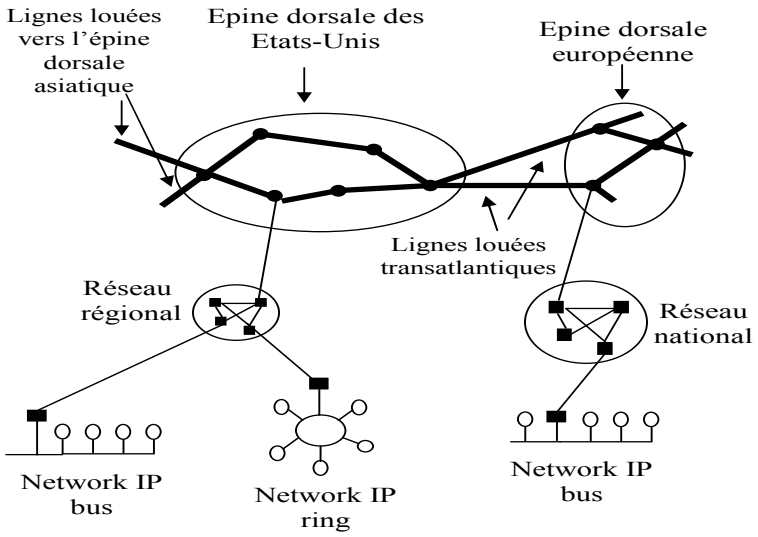


FIG. 1.1 – *Structure générale simplifiée du réseau Internet*

Toute la force d'Internet réside dans cette interconnection de systèmes autonomes (appelés aussi sous-réseaux autonomes) dans une vaste infrastructure de niveau mondial.

1.3 Logiciel de réseau

Avec la gamme d'offres de services proposée par Internet et vu les attentes grandissantes des utilisateurs des réseaux, il est aisé d'imaginer qu'il est très complexe de mettre en place l'architecture d'un réseau et encore plus de mettre en oeuvre des protocoles qui doivent gérer d'une part l'hétérogénéité des réseaux interconnectés et d'autre part les différentes qualités de service requises pour chaque type de service. De ce fait, et pour réduire la complexité des réseaux, la plupart d'entre eux sont organisés en **couches** superposées et indépendantes. Chaque couche possède sa propre fonctionnalité et peut offrir 2 types de services à sa couche supérieure: un service en mode connexion et un service en mode sans connexion.

1.3.1 Services en mode connexion et en mode sans connexion

Les couches peuvent offrir deux types de services différents aux couches qui leur sont supérieures. Le **mode connexion** et le **mode sans connexion**. Le mode connexion est comparable au téléphone. On doit d'abord établir la connexion, l'exploiter et enfin l'interrompre. Le mode sans connexion serait comparable plutôt au courier postal. Lorsqu'on désire envoyer une lettre, on n'a pas à avertir le destinataire au préalable. On mentionne l'adresse sur la lettre et on est sûr que celle ci arrivera à destination. Chaque message (ou lettre) est acheminée indépendamment des autres messages. Il en est de même pour les réseaux où l'émetteur envoie ses données sans vérifier à l'avance si l'équipement à atteindre ainsi que les noeuds intermédiaires éventuels sont bien actifs. C'est alors aux équipements gérant le réseau d'acheminer les messages étape par étape et en assurant éventuellement sa temporisation jusqu'à ce que le destinataire soit actif. La

particularité de ce mode est que les messages peuvent prendre chacun un chemin différent si bien que les premiers envoyés peuvent être retardés et arriver en dernier. Chose qui ne peut se produire en mode connexion où la transmission se passe comme si l'on disposait d'un "tuyau" où l'on envoi une suite de bits: les premiers envoyés sont toujours les premiers arrivés.
Dans le contexte des opérations internes des sous-réseaux, les messages indépendants du mode sans connexion sont appelés *datagrammes*. On parle alors de sous-réseau datagramme. Par contre, une connexion sera appelée *circuit virtuel*.
A titre d'exemple, les réseaux IP sont des réseaux datagrammes. Un exemple de réseau orienté connexion est le réseau ATM (Asynchronous Transfer Mode).

Le service offert (avec ou sans connexion) et la structure du sous-réseau (datagramme ou circuit virtuel) peuvent être différents. Le tableau 1.2 montre des exemples des 4 combinaisons possibles.

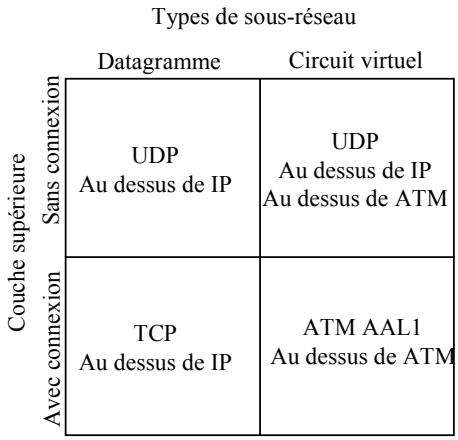


FIG. 1.2 – *Services réseaux*

1.3.2 Service fiable et non fiable

Un service offert par une couche peut être **fiable** ou **non fiable**. On dit d'un service qu'il est fiable lorsqu'on n'a pas de pertes de données. La mise en oeuvre de cette qualité de service se fait par l'acquittement des données bien reçues. Le récepteur envoie un acquittement pour chaque donnée arrivée, ainsi, si l'émetteur ne reçoit pas d'acquittements pour un message envoyé, il sait que ce dernier s'est perdu et le réemet. Un exemple d'utilisation d'un service fiable est le transfert de fichiers où l'on veut être sûr de l'arrivée correcte de tous les bits. Le principal avantage de ce service est qu'il garantit l'arrivée de tout les messages envoyés.
À l'opposé, un service non fiable est un service pour lequel il n'est pas primordial de ne jamais avoir de pertes. Ainsi, les messages sont envoyés et sont considérés comme bien reçus. Le principal avantage de ce service est sa rapidité vu que l'on ne se préoccupe pas de réemettre les données non reçues. Ce service trouve ses applications dans des domaines tels que la projection de film vidéo où il est préférable que quelques pixels soient erronés plutôt que le film se déroule en saccade sous prétexte de corriger des erreurs de transmissions. Il en est de même pour la transmission du son.

1.3.3 Couches et protocoles

Les réseaux sont organisés en couches superposées où chaque couche possède une fonction bien définie et fonctionne selon un protocole propre à elle. Il en est ainsi pour les deux modèles de références ISO et TCP/IP. Le nombre de couches peut varier d'un réseau à un autre mais l'objectif cité ci-dessus d'une structure en couche reste commune pour tous les modèles. Les réseaux OSI sont structurés en une série de 7 couches. Le modèle TCP/IP ne comprend que les quatres premières couches de l'ISO en plus de la couche application car il s'est avéré après l'expérience de l'ISO, que ce sont les couches le plus utilisées dans une transmission. Chaque couche est constituée d'éléments matériels et logiciels et offre un service à la couche située immédiatement au dessous d'elle en lui épargnant les détails d'implémentations nécessaires. Il est à noter que le modèle ISO a été élaboré avant l'apparition de l'Internet. C'est un modèle général qui devait donc s'appliquer à plusieurs sorte de réseaux. Le TCP/IP quand à lui, a été conçu spécialement pour gérer la communication entre réseaux hétérogènes interconnectés comme c'est le cas pour Internet.

Comme illustré dans la figure 1.3, chaque couche d'une machine émettrice va "discuter" avec la couche de même niveau de la machine destinataire. C'est cet ensemble de règles de langage qui font que deux couches de même niveau peuvent communiquer qui s'appelle **protocole** (figure 1.3). Cependant, le dialogue entre une paire de couches de niveau n ne se fait pas directement d'une couche à une autre mais la couche émettrice (soit la couche n) transfert ses informations à la couche qui lui est immédiatement inférieure (soit la couche $n-1$) qui le transfert à son tour à la couche suivante jusqu'à arriver au support physique. Le message est alors véhiculé sous forme de suite de bits puis va remonter les couches de la machine destinataire jusqu'à arriver à la couche n. À l'emission, chaque couche ajoute un en-tête au message. Cet en-tête portera des informations nécessaires à la même couche de la machine réceptrice. Ces informations peuvent être relatives à la taille du message émis, à sa durée de vie ou encore aux adresses sources et destinations. À la réception, chaque couche lit l'en-tête qui lui est réservé, traite l'information, se débarrasse de cet en-tête puis transmet le message à la couche supérieure. D'où l'importance que chaque paire de couches adjacentes conviennent d'un même protocole de transmission. Toutes ces opérations étant évidemment transparentes à l'utilisateur.

Par ailleurs, si une couche possède une limitation en taille de messages par rapport à la couche qui lui est immédiatement supérieure (par exemple la couche réseau par rapport à la couche transport), ce message devra être *fragmenté* (ou segmenté, la terminologie étant différente selon les réseaux) puis acheminé comme un paquet indépendant.

1.3.3.1 Architecture des protocoles TCP/IP

Comme dit précédemment, les logiciels TCP/IP sont structurés en 4 couches comme illustré sur la figure 1.4. La couche accès réseau, la couche internet (ou réseau selon l'appellation OSI), la couche transport et la couche application. La couche accès réseau spécifie la forme d'envoi des données. Elle prend en charge des notions telles que le format des données ou la conversion des données (analogique/numérique). La couche internet gère la circulation des paquets à travers le réseau en assurant leur routage. Le protocole qui y est utilisé est le IP (Internet

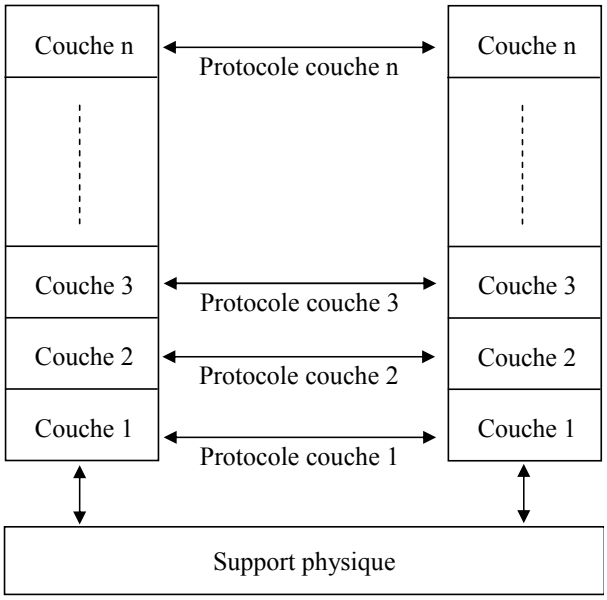


FIG. 1.3 – *Protocole entre deux couches de même niveau*

Protocol). La couche transport quand à elle permet à deux machines distantes de communiquer (communication de bout en bout). Elle régule le flux et permet un transport fiable (pour la cas du protocole TCP) ou non fiable (pour la cas du protocole UDP). Enfin, la couche application qui est celle des programmes utilisateurs.

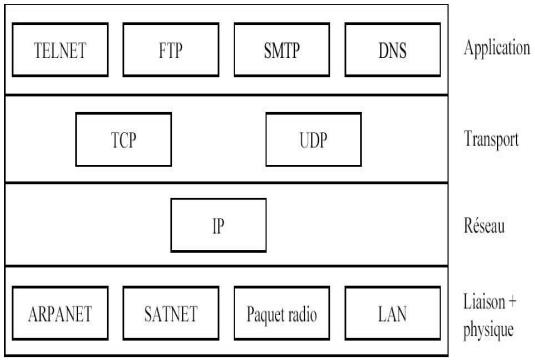


FIG. 1.4 – *Protocoles utilisés dans le modèle TCP/IP*

Le protocole IP présente un service non fiable. En plus de sa rapidité, il présente l'avantage d'avoir été conçu spécialement pour permettre à plusieurs réseaux hétérogènes de coopérer (l'interréseautage). Le protocole TCP, lui, fonctionne en mode connexion et présente un service fiable. Monté sur le IP, il permet de traiter de bout en bout des données de manière fiable sur un ensemble de réseaux non fiable. L'UDP fonctionne en mode non connecté. Monté sur le IP, il se contente d'encapsuler des datagrammes et de les envoyer sans établir de connexion. On peut dire que l'avantage de TCP est la fiabilité car il soutient la *livraison garantie*, tandis que l'avantage de l'UDP est la rapidité. TCP à été formellement défini dans les RFC 793, 1122 et

323. Les spécifications de l'UDP sont dans le RFC 768. De plus amples détails sont donnés sur les couches des modèles OSI ainsi que ceux du TCP/IP sont donnés en annexe A.

1.3.3.2 Encapsulation des données par une pile de protocoles: exemple des protocoles TCP/IP

Comme pour le modèle OSI, pour que des données passent d'une machine A à une autre machine B, elles doivent descendre l'ensemble des couches, chacune lui joignant un en-tête. Les données sont transmises par le support physique puis remontent l'ensemble des couches. Ce processus d'ajout d'en-tête est nommé encapsulation et est illustré sur la figure 1.5.

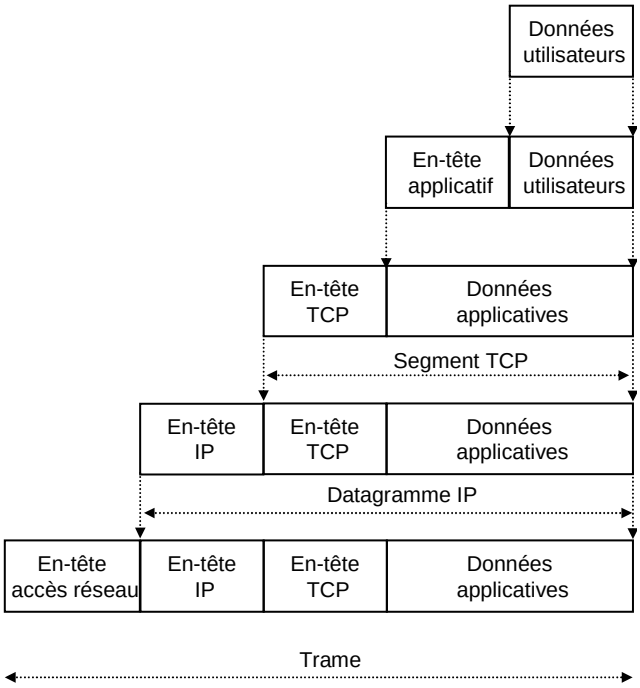


FIG. 1.5 – *Encapsulation des données par la pile des protocoles TCP/IP*

A chaque niveau, le paquet de données change d'aspect, car on lui ajoute un en-tête, ainsi l'appellation de cette donnée change suivant les couches :

- Le paquet de données est appelé **message** au niveau de la couche application.
- Le message est ensuite encapsulé sous forme de **segment** dans la couche transport.
- Le segment une fois encapsulé dans la couche Internet prend le nom de **datagramme** (ou paquet).

– Enfin, on parle de **trame** au niveau de la couche accès réseau.

Exemple d'en-tête: l'en-tête du segment TCP La figure 1.6 illustre l'en-tête d'un segment TCP. Celui-ci possède une longueur fixe de 20 octets, il peut être suivis par des options. Chaque champ d'en-tête apporte sa propre information. Á titre d'exemple, les champs *port source* et *port destination* identifient les extrémités locales de la connexion. Le champ *numéro de séquence* spécifie le numéro de séquence du premier octet de données envoyées. En effet, au sein d'un flot TCP, chaque octet de donnée est numéroté. Le champ *numéro d'accusé de réception* spécifie le numéro du prochain octet attendu. Le champ *longueur d'en-tête* indique combien de mots de 32 bits contient l'en-tête. La longueur du champs *options* étant variable, ce champ est important pour savoir le point de départ des données au sein du segment. Le contrôle de flux dans le TCP est réalisé au moyen de *fenêtre d'anticipation à tailles variables* (voir chapitre 2). Cette fenêtre représente la taille de données maximales que l'on peut envoyer après un octet bien reçu. Le champ *Taille de fenêtre* indique cette taille maximale. Cette taille étant exprimée sur 16 bits, la taille de fenêtre maximum ne peut excéder 64 Ko.

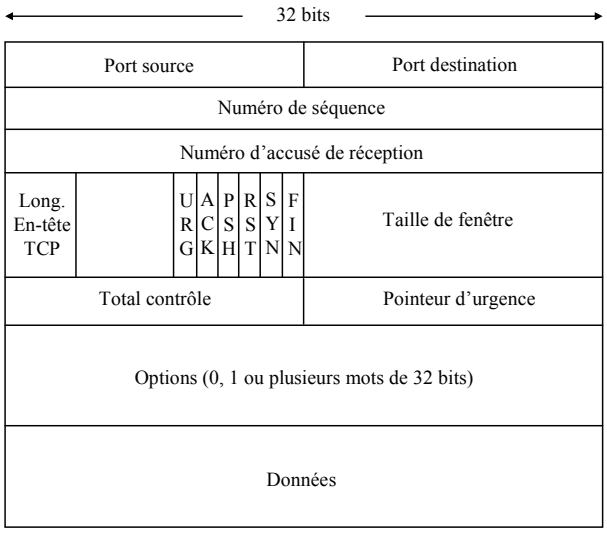


FIG. 1.6 – *L'en-tête TCP*

1.3.4 Quelques notions des indices de performances des réseaux

Après avoir passé en revue l'architecture des réseaux en terme de superposition des couches, de protocoles de communication et de liaison par les routeurs, nous allons nous intéresser maintenant à quelques indices de performances comme les débits d'une ligne, les délais de transmission ou encore les pertes.
Les débits, exprimés en bits/sec, représentent la vitesse à laquelle le récepteur reçoit les données. Généralement, toutes les applications requièrent un débit minimal disponible en permanence. Si, par exemple, une application de téléphonie par Internet effectue un codage de la voix à

32Kbits/s, elle soit être en mesure d'envoyer des données et de s'assurer de leur bonne arrivée à ce même débit. Si le débit requis n'est pas disponible, les applications doivent être capables de coder à des rythmes différents et disposer de suffisamment de ressources pour maintenir la vitesse requise.
Les paquets partent d'un serveur (source) pour rejoindre un autre serveur (destination). Ils peuvent, pour cela, traverser une série de routeurs. Pendant leur transmission, les données sont donc affectées de plusieurs types de retards possibles, les plus importants d'entre eux sont les temps de transmission, de propagation et les temps d'attente. Le temps de transmission correspond au temps requis pour transmettre tous les bits d'un paquet sur un canal de transmission. Par exemple, si la capacité d'une liaison (entre deux noeuds A et B) est de R (bits/s) et la longueur d'un paquet est L (en bits), alors le temps de transmission sera L/R. Ce délai est à différencier du délai de propagation qui est le temps nécessaire au trajet entre l'origine (noeud A) et la destination (noeud B). Les données se propagent à la vitesse de propagation d'une liaison qui est déterminée par son support physique, elle varie entre $2 \cdot 10^8$ et $3 \cdot 10^8$, ce qui équivaut à peu prés à la vitesse de la lumière. Le temps de propagation correspond à la distance séparant deux noeuds (d) divisée par la vitesse de propagation (s). Lorsqu'on a affaire à des réseaux longues distances, ce délai peut atteindre l'ordre de la milliseconde. Le temps d'attente quant à lui dépend de la file d'attente présente dans un routeur. En effet, les données emmagasinées dans un routeur en sortent les unes à la suite des autres (généralement selon la gestion FIFO "First In First Out"). La dernière donnée entrée dans un routeur attend donc la sortie de toutes les données présentes pour en sortir. Ce temps dépend du nombre de paquets en attente dans le routeur qui dépend lui même de l'intensité du traffic et du rapport des trafics à l'entrée et à la sortie du routeur. Le temps total de transmission est constitué de la somme de tous ces délais.
Les files d'attente sont de capacité limitée, ainsi, des paquets qui arrivent au routeur peuvent trouver une file d'attente complète et fermée. En absence d'espace libre pour stocker un paquet, le routeur l'abandonne et le paquet est dit perdu. Dans des applications qui nécessitent un transfert fiable (sans pertes), il est alors nécessaire de retransmettre ces données perdues. Cela altère le temps total de transmission, augmente la charge du réseau et baisse donc les performances de la communication.

Il apparaît que les performances offertes par les protocoles sont de natures différentes. Le choix d'un service se fera donc en fonction de l'application vers laquelle on s'oriente. Si l'application est le transfert de la vidéo ou de la voix, la garantie de temps de transmissions courts est importante alors que l'évitement des pertes de données l'est moins. Si l'application est un transfert de fichier (ou page web ou courier électronique), des temps de transmission courts sont moins importants alors que les pertes de données ne sont pas tolérées.

Le travail présenté ici se penche essentiellement sur l'analyse du comportement d'une ligne de communication TCP/IP et plus particulièrement sur le problème de congestion dans ces mêmes réseaux qui sera présenté en détails dans le chapitre 2.

Chapitre 2

Le contrôle de congestion

2.1 État de l'art

Dans un premier temps était définie la congestion comme un émetteur qui envoie des données à des vitesses trop élevées pour le récepteur (voir figure 2.1). La solution apportée alors fut d'informer l'émetteur, par le récepteur, de la capacité de ce dernier, l'émetteur étant alors tenu d'envoyer une quantité de donnée au plus égale à la capacité du récepteur.
Cependant, cette méthode, certes respecte la capacité du récepteur, mais pas celle du réseau. Si

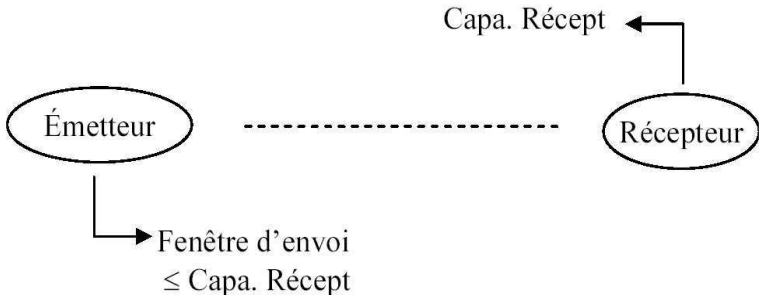


FIG. 2.1 – *Capacité du récepteur*

l'on se penche sur l'exemple de la figure 2.2, le récepteur R n'est pas saturé, cependant la capacité des routeurs est dépassée, d'où une congestion et donc des pertes de données. Un contrôle de congestion efficace passe donc par le respect de la capacité du récepteur et par celle du réseau qu'il utilise.

Dans les années 80 avec l'expansion grandissante d'Internet, ont commencé des séries de "congestion collapse"[1] qui causaient parfois une baisse soudaine de bande passante de l'ordre de plusieurs milliers d'octets par seconde. Des équipes de chercheurs (notamment de l'université de Berkeley) se sont penchées sur ce problème et ont introduit dans le 4.3 BSD (Unix de

1. Séries de congestions qui ont été appelées à ce moment là congestion collapse.

Berkeley) TCP de nouveaux algorithmes qui ont permis la prise en compte constante de l'état des réseaux et ont établi des méthodes pour réagir à une congestion. Notons que ces mécanismes de contrôle de congestion agissent en fonction des informations présentes au niveau de l'émetteur et du récepteur et sont dits, pour cela, algorithmes *point à point* (*end-to-end*). Ils opèrent généralement de sorte à forcer les connexions TCP à baisser leur débits lors d'une congestion et on dit des flots soumis à ces algorithmes qu'ils sont "sensibles aux signaux de congestion" (responsive to congestion signals). C'est principalement ces algorithmes qui préviennent de la congestion dans l'Internet actuel.

Les principaux algorithmes de contrôle de congestion sont implantés dans la couche transport,

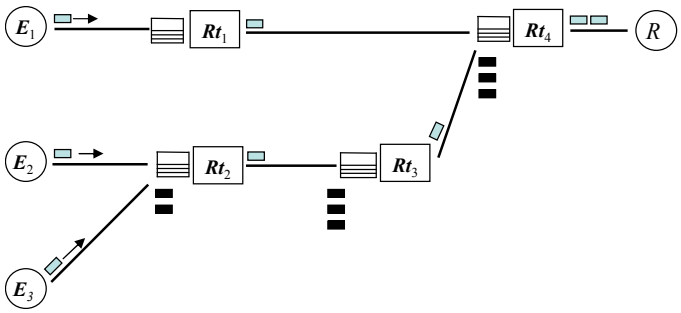


FIG. 2.2 – *Réseau congestionné*

donc dans le TCP. Cette couche offre un service fiable qui réemet les données en cas de pertes. Naturellement, pour réagir à une perte, il faut commencer par la détecter. Pour cela, la notion d'acquittement est utilisée. Un acquittement est un message que le récepteur envoie à l'émetteur lorsqu'il a bien reçu une donnée (c'est cela même qui indique qu'un service est fiable comme c'est le cas du TCP). Ainsi, grâce aux acquittements, l'émetteur sait si une donnée a été bien transmise ou doit être réémise. Ces algorithmes ont été élaborés par V. Jacobson en 1988 et 1990 ([Jac88], [Jac90]). Ils se basent sur le principe physique de la "conservation de paquets". Conservation de paquets dans les réseaux de communication signifie qu'à l'état d"équilibre', i.e., en transmission stable avec un maximum de données en transit, une nouvelle donnée peut entrer dans le réseau si une ancienne donnée en sort. L'arrivée d'un acquittement à l'émetteur lui indique qu'une donnée est sortie du réseau et qu'il peut de ce fait en envoyer une autre. Une fois l'équilibre atteint, les acquittements jouent donc le rôle d'"horloge" qui cadence l'envoi des données. D'après V. Jacobson, si le principe de conservation de paquets peut être appliqué aux réseaux, alors une congestion ne pourra se produire qu'à titre exceptionnel. Cependant, la réalité est quelque peu différente comme nous allons le voir dans ce qui suit.

2.2 Techniques d'analyse existantes

2.2.1 Les algorithmes point à point

De façon générale, une transmission sur une ligne de communication se base sur l'évolution de la taille des données envoyées selon la charge apparente du réseau. Cette taille maximale envoyée est appelée fenêtre de congestion. Elle est petite au début de la connexion, puis évolue tout au long de la transmission selon des indices de charge de réseau donnés par le récepteur (en l'occurrence les acquittements). Si une perte est détectée sur la ligne, le système est réinitialisé et la taille de la fenêtre de congestion aussi, puis la transmission reprend de nouveau.

Jacobson dans ([Jac88]) soutient que 3 raisons peuvent provoquer l'échec de la conservation de paquets:

1. la connexion n'est pas équilibrée,
2. l'émetteur envoie 1 nouveau paquet avant qu'un ancien paquet n'ait quitté le réseau,
3. l'équilibre ne peut être atteint parce que les resources sont limitées sur le réseau.

La première cause d'échec provient du fait que l'on soit tout au début d'une transmission ou après une réinitialisation suite à des pertes. L'équilibre peut être vu comme l'utilisation des acquittements comme "horloge". Lorsqu'il est atteint, la connexion s'auto cadence ajustant l'envoi des messages à l'arrivée des acquittements. Cependant, cette cadence est difficile à obtenir car elle stipule que pour envoyer des données sur la ligne, on doit disposer d'acquittements, or, pour recevoir des acquittements, on doit envoyer des données. La première idée fut de commencer la transmission directement par l'envoi d'une forte quantité de données (comme si l'on voulait atteindre l'équilibre dés le début de la transmission) et d'appliquer alors la conservation. Cette méthode ne peut pas marcher car l'envoi de beaucoup de données dans un réseau pour lequel on ne connaît pas l'état de charge ni les ressources a beaucoup de chance de causer une congestion. Un algorithme, slow start, a été défini pour atteindre cet équilibre de façon graduelle.
Avant d'entamer l'explication des différents algorithmes, nous allons commencer par expliquer ce qu'est une fenêtre d'envoi et comment on fait évoluer la taille de cette fenêtre (ajouter fenêtre et n° d'acquittement).
Les segments envoyés dans un flot TCP portent des numéros de séquence (numéro présent dans l'en-tête. Voir la section 1.3.3). Lorsque le récepteur reçoit un message numéroté, il renvoie un acquittement portant le numéro de la prochaine séquence attendue (présent dans le champ Numéro d'accusé de réception, voir la figure1.6). Ainsi, si le numéro du segment reçu est x, celui de l'acquittement envoyé sera $x + 1$. Par ce numéro d'acquittement, le récepteur signifie à l'émetteur que toutes les séquences jusqu'à x ont été bien reçues et que la séquence attendue est la $x + 1$ème. Or, un segment peut se perdre ou être retardé parce qu'il est passé par un chemin plus long. Supposons que le dernier segment reçu soit le numéro x, le prochain segment attendu est de ce fait le $x + 1$ème. L'émetteur envoie le segment $x + 1$ mais ce dernier est en retard. Le segment reçu aura donc le numéro $x + 2$ (s'il n'a pas de retard non plus). Le récepteur accepte ce segment arrivé mais envoi à l'émetteur un acquittement portant le même numéro que précédemment, i.e., $x + 1$.

2.2.1.1 L'algorithme slow start

L'algorithme, slow start (ou démarrage lent) a été établi pour atteindre l'état d'équilibre et pouvoir utiliser le principe de conservation. L'étape slow start sert à graduellement augmenter la quantité de données en transit en commençant par une quantité initiale généralement petite (généralement 1 ou 2 ou 4 segments).
Pour mettre en place cet l'algorithme, une nouvelle variable a été introduite: la fenêtre de congestion notée *cwnd* (*cwnd* par rapport à congestion window en anglais). Cette variable représente la quantité de segments que l'on peut envoyer sur une ligne de transmission avant de recevoir le premier acquittement.

Remarque 1 *La fenêtre de congestion :* cwnd*, varie tout au long de la connexion. En cas de réinitialisation de la transmission, cette fenêtre prend une même valeur fixée à l'avance. Cette valeur initiale de* cwnd *est appelée Initial Window* IW *et on la prend ici égale à un segment.*

Remarque 2 *Les termes relatifs aux différentes tailles des fenêtres et des segments sont pris de [[APS99]] où des définitions ont été données de façon claire et précise. Généralement, se sont les termes définis dans ce papier que l'on retrouve dans la littérature, c'est pourquoi nous avons choisi de les utiliser dans ce présent travail.*

Différentes versions de TCP existent. Selon la version utilisée, slow start commence par envoyer 1 segment ou 2 ou 4. Nous nous contenterons de la version classique de TCP qui stipule l'envoi d'un segment en début de transmission. La fenêtre de congestion initiale IW est donc égale à un. Cette première fenêtre est envoyée et si le segment est bien reçu par le récepteur, celui-ci envoi un acquittement à l'émetteur. Lorsque l'émetteur reçoit cet acquittement, il déduit que la transmission s'est bien déroulée et met dans la nouvelle fenêtre 2 segments par acquittement reçu, donc ici 2 segments. La nouvelle valeur de $cwnd$ est 2. Cette deuxième fenêtre de congestion est de nouveau envoyée. Considérons pour l'instant qu'il n'y a pas de congestion sur la ligne. Les deux segments envoyés seront donc reçus par le récepteur et les deux acquittements vont être reçus par l'émetteur. A chaque acquittement reçu, la 3^{eme} fenêtre sera augmentée de deux segments. Ainsi, au 2^{eme} acquittement reçu, la nouvelle $cwnd$ sera égale à 4, et ainsi de suite. La fenêtre de congestion va croître (elle sera multipliée par deux) à chaque transmission. C'est la phase exponentielle de la transmission. En réalité, l'algorithme 'démarrage lent' n'est pas si lent que ça. En effet, il prend un temps égal à $Rlog_2W$ ([Jac88]) où R est le temps de boucle (calculé de l'instant d'envoi d'une donnée jusqu'au moment de la réception de son acquittement) et W la taille de la fenêtre en paquet ([Jac88]). Ainsi, la fenêtre croît suffisamment vite pour que l'effet soit négligeable sur les performances.

Lorsqu'une donnée est envoyée, l'émetteur attend l'acquittement pendant un temps bien défini. Ce temps est donné par un temporisateur de retransmission appelé souvent *Tempo*. A l'envoi d'un message, *Tempo* est enclenchée. Si à l'expiration de *Tempo*, l'acquittement n'est pas encore arrivé, on en déduit que le message s'est perdu et l'émetteur le réemet. Cependant, si $Tempo$ est mal calibrée, il peut arriver qu'une donnée soit envoyée trop tôt ou trop tard, la deuxième raison d'échec de la conservation de paquets telle que présentée par Jacobson est un mauvais calibrage du temporisateur de retransmission.

2.2.1.2 Temporisateur de retransmission:

Le calibrage de *Tempo* et donc extrêmement important et longtemps, le choix de $Tempo$ a été le point crucial d'un algorithme de contrôle de congestion efficace. Le calcul de $Tempo$ fait parti des algorithmes introduit dans le TCP. Le corps de ce temporisateur de retransmission est l'estimation du temps de boucle. Le temps de boucle change selon l'état du réseau (notamment la charge des routeurs). Cette variation doit être prise en compte dans l'estimation du temps de boucle. Les spécifications du protocole TCP données dans [Jac88] suggèrent de mettre à jour le temps de boucle de la façon suivante: Lorsqu'un acquittement arrive, on calcule le temps de boucle réel et on s'en sert pour estimer le temps de boucle de la prochaine donnée à envoyer. Ce nouveau temps de boucle sera considéré constant jusqu'à l'arrivée du prochain acquittement et ainsi de suite.
Ainsi, si RTT (Round Trip Time) représente le temps de boucle estimé et M est le temps de boucle réel calculé, alors:

$$RTT(i+1) = \alpha RTT(i) + (1-\alpha)M(i) \tag{2.1}$$

où α est un facteur de pondération égal à 7/8 qui détermine le poids attribué à l'ancienne valeur de RTT. $RTT(i+1)$ est la nouvelle valeur de RTT. Elle va dépendre à la fois de sa précédente valeur ($RTT(i)$) et de la dernière valeur de M ($M(i)$).
Même avec une valeur correcte de RTT, il n'est pas facile de déterminer une valeur juste pour $Tempo$. La deuxième cause d'échec de la conservation de paquet est due à un mauvais calibrage de $Tempo$. Dans les premières implémentations, $Tempo$ était prise égale à $\beta Tempo$ avec β égal à 2. Une valeur constante de β ne permettait pas une bonne adaptation de $Tempo$ à la ligne de transmission. Jacobson proposa alors de rendre β proportionnelle à l'écart type de la fonction de densité de probabilité de retour d'un accusé de réception en fonction du temps. Il utilisa l'écart moyen (nommé D) comme estimation simple de l'écart type.
D est calculé en fonction de la différence, en valeur absolue, entre la valeur estimée et celle observée du temps de boucle ($|RTT(i+1) - M(i)|$). D est donné par l'équation:

$$D(i+1) = \alpha D(i) + (1-\alpha)|RTT(i+1) - M(i)| \tag{2.2}$$

La valeur associée à $Tempo$ est alors calculée de la façon suivante:

$$Tempo(i+1) = RTT(i+1) + 4D(i+1) \tag{2.3}$$

Le choix de multiplier M par 4 s'est fait suite à des tests. Il s'est avéré qu'un délai 4 fois supérieur à l'écart type permettait de minimiser les expirations de temporisateurs suivies de retransmissions inutiles.
$Tempo$ est donc recalculée de façon dynamique pour suivre au mieux les fluctuations de l'état de la ligne de transmission. En effet, $Tempo$ sert à dire s'il y a perte ou pas. si $Tempo$ est trop courte, le risque est qu'elle détecte prématurément une congestion et que l'algorithme retransmette des données probablement déjà reçues par le récepteur. Ceci aurait aussi pour effet de charger inutilement le réseau. A l'opposé, une valeur de $Tempo$ trop longue détecterait en retard une congestion. Il en résulterait une augmentation du nombre de pertes et une baisse de performances. La figure 2.3 montre la probabilité d'arrivée des acquittements par temps de

boucle.

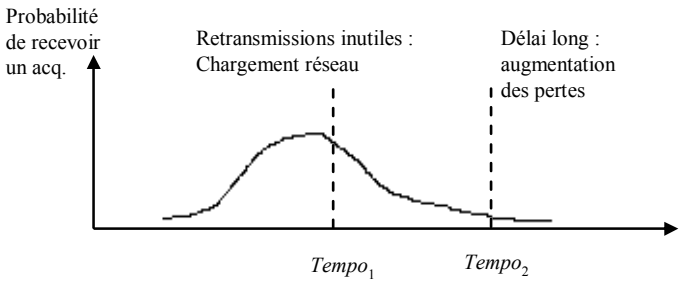


FIG. 2.3 – *Choix du temporisateur de retransmission* $Tempo$

2.2.1.3 L'algorithme congestion avoidance:

La troisième raison d'échec de la conservation de paquets est la limitation de ressources. La prochaine étape dans le contrôle de congestion est l'algorithme de contrôle de congestion lui même (congestion avoidance en anglais) qui a pour objectif d'adapter la transmission aux ressources de la ligne. Autrement dit, on envoie une nouvelle donnée que si une ancienne donnée sort du réseau. On peut se servir des acquittements pour enclencher l'envoi d'une donnée. Pendant la phase congestion avoidance, si une fenêtre est reçue, alors la suivante sera plus grande de l'équivalent d'un segment. Supposons que $cwnd$ soit donnée en nombre de segments[2], cela revient à dire qu'à chaque acquittement reçu on met dans la nouvelle fenêtre une quantité égale à 1 segment plus l'équivalent de ($\frac{1}{cwnd}$) segment. De la sorte, si tous les acquittements arrivent, i.e., $cwnd$ acquittements, la prochaine fenêtre envoyée aura exactement ($cwnd$+1) segments.
Une deuxième variable est nécessaire à la mise en oeuvre de cet algorithme: le seuil d'évitement de congestion (slow start threshold) noté $ssth$. Ce seuil va délimiter lequel des deux algorithmes slow start ou congestion avoidance sera appliqué. Tant que $cwnd$ est inférieur au seuil $ssth$, c'est slow start qui est appliqué, si la taille de la fenêtre est supérieure à $ssth$, c'est congestion avoidance qui est appliqué[3].

Lorsqu'une congestion est détectée, le seuil $ssth$ est mis à jour à la moitié de la fenêtre courante d'envoi.
Si flightsize représente la taille des données envoyées et pas encore acquittés,
et SMSS (Sender Maximum Sender Size) représente la taille du plus grand segment que l'émetteur puisse envoyer[4].
Alors:

$$ssth = max(flightsize/2, 2^{*}SMSS) \tag{2.4}$$

2. $cwnd$ peut aussi être donnée en nombre d'octet. Voir [APS99]
3. lorsque $cwnd = ssth$, les deux algorithmes peuvent être appliqués indifféremment.
4. Les définitions détaillées de SMSS et de Flightsize sont données dans [APS99].

et la taille de $cwnd$ elle est remise à IW, ici à 1 segment.
Après avoir remis à jour ces différentes valeurs, la transmission redémarre en appliquant l'algorithme slow start. L'évolution de la taille de $cwnd$ tout au long de la transmission est donnée par la figure 2.4.

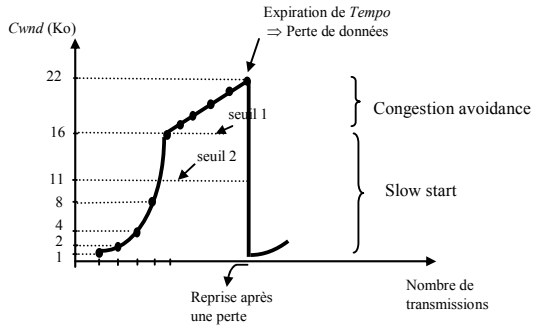


FIG. 2.4 – *Evolution d'une transmission TCP/IP sous les algorithmes slow start et congestion avoidance*

2.2.1.4 Algorithmes fast retransmit et fast recovery:

Par la suite ont été établis, toujours par Jacobson, deux autres algorithmes: fast retransmit et fast recovery. Ils servent respectivement à déterminer une perte et à retrouver l'état d'équilibre plus rapidement qu'avec les deux premiers algorithmes slow start et congestion avoidance.

Algorithme fast retransmit Tout d'abord, la détermination d'une perte n'est plus donnée uniquement par l'expiration du temporisateur de retransmission $Tempo$ mais aussi par ce qu'on appelle les acquittements dupliqués. Expliquons en premier ce que sont les acquittements dupliqués. Comme expliqué précédemment (dans le paragraphe 2.2.1: algorithmes point à point), lorsqu'un récepteur reçoit une donnée, il envoie un acquittement portant le numéro de la prochaine séquence attendue. Si le dernier segment envoyé se perd ou est en retard, le récepteur accepte les autres segments qui arrivent mais va continuer d'envoyer des acquittements portant ce même numéro de séquence jusqu'à ce qu'il ait reçu le bon segment ou que le temporisateur $Tempo$ expire. La théorie des acquittements dupliqués soutient qu'au delà d'un nombre fixé d'acquittements portant le même numéro de séquence, on peut conclure une perte sans attendre que $Tempo$ expire. C'est le changement apporté dans fast retransmit où après l'arrivée de 3 acquittements successifs dupliqués (4 acquittements portant le même numéro), TCP enclenche une réinitialisation suite à ce qui parait être la perte d'un message. La fenêtre de congestion $cwnd$ est remise à 1 et le seuil $ssth$ est mis à jour selon l'equation 2.4, puis la transmission reprend selon l'algorithme slow start.

Notons cependant que les récepteurs ne renvoient pas toujours les acquittements immédiatement après la réception d'un segment car ils peuvent être occupés à traiter d'autres données.

Pour accélérer le processus de recouvrement basés sur les acquittements dupliqués, il faut que les récepteurs émettent immédiatement les acquittements des segments qu'ils ont déjà réceptionnés.

L'algorithme fast retransmit est implémenté dans le TCP Tahoe ([FF96]).

Algorithme fast recovery Le dernier algorithme le plus couramment implémenté dans le TCP est fast recovery. Celui ci permet de retrouver rapidement l'état d'équilibre en évitant comme dans les versions précédentes de TCP de passer par l'algorithme slow start pout toute réinitialisation suite à des pertes. Cet algorithme est comme on le verra un peu plus loin, surtout utile lorsque les fenêtres de transmission sont grandes et qu'il n'y a que très peu de pertes par temps de boucle. L'idée ajoutée dans cet algorithme est donc de ne pas reprendre une transmission à slow start après une perte détectée par les acquittements dupliqués. La transmission reprend directement avec congestion avoidance. En effet, d'une part, slow start est une phase où la transmission n'est pas stable et où la conservation n'est pas encore appliquée. D'autre part, le fait qu'il y ait des acquittements qui arrivent à l'émetteur signifie que des segments arrivent au récepteur, donc, il se produit peut être des pertes sur la ligne mais celle-ci continue malgré tout de transmettre des messages et il n'est pas nécessaire de diminuer trop brutalement la cadence de la transmission.

Les algorithmes fast recovery et fast retransmit sont implémentés dans TCP Reno ([FF96]).

Fast recovery ne peut réemettre qu'un seul paquet perdu par cycle. Ceci nous conduit au principal inconvénient de TCP Reno (l'ensemble des inconvénients de Reno est cité dans [FF96]) qui est que ce dernier ne détecte et ne réemet qu'un seul paquet dans une même phase fast recovery. Cet inconvénient ne le rend performant que qu s'il n'y a pas plus d'une perte par RTT. S'il y a 2 voir 3 pertes dans un même temps de boucle, ses performances sont très mauvaises comme cela a été montré dans [FF96] où des simulations ont été élaborées pour comparer les performances des différentes versions de TCP lorsqu'il y a 1 perte puis 2 puis 3. A 3 pertes, l'émetteur est incapable d'envoyer des données et est finalement obligé d'attendre que $Tempo$ expire pour reprendre la transmission avec slow start.
Pour pallier à cet inconvénient, Reno a subi une modification suggérée par Janey Hoe ([Hoe95] et [Hoe96]). La version Reno modifiée est appelée New Reno ([FH99]). La nouveauté dans le New Reno est qu'on arrive à détecter si une ou plusieurs segments se sont perdus en un seul cycle fast recovery. Cependant, la réemission des pertes se fait d'un paquet au plus par RTT. D'autres nouveaux protocoles sont proposés où il serait possible de pallier complètement aux problèmes d'une seule réemission au plus de segments perdus par RTT. C'est le cas du TCP SACK (Selective ACKnowledgement). Cependant, cette option proposée dans [JB88] et [MMFR96] est implémentée dans certains réseaux comme Netblt [CLZ88], XTP [SDW92] et RDP [VH84] mais elle n'est pas encore utilisée pour Internet.

2.2.2 Les algorithmes basés sur les routeurs: Active Queue Management

Les algorithmes, améliorations et changements vus jusque là concernent tous des méthodes de contrôle de congestion point à point, i.e., émetteur-récepteur. En effet, la seule information

pertinente utilisée pour déduire ou non une congestion est la réception ou non d'un acquittement. L'émetteur "déduit" alors l'état du réseau en fonction soit d'un acquittement arrivé en retard (expiration de $Tempo$) ou de plusieurs acquittements portant le même numéro (TCP Reno) et décide de l'évolution de la taille des données transmises. En aucun cas, l'information "congestion dans le réseau" ne vient du réseau lui même. Des mécanismes de contrôle doivent être ajoutés aux routeurs pour compléter les contrôles point à point.
Il existe deux classes d'algorithmes basés sur les routeurs:

- Les files d'attentes (Queue Management)
- Les algorithmes d'ordonnancement (scheduling algorithms)

Les files d'attente contrôlent la longueur de la file d'attente dans les buffers en rejetant certains paquets lorsque nécessaire. Les algorithmes d'ordonnancement déterminent quel prochain paquet doit être envoyé et sont surtout utilisés pour partager la bande passante aux différents flux. Ces 2 types d'algorithmes visent donc des performances différentes.

- **Tail drop**:
  La technique traditionnelle de contrôle de file d'attente dans les routeurs est de fixer une longueur maximale de bande passante pour chaque flot. Tant que cette taille n'est pas atteinte, les paquets venant de ce flot sont acceptés. Si elle est atteinte, les paquets sont rejetés en attendant que la longueur de la file d'attente baisse. Cette technique est appelé tail drop ([BCC$^+$98]). Les derniers paquets arrivés sont ceux qui sont rejetés. L'inconvénient de cette méthode est qu'elle peut induire la monopolisation de la bande passante par un flot unique ou par peu de flux. D'autre part, l'objectif principal des gestions de files d'attente est d'atteindre une connexion équilibrée tout en ayant un file d'attente faible dans les buffers. En effet, une file d'attente faible dans les buffers permet à ces derniers de protéger une ligne de transmission d'une arrivée brusque de donnée. Une file faible à l'équilibre traduit la capacité du buffer à absorber une forte arrivée de message. Or, tail drop agit lorsqu'il y a perte de donnée et donc lorsque la file est pleine. D'où l'apparition de nouvelles techniques de file d'attente appelées Active Queue Management (AQM) qui permettent de garder des files d'attente faibles.

- **Random Early Detection: RED**
  L'algorithme de contrôle de congestion Random Early Detection (RED) est un algorithme basé sur les files d'attente et implanté dans les routeurs ([BCC$^+$98], [FJ93]). Contrairement aux algorithmes de files d'attente classiques qui rejettent les paquets lorsque les buffers sont pleins, RED rejette les paquets entrants avec une certaine probabilité avant que les buffers ne soient pleins. Cette probabilité augmente lorsque la taille moyenne de la file d'attente augmente. L'objectif de RED est d'une part de contrôler la longueur de la file d'attente dans les buffers mais aussi d'éviter les inconvénients des autres algorithmes de contrôle de congestion (comme tail drop) où un buffer saturé entraîne la baisse des débits de toutes les sources qui l'utilisent (c'est ce qui est appelé le problème de synchronisation). L'algorithme RED comprend deux étapes: La première est d'estimer la valeur moyenne de la file d'attente (les détails de calculs de cette file d'attente sont donnés dans ([FJ93])) et la seconde où RED décide du rejet ou pas d'un paquet entrant. Deux para-

mètres sont considérés dans cet algorithme pour ce processus de décision: $minth$ (minimum threshold) et $maxth$ (maximum threshold). $minth$ est le seuil de la valeur moyenne de la file d'attente au dessous de laquelle toutes les données qui arrivent au buffer sont acceptées. $maxth$ est le seuil de la valeur moyenne de la file d'attente au delà de laquelle toutes les données qui arrivent au buffer sont rejetées. Si la taille de la file d'attente est comprise entre $minth$ et $maxth$, chaque paquet est marqué avec une probabilité p_a qui dépend de la longueur moyenne de la file d'attente (([FJ93])) (figure 2.5).

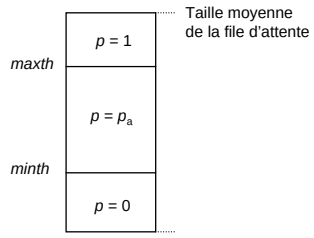


FIG. 2.5 – *Seuils min et max de l'algorithme RED.*

Cependant, à ce jour, l'algorithme RED n'a pas le degré d'universalité du protocole TCP. Il n'est pas installé sur la majorité des routeurs. L'algorithme tail drop, qui ne rejette les paquets que lorsque la mémoire tampon est pleine, est encore majoritaire. L'utilité de RED est même contestée (voir par exemple [BZI+01] et [MBD+99]) en raison des complications algorithmiques engendrées au niveau des routeurs. En effet, pour être pleinement efficace (i.e. meilleur que l'algorithme tail drop qualifié de fainéant), les seuils $maxth$ et $minth$ doivent être optimisés. Les valeurs optimales sont malheureusement trop sensibles aux conditions de trafic qui fluctuent beaucoup dans le réseau; elles sont difficiles à réactualiser en raison du coût algorithmique.

2.2.3 Quelques travaux sur l'évaluation de performances des réseaux de communication

Des études sur la stabilité des réseaux de communication ont été effectuées et notamment ceux de [BB98] où les auteurs modélisent chaque noeud d'un chemin par une file d'attente fifo et déterminent des bornes aux débits maximaux admis par le contrôle de flux. Leurs travaux montrent que les performances du contrôle de flux à fenêtre dépendent non seulement de l'intensité du traffic mais aussi de la variabilité de ces flux.
D'autres travaux [PFTK98] se sont penchés sur l'évaluation des performances des réseaux en élaborant leurs recherches sur la détermination d'équations analytiques modélisant le débit de transmission en fonction des pertes sur une ligne et des temps de boucle. Ces travaux rejoignent une ensemble d'autres travaux ([MMFR97], [LPD02]). L'originalité des travaux de [PFTK98] réside dans la prise en compte d'un débit dit "équitable" entre les flux non-TCP qui interagissent avec une connection TCP (cette équité est appelée TCP-friendly, [MMFR97]). Ces travaux sont faits sur la base de certaines hypothèses sur les protocoles (temps de l'algorithme slow start court par rapport au temps de simulation), sur des variables statistiques (pertes corrélées, pertes dans un temps de boucle indépendantes des pertes pendant les autres temps de boucle, temps

de boucle indépendants de la taille des fenêtres) et enfin les protocoles utilisés par tous les flux sont supposés être le TCP-Reno.
D'autres travaux de la communauté automatique tentent de trouver un modèle analytique qui modélise le débit afin d'agir avant que le pertes ne se produisent et ainsi de ne pas perdre en terme de performances de la liaison réseaux ([LPD02],[HC01]). Ils proposent de nouveaux protocoles ("scalar control") qui pallient aux inconvénients des protocoles classiques utilisés dans l'Internet d'aujourd'hui. Les travaux tels que ceux de [HC01] se concentrent sur l'analyse de la stabilité des réseaux TCP/AQM en commençant par établir un modèle analytique d'une transmission TCP puis en linéarisant le modèle (les flux de TCP étant de nature non linéaire).
Des travaux d'évaluation de performances ont été élaborés pour les réseaux dans lesquels circulent des traffics de données sans contrôle de flux (UDP par exemple). Parmi ces travaux ceux qui traitent de la résolution des réseaux des files d'attente par des méthodes approximatives (comme l'approximation de la distribution jointe de probabilité de l'état du réseau, [Roy05]) et en utilisant des modèles fluides. Ces techniques de résolution de réseaux de files d'attente s'apparentent à celles usuellement utilisées dans les systèmes de production. Ces modèles fluides apportent de bonnes solutions en ce qui concerne les systèmes mono-buffer ([DF82]) mais les solutions restent inconnues dans le cas multi-buffer. Dans ce cas là, sous certaines hypothèses, on suppose l'indépendance des variables aléatoires qui représentent l'état des buffers et on peut construire une solution ([DDX89]).

De façon générale, les travaux d'évaluation de performances sont surtout partagés entre la communauté réseaux de communication et la communauté automatique. La communauté réseaux de communication se concentre, pour améliorer les performances des réseaux, sur les protocoles de contrôle de congestion soit en baissant les débits lors de l'occurrence des pertes soit en améliorant le temps de recouvrement de la transmission suite à ces pertes ([BB98], [Ste97], [MMFR96], [MMFR97], [BCC$^+$98], [FJ93]). Les travaux de la communauté automatique se concentrent sur l'obtention de modèles analytiques des flux afin d'appliquer les techniques de contrôles par retour d'état et d'obtenir un système stable ([HC01], [LPD02], [Roy05], [PFTK98]).

Chapitre 3

Modélisation d'une ligne de communication TCP/IP

L'état d'un système dynamique peut être décrit par une combinaison de variables continues (qui évoluent dans un intervalle de temps continu), discrètes (qui évoluent dans un ensemble dénombrable) ou symbolique (qui caractérisent un état d'un élément du système qui évolue dans un ensemble fini non structuré).
Un système physique peut être décrit par un modèle à dynamique continue ou par un système à événement discret (SED) selon les phénomènes auxquels on s'intéresse. Si une description des états d'un système peut suffire, on se tourne vers les SED pouvant être décrits par les outils de modélisation discrets. Si l'on s'intéresse à la description de phénomènes tels que remplissage de bac, variation de température ou de concentration, on s'oriente vers des modèles continus décrits par des équations. Ainsi, il est suffisant dans certains cas d'utiliser l'un ou l'autre de ces modèles pour décrire de façon satisfaisante un système donné. Cependant, les systèmes à commander ne peuvent généralement pas être classés dans l'une de ces catégories et nécessitent une prise en compte à la fois de variables d'états continues, discrètes et symboliques et des évolutions à la fois continues et événementielles. Ces types de systèmes sont appelés systèmes à dynamiques hybrides (SDH) et couvrent des domaines d'applications variés tels que les systèmes de transport, l'industrie agro-alimentaire et les réseaux de communication.
De façon générale, les approches de modélisation des systèmes hybrides peuvent être classées en 3 catégories:

1. Les approches basées sur une extension des modèles continus. Parmi elles, une extension des modèles bond graph.
2. Les approches basées sur une extension des modèles pour les systèmes à événements discrets, l'exemple type de cette extension est celui des RdP hybrides.
3. Les approches mixtes combinant les parties continues et discrètes dans une même structure. Parmi ces approches, les RdP mixtes et les statecharts hybrides.

Les bond graphs sont très utilisés dans la modélisation de systèmes physiques tels que les systèmes hydrauliques ou électriques. La modélisation est basée sur la décomposition du système ou sous système en identifiant les échanges énergétiques puis en déterminant les éléments de chaque système (moteur, vérin, charge, etc) et les variables qui définissent l'interaction (échange d'énergie) entre les systèmes. Un formalisme graphique permet de décrire des

éléments non linéaire qualifiés de "tout ou rien" tels que les diodes, les thyristors ou les embrayages. Il est complété d'une démarche qui permet de déduire les équations mathématiques. Ce formalisme utilisé est surtout appliqué dans les systèmes électriques ou hydrauliques.
Dans les RdP discrets, le marquage d'une place peut correspondre soit à un l'état booléen d'un élément du système (ressource disponible ou pas), soit à un nombre entier naturel qui compte une capacité d'objets (nombre de pièces dans un stock). Cependant, lorsqu'un RdP contient un grand nombre de jetons, le nombre d'états atteignables explose. Cette limitation des RdP a conduit à l'apparition des RdP continus ([AD98], [All87]). Les marquages de places dans un RdP continu sont des nombres réels et le franchissement des transitions est un processus continu. Ainsi, le franchissement des transitions consiste à retirer une quantité de marquage q des places d'entrées de la transition et de rajouter la même quantité q aux places de sorties de cette transition. Cette modélisation fournit une très bonne approximation pour certains modèles discrets. Cependant, si le nombre de pièces dans un stock peut être approximé à un processus continu, l'état d'un élément du système tel que vanne ouverte ou fermée ne peut être modélisé par un nombre réel. La modélisation d'un système hybride conduit naturellement aux RdP hybrides contenant une partie discrète et une partie continue ([DA01], [DA05]).
Des modèles dits mixtes reposent sur la collaboration de deux sous-modèles, l'un pour l'aspect événementiel, basé par exemple, sur les automates à états finis ou les RdP et l'autre pour l'aspect continu basé sur des équations d'états. L'aspect hybride est pris en compte dans l'interface entre les deux sous-modèles.
Un des modèles basé sur cette approche est les automates hybrides. Les automates à états finis sont constitués de sommets représentant les états discrets, de transitions entre sommets définies par un triplet: sommet source, sommet destination et événement, d'un ensemble fini de symboles décrivant les événements (appelé alphabet) et d'un état initial. Le comportement global d'un système à événement discret est décrit par l'ensemble des trajectoires d'événements qui peuvent être exécutées en parcourant l'automate à états finis à partir de l'états initial. Les automates temporisés sont des automates finis étendus par un ensemble de variables réelles, appelées *horloges*. Les automates hybrides sont une extension des automates temporisés où la dynamique continue n'est plus représentée par des horloges mais par des équations différentielles quelconques. De façon informelle, un automate hybride apparaît comme l'association d'un automate d'état fini pilotant un ensemble d'équations dynamiques continues [Zay01]. Cependant, il est rapidement apparu que même pour traiter des problèmes simples, il est nécessaire d'introduire un mécanisme permettant de structurer les modèles et de les définir de manière modulaire. Ainsi, la composition des automates hybrides est basée sur la composition synchrone. Cet état de fait rend difficile la conception et la description des automates hybrides pour la modélisation du comportement d'un système. Notons que cette difficulté n'altère pas l'intérêt des automates hybrides pour les traitements théoriques tels que la vérification, la synthèse de la commande ou la définition des interactions entre la partie événementielle et la partie continue [Zay01].
Un second exemple d'outils mixtes est les statecharts hybrides. Ces derniers ont été définis comme une solution pour améliorer la structuration des spécifications à base d'automates afin de faciliter leur description, leur lecture et leur modification.
Les statechart sont un outil de modélisation graphique pour les système réactifs complexes. Les transitions sont étiquetées par des expressions de la forme "événement déclenchant/action"; où l'action est synchrone avec l'événement déclenchant. Les statecharts sont habituellement carac-

térisés par l'expression: statechart = automates + profondeur + orthogonalité + communication par diffusion. La notion de profondeur sert à décomposer des super-états en sous-états afin de simplifier la représentation des graphes dans une approche ascendante et la décomposition par niveaux d'abstraction dans une approche descendante. La notion d'orthogonalité permet de décomposer un état en sous-état actifs simultanément et de diminuer ainsi le nombre d'états manipulés en considérant des automates localement indépendants. Afin d'éviter des problèmes temporels d'évolution, les événements sont émis en diffusion et sont disponibles partout en même temps. Les statecharts permettent donc d'augmenter la lisibilité des modèles ([Zay01]).
Un autre exemple d'outils mixtes les RdP mixtes (RdPM). Comme pour les statecharts, les RdP mixtes reposent essentiellement sur la définition de l'interface entre la dynamique continue et celle événementielle d'un système. Les RdP mixte consiste à donner une interprétation aux RdP par un ensemble d'équations différentielles. Un RdP mixte complet d'un système est modulaire et est constitué d'un ensemble de RdP interprétés par des équations algébro-différentielles, chaque RdP modélisant les changements d'états discrets d'un phénomène physique. Une place d'un RdPM peut modéliser la configuration d'une ressource du système à laquelle est associée un comportement dynamique. Dans ce cas, une ou plusieurs équations différentielles peuvent être associées à cette place. Une place peut aussi modéliser un phénomène discret et dans ce cas, aucune équation différentielle ne lui sera associée. Pour certains systèmes complexes, une équation différentielle peut être associée à un marquage global plutôt qu'à une seule place. Enfin et de même que pour les automates ou les statecharts hybrides, l'aspect continu agit sur l'évolution du marquage par l'intermédiaire de conditions de validations associées aux transitions ([Zay01]).
Une des caractéristiques des ces différents formalismes mixtes est leur généralité puisqu'ils ne font pas d'hypothèses sur le type de phénomènes à modéliser. La contre partie de cette liberté de description des modèles est que d'une part, ils ne contiennent pas d'éléments méthodologiques pour guider l'élaboration des modèles et d'autre part, la validité du modèle n'est pas assurée par construction et doit donc être vérifiée.

Nous nous intéressons ici à un domaine d'application particulier qui est les réseaux de communication. Nous n'avons donc pas d'intérêt particulier à utiliser un formalisme trop général. Par ailleurs, nous désirons avoir une bonne lisibilité et compréhension du modèle. Ainsi, nous avons fixé notre choix sur les RdP hybrides comme outils de modélisation et nous pensons que cet outil est bien adapté au cas des réseaux de communication car il s'agit de systèmes à flux continus où toutes les variables sont positives. Nous allons voir un peu plus loin que cet outil représente une bonne approximation des phénomènes rencontrés dans les réseaux et que, tout en étant un outils intuitif et simple d'utilisation, il permet de mettre en évidence les différentes dynamiques de façon juste et précise.

3.1 Outils de modélisation: Les réseaux de Petri hybrides

Les Réseaux de Petri (RdP) sont un outil qui permet de modéliser de façon simple et intuitive des systèmes de nature très variée. Ils permettent de visualiser des comportements tels que la synchronisation des tâches, le parallélisme ou encore le partage de ressources.
Les RdP sont de nature essentiellement discrète. En effet, ils permettent de modéliser des états

discrets d'un systèmes tels que la marche ou l'arrêt d'une machine ou encore le nombre de pièces présentes dans un stock. Cependant, ces modèles discrets se sont rapidement vus limités car ils explosent rapidement lorsque le nombre d'états croit. Sont apparus alors les RdP continus (ref).
Nous allons présenter brièvement les RdP discrets car ils sont désormais connus en ne rappelant d'ailleurs que la partie formelle. Celle-ci nous sera utile dans la définition des RdP temporisés et continus sur lesquels nous nous attarderons davantage.

Définitions et concepts de base des RdP: Un RdP marqué est un quadruplet:

$$R =< P, T, Pré, Post, M_0 > \tag{3.1}$$

où

- P est l'ensemble fini non vide des places du RdP;
- T est l'ensemble fini non vide des transitions du RdP;
- $P \cap T = \emptyset$, c'est à dire que les deux ensembles sont disjoints;
- *Pré* : $P \times T \rightarrow N$ est l'application *d'incidence avant*.
  Pré(P_i, T_j) est le poids de l'arc $P_i \rightarrow T_j$;
- $Post$: $P \times T \rightarrow N$ est l'application *d'incidence arrière*.
  $Post$(T_j, P_i) est le poids de l'arc $T_j \rightarrow P_i$.
- M_0 est le marquage initial tel que $M_0(P_i)$ est le nombre de marques contenues dans la place P_i.

On note $^\circ T_j$ et T_j° l'ensemble des places d'entrées et de sorties respectivement de la transition T_j. De même, on note $^\circ P_i$ et P_i° l'ensemble des transitions d'entrées et de sorties respectivement de la place P_i.

On appelle **degré de validation** d'une transitions T_j la valeur q telle que:

$$q \leq min_{i:P_i \in {}^\circ T_j}(\frac{m(P_i)}{Pre(P_i, T_j)}) < q + 1. \tag{3.2}$$

Dans le cas des RdP discrets, *une transition T_j est validée si $q \geq 1$*. On dit que T_j est q-validée. Le vecteur de validation e du RdP de la figure 3.1 pour le marquage initial $M_0 = (1001)$ est: $e = (q(T_1, M), q(T_2, M), q(T_3, M)) = (100)$.

On appelle *matrice d'incidence avant* la matrice:
W^- = [wij^-], où wij^- = Pré(P_i, T_j)

et *matrice d'incidence arrière* la matrice:
W^+ = [wij^+], où wij^+ = $Post(P_i, T_j)$

Pour l'exemple du RdP de la figure 3.1.a, on a:

$$W^- = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad et \quad W^+ = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.3}$$

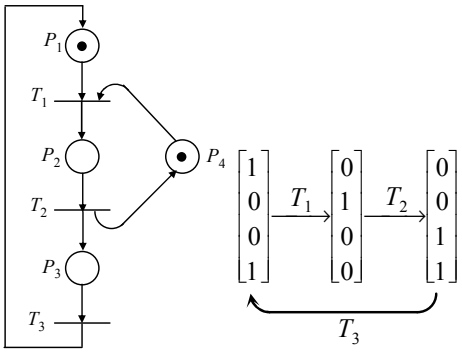


FIG. 3.1 – *a) Exemple de RdP ordinaire. b) Graphe de marquage équivalent*

On appelle *matrice d'incidence* la matrice:

$$W = W^+ - W^- = [w_{ij}] \tag{3.4}$$

Une colonne de cette matrice correspond à la modification du marquage apportée suite au franchissement de la transition correspondante.
Un *graphe des marquages* (figure 3.1.b) est composé de noeuds correspondants aux marquages accessibles et d'arcs qui correspondent aux franchissements des transitions qui font passer d'un noeud à un autre.

Ces notions d'applications d'incidence *Pré* et $Post$ et de W est commune aux différents types de RdP (les RdP colorés, les RdP continus, -que l'on verra dans les paragraphes suivants-). En effet, au lieu que les seules valeurs prises par ces applications soient des uns et des zéros, celles-ci peuvent être des nombres rationnels, des vitesses (cas des RdP continus) ou encore des fonctions (cas des RdP colorés).
Le graphe des marquages (ou d'évolution), de même, se construit aussi pour les RdP temporisés, continus ou hybrides, à la différence que ce sont des graphes plus complexes que ceux obtenus pour les RdP ordinaires.

Extensions et abréviations des RdP: Aux réseaux de Petri "ordinaires" sont venus s'ajouter des extensions qui, sans modifier les caractéristiques de ces RdP, permettent d'aborder un plus grand nombre d'application. Parmi eux, on trouve les RdP à arc inhibiteur, appelés aussi test à zéro, qui permettent le franchissement d'une transition lorsqu'une de ses places d'entrées est vide. A l'opposé, des abréviations des RdP existants viennent simplifier le graphisme des réseaux sans pour autant diminuer la complexité du système modélisé. On peut citer comme exemple les RdP à capacité, les RdP colorés et les RdP généralisés. Nous allons voir plus en détails dans la section 3.3.1 les RdP colorés car nous les utilisons pour établir le modèle d'une ligne de communication.

3.1.1 Les RdP temporisés

Les RdP ordinaires permettent de modéliser le cours d'une opération mais ils n'apportent pas l'information "durée" de cette opération. Il est possible de représenter cette notion de temps

grâce aux RdP temporisés qui permettent non seulement de dire ce qui se passe, mais quand ça se passe.
Pour cela, des durées peuvent être associées à chaque place d'un RdP. On parle alors de RdP P-temporisés. Ces durées peuvent également être associées à chaque transition d'un RdP, on parlera alors de RdP T-temporisés. Le modèle retenu pour la suite de notre travail est le T-temporisé.

Principe de fonctionnement des RdP T-temporisés Un RdP T-temporisé est défini par le couple <R, Tempo> où R est un RdP non marqué et Tempo une application telle: Tempo(T_j) = d_j = durée associée à T_j (d_j rationnel positif ou nul).
Prenons l'exemple de la figure 3.2. Si, à t_0, la transition T_1 est validée, alors il restera une durée d_1 avant son franchissement. Cette durée est appelée *temps résiduel de franchissement*. De même, si T_2 est validée, il s'écoulera un temps d_2 avant son franchissement.

Remarque 3 *On supposera dans tous les modèles RdP T-temporisés qui vont suivre, qu'à l'état initial, toutes les durées résiduelles sont maximales (noté $m_{0,max}$ dans [DA05]), i.e., égales à d_i pour chaque transition validée T_i.*

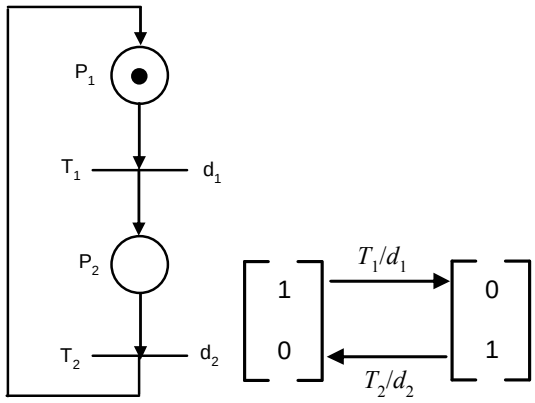


FIG. 3.2 – *a- RdP T-temporisé. b- Graphe de marquage équivalent.*

3.1.2 Les RdP continus et hybrides

La limitation des RdP discrets réside dans le fait que lorsque le nombre d'états d'un système augmente, le nombre de marquages accessibles du RdP qui le représentent explose et devient difficile voire impossible à exploiter. A titre d'exemple, on peut considérer le système de la figure 3.3 composé de 3 machines M_1, M_2 et M_3 et de deux stocks S_1 et S_2 de capacité C_1 et C_2 respectivement ([DA01] et [AD98]). Nous admettons qu'il y a toujours des pièces brutes qui arrivent à l'entrée de M_1 et de la place disponible pour les pièces en sortie de M_3. Le nombre d'états atteignables est $N = 2^3(C_1 + 1)(C_2 + 1)$. Soit, pour C_1 = C_2 = 12, N = 1 352. Si le nombre de machines passe à 10 et le nombre de stock à 9, alors pour les mêmes valeurs de C_1 et C_2, N est supérieur à 10^{13} états. Cette réalité a engendré l'apparition des RdP continus [AD98].

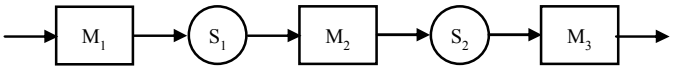


FIG. 3.3 – *Ligne de production.*

3.1.2.1 Idée de base des RdP continus autonome

Nous pouvons introduire les RdP continus par l'idée de la division du marquage M d'un RdP discret en k parties égales [DA92].
Prenons l'exemple du RdP R de la figure 3.4. Le vecteur de marquage, noté M d'un RdP représente le marquage à instant donné, de toutes les places du RdP. Le vecteur de marquage M de R est (1, 0). Si on divise M par deux ($k = 2$), le marquage M' du nouveau RdP R' est (2, 0). Si on divise M par 4 ($k = 4$), M' sera égal à (4, 0), etc. Si k est quelconque, alors le marquage M' sera (k, 0). Supposons que k est infini, l'ensemble des marquages accessibles devient aussi infini (figure 3.4). Le RdP ainsi obtenu est dit **continu autonome**.

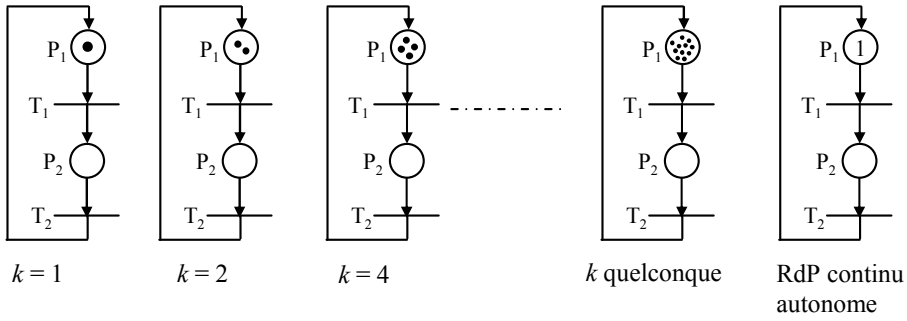


FIG. 3.4 – *Transformation d'un RdP ordinaire en RdP continu: division de marques.*

DÉFINITION 1 *Dans les RdP continus autonomes, une transition est validée si ses places en amont possèdent un marquage non nul. De façon plus formelle, le degré de validation d'une transition T_j continue est donnée par:*

$$q = min_{i:P_i \in {}^\circ T_j}(\frac{m(P_i)}{Pre(P_i, T_j)}) < q + 1. \tag{3.5}$$

Si $q > 0$, alors T_j est validée [DA05].

3.1.2.2 Les RdP continus temporisés

On a utilisé la division des jetons en k marques pour passer d'un RdP discret à un RdP continu autonome. De même, nous allons utiliser le même concept pour passer d'un RdP discret temporisé à un RdP continu temporisé.
Pour cela, nous allons commencer par expliquer les notions de limitations de franchissements *implicite* et *explicite*. On prend l'exemple du RdP de la figure 3.5.a. A l'expiration de la durée

d_1, la transition T_1 est franchie et le jeton passe de P_1 à P_2. Quel que soit le nombre de jetons dans P_1, ils vont tous franchir simultanément la transition T_1 pendant la durée d_1. Si l'on désire limiter le nombre de franchissements simultanés de la transition T_1 à 1, on ajoute la place P_3 marquée d'un jeton comme dans la figure 3.5.b. Une seule marque va franchir T_1 à chaque intervalle de temps d_1. En effet, on considère que lorsque la durée résiduelle associée à une transition est écoulée, celle-ci peut être franchie. Il s'agit là d'un fonctionnement à *vitesse maximale* (les transitions sont franchies dès que possible) différent d'un fonctionnement général où les transitions peuvent être franchies à un instant quelconque après l'écoulement du temps résiduel (à cause d'une procédure d'ordonnancement par exemple) [DA05]. Nous considérons dans tous nos modèles un fonctionnement à vitesse maximale. Lorsque l'on considère implicitement que le nombre de franchissements simultanés d'une transition est limitée à un, on parlera de limitation *implicite*. Lorsqu'on considère une place supplémentaire qui force la limitation comme c'est le cas sur la figure 3.5.b, on parlera de limitation *explicite*.

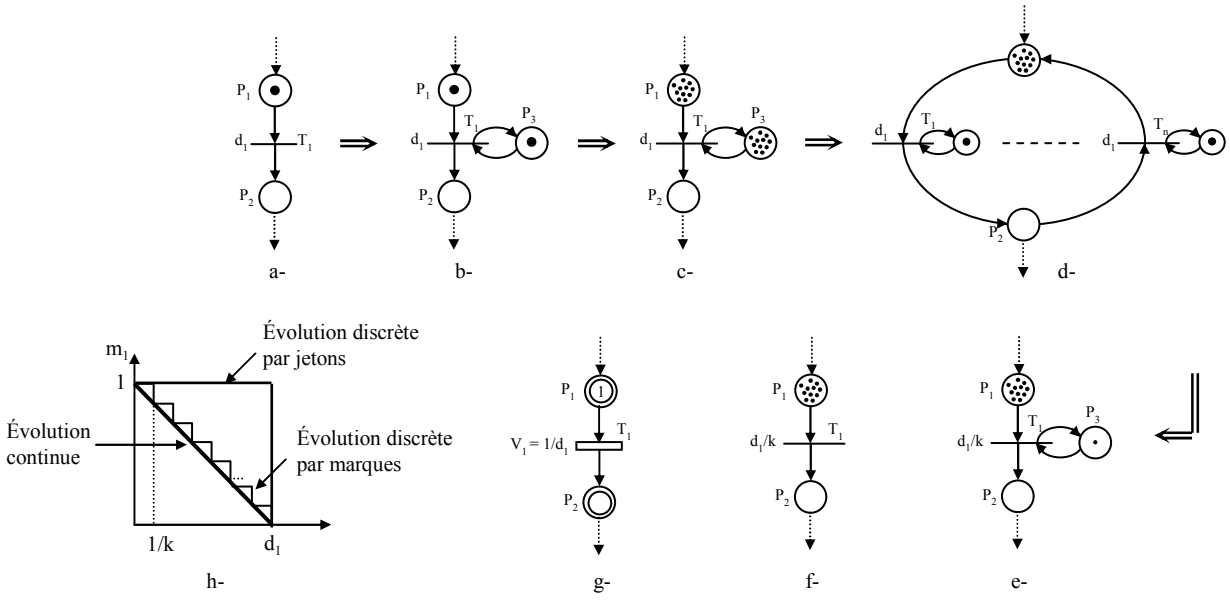


FIG. 3.5 – *a - e) Transformation d'un RdP discret temporisé en RdP continu temporisé. g) Evolutions du RdP temporisé discret et continu.*

Considérons la figure 3.5.b et divisons les jetons de P_1 et de P_2 en k marques, k étant un réel quelconque. On obtient le modèle de la figure 3.5.c. Ainsi, les k marques vont franchir T_1 simultanément et ce franchissement va prendre le temps d_1. Si on veut limiter le franchissement à une marque, on transforme le RdP comme indiqué sur la figure 3.5.d. La transition T_1 est répétée k fois avec une limitation explicite de chacune d'elle à une marque. Au lieu qu'il y ait un franchissement simultané de k marques en un temps d_1 (figure 3.5.c), il y aura k franchissements distincts d'une seule marque pendant le même délai d_1 (figure 3.5.d).
Une estimation de ce modèle (figure 3.5.d) nous amène au RdP de la figure 3.5.e. Cette estimation consiste à dire que k marques franchissant en même temps k transitions différentes (une marque par transition) en un délai d_1 est équivalent à k marques franchissant les unes à la suite

des autres une seule transition, chaque franchissement étant séparé du suivant d'un délai de d_1/k [DA92]. Le modèle de la figure 3.5.f est identique à la figure 3.5.e sauf que la limitation à une marque de la transition T_1 y est implicite.
Finalement, si l'on considère que k tend vers l'infini, on considère aussi que le temps d_1/k tend vers zero. Il est alors plus juste de considérer le franchissement de T_1 comme un débit continu et de parler non plus de temps mais de vitesse de franchissement. La vitesse associée à T_1 est donc $V = 1/d_1$ (exprimée en marques par unité de temps).
Afin de différencier les RdP continus de ceux discrets, nous représenterons dorénavant les places continues par un double cercle et les transitions continues par un rectangle. Le RdP continu équivalent au RdP de la figure 3.5.a est donné par la figure 3.5.g.

Remarque 4 *Prenons l'exemple d'un RdP temporisé discret qui comporte une transition T_i associée à un délai d_i. Si toutes les places en amont de T_i sont constamment marquées, alors le délai qui sépare deux franchissements successifs de T_i sera d_i. Autrement, ce délai est supérieur à d_i (fonctionnement à vitesse maximale).*
Considérons le même RdP, mais cette fois continu temporisé. Il sera associé à la transition T_i la vitesse V_i. Si toutes les places en amont de T_i possèdent un marquage toujours positif, alors la vitesse de T_i est égale V_i (sous réserve de présence d'autres contraintes comme le conflit par exemple), sinon elle sera inférieure à V_i. Il s'agit là de la vitesse instantanée $v_i(t)$ de T_i qui est au plus égale à V_i. *Toutefois, cette vitesse ne dépend pas de la valeur du marquage en amont, elle reste constante tant que le marquage n'est pas nul. Pour cela, ces RdP sont dits RdP continus à vitesses constantes* **RdPCC**.

Remarque 5 *Un premier calcul des durées présenté ci dessus nous a permis de passer de la figure 3.5.d à la figure 3.5.e. Il existe un second calcul qui conduit à un RdP continu où les vitesses maximales de franchissement sont fonctions des marquages des places continues. On parle alors de RdP continus à vitesses variables, les* **RdPCV** *[DA05].*

3.1.2.3 Le marquage 0^+

Dans les RdP discrets, un test à zéro est simple à réaliser car une place vide est une place qui contient moins d'un jeton. Un arc inhibiteur dont le poids est 1 permet ainsi d'effectuer ce test. Si l'on veut introduire la notion de test à zéro pour les RdP continus, le problème est autre. En effet, une transition continue est validée si sa place en amont possède un marquage réel strictement positif. Or, il n'y a pas de marquage réel positif au dessous duquel on peut dire qu'un marquage continu est nul. La notion du 0^+ a alors été introduite. Elle représente une *quantité infiniment petite mais non nulle* [DA01]. L'association du poids 0^+ à un arc inhibiteur permet d'effectuer un test à zéro d'une place continue (Figure 3.6).

Notation 0^+: Un nombre α/k, où α est un nombre positif fini et $k \rightarrow \infty$, est noté 0^+ ([DA05]).

Il s'ensuit que $0 < 0^+$ et qu'il n'existe pas de réel positif x tel que $x < 0^+$.
Le nombre 0^+ est un nombre non standard ([Rob66]). Ceux-ci sont notés ${}^*\Re$. Dans ce qui suit,

tout nombre $x \in^* \Re$ est noté ${}^*x$ et tout vecteur de marquage contenant un nombre non standard sera noté $\tilde{M}$.

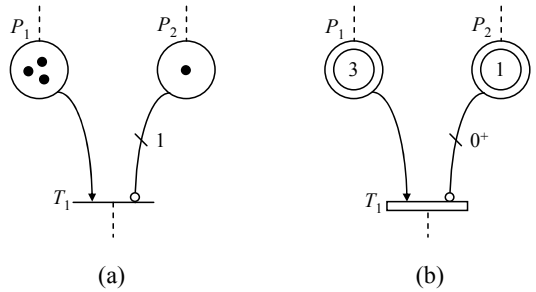


FIG. 3.6 – *Un test à zéro pour a) Un RdP discret temporisé, b) Le RdP continu équivalent.*

3.1.2.4 Quelques règles et définitions des RdP continus temporisés

Soit un RdP R continu à vitesses constantes.

DÉFINITION 2 *Une place P_i de R est* **alimentée** *s'il y a au moins une transition T_j parmi les transitions d'entrées de P_i qui est validée.*
De façon formelle,P_i est alimentée si $(\sum_{T_j \in {}^\circ P_i} Post(P_i, T_j)v_j) > 0$.

DÉFINITION 3 *Une place T_j est* **validée** *si chaque place P_i d'entrée de T_j satisfait au moins une des deux conditions suivantes:*

1. $m_i(t) > 0$
2. P_i *est alimentée*

Remarque 6 *Une transition est* fortement validée *si toutes ses places d'entrées satisfont la première condition, autrement elle est* faiblement validée.

Un algorithme est défini dans [DA05] pour déterminer l'ensemble des transitions validées d'un RdP continu.

Vitesse de franchissement d'une transition Notons:
I_i = Pré$(P_i) = \sum_{T_k \in {}^\circ P_i} Post(P_i,T_k) \cdot v_k$ = vitesse d'alimentation de la place P_i
et O_i = Post$(P_i) = \sum_{T_k \in P_i^\circ}$ Pré$(P_i,T_k) \cdot v_k$ = vitesse de vidage de la place P_i.

L'écart I_i - O_i, noté B_i représente le bilan dynamique de la place P_i. B_i peut être positif, négatif ou nul. Cependant, prenons un intervalle [a, b] où m_i reste égal à zéro. D'une part, si $m_i = 0$, B_i ne peut être négatif. D'autre part, si m_i reste nul tout au long de cet intervalle, cela signifie que B_i n'est pas positif. Ainsi, m_i constant égal à zéro sur un intervalle donné signifie que B_i

ne peut être que nul durant ce même intervalle.

Propriété 1 *La vitesse de franchissement d'une transition* T_j fortement validée *est égale à sa vitesse maximale de franchissement* $v_j(t) = V_j$.

Propriété 2 *La vitesse de franchissement d'une transition* T_j faiblement validée *est donnée par la relation:* $v_j(t) = min[V_j, min_i(B_i + v_j(t))]$
avec i *tel que* P_i *est une place d'entrée de* T_j *et* $m_i(t) = 0$.

Propriété 3 *Sur un intervalle [a, b] correspondant à un IB-état, le marquage d'une place* P_i *à la borne supérieure b est égal à* $\tilde{m}_i(b) = 0^+$ *(*$\tilde{m}$ *signifie que le marquage est un nombre non standard, ici* 0^+*) si et seulement si:*

- $\tilde{m}_i(t) = 0^+, \forall t \in [a, b]$, *ou*
- $m_i(t)$ *décroît jusqu'à 0 ET* $I_i(t) > 0\ \forall t \in [a, b]$.

Propriété 4 *Pour un intervalle [a, b] correspondant à un IB-état, le marquage d'une place* P_i *est égal à* $\tilde{m}_i(t) = 0^+, \forall t \in [a, b]$ *si et seulement si:*

- $m_i(t) = 0$, $I_i(t) > 0$ *et* P_i *n'est pas vidée par une transition immédiate*[1], *ou*
- $\tilde{m}_i(a) = 0^+$ *et aucune transition de* P_i° *n'est validée à l'instant* a.

L'expression des vitesses de franchissement dans quatre RdP différents est montré sur la figure 3.7. De manière générale, les vitesses de franchissement sont calculées à partir d'un algorithme défini dans [DA05].

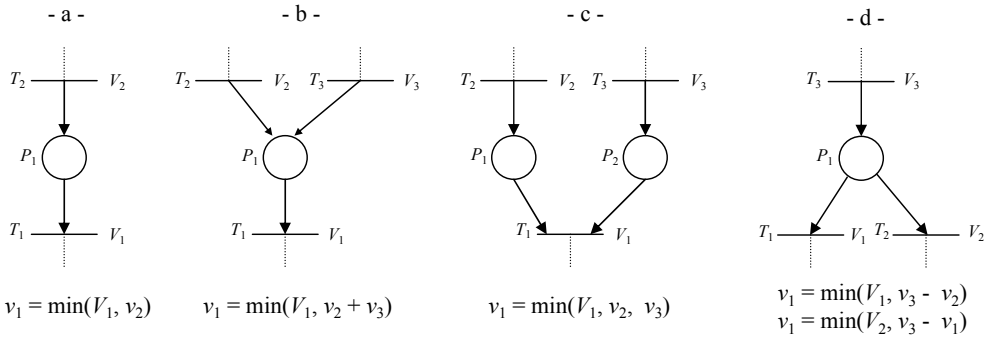


FIG. 3.7 – *Calcul des vitesses de franchissement.*

1. Une transition immédiate est une transition franchie *q fois à partir du moment où elle est q-validée*, voir définition détaillée dans ([DA05], p.67)

3.1.2.5 Les conflits dans un RdP continu

De façon informelle, on parle de conflit dans un RdP continu lorsqu'une place P_i de ce RdP possède un débit d'entrée I_i qui n'est pas suffisant pour franchir les transitions de sortie de cette même place à leurs vitesses maximales.

DÉFINITION 4 *Soit $Q_j(t)$ le sous ensemble des places P_h dans $^\circ T_j$ tel que $m_h(t) = 0$. Étant donné un marquage M et un débit d'alimentation $I_i(t)$, On dit qu'il y a un conflit par rapport à la place P_i à l'instant t si:*

1. $m_i(t) = 0$, *et*
2. $I_i(t) < \sum_{P_h \in Q_j(t)} min(\frac{1}{Pre(P_h,\, T_j)} \sum_{T_k \in {}^\circ P_i} Post(P_h, T_k) \cdot v_k(t), V_j)$

Pour résoudre un tel conflit, nous avons le choix entre deux solutions: la *priorité* ou le *partage*.

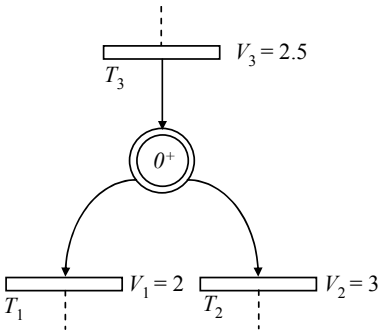


FIG. 3.8 – *Cas d'un conflit dans un RdP continu temporisé.*

Prenons l'exemple de la figure 3.8. Les vitesses instantanées associées à T_1 et T_2 sont respectivement $v_1 \leq min(V_1,v_3) = 2$, $v_2 \leq min(V_2,v_3) = 2.5$ et $v_3 < V_1 + V_2$. Ainsi, il faut trouver les valeurs adéquates pour v_1 et v_2 telles que:

$v_1 \leq 2$,
$v_2 \leq 2.5$,
et $v_1 + v_2 = 2.5$,

Si la solution choisie est celle de la priorité et en prenant, par exemple, T_1 prioritaire à T_2 (noté $T_1 < T_2$), alors on prendra $v_1 = 2$ et $v_2 = 0.5$.
Si la solution choisie est celle du partage, on posera $v_1 = \alpha v_2$, α fixé selon le poids que l'on désire accorder à chacune des deux vitesses. Ces notions de conflits et de priorités seront reprises dans les sections suivantes.

3.1.2.6 Fonctionnement d'un RdP à vitesses constantes: RdPCC

Présentation intuitive

Nous allons d'abord présenter de manière intuitive le fonctionnement d'un RdPCC. L'exemple du système à réservoir de la figure 3.9.a comprend deux réservoirs et une pompe. Le réservoir 1 se vide dans le second par l'effet de la gravité à la vitesse V_1. Le réservoir 2 se vide à la vitesse

V_2 grâce à une pompe P qui a son tour remplit le réservoir 1 et ainsi de suite.

Ce système est continu et peut être représenté par le RdP continu de la figure 3.9.b. Deux places continues représentent les réservoirs 1 et 2. Le marquage de ses places indique la quantité d'eau initiale présente dans les réservoirs. Les vitesses V_1 et V_2 sont associées aux transitions T_1 et T_2. Dans un premier temps, la place P_1 se remplit à la vitesse V_2 et se vide à V_1. La place P_2 a exactement la dynamique opposée. Il en découle les équations suivantes:

$$\dot{m_1} = V_2 - V_1; \tag{3.6}$$
$$\dot{m_2} = V_1 - V_2. \tag{3.7}$$

et

$$m_1(t + \mathsf{d}t) = m_1(t) + (V_2 - V_1)\mathsf{d}t; \tag{3.8}$$
$$m_2(t + \mathsf{d}t) = m_2(t) + (V_1 - V_2)\mathsf{d}t. \tag{3.9}$$

m_1 et m_2 étant les marquages des places P_1 et P_2 respectivement.

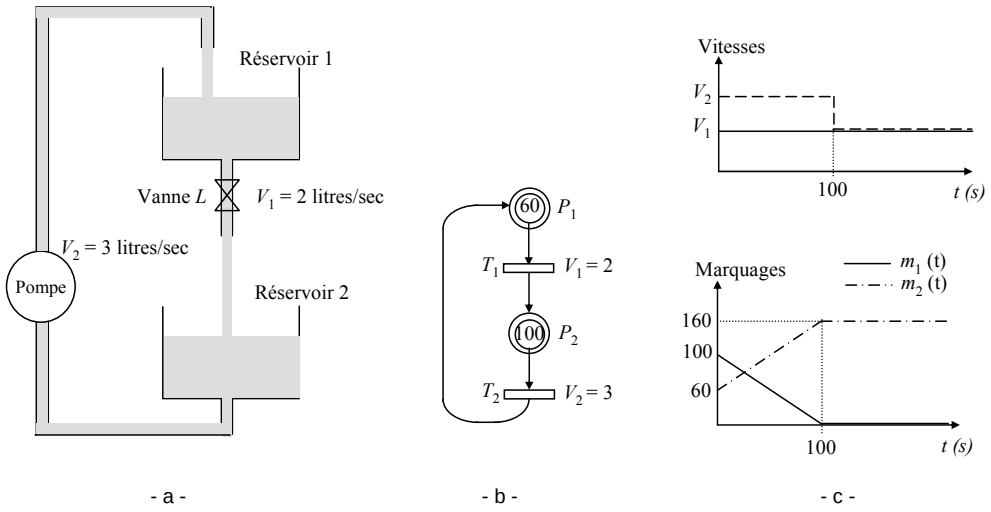


FIG. 3.9 – *a) Système à réservoir d'eau. b) RdP continu équivalent. c) Dynamique des marquages et évolution des vitesses.*

Les vitesses $v_1(t)$ et $v_2(t)$ représentent les vitesses instantanées de franchissement des transitions T_1 et T_2. Elles ont pour valeurs maximales V_1 et V_2. Les vitesses associées aux transitions d'un RdP continu sont les vitesses maximales (figure 3.9.b).

D'une part, selon que V_1 est supérieure à V_2 ou inversement, un des marquages, m_1 ou m_2 va s'annuler (équations 3.8 et 3.9). D'autre part, Un marquage ne peut pas être négatif. Il s'ensuit qu'*au moment du passage d'un marquage à zéro, les dynamiques vont changer*.

Prenons V_1 = 2 litre/sec, V_2 = 3 litre/sec, $m_1(0)$ = 60 litres et $m_2(0)$ = 100 litres:

$$m_1(t) = 60 + t; \tag{3.10}$$
$$m_2(t) = 100 - t. \tag{3.11}$$

À $t = 100sec$, $m_1(t)$ = 0 et $m_2(t)$ = 160 litres (figure 3.9.c).
Si le réservoir 1 (représenté par m_1) est vide, alors il ne pourra transmettre de l'eau au réservoir 2 que s'il en reçoit de ce dernier par la pompe P. De façon générale, le débit V_1 du réservoir 1 s'aligne sur celui de la pompe.
À partir de $t = 100sec$, les vitesses étant désormais égales, la dynamique des deux marquages est nulle ($v_2 - v_1$ = 0) et les marquages restent constants (figure 3.9.c).

Pour t < 100 sec, $m_2(t)$> 0. La transition T_2 est franchie parce que sa place en amont est marquée. Sa vitesse de franchissement est maximale, on dit que T_2 est *fortement validée*. Son franchissement induit le passage de la quantité V_2dt (pour un intervalle de franchissement égal à dt).

À $t \geq 100$sec, $m_2(t)$ = 0. La transition T_2 continue d'être franchie car sa place est amont P_2 est alimentée par T_1. La transition T_2 est franchie à la vitesse V_1 ($v_2(t) = V_1$) qui est inférieure à sa vitesse maximale de franchissement (V_2). On dit que T_2 est *faiblement validée*.

Le comportement du système à réservoir possède donc 2 phases distinctes. Une première phase pour $t \in [0,100]$ où les vitesses sont constantes égales à (v_1,v_2) = (V_1,V_2) et où l'évolution des marquages se fait selon les équations 3.10 et 3.11. Une deuxième phase où le vecteur de vitesses est également constant à (v_1,v_2) = (V_1,V_1), la dynamique des marquages devenant alors nulle.
Les vitesses sont constantes par intervalle de temps, elles changent lors du passage d'un marquage à zéro. La dynamique des marquages de ce système est donc **constant par morceaux**.

3.1.2.7 Graphe d'évolution d'un RdPCC

Le nombre de marquages d'une place étant infini dans les RdP continus, on ne peut pas établir de graphe de marquages. Cependant, les vecteurs vitesses instantanés de franchissement restent constants dans des intervalles donnés (fonctionnement constant par morceaux). Lors du passage d'un marquage à zéro, les vitesses du RdPCC changent. De ce fait, nous pouvons nous baser sur les vecteurs de franchissement ainsi que sur les vecteurs de marquages du RdP à chaque changement de dynamique pour définir complètement l'évolution de ce RdP. On construit donc un graphe d'évolution comme un RdP où chaque place représente un **IB-state** (ou **IB-état**) et où à chaque transition est associé l'événement qui provoque le changement de l'état des vitesses (un changement de marquage). Un **IB-état** (ou état comportemental invariant, IB-state en anglais pour: *Invariant Behavior state*) est une phase d'évolution du RdP où , *le vecteur de vitesses instantanées reste constant*. Autrement dit, tant qu'on est dans un même IB-état, les marquages évoluent de façon linéaire.
Prenons l'exemple du RdP de la figure 3.9 et déterminons le graphe d'évolution correspondant. Le marquage initial est $M_0 = (60,100)$ et à cet instant $v = (V_1, V_2) = (2, 3)$ (figure 3.10).

Ce vecteur vitesse restera constant jusqu'à l'instant $t_1 = 100s$ où m_2 *atteint la valeur zéro* (noté $m_2 = 0$ devant la transition correspondante). L'instant d'occurrence de cet événement ($t_1 = 100s$) est noté après la barre /. Ce marquage nul va entraîner un changement du vecteur vitesses en $v = (V_1, V_1) = (2, 2)$. Lorsque tous les bilans dynamiques B_i sont positifs ou nuls, les marquages ne peuvent plus s'annuler et le graphe d'évolution est obtenu [2].

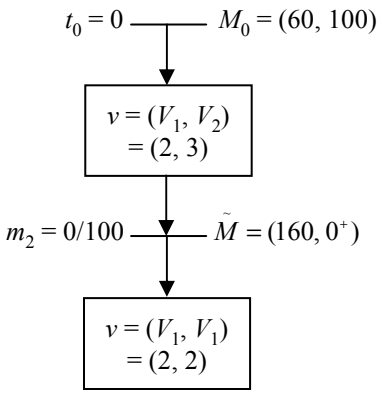


FIG. 3.10 – *Graphe d'évolution d'un RdPCC.*

3.1.3 Les RdP hybrides

Un RdP hybride contient un ensemble non vide de places et de transitions continues et un ensemble non vide de places et de transitions discrètes.
Un RdP hybride est dit autonome si sa partie continue est autonome (quantité de franchissement τ des transitions, voir paragraphe 3.1.2.1) et sa partie discrète est non temporisée. Il sera dit temporisé sinon. La modélisation du temps dans les RdP hybrides est donnée en détail dans [DA01] et [DA05]. Elle conduit à considérer des vitesses de franchissement sur les transitions continues (C-transitions) et des délais de franchissement sur les transitions discrètes (D-transitions).

Définition formelle des RdP hybrides Un RdP hybride autonome marqué est un sextuple $< P, T, Pr, Post, M_0, h >$ tel que:

- $P = \{P_1, P_2, ..., P_n\}$ est un ensemble fini non vide de places.
- $T = T_1, T_2, ..., T_m$ est un ensemble fini non vide de transitions.
- $P \cap T = \varnothing$ i.e., les ensembles de places et de transitions sont disjoints.
- h: $P \cup T \rightarrowtail \{D,C\}$ est une application dite *fonction hybride*. Elle indique pour chaque noeud du RdP s'il s'agit d'un noeud discret (P^D, T^D) ou continu (P^C, T^C).
- Pré: $P \times T \rightarrow Q_+$ (si $P_i \in P^C$) ou $\aleph$ (si $P_i \in P^D$) est l'application d'incidence avant.
- Post: $P \times T \rightarrow Q_+$ (si $P_i \in P^C$) ou $\aleph$ (si $P_i \in P^D$) est l'application d'incidence arrière.
- $M_0 : P \rightarrow \Re_+$ (si $P_i \in P^C$) or $\aleph$ (si $P_i \in P^D$) est le marquage initial.

2. On peut construire un graphe d'évolution si le RdP correspondant est déterministe, autrement, il faudra d'abord régler les conflits. Voir [DA05]

– Pré et Post doivent vérifier le critère:
si P_i et T_j telles que $h(P_i)$ = D et $h(T_j)$ = C, alors Pré(P_i,T_j) = Post(P_i,T_j).
Cette condition nous permet de garantir que le marquage d'une D-place reste entier quelque soit l'évolution du RdP hybride.

Un RdP hybride temporisé est un couple <H, Tempo> tel que H est un RdP hybride autonome marqué (définition ci-dessus) et Tempo une application qui associe à chaque transition un nombre rationnel positif.
- Si $T_j \in T^D$, alors Tempo(T_j) = d_j, d_j étant un temps associé à T_j.
- Si $T_j \in T^C$, alors V_j = 1/Tempo(T_j) = $1/d_j$, V_j étant la vitesse maximale de franchissement de T_j.

DÉFINITION 5 *Une* **C-transition** T_j *est* **validée** *si chacune de ses places d'entrée* P_i *satisfait les conditions suivantes:*
Si P_i *est une place discrète:*
$m_i \geq Pre(P_i,T_j)$
Si P_i *est une place continue:*
Soit $m_i > 0$,
ou bien P_i *est alimentée, i.e., il existe au moins une transition d'entrée de* P_i *dont la vitesse de franchissement est non nulle.*

Une C-transition T_j *est dite* fortement validée *si* $m_i > 0$ *pour chacune de ses places d'entrées* P_i*, sinon elle est dite* faiblement validée.

DÉFINITION 6 *On dit d'une* **D-transition** T_j *qu'elle est* **validée** *si chacune de ses places d'entrées* P_i *satisfait la condition:*
$m_i \geq Pre(P_i,T_j)$.

DÉFINITION 7 *Le* **D-degré de validation** *d'une transition continue* T_j *pour un marquage* M *est donné par:*

$$D(T_j, M) = min_{P_i \in T_j \cap P^D} \lfloor m_i / Pre(P_i, T_j) \rfloor. \tag{3.12}$$

Dans le cas particulier où tous les arcs reliant les D-places aux C-transitions sont de poids un, alors:

$$D(T_j, M) = min_{P_i \in T_j \cap P^D} m_i. \tag{3.13}$$

Dynamiques d'un RdP hybride
Reprenons l'exemple de la figure 3.9. On suppose que la capacité maximum du réservoir 1 est de 150 litres. Lorsque le réservoir 1 atteint ce niveau d'eau, la pompe s'arrête automatiquement pendant une durée de 100 secondes puis reprend de nouveau. Pour représenter le système sous cette nouvelle contrainte, il est nécessaire de représenter l'état de la pompe. Celle-ci possède deux états discrets: un état de marche et un état d'arrêt. Elle sera donc représentée par des places discrètes. Le flux d'eau dans les réservoirs est continu et est représenté comme précédemment

dans la figure 3.9.b. Ces deux parties interagissent et forment un modèle hybride. Ce modèle est donné par le RdP de la figure 3.11.a.

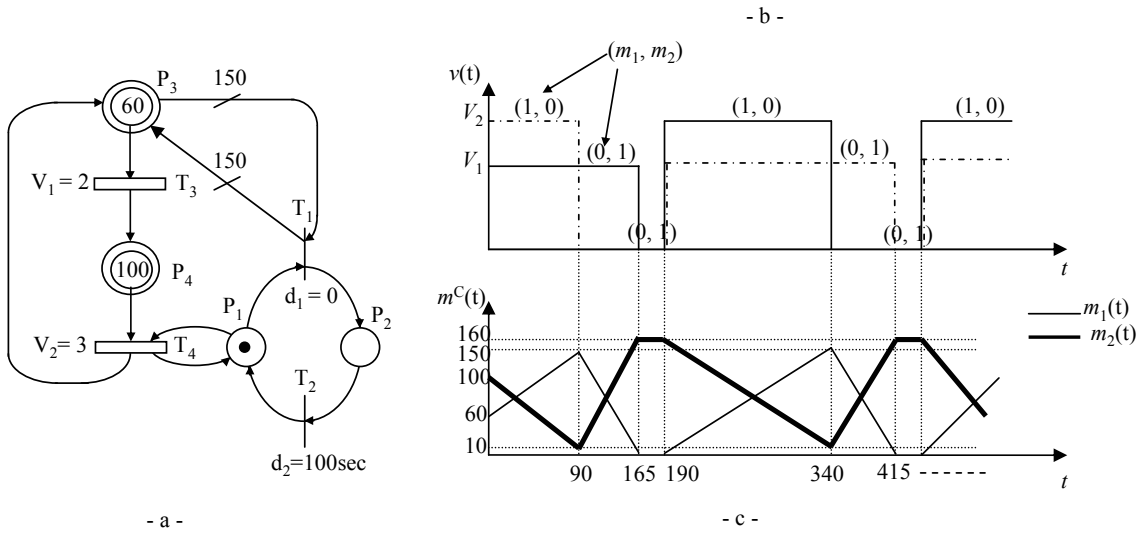


FIG. 3.11 – *Dynamique d'un RdP hybride.*

Les places P_1 et P_2 de la figure 3.11.a représentent les états de marche et d'arrêt de la pompe respectivement. La partie continue P_3 - T_3 - P_4 - T_4 reste telle qu'elle était donnée sur la figure 3.9.b. A l'état initial, la pompe est en état de marche. La transition T_1 représente le passage de la pompe de l'état de marche à l'état d'arrêt. Ce passage est conditionné par le niveau d'eau du réservoir 1 représenté par le poids de l'arc $P_3 \rightarrow T_1$. La transition T_2 représente le passage de l'état d'arrêt à l'état de marche. À la transition T_1 est associée un temps de franchissement de durée nulle (d_1 = 0); la transition est immédiate. À la transition T_2 est associée un temps de franchissement de durée d_2 = 100. Cette durée correspond à la durée d'arrêt de la pompe. Lorsque P_2 est marquée (pompe à l'arrêt), on commence à compter le temps (voir section 3.1.1), une fois la durée d_2 écoulée la transition T_2 est franchie.

La partie P_1 - T_4 représente l'effet du discret sur le continu. En effet, tant que P_1 n'est pas marquée, la transition T_4 ne peut être franchie (règle de franchissement des RdP discrets).
Par contre, en un temps dt, le franchissement de T_4 va induire le passage d'une quantité V_2dt de chacune des places P_1 et P_4. L'arc ajouté de T_4 vers P_1 garantit qu'à chaque franchissement de T_4, la même quantité V_2dt est enlevée puis ajoutée à P_1 de telle sorte que le jeton présent dans P_1 reste entier (voir les conditions sur fonctions Pré et Post d'un RdP hybride). En d'autre termes, le franchissement d'une C-transition ne doit pas modifier le marquage d'une D-place.

La partie P_3 - T_1 représente l'effet du continu sur le discret. En effet, tant que le marquage de P_3 est inférieur à 150, la transition T_1 n'est pas franchissable. Lorsque m_3 atteint 150, T_1 est franchie et c'est P_2 qui devient marquée (pompe à l'arrêt).

Dynamique des marquages et vitesses de franchissement Le marquage à l'état initial du

RdP de la figure 3.11.a est (1, 0, 60, 100). Les marquages m_3 et m_4 sont positifs donc T_3 et T_4 sont fortement validées (Définition 1) et la dynamique des marquages m_3 et m_4 est identique à celle donnée par les équations 3.10 et 3.11. Les transitions T_1 et T_2 ne sont pas franchissables donc les marquages de m_1 et m_2 ne changent pas. Les marquages m_3 et m_4 évoluent selon les courbes de la figure 3.11.c
A t = 90 sec (figures 3.11.a et 3.11.b), m_3 = 150 et T_1 est franchie car $d_1 = 0$. La place P_1 n'est plus marquée et de ce fait, T_4 n'étant plus validée, sa vitesse de franchissement s'annule ($v_2(t)$ = 0). La nouvelle dynamique des marquages est:

$$\dot{m}_3 = -V_1 = -2litres/sec, \tag{3.14}$$

$$\dot{m}_4 = V_1 = 2litres/sec. \tag{3.15}$$

et l'évolution des marquages est donnée par :

$$m_3(t) = 150 - 2t, \tag{3.16}$$

$$m_4(t) = 10 + 2t. \tag{3.17}$$

A partir de t = 90sec, le marquage de la place P_3 se vide jusqu'à atteindre zéro à t = 165 secondes. La transition T_3 n'est plus validée car P_3 n'est plus alimentée (voir la définition 1), il s'ensuit que $v_1(t)$ = 0 . Ainsi, les deux vitesses de franchissement $v_1(t)$ et $v_2(t)$ sont nulles et les marquages restent constants.
A t = 190 sec, T_2 est franchie (la pompe reprend après 100 secondes d'arrêt). Le marquage m_4 est positif, donc T_4 est fortement validée et $v_2(t) = V_2$ = 3 litre/sec.
Le marquage m_3 devient également positif. Ainsi, T_3 est fortement validée et $v_1(t) = V_1$ = 2 litre/sec. La dynamique du système continue selon la figure 3.11.c.

Graphe d'évolution d'un RdP hybride Comme nous l'avons vu précédemment dans la section 3.1.2.6, un RdPCC peut être défini par son marquage et les vecteurs vitesses instantanées de franchissement. Ce qui définit, dans les RdP hybrides, un état comportemental invariant, c'est l'IB-state (Invariant Behavior state). Un IB-state (ou IB-état) représente des phases où l'évolution des marquages du RdP hybride est constante et continue.
De manière formelle, un IB-état correspond à un intervalle de temps dans lequel:

1. le marquage M^D des D-places est constant,
2. Le vecteur de validation e^D des D-transitions est constant.
3. le vecteur vitesses instantanées des C-transitions est constant,
4. le vecteur des marques réservées à la fois dans les D-places et les C-places est constant.
5. Lorsqu'un IB-état est atteint, le marquage des places continus M^C est constant.

Un graphe d'évolution d'un RdP hybride est donc constitué d'une succession d'IB-états représentés par des noeuds (figure 3.12). On passe d'un noeud à un autre par l'occurrence d'un des événements suivants:

1. Le marquage d'une C-place s'annule (appelé C1-événement),
2. Le franchissement d'une D-transition (appelé D1-événement),
3. Le degré de validation d'une D-transition qui change à cause du marquage d'une C-place (appelé D2-événement).

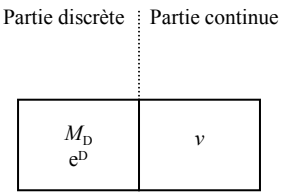


FIG. 3.12 – *Représentation d'un IB-état.*

Chaque noeud comprend deux parties; une discrète et une continue. Dans la partie discrète sont indiqués le marquage total et le vecteur de validation des D-places, notés respectivement M_D et e^D. Dans la partie continue est indiqué le vecteur vitesses instantanées (v).
Afin de mieux comprendre la construction d'un graphe d'évolution d'un RdP hybride et les IB-états, nous allons illustrer ces notions à travers l'exemple de la figure 3.11. Le graphe d'évolution obtenu correspond à celui de la figure 3.13. A l'instant initial, le marquage des places continues est M_C = (60 100), celui des places discrètes est M_D = (1 0) et les vitesses sont maximales ($v = (2,3)$). A t = 90s, le marquage de P_3 atteint 150, $m_3 = 150$. T_1 peut être franchie rendant P_1 non marquée et T_4 infranchissable ($v_2(t) = 0$). Le vecteur vitesse change ($v = (2,0)$) ainsi que l'IB-état. Sur la transition de passage de l'IB1 à l'IB2 est indiqué l'événement qui a engendré ce changement de phase. L'écriture $m_3 = 150/90$ signifie que 90s après le début de l'IB1, le marquage de P_3 a atteint 150. Après un délai de 75s passé dans l'IB2, le marquage de P_3 se vide ($m_3 = 0$). On entre alors dans l'IB3 et ainsi de suite jusqu'à construire tout le graphe d'évolution.

3.2 Le modèle d'une ligne de communication TCP/IP

Après avoir présenté l'outil de modélisation RdP hybride, nous allons passer à la partie modélisation d'une ligne de communication TCP/IP dans un environnement Internet.
Une ligne de communication comprend généralement une émetteur, un récepteur et un certain nombre de routeurs intermédiaires. Nous allons étudier l'exemple d'une transmission de données à travers une ligne de communication composée d'un émetteur E et d'un récepteur R reliés par un routeur Rt et deux canaux de transmission ayant les débits respectifs V_1 et V_2. Le routeur possède un buffer de capacité maximale que l'on note C. Au delà de cette valeur, le buffer ne peut plus recevoir les données et celles-ci sont perdues. La composition globale de cette ligne de communication considérée est donnée par la figure 3.14.

Deux parties distinctes vont apparaître dans la modélisation d'une telle ligne. D'une part, les données (succession de bits) sont transmises à des débits généralement très élevés (de l'ordre du mega bits / secondes). Ce rapport débit/taille de données transmises nous permet de considérer qu'un modèle continu peut représenter une très bonne approximation de ce système de nature discrète ([DF82] et [GS80]). Ainsi, il ne s'agit plus de représenter un passage discret de bits (ou d'octets) mais l'émission d'un flux continu d'un ordinateur 1 vers un ordinateur 2 [BA03].
D'autre part, la transmission est régie par les protocoles de contrôle de flux du TCP/IP. Protocoles qui, comme on l'a vu au chapitre 2, gèrent la transmission des données par un ensemble de consignes discrètes qui dépendent essentiellement de la taille des données transmises, de

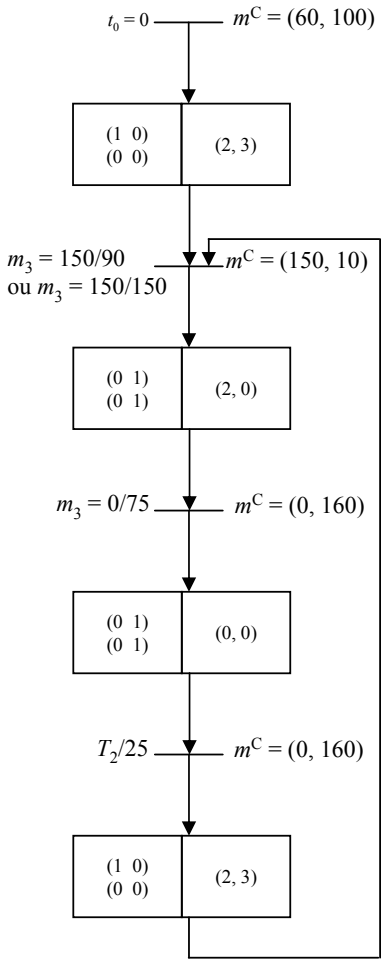


FIG. 3.13 – *Graphe d'évolution d'un RdP hybride.*

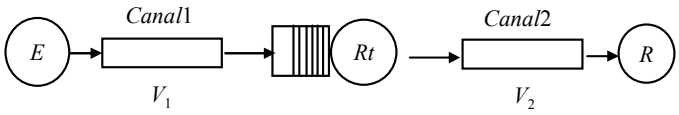


FIG. 3.14 – *Composition globale de la ligne de communication considérée.*

l'arrivée ou non d'acquittements, de temporisateurs. Ces consignes discrètes prennent leurs valeurs selon l'état du flux continu présent sur la ligne. De même, la dynamique de ce flux dépend étroitement des consignes des protocoles. Nous sommes donc en présence de deux dynamiques, l'une discrète et l'autre continue, les deux interagissant entre elles pour donner une dynamique continue ponctuellement modifiée par une dynamique discrète. Ceci est la définition même d'un système hybride continu-discret.

L'outil de modélisation choisi est les RdP hybrides car outre la partie discrète des réseaux que l'on modélise aisément par les RdP discrets, nous allons voir que la partie continue des réseaux de communication est tout à fait assimilable à des systèmes à flux continus déjà parfaitement modélisés par des RdP continus [BA03].

3.2.1 Analogie systèmes à flux continus - réseaux de communication

Prenons l'exemple du système à réservoir de la figure 3.15.a. Ce système est composé d'un réservoir qui peut être alimenté par deux sources différentes. Une première source, naturelle (pluie) et une seconde par les nappes phréatiques grâce à une pompe P. Dans le système tel que décrit par [Méz94], le niveau du réservoir doit être compris entre deux hauteurs h_1 et h_2, mais cette information n'étant pas utile en ce qui nous concerne, nous considérons un modèle simplifié où il n'y a pas de contraintes sur le niveau du réservoir.
La quantité d'eau provenant de la source naturelle est complètement aléatoire car elle dépend de la saison et de la météorologie. Celle provenant des nappes phréatiques, par contre est contrôlée par l'action de la pompe. Le réservoir est utilisé pour alimenter des consommateurs dont la demande fluctue dans le temps.
Le modèle RdP du système à réservoir est donné par la figure 3.15.b. Le réservoir est représenté par la place continue P_1 dont le marquage indique la quantité d'eau présente dans le réservoir. Deux transitions continues T_1 et T_2 à l'entrée de P_1 représentent les deux débits d'eau par les sources naturelle et phréatiques respectivement. L'arrivée naturelle d'eau étant aléatoire, nous représentons la vitesse maximale de franchissement associée à T_2 par une fonction quelconque dans le temps comme donné dans la figure 3.15.b. Le marquage discret de P_2 ou P_3 indique que la pompe est ouverte ou fermée respectivement rendant ou pas la transition T_2 franchissable ou pas. Une transition continue en sortie de P_1, T_3, représente la consommation d'eau. Celle ci étant également inconnue, la vitesse maximale de franchissement de T_3 est donnée par une fonction du temps (figure 3.15.b).

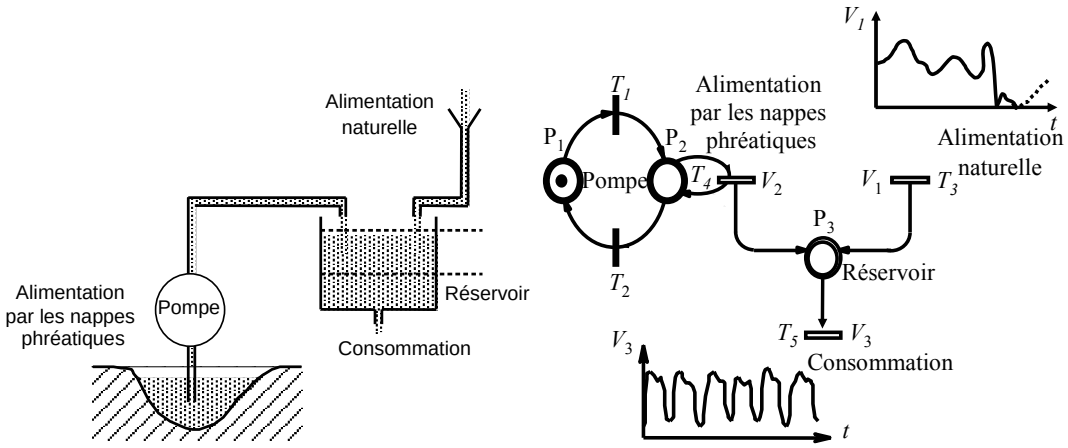


FIG. 3.15 – *a) Système à réservoir d'eau. b) Modèle RdP hybride équivalent.*

Revenons aux réseaux de communication. Ceux-ci sont composés de canaux de transmissions, de buffers et évidemment d'émetteurs/récepteurs. Par analogie entre le système à flux continu décrit ci dessus et les réseaux, la place continue P_3 représente le buffer d'un réseau de communication, la C-transition d'entrée T_3 représente une source émettrice vers ce réseau et la C-transition de sortie T_5 représente une source réceptrice. Les vitesses associées à ces deux transitions restent aléatoires, les arrivées et les capacités de sorties des réseaux n'étant pas connues au préalable. Ce qui représentait auparavant un flux d'eau dans le réservoir (V_1 et V_3)

représente dorénavant un flux de messages dans le réseau. On voit très bien qu'une transmission de message via un réseau est très semblable à l'écoulement d'un flux à travers un système à réservoirs. Ces deux phénomènes étant semblables et les systèmes à flux continus étant très bien modélisés par les RdP continus, nous pouvons conclure que les réseaux de communication sont parfaitement modélisable par RdP continus aussi [BA03]. Nous allons voir dans ce qui suit que cette conclusion est vérifiée.

3.2.2 Le modèle *continu* d'une ligne de communication TCP/IP

La partie continue représente le passage continu de flux de messages et ne tient pas compte des protocoles de transmission. Le modèle de la ligne sera donc représenté par un RdP continu. Afin d'introduire les idées de façon progressive, nous allons modéliser la ligne partie par partie en commençant par le routeur ([BA03], [BA05b]).

3.2.2.1 Le modèle du routeur

Un routeur se comporte comme un réservoir à capacité limitée. Il reçoit des données en entrée et leur trouve une ligne de sortie. Sur le routeur de la figure 3.14, les données arrivent à la vitesse V_1 et sortent à la vitesse V_2. Ce flux de bits qui arrive et qui sort à des débits importants est approximé à un flux continu. Ainsi, le modèle du routeur est donné par le RdP continu de la figure 3.16.a. La place P_1 représente la charge du routeur, la place complémentaire P_2 représente l'espace libre restant dans le buffer. La somme m_1 + m_2 reste toujours constante égale à la capacité maximale du routeur C. C'est un invariant de marquage.

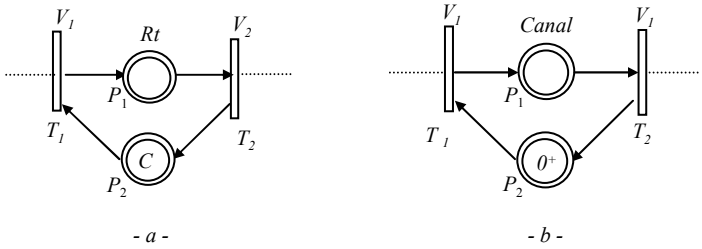


FIG. 3.16 – *a) Modélisation d'un routeur par RdP continu. b) Modélisation d'un canal de transmission par RdP continu.*

3.2.2.2 Le modèle du canal de transmission

Un canal de transmission transmet des données mais ne les stocke pas. Il se comporte comme un réservoir à capacité nulle. On pourrait, pour cela, reprendre le modèle RdP du routeur et l'appliquer au canal, la capacité nulle du canal étant représentée par un marquage nul de la place complémentaire P_2. Cependant, si la somme des marquages de P_1 et P_2 est nulle, aucune des transitions T_1 ou T_2 n'est franchissable. Pour modéliser une capacité de stockage nulle, on va utiliser un nouveau concept, celui du 0^+, introduit par David et Alla dans [DA01]. Un marquage 0^+ associé à une place continue indique une *quantité infiniment petite mais non nulle*. Un tel marquage permet alors non seulement de valider chacune des deux transitions T_1 et

T_2 mais aussi d'assurer que la vitesse d'entrée du canal est toujours égale à sa vitesse de sortie. Nous allons voir l'utilité de cette dernière remarque dans les paragraphes suivants.
On représente le canal 1 de la figure 3.14 par la place P_1 de la figure 3.16.b. La vitesse d'entrée du canal est égale à sa vitesse de sortie. Les deux vitesses associées à T_1 et T_2 sont donc identiques égales à V_1. La place P_2 représente la place complémentaire du canal. De même que pour le routeur, la somme $m_1 + m_2$ doit rester constante égale à la capacité maximale du canal, 0^+. Chacun des canaux 1 et 2 de la figure 3.14 peut être représenté par le RdP de la figure 3.16.b à la différence près que la vitesse V_1 est associée aux transitions du canal 1 et la vitesse V_2 au canal 2.

Délai de propagation Cependant, il arrive que les canaux aient des temps de transmissions qui ne sont pas négligeables. Auquel cas, on considère que les données *restent* un certains temps dans le canal. Voyons ce qui se passe dans ce cas là. Les données entrent dans le canal à la vitesse qui est le débit du canal et mettent un temps égal au temps de propagation pour en sortir. Ce comportement peut être modélisé comme sur la figure 3.17.b. Prenons l'exemple d'un canal dont le débit est de V = 50kbits/s avec un délai de propagation de $t_{propa} = d = 250ms$ (figure 3.17.a). Les données entrent et sortent du canal à la vitesse V (figure 3.17.b). La transition discrète temporisée T_3 modélise le délai d'attente dans le canal. Un poids de 0^+ est associé aux arcs $P_1 - T_3$ et $T_3 - P_2$ signifiant que dés qu'une petite quantité de données entre dans le canal, celle ci valide le franchissement de la transition T_3 et attend une durée de propagation d avant d'être disponible à la sortie du canal. Enfin, la place P_3 indique la *capacité* maximale du canal, i.e., $V.d$ au delà de laquelle on ne peut plus envoyer de messages (Ce cas de figure est tout à fait semblable à celui d'un flux de pièces sur un convoyeur traité dans [DA01]).
Dans ce cas de figure, le canal de transmission est modélisé par un RdP hybride temporisé. Dans les modèles qui vont suivre, nous considérons des délais de transmission relativement faibles par rapport aux débits ($d \to 0$ et $d.V \to 0$) et de ce fait, le modèle des canaux de transmission considéré est celui de la figure 3.16.b.

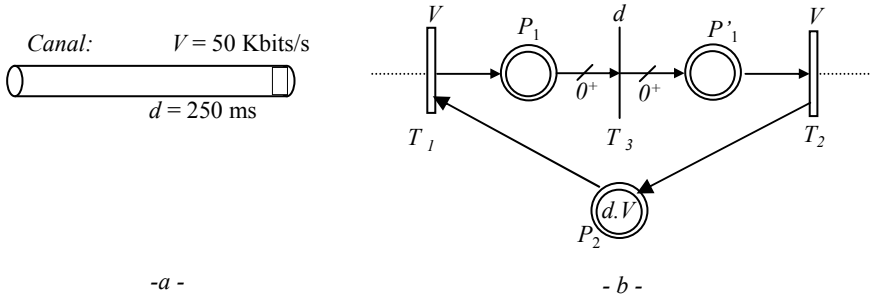


FIG. 3.17 – *a) Canal avec délai de propagation, b) Modélisation RdP continu équivalente.*

3.2.2.3 Le modèle continu d'une ligne de communication TCP/IP

Finalement, le modèle de la ligne *Emetteur - Routeur - Récepteur* est donné par la figure 3.18. Il comprend, en plus des modèles du routeur et des canaux (places Rt, $Cnl1$ et $Cnl2$), les places E et R qui représentent respectivement, les buffers de l'émetteur et du récepteur.

Le marquage de ces deux places indique la quantité de données présentes dans l'émetteur et le récepteur respectivement.

Etude de la dynamique de la ligne Analysons le comportement de la ligne telle que donnée par la figure 3.18. Nous supposons que l'émetteur possède une quantité de données K à envoyer

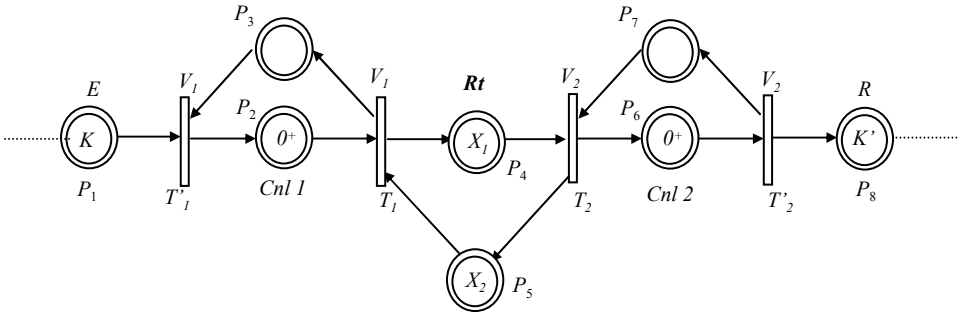


FIG. 3.18 – *Premier modèle RdP continu d'une ligne de communication.*

et que le récepteur en a reçu une quantité K'. Le marquage du routeur est tel que $X_1 + X_2 = C$. La dynamique de la place P_2 est $\dot{m}_2 = v'_1 - v_1 = 0$ (v'_1 et v_1 les vitesses instantanées associées à T'_1 et T_1 respectivement). Il en est de même pour les places P_3, P_6 et P_7 (invariant de marquage 0^+). La dynamique des places P_4 et P_5 est donnée par les relations:

$$\dot{m}_4 = V_1 - V_2; \tag{3.18}$$
$$\dot{m}_5 = V_2 - V_1. \tag{3.19}$$

Ainsi:

$$m_4(t) = (V_1 - V_2)t; \tag{3.20}$$
$$m_5(t) = C + (V_2 - V_1)t. \tag{3.21}$$

1. *Lorsque* $V_1 > V_2$:
   Un flux d'un débit V_1 arrive dans le routeur par le canal 1. Initialement, chacune des deux transitions T_1 et T_2 est fortement validée. Le vecteur de franchissement est donc égal à $v = (V_1, V_2)$. Les places P_4 et P_5 se remplissent et se vident respectivement selon les équations 3.20 et 3.21. Lorsque le marquage de P_5 s'annule, la transition T_1 devient faiblement validée et $v_1(t) = min[V_1, V_2] = V_2$ (voir la définition des transitions faiblement validées).
   L'instant de changement de dynamique, $t_c = C/(V_1 - V_2)$, est calculé en posant $m_5(t_c) = 0$ (ou $m_4(t_c) = C$) (Figure 3.19.b).
   Cette évolution correspond au graphe d'évolution de marquages donné sur la figure 3.19.a.

2. *Lorsque* $V_1 < V_2$:
   Dans ce cas de figure, la place P_4 se vide plus vite qu'elle ne se remplie, i.e., T_1 est fortement validée ($v_1(t) = V_1$) tandis que T_2 est faiblement validée ($v_2(t) = min(V_1, V_2) =$

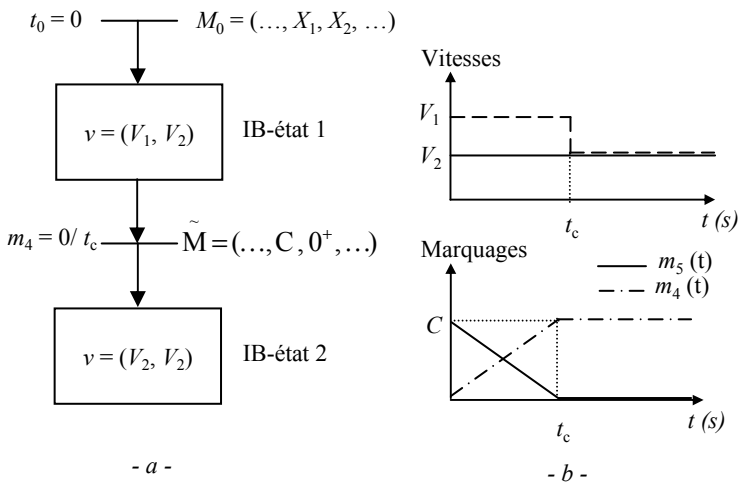


FIG. 3.19 – *a) Dynamique de la ligne de communication dans le cas $V_1 > V_2$. b) Graphe d'évolution correspondant.*

V_1, voir définition 3 et propriété 2 du franchissement des transitions continues). Les dynamiques de P_4 et P_5 deviennent $\dot{m}_4 = \dot{m}_5 = 0$ et les marquages $m_4(t)$ et $m_5(t)$ restent constants (figure 3.20)). Le graphe d'évolution associé ne comporte qu'un seul noeud et est donné par la figure 3.20.

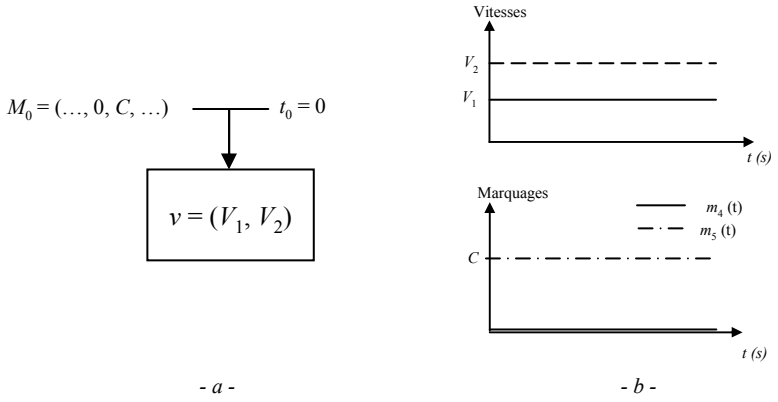


FIG. 3.20 – *Dynamique de la ligne de communication dans le cas $V_1 < V_2$. b) Graphe d'évolution correspondant.*

De façon générale, et pour mettre en évidence des dynamiques plus riches, nous ne considérerons dorénavant que les cas où $V_1 > V_2$ (débit d'entrée supérieur au débit de sortie).

Toutefois, remettons le modèle du routeur dans son vrai contexte qui est la ligne de communication avec des émetteurs et des récepteurs. Lorsque le buffer se charge ($m_5 = 0$ à $t = t_c$, Figure 3.18), il ne peut plus recevoir de données à la vitesse V_1 et sa vitesse d'entrée instantanée $v_1(t)$ s'aligne à V_2. Ce débit d'entrée vers la place Rt représente ce que le routeur *peut* prendre

comme données. Cependant, du côté de l'émetteur, rien, jusque là, n'est venu lui signaler qu'il fallait baisser le débit de V_1 à V_2. Il continue donc à émettre à la vitesse V_1 et la différence entre ce qui arrive au routeur ($v'_1(t)$) et ce qui est réellement pris par ce dernier ($v_1(t)$) est perdu. C'est le phénomène de congestion dans le réseau dû à un débit d'entrée supérieur aux capacités de la ligne.

3.2.2.4 Modélisation des pertes dans une ligne de communication

Regardons de plus prés les pertes de données décrites ci-dessus. Ces pertes se produisent à l'entrée du routeur. Nous savons aussi que ce qui est débité sur la ligne est égal à ce qui est pris par le routeur plus ce qui est perdu. Les pertes peuvent donc être modélisées comme indiqué sur la figure 3.21 où une transition, $T"_1$, de priorité inférieure à T_1, est ajoutée à la sortie du canal 1. Le marquage 0^+ d'une place assure qu'à tout instant la somme de ce qui entre dans cette place est égale à la somme de ce qui en sort. Ainsi, $v'_1(t) = v_1(t) + v''_1(t)$ est toujours vrai. La vitesse maximale associée à $T"_1$ est V_1 car il peut se perdre, au plus, toute les données qui arrivent au canal.

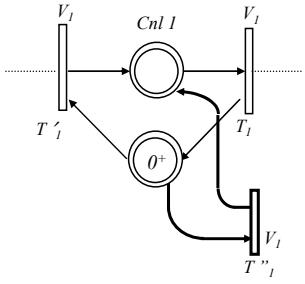


FIG. 3.21 – *Modélisation des pertes d'une ligne de transmission.*

Finalement, le modèle de la ligne de communication soumise à des pertes est représentée par la figure 3.22.
Tant que $m_5 \neq 0$, on a $v_1 = min(V_1, v'_1) = V_1$. La transition T_1 est prioritaire à T''_1, donc $v''_1 = V_1 - V_1 = 0$. A partir du moment où $m_5 = 0$, le nouveau calcul de v_1 donne: $v_1 = min(V_1,v'_1,v_2)$. Si $V_2 < V_1$, alors on trouve $v_1 = V_2$ et $v''_1 = v'_1 - v_1 = V_1 - V_2$.

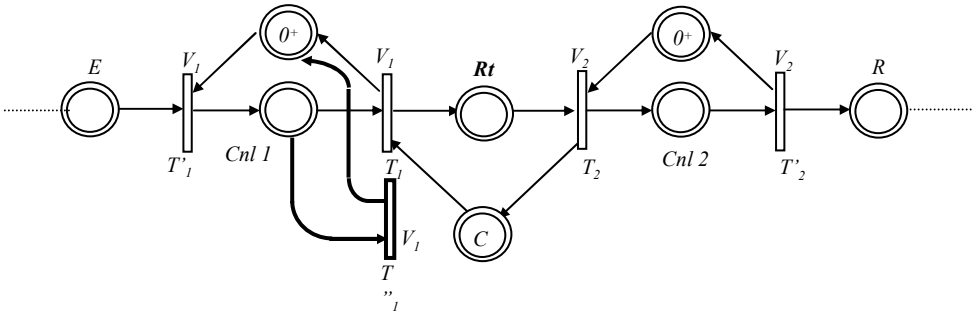


FIG. 3.22 – *Modélisation d'une ligne de transmission avec pertes.*

3.2.2.5 Modélisation d'une ligne de communication dans un environnement extérieur

La ligne décrite par la figure 3.22 ne tient compte que des éléments de la ligne elle-même et non d'un environnement extérieur qui peut partager certaines ressources de la ligne. Pour cela, cette ligne est dite isolée. Cependant, il est naturel de voir une ligne de communication partager ses ressources, et notamment son buffer, avec d'autres ressources extérieures. Dorénavant et afin de différencier notre ligne des sources extérieures, la ligne décrite sur la figure 3.22 sera appelée ligne d'émission tandis que toute source extérieure se partageant la ligne sera appelé ligne extérieure. Les mêmes appellations seront données aux flux, à savoir, flux d'émission et flux extérieurs.

Considérons des émetteurs extérieurs, notés E_{ext}, qui envoient des données vers d'autres récepteurs R_{ext} via le routeur Rt de la ligne d'émission. La capacité du récepteur et partagée entre les données qui arrivent de E et celles qui arrivent de E_{ext} (figure 3.23.a). On peut choisir de considérer le cas d'un seul émetteur extérieur connecté, ou celui de deux émetteurs extérieurs ou trois, sauf que ce nombre n'est jamais connu à l'avance. Il en est de même d'ailleurs des instants de connexion et des débits respectifs de ces émetteurs extérieurs. Afin de considérer le cas général où nous n'avons aucune donnée précise sur l'environnement où baigne la ligne Internet utilisée pour notre communication (à savoir, nombre d'émetteurs extérieurs connectés, instants de connexion, débits respectifs), nous allons considérer une seule entrée vers le routeur Rt qui représente la somme de toute les entrées possibles (figure 3.23.b). Cette entrée aura donc une allure aléatoire à l'image de ce qui peut se produire sur Internet comme connexion extérieures. De même, la sortie de ces flux est inconnue et la vitesse associée à cette sortie est aléatoire. Cependant, dans la suite de ce travail, nous ne considérerons aléatoires que les entrées extérieures et non les sorties. Cette hypothèse nous permet de simplifier le cas considéré sans pour autant enlever le caractère aléatoire des flux extérieurs de données dans le routeur. Celui-ci est garantit par le flux d'entrées appelé V_3 (figure 3.24).

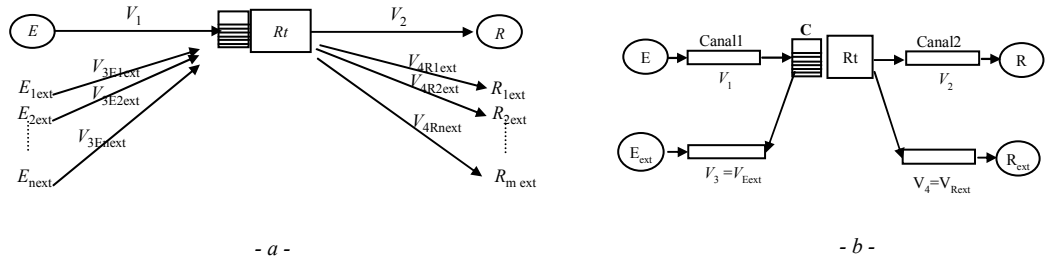


FIG. 3.23 – *Arrivées de flux extérieurs.*

Les places Rt_C et Rt_E de la figure 3.24 sont toutes deux relatives aux flux présents dans le routeur et la place P_5 représente la quantité de données que peut encore prendre le buffer. En effet, considérer deux places distinctes pour représenter les données présentes dans le routeur permet de différencier le flux d'émission du flux extérieur.
Selon que l'on considère le flux d'émission prioritaire au flux extérieur ou inversement, nous considérerons la transition T_1 prioritaire à T_3 ou inversement. Cette modélisation nous permet donc de prendre en compte d'éventuels flux prioritaires sur une ligne. Notons que tant que le

buffer n'est pas plein, les données peuvent entrer dans le routeur librement, seul le cas où le buffer est plein fait appel à la gestion de priorité.

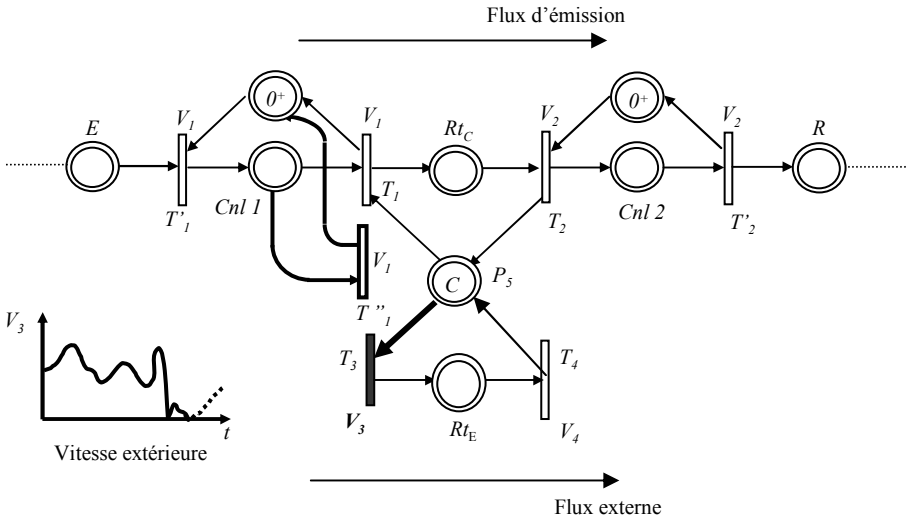


FIG. 3.24 – *Modèle de la ligne partagée avec les flux extérieurs.*

3.2.2.6 Dynamique de la ligne de communication

Nous allons considérer qu'en moyenne, les flux d'entrés du routeur sont supérieurs aux flux de sortie (dans le cas contraire, il n'y a ni remplissage du buffer ni pertes sur la ligne, voir paragraphe précédent).

La dynamique du routeur peut d'emblée être divisée en deux parties différentes : soit le routeur n'est pas plein, ($m_5 \neq 0$) et les messages qui arrivent peuvent être stockés dans le buffer, soit le routeur est plein et il y a alors conflit entre les transitions T_1 et T_3.

1. Tant que la relation $m_5 \neq 0$ est vérifiée, chacune des transitions T_1, T_2, T_3 et T_4 est fortement validée. Toutes les vitesses instantanées de franchissement associées à ces transitions sont égales à leurs vitesses maximales respectives. Les dynamiques des places P_4, P_5 et P_9 s'écrivent sous la forme :

$$\dot{m_4} = V_1 - V_2, \tag{3.22}$$
$$\dot{m_5} = (V_2 + V_4) - (V_1 + V_3), \tag{3.23}$$
$$\dot{m_9} = V_3 - V_4. \tag{3.24}$$

L'évolution de marquages de ces places s'écrit:

$$m_4(t) = (V_1 - V_2) * t, \tag{3.25}$$
$$m_5(t) = C + [(V_2 + V_4) - (V_1 + V_3)] * t, \tag{3.26}$$
$$m_9(t) = (V_3 - V_4) * t. \tag{3.27}$$

Avec: $m_4(t) + m_5(t) + m_9(t) = C$ toujours vérifié.

Les dynamiques des places relatives au canal 2, P_6 et P_7 sont nulles et les marquages associés à ces places restent constants.

L'équation de vitesse qui régit le canal 1 s'écrit sous la forme: $v_1'(t) = v_1(t) + v_1''(t)$. $v_1'(t)$ et $v_1(t)$ étant égales à V_1, $v_1''(t) = V_1 - V_1 = 0$. Cela s'explique aisément par le fait qu'il n'y a pas de pertes sur la ligne tant qu'il y a suffisamment de place dans le buffer.

2. Si $m_5 = 0$, T_1 et T_3 sont faiblement validées et il y a conflit entre ces deux transitions. Selon la définition 3 du même chapitre les expressions de v_1 et v_3 donnent:
   $v_1 \leq min(V_1, v_1', v_2 + v_4 - v_3)$,
   $v_3 \leq min(V_3, v_2 + v_4 - v_1)$,
   $v_1 + v_3 = v_2 + v_4$.
   T_3 étant prioritaire à T_1, on a $v_3 = min(V_3, v_2 + v_4)$ et v_1 est calculé en fonction de la valeur de v_3. Ainsi, si $v_3 = V_3$, $v_1 = (v_2 + v_4) - V_3$ et $v_1' = V_1 - v_1$ et si $v_3 = v_2 + v_4$ alors $v_1 = 0$ et $v_1' = V_1$. En d'autres termes, le débit extérieur étant prioritaire, il prend toute la place que lui permet sa vitesse extérieure V_3 et le flux d'émission occupe la place restante dans le buffer. La vitesse des pertes varie selon celle des précédentes vitesses.

Nous allons garder dans la suite de notre travail le RdP continu de la figure 3.24 pour modéliser la ligne de communication sans les protocoles TCP/IP.

Graphe d'évolution du RdP continu Les dynamiques citées ci-dessus peuvent être représentées par le graphe d'évolution de la ligne. Cependant, nous ne pouvons pas prévoir les évolutions des marquages en considérant le caractère aléatoire de la vitesse extérieure v_3. Supposons que v_3 est constante (en considérant, par exemple, le système sur un intervalle de temps assez petit pour pouvoir assimiler v_3 à une valeur fixe) et prenons les cas où $V_3 > V_4$ et celui où $V_3 < V_4$ (dans les deux cas, $V_1 > V_2$).

1. $V_3 > V_4$:
   Le graphe d'évolution correspond à celui de la figure 3.25. Le vecteur de marquage correspond aux places m_2, m_3, m_4, m_5 et m_9 respectivement (Figure 3.24), le vecteur des vitesses correspond aux vitesses v_1, v_1', v_1'', v_2, v_3 et v_4 respectivement (Figure 3.24). Les deux branches correspondent aux deux cas possibles où $v_3 = V_3$ et où $v_3 = V_2 + V_4$ lorsque $m_5 = 0$. Dans les deux cas, on obtient : $\dot{m}_4 = V_4 - V_3 < 0$ et $\dot{m}_9 = V_3 - V_4 > 0$. Ainsi, la place m_4 se vide tandis que m_9 se remplie. Lorsque $m_4 = 0$ (à t_2 par l'IB2 et à t_3 par IB3), les dynamiques se stabilisent aux vitesses données par l'IB4.
2. $V_3 < V_4$:
   Ce cas correspond à celui où seule la ligne émettrice charge le buffer (car $V_1 > V_2$). Le graphe d'évolution correspond à celui de la figure 3.26. Tant que $m_5 \neq 0$ (IB1), les vitesses sont maximales excepté le débit de sortie extérieur qui s'aligne sur le débit d'entrée ($v_4 = V_3$). Les dynamiques des places du routeur sont:

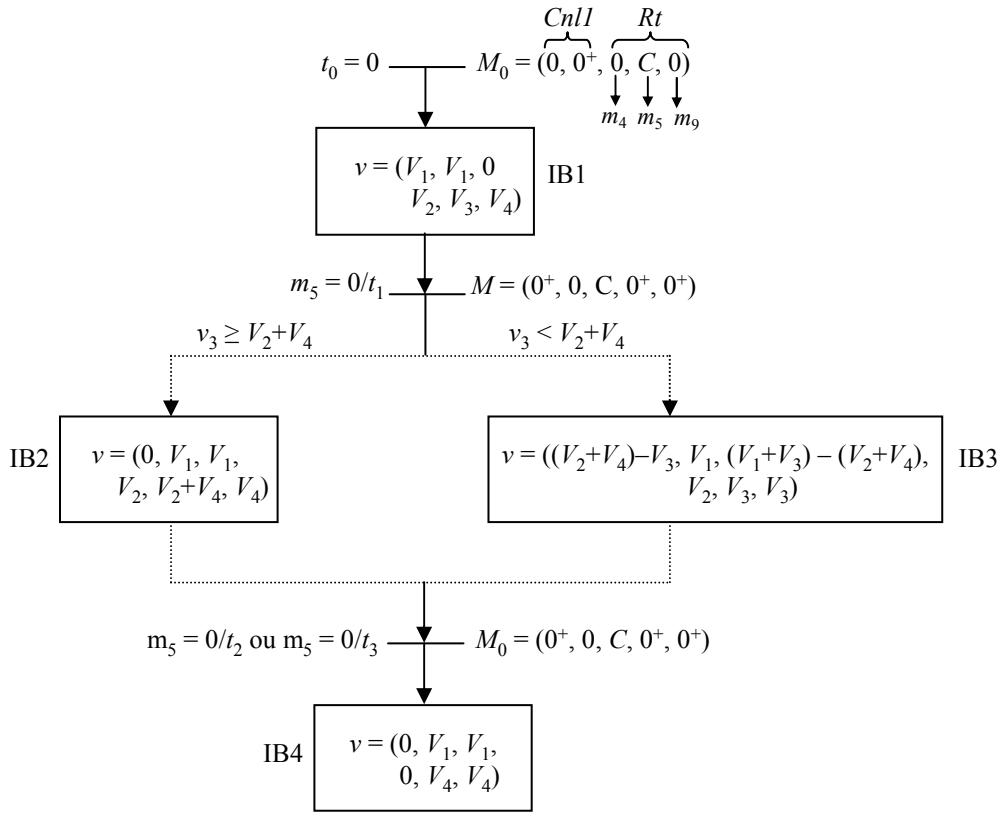


FIG. 3.25 – *Graphe d'évolution de la ligne de comunication avec* $V_3 > V_4$.

$\dot{m}_9 = 0$,
$\dot{m}_4 = V_1 - V_2 < 0$,
et $\dot{m}_5 = V_2 - V_1 > 0$.

Ainsi, le marquage de m_5 diminue tandis que celui de m_4 augmente. Lorsque $m_5 = 0$ (IB2), $v_3 = min[V_3, v_2 + v_4] = V_3$, $v_1 = min[V_1, v_2 + v_4 - v_3] = min[V_1, V_2] = V_2$ (car $v_4 = v_3 = V_3$ et $V_2 < V_1$) et $v_1'' = V_1 - V_2$.

Il apparaît dans ce deuxième exemple que malgré la priorité du flux extérieur sur le flux d'émission, le buffer se charge des données d'émission et non du flux extérieur.

3.3 Le modèle RdP *discret* d'une ligne de communication TCP/IP

Comme nous l'avons dis précédemment, la ligne telle que modélisée sur la figure 3.24 ne prend pas en compte les consignes discrètes des protocoles de transmission TCP/IP. La modélisation de cette partie discrète fera donc l'objet de cette section [Bit04].

Tout d'abord, commençons par dire que les protocoles modélisés ici seront slow start, congestion avoidance, le temporisateur de retransmission et le processus de reprise après des pertes. Ces quatre processus ont été détaillés dans le chapitre 2. Nous avons choisi ces pro-

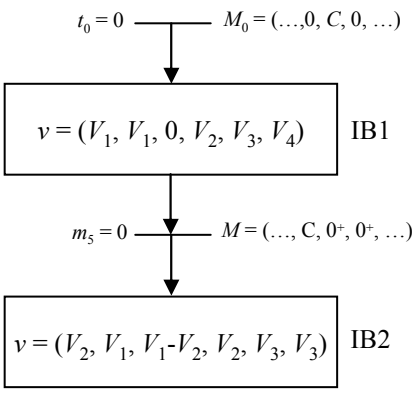


FIG. 3.26 – *Graphe d'évolution de la ligne de communication avec $V_3 < V_4$.*

tocoles car se sont ceux implantés dans toutes les versions TCP, anciennes ou nouvelles. Les nouveaux protocoles fast retransmit, fast recovery ou encore la nouvelle option SACK (chapitre 2) n'étant implantés que dans certaines nouvelles versions du TCP.

Les deux informations pertinentes sur lesquelles se basent les protocoles sus cités sont l'arrivée ou non d'acquittement et l'expiration du temporisateur de retransmission. Ces deux informations servent à estimer l'état du réseau pour moduler au mieux la taille des données envoyées afin d'éviter l'état de congestion qui coûte chère à une transmission en terme de temps et de bande passante. Nous allons reprendre les algorithmes un à un et présenter leurs modèles respectifs en commençant par slow start.

3.3.1 Chevauchement des données : modèle coloré

D'une part, lorsqu'une fenêtre a été complètement envoyée, l'envoi de la prochaine fenêtre commence. Plusieurs fenêtres peuvent donc se chevaucher sur une ligne. D'autre part, lorsque des données sont envoyées, les temporisations correspondantes sont lancées, donc plusieurs temporisateurs peuvent être enclenchés simultanément. Par ailleurs, le nombre de fenêtres présentes sur une ligne ne peut être connu à priori. En effet, il dépend étroitement des instants d'arrivés des acquittements. Instants que l'on ne peut que calculer à posteriori. Une première solution consiste à considérer une place différente pour chaque temporisation lancée. Le modèle de la ligne risque donc de se répéter autant de fois qu'il y a de données envoyées donnant un modèle global très lourd. Finalement, afin de gérer ce chevauchement des informations et des temporisations, un modèle RdP coloré est utilisé.

Les RdP colorés Les RdP colorés permettent d'obtenir une forme simplifiée d'un RdP de structure complexe. En effet, ces RdP sont très pratiques lorsqu'il s'agit, par exemple, d'un système qui possède plusieurs composantes aux comportements identiques. La représentation de tels systèmes par des RdP ordinaires peut donner des modèles de taille inexploitable. Il s'agit alors, par RdP colorés, d'associer une *couleur* à chaque partie et d'effectuer ensuite un *pliage* de ces différentes parties.

Commençons par définir ce qu'est une couleur. Une couleur est une façon de différencier des jetons lorsqu'on a besoin d'informations détaillées sur un marquage. Prenons l'exemple d'un stock qui contient 3 pièces de 2 types différents (les pièces de type 1 sont notées p_1 et les les pièces de type 2 sont notées p_2). Modéliser ce stock par un RdP ordinaire apportera l'information nombre de pièces dans le stock (figure 3.27.a). Ce même stock modélisé par un RdP coloré apportera l'information nombre et types de pièces dans le stock. Si, sur les 3 pièces présentes dans le stock, 2 sont de type p_1 et une de type p_2, l'ensemble des couleurs sera (p_1,p_2) et le RdP coloré est donné par la figure 3.27.b. L'ensemble de couleurs d'un RdP coloré est également associé aux transitions. La règle de franchissement des transitions dans les RdP colorés est la même que pour un RdP ordinaire mais s'applique par couleur. Ainsi, une transition T_i est franchissable par rapport à la couleur c_i si toutes ses places en amont contiennent au moins une marque portant la couleur c_i.

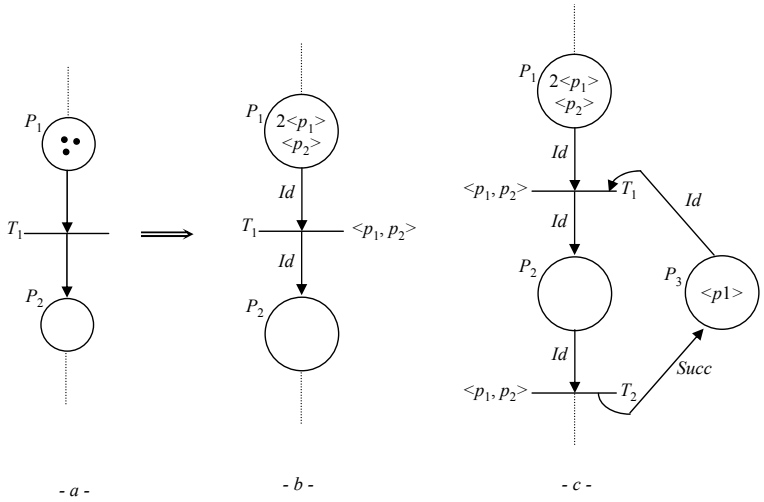


FIG. 3.27 – *a) Modèle RdP ordinaire, b)et c) Modèle coloré.*

Dans le RdP coloré de la figure 3.27.b une *fonction identité*, notée Id est associée à chacun des arcs $P_1 \rightarrow T_1$ et $T_1 \rightarrow P_2$. Cette fonction signifie que le franchissement de T_1 par rapport à la couleur p_1 (ou p_2) induit le retrait d'une pièce de couleur p_1 (ou p_2) et le dépôt d'une pièce de la même couleur dans la place P_2. Cependant, les fonctions associées aux arcs peuvent être différentes de la fonction Id. Supposons que ces mêmes pièces p_1, p_2 sont prises par une machine M_1 de façon alternative, i.e., une pièce p_1 suivie d'une pièce p_2, etc. Ce système peut être représenter par le RdP coloré de la figure 3.27.c. Le marquage de la place P_2 indique la pièce que la machine M_1 usine tandis que le marquage de la place P_3 indique la pièce que M_1 est prête à recevoir. La *fonction successeur* notée Succ est associée à l'arc $T_2 \rightarrow P_3$ telle que Succ(p_i)=p_{i+1} (modulo 2). Initialement, le seul franchissement possible de T_1 est par rapport à la couleur p_1. Lorsque cette pièce sort de la machine (franchissement de T_2), une marque de couleur p_2 est mise dans P_3: Succ($p_1 = p_2$). La fonction successeur permet ainsi l'ordonnancement de l'usinage dans la machine: pièce de type p_1 puis pièce de type p_2, etc. D'autres fonctions peuvent être associées aux arcs telles que Prec (précédent pour la décrémentation) ou alors Dec (Décoloration). Notons simplement que ces fonctions peuvent être complexes et induire le changement ou la suppression des couleurs.

En ce qui concerne le modèle de la ligne de communication, une couleur sera associée à chaque fenêtre de données envoyée. Ainsi, les places contiendront des marquages associés à des numéros de fenêtres. Un marquage $< m,\ f_i >$ signifiera que cette place contient la quantité m de donnée correspondant à la fenêtre d'envoi numéro i. Un marquage $< 3.5m,\ f_j >$ signifiera que cette place contient la quantité ($3.5m$) de données correspondant à la fenêtre d'envoi numéro j, etc. L'ensemble des couleurs associées aussi bien aux places qu'aux transitions est nommée $F = (f_1,f_2,...,f_n)$, n étant le nombre maximal représentant le nombre de fenêtres présentent sur la ligne.

3.3.2 Modélisation de l'algorithme slow start

La phase slow start est la phase dite exponentielle. Durant cette partie de la transmission, chaque acquittement reçu augmente la nouvelle fenêtre de congestion de deux nouveaux segments. Commençons par modéliser l'arrivée des acquittements.

Arrivée des acquittements Un acquittement est envoyé par le récepteur à chaque fois qu'il reçoit un segment. La taille d'un segment peut varier d'un flux à un autre et, selon les routeurs, peut être fragmentée si elle est trop importante. Cependant, un segment varie souvent entre 500 octets pour les segments de petite taille et 1500 octets pour les segments les plus gros. Ainsi, nous adopterons pour notre part, des segments de taille fixe égale $m = 1000$ octets.
S'il y a l'équivalent d'une taille de donnée (m) dans le buffer du récepteur, un acquittement est envoyé. Au niveau du modèle, cela se traduit par le franchissement de la D-transtion T_5 lorsque $m_8 = m$ (Figure 3.28). Une quantité d'information continue, ici m, se transforme en nombre discret, ici nombre d'acquittement, par le biais de la D-transition T_5.

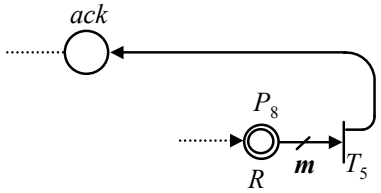


FIG. 3.28 – *Envoi des acquittements.*

La place ack représente le nombre d'acquittements reçus par l'émetteur. En effet, le flux des acquittements n'est pas représenté ici dans le sens où l'on considère que les acquittements ne sont ni perdus dans le réseau ni retardés par celui-ci. Cette hypothèse de travail est envisageable car le flux des acquittements est considéré d'une part prioritaire sur les autres flux et d'autre part, les acquittements étant des messages de petite taille, leur acheminement sur le réseau est plus rapide que celui d'un flux ordinaire. En outre, l'on peut considérer aisément un retard de l'arrivée des acquittements sur la ligne en posant une temporisation sur la D-transition T_5.

Modélisation de l'évolution exponentielle de la fenêtre de congestion Durant cette partie de la transmission, chaque acquittement reçu augmente la nouvelle fenêtre de congestion de

deux nouveaux segments. Observons la figure 3.29. Elle représente le modèle (que l'on va compléter plus loin) d'une seule fenêtre lancée sur la ligne. Une fenêtre est activée (place $Autss1$ marquée) et sa temporisation associée est lancée (place $Tempo1$ marquée) par le franchissement de T_1. Un acquittement est envoyé par le récepteur à chaque fois qu'il reçoit un segment. S'il y a l'équivalent d'un segment (m) dans le récepteur R, un acquittement est envoyé. Au niveau du modèle, cela se traduit par le franchissement de la transition T_3 lorsque le marquage de la place $R = m$. Une quantité d'information continue, ici m, se transforme en nombre discret, ici nombre d'acquittement, par le biais de la D-transition T_3. La place $ack1$ représente ainsi le nombre d'acquittements reçus par l'émetteur.

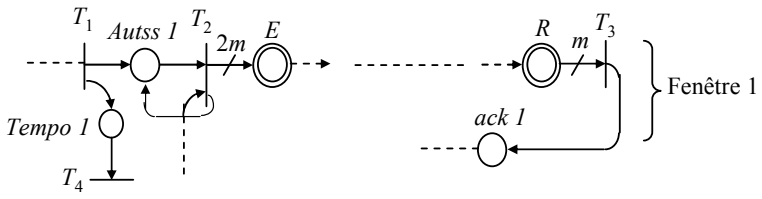


FIG. 3.29 – *Envoi d'une seule fenêtre de donnée sur la ligne de transmission.*

Nous allons considérer maintenant le cas où plusieurs fenêtres se chevauchent sur la ligne de communication. La figure 3.30 représente le cas où 3 fenêtres au plus peuvent se chevaucher. Ainsi, trois parties identiques se répètent où chaque partie (ou étage) représente l'envoie d'une fenêtre. La ligne continue est représentée en pointillé (mis à part E et R) pour ne pas encombrer le modèle. Supposons que l'on active et lance la première fenêtre (les places $Tempo1$ et $Autss1$ marquées). Les acquittements de cette première fenêtre déterminent la taille de la seconde fenêtre. Remarquons que pour chaque acquittement arrivé, deux segments sont créés dans la seconde fenêtre pour assurer l'évolution exponentielle des données. La seconde fenêtre est activée puis lancée lorsque la première fenêtre est complètement envoyée (arc inhibiteur de la place E vers la transition T_1'). De même, les acquittements de cette seconde fenêtre vont déterminer la taille de la troisième fenêtre que je vais activer et lancer à son tour lorsque j'aurais fini d'envoyer la seconde fenêtre. Si l'on considère qu'il n'y a que trois fenêtres au plus qui se chevauchent, nous pouvons boucler le schéma (arc de la place $ack3$ vers T_1) à la fin de la 3ème fenêtre. D'une part, nous ne savons pas à l'avance le nombre de fenêtres qui vont se chevaucher. D'autres part, s'il y a beaucoup de fenêtres qui se chevauchent, le modèle risque d'être très gros. La 2ème solution proposée alors est de considérer un modèle coloré avec l'ensemble des couleurs donné par les différentes fenêtres présentes simultanément sur la ligne. Le modèle RdP hybride coloré d'une ligne de transmission est donné par la figure 3.31. Une couleur est associée à chaque fenêtre de données envoyée. L'ensemble des couleurs associées aussi bien aux places qu'aux transitions est nommée $F = (f_1,f_2,...,f_n)$, n étant le nombre maximal représentant le nombre de fenêtres présentes sur la ligne.

Au niveau des transitions T_5 et T_7, on voit donc l'interaction directe, d'une part entre le flux continu sur la ligne qui se transforme en information discrète pour le protocole et d'autre part, ces mêmes informations discrètes (nombre d'acquittements) servant à gérer la quantité de flux sur la ligne. Ce modèle montre les parties continues et discrètes sur une ligne de communication et leur constante interaction formant ainsi un modèle hybride.

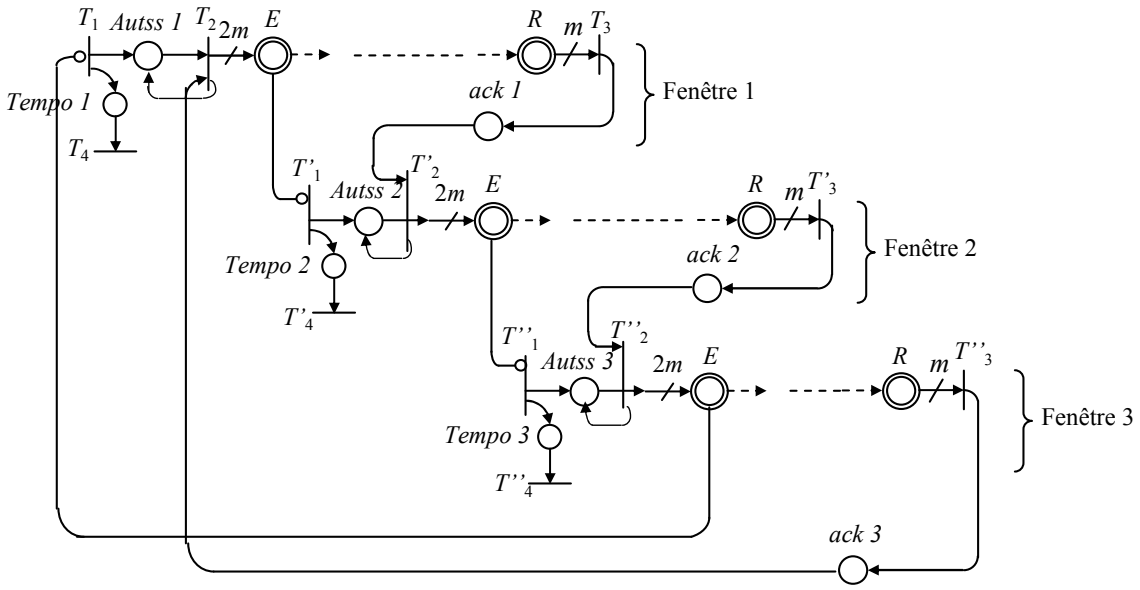


FIG. 3.30 – *Modèle non coloré du chevauchement de trois fenêtres de transmission en évolution exponentielle.*

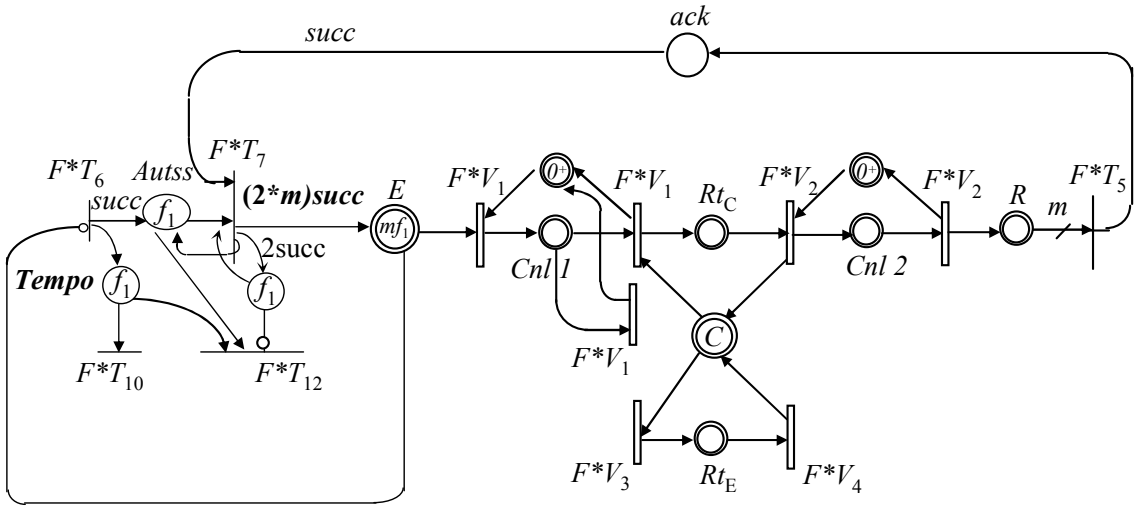


FIG. 3.31 – *Modèle coloré d'une ligne soumise au protocole slow start.*

3.3.3 Modélisation de l'algorithme congestion avoidance

Pendant la phase congestion avoidance, pour chaque ack arrivé, on considère un segment dans la nouvelle fenêtre (transition T_9, Figure 3.32). L'augmentation linéaire de la taille des fenêtres est obtenue en considérant un acquittement de plus dans ack (franchissement de la transition T_8) dès qu'on décide d'envoyer une fenêtre répondant à une évolution linéaire (place $Autca$ marquée).

De même que pour le modèle de la figure 3.31, les transitions T_5 et T_9 (T_5 et T_7 sur la figure 3.31) représentent le passage du continu au discret et du discret au continu respectivement.

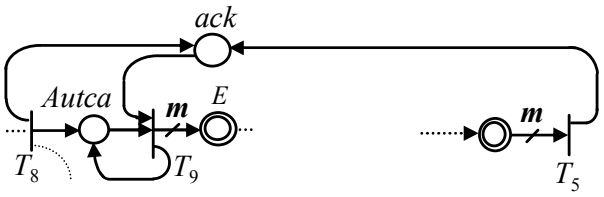


FIG. 3.32 – *Evolution linéaire de la transmission.*

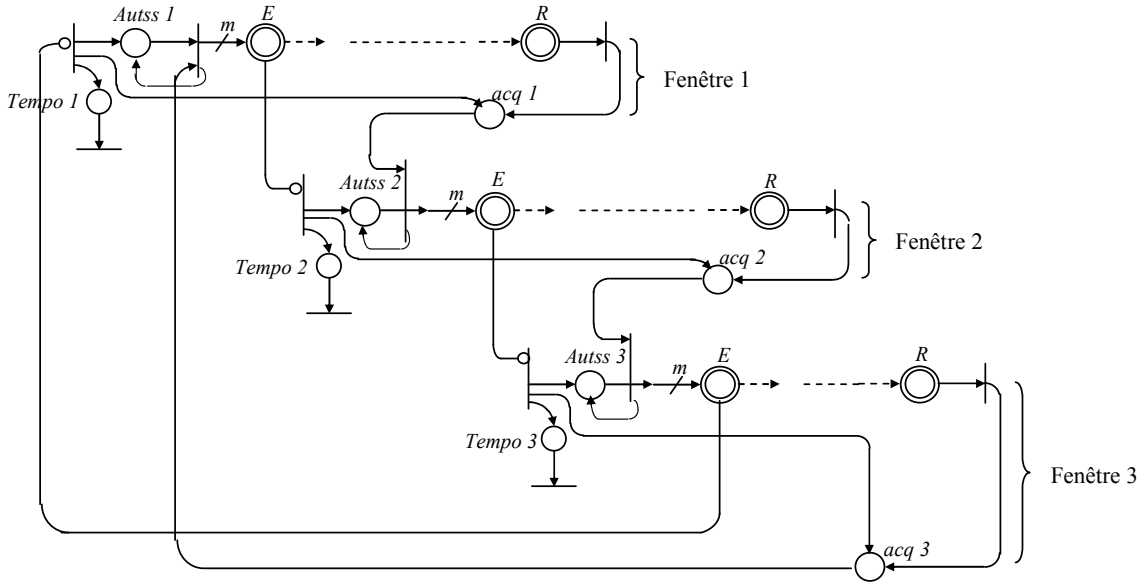


FIG. 3.33 – *Modèle non coloré du chevauchement de trois fenêtres de transmission en évolution linéaire.*

La figure 3.33 représente l'évolution linéaire d'une fenêtre de congestion en supposant que trois fenêtres peuvent se chevaucher. Pour les mêmes raisons que dans le cas de l'évolution exponentielle, i.e., nous ne connaissons pas à priori le nombre de fenêtres présentent simultanément sur une ligne de communication, nous allons adopter la solution d'un modèle RdP hybride coloré. Le modèle est finalement donné par la figure 3.34.

3.3.4 Gestion des temporisateurs

Lorsqu'un segment est envoyé, on attend l'arrivée de son acquittement pour augmenter la taille de la prochaine fenêtre envoyée. Cette durée d'attente est fixée par une temporisation appelée $Tempo$. Si, à échéance de $Tempo$, les acquittements ne sont toujours pas arrivés, les données sont considérées perdues. Pour modéliser les temporisations, nous utilisons une nouvelle place nommée $Tempo$ en amont d'une D-transition temporisée T_{10} (figure 3.35). La valeur associée à cette transition n'est autre que la valeur de la temporisation. Dès que la décision d'envoi de données est prise (selon le protocole slow start : place $Autss$ marquée ou selon le protocole congestion avoidance : place $Autca$ marquée), la place $Tempo$ est marquée. Si à échéance de

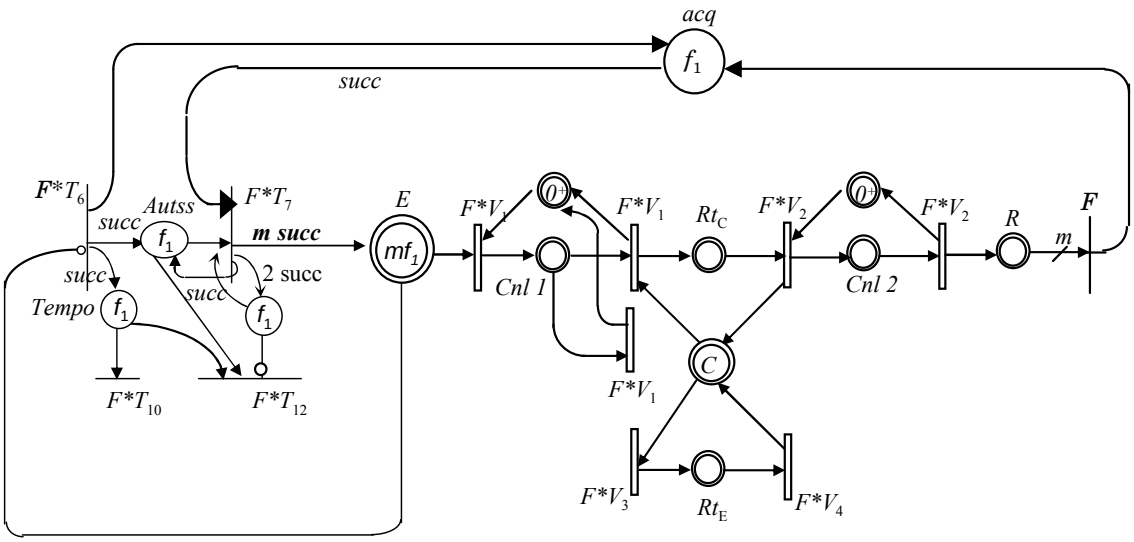


FIG. 3.34 – *Modèle coloré d'une ligne soumise au protocole congestion avoidance.*

$Tempo$, les acquittements ne sont pas arrivés, un procédure de réinitialisation est enclenchée. Notons que la variation mathématique de la valeur de la temporisation (donc la durée associée à T_{10}) (voir chapitre 2) ainsi que la procédure de réinitialisation après expiration du temporisateur ne sont pas représentées dans le modèle RdP car cela ne ferait que compliquer le modèle global sans pour autant apporter de réelles contributions du point de vue d'une modélisation pas RdP. Ces deux procédures sont réalisées de façon logicielle.

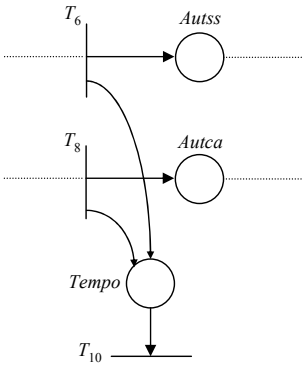


FIG. 3.35 – *Modélisation du temporisateur de retransmission.*

3.3.5 Passage de slow start à congestion avoidance

Slow start et congestion avoidance sont deux algorithmes différents mais on doit pouvoir passer de l'un à l'autre selon l'état de la transmission. Le passage d'un algorithme à un autre se fait par un test du seuil d'évitement de congestion appelé $ssth$. Si la taille de la fenêtre envoyée est inférieure à ce seuil, l'algorithme appliqué est slow start, sinon, on applique congestion

avoidance. Pour cela, nous nous servons d'un nouvelle place, appelée Mem, pour mémoriser la taille de la fenêtre à envoyer. Avant de décider de l'envoi d'une fenêtre, Mem est comparée à $ssth$ (figure 3.36) et selon le résultat, l'un ou l'autre des algorithmes est appliqué.

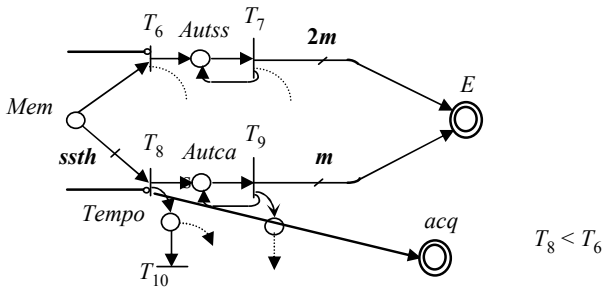


FIG. 3.36 – *Test de seuil.*

Finalement, le modèle de la ligne de communication, sous les protocoles de contrôles de congestion du TCP est donné par un RdP hybride temporisé coloré et est représenté par la figure 3.37.

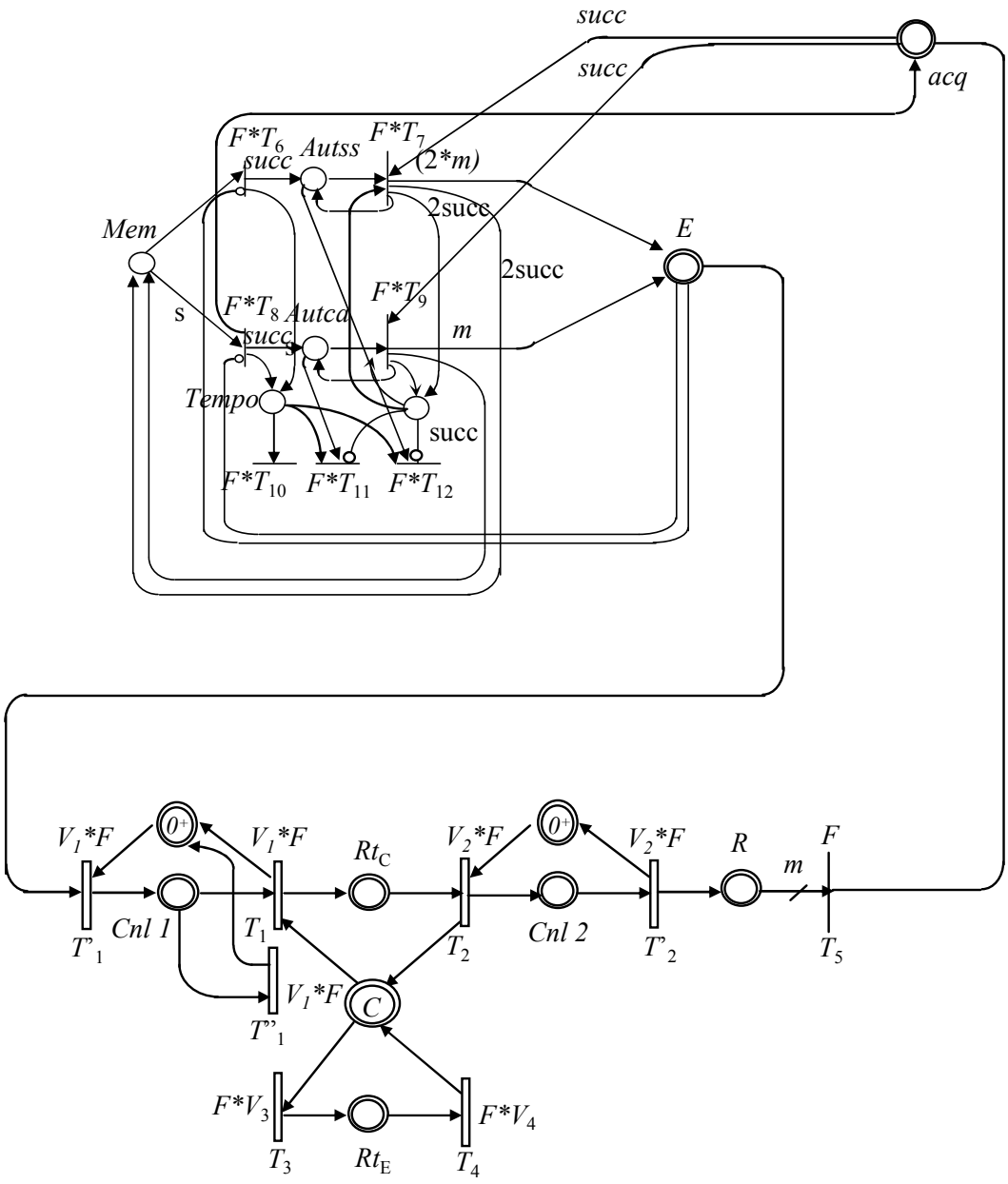


FIG. 3.37 – *Modèle RdP hybride coloré d'un ligne de communication transmises aux protocoles TCP/IP dans un environnement Internet.*

Chapitre 4

Evaluation de performances d'une ligne de communication TCP/IP

Après avoir vu le modèle complet RdP hybride d'une ligne de transmission soumise aux protocoles TCP/IP, nous allons passer à l'étude de ce modèle et à l'extraction des performances de cette ligne. Pour ce faire, le modèle de la figure 3.37 est implanté sous Matlab. L'objectif des simulations est dans un premier temps de vérifier la validité du modèle tel que décrit par la figure 3.37. Cette validité sera vérifiée en étudiant les évolutions des données envoyées sur la ligne de communication et en comparant ces évolutions avec celles prévues par la théorie. Dans un second temps, on observera le comportement de la ligne en termes de remplissage du buffer, d'instants d'occurrence des pertes, de proportion d'occupation du buffer par les flux selon les priorités ainsi que certains critères de performances tels que débit de la ligne et temps de transmission. Les simulations nous permettront, comme nous le verrons un peu plus loin, de tirer des conclusions pertinentes en observant certaines variables et notamment le temps de boucle sur la ligne ([BA05b], [BA05c]).

4.1 Algorithme de simulation de la ligne de transmission

Avant d'entrer dans les détails de l'algorithme mis en place pour traduire le comportement du RdP de la figure 3.37, regardons de plus près la nature des variables de ce système.

4.1.1 Caractère événementiel de la ligne de communication

Nous avons exposé une première idée de l'évolution rencontrée sur la ligne dans les sections 3.2.2.3 et 3.2.2.6. Nous y avons vu que les débits des canaux de transmission sont à vitesses maximales constantes et que les vitesses instantanées correspondantes changent lorsqu'un marquage continu atteint la valeur zéro (Figure 3.18). Cette évolution correspond à celle des RdP Continus à vitesses Constantes (RdPCC, section 3.1.2.6). Si l'on omet de considérer la vitesse extérieure $v_3(t)$, qui elle, est aléatoire, nous avons affaire à un système événementiel. Cependant, $v_3(t)$ possède une valeur maximale qui varie dans le temps. Le système complet ne peut donc pas être considéré événementiel. Afin de rendre la dynamique complète à caractère événementiel, nous avons choisi de discrétiser le temps. Le pas de discrétisation est choisi

suffisamment petit pour ne pas altérer les performances du système. Ainsi, sur chaque pas de discrétisation, appelé δ, la valeur maximale de la vitesse $v_3(t)$ reste constante. Deux événements peuvent maintenant changer la dynamique du flux continu: d'une part, le marquage d'une place continue qui atteint zéro et d'autre part, la variation de $v_3(t)$ à chaque pas δ. L'algorithme implanté de la ligne se base sur l'occurrence d'un de ses deux événements.

4.1.2 Algorithme de simulation de la ligne TCP/IP

La simulation se fait sur un horizon donné (noté $[0, t_f]$ avec t_f assez grand pour voir toutes les dynamiques du système) par tranche de δ (rappelons que δ est le pas de discrétisation de la vitesse extérieure $v_3(t)$). On introduit pour cela un intervalle appelé $[t_{start}, t_{final}]$ tel que $t_{final} = t_{start} + \delta$ et tel que $v_3(t)$ reste constante sur cet intervalle. Ainsi, sur un $i$ème intervalle $[t_{start}, t_{final}]$, la valeur maximale V_3 sera égale à $v_3(i)$, i étant le numéro de l'intervalle (Figure 4.1).

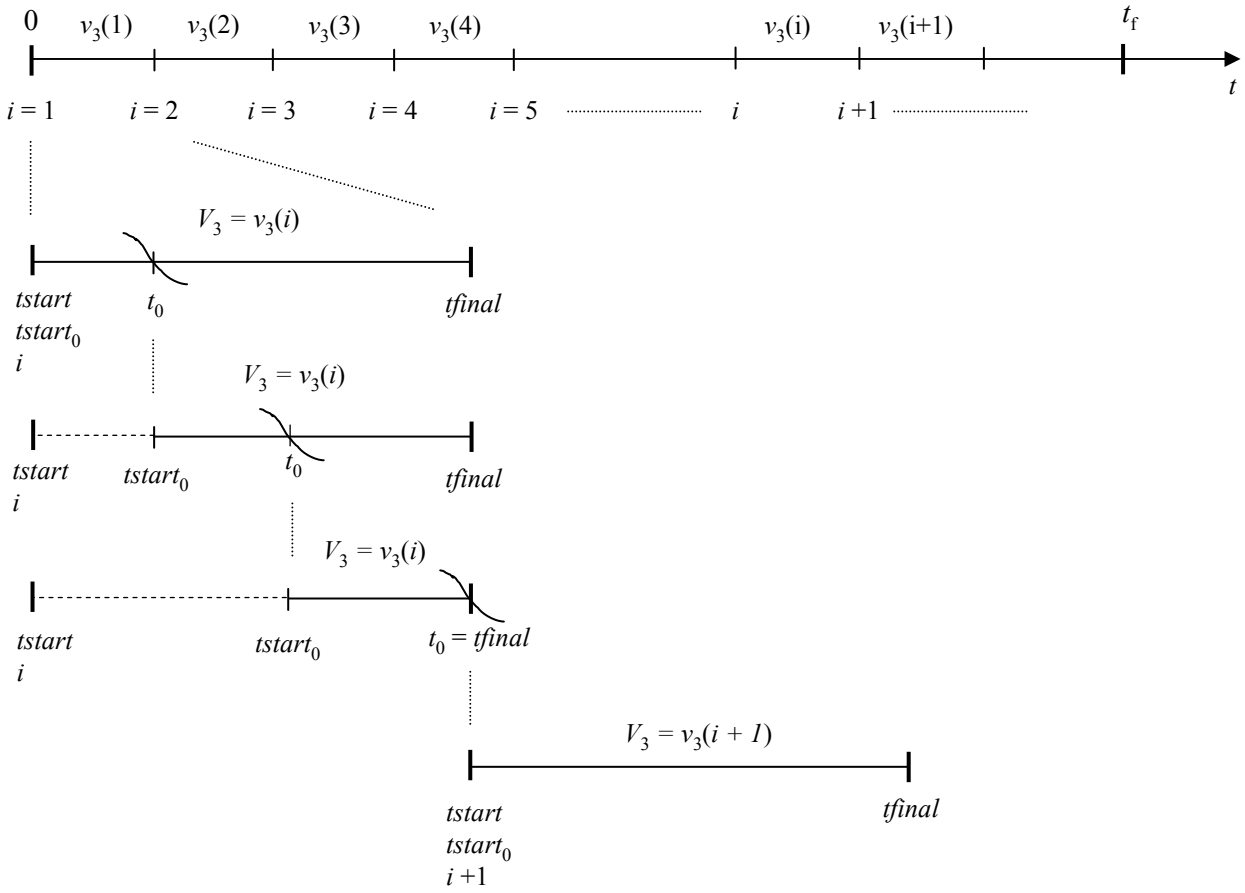


FIG. 4.1 – *Procédure de discrétisation du temps*

Les même procédures de calculs de dynamiques et de marquages vont se répéter sur les intervalles successifs $[t_{start}, t_{final}]$ jusqu'à balayer tout l'horizon de simulation $[0, t_f]$. Rappelons

qu'à ces dynamiques continues vont se rajouter les contraintes discrètes dues aux protocoles de communication considérés (pris en compte dans le modèle RdP hybride).
Ainsi, sur chaque intervalle de temps et à partir des valeurs des marquages au début de cet intervalle, on calcule les vitesses instantanées qui vont déterminer la dynamique des marquages tout au long de l'intervalle de travail.
Les changements de dynamique se font à un instant appelé t_0. Cet instant t_0 peut soit être égal à t_{final} (si l'instant de passage à zéro coincide avec le changement de valeur de v_3) soit représenter l'instant de passage à zéro d'un des marquages continus de la ligne. Si, sur un intervalle donné, plusieurs marquages atteignent le zéro à des instants différents, t_0 sera relatif au plus petit d'entre eux. Ainsi, lorsque $t_0 < t_{final}$, les différents marquages atteint à t_0 sont sauvegardés et on reprend le calculs des vitesses pour le temps restant de l'intervalle, soit $[t_0, t_{final}]$.
Lorsque $t_0 = t_{final}$, les valeurs de marquages sont de même sauvegardés et on reprend les calculs sur un nouvel intervalle $[t_{start}, t_{final}]$ (avec: t_{start}(nouveau) devient égal à t_{final}(ancien) et t_{final}(nouveau) prend la valeur $t_{start} + \delta$). A chaque changement d'intervalle, la valeur maximale de la vitesse extérieure $v_3(t)$ est égale à sa valeur correspondante sur cet intervalle.
Notons que cette dynamique doit être calculée pour toutes les fenêtres présentes sur la ligne, ainsi, un balayage par couleur est aussi effectué.

4.1.2.1 Les temporisateurs

Nous désignons un vecteur relatif à la valeur des différentes temporisations lancées en même temps. Rappelons d'abord comment sont calculées et remises à jour les temporisations. Les temporisations sont relatives aux fenêtres. Une fenêtre lancée enclenche sa temporisation correspondante. A l'état initial, nous supposons une valeur initiale de la temporisation (valeurs initiales appelées: $Tempo(1)$, $RTT(1)$, $D(1)$ et $M(1)$). Lorsque tous les acquittement de cette fenêtre sont arrivés, on remet à jour la valeur de $Tempo$ pour l'appliquer à la prochaine fenêtre envoyée. On procède selon les équations citées dans le chapitre 2, section 3.1.1 et on écrit:

$$Tempo(2) = RTT(2) + 4D(2),$$
avec: $$RTT(2) = \alpha RTT(1) + (1-\alpha)M(1);$$
et $$D(2) = \alpha D(1) + (1-\alpha)|RTT(1) - M(1)|;$$

À partir de ce moment, les données envoyées (au moins la 2ème fenêtre) seront soumises au nouveau délai donné par $Tempo(2)$. Par la suite, à chaque fois que tous les acquittement d'une même fenêtre sont arrivées, on remet à jour $Tempo$ et on l'applique aux nouvelles données envoyées etc. De façon générale, on écrit:

$$Tempo(i+1) = RTT(i+1) + 4D(i+1),$$
avec: $$RTT(i+1) = \alpha RTT(i) + (1-\alpha)M(i);$$
et $$D(i+1) = \alpha D(i) + (1-\alpha)|RTT(i) - M(i)|;$$

L'indice i indique les valeurs calculées pour la précédente fenêtre reçue et l'indice $i+1$ indique les valeurs estimées appliquées aux prochaines fenêtres envoyées.
Pendant la simulation, à chaque changement de dynamique (à chaque nouvel instant t_0), on soustrait à la valeur courante de l'ensemble des temporisations lancées l'intervalle $(t_0 - t_{start})$ pour tenir compte du temps écoulé. On écrit:

$$tempo_k = tempo_k - (t_0 - t_{start}) \quad \text{pour chaque fenêtre } k \text{ lancée.}$$

Notons que les temps de boucle des données sont calculées d'une façon similaire. En effet, les temps de boucle sont initialisés à zéro au début de l'envoi des fenêtres puis sont augmentés de

la valeur $(t_0 - t_{start})$ à chaque instant t_0. Cette procédure nous permet de récupérer le temps exact mis entre l'envoi des données et la récupération de leurs acquittements. Une procédure de sauvegarde des temps de boucle est aussi effectuée.

4.1.2.2 Sauvegarde et récupération de données

Afin de récupérer les valeurs des données qui nous intéresse (par exemple, les marquages ou les vitesses), on effectue une sauvegarde des différentes données à chaque t_0. Prenons l'exemple de la place Mem et regardons ce que donne la sauvegarde.
La sauvegarde se fait à chaque instant (j) pour les n fenêtres gérées simultanément.
À l'instant j: $Mem = (x_{j1}\ \ x_{j2}\ \ x_{j3}\ ...\ \ x_{jn})$
À l'instant $j+1$: $Mem = (x_{(j+1)1}\ \ x_{(j+1)2}\ \ x_{(j+1)3}\ \ ...\ \ x_{(j+1)n})$
$\Rightarrow Mem$ =

$$\begin{array}{llll} j=1 & x_{11} & \dots & x_{1n} \\ j=2 & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ j=r & x_{r2} & \dots & x_{rn} \end{array}$$

Ainsi, une ligne de la matrice Mem nous indique le marquage de toute les couleurs à un instant j et une colonne de cette matrice nous indique l'évolution d'un même numéro de fenêtre (couleur) dans le temps.

$\Rrightarrow$ ***Programme principal***

1. ***INITIALISATION*: Déclaration des valeurs initiales des:**
   Valeurs maximales des débits (V_1, V_2, V_4,v_3),
   Capacité du buffer C,
   Taille d'un segment (m),
   Marquages initiaux, valeur initiale de Tempo, pas de discrétisation δ,
   Quantité initiale de données à envoyer: $U1$
   Valeur courante de cette quantité de donnée: U. Initialement $U = U_1$,
   R: Quantité de données arrivée au récepteur. Initialement $R = 0$, Pertes: pertes=0
   Valeurs totale des pertes: tot_{pertes}=0
   Temps de boucle $M = 0$
   Numéro de la fenêtre $k = 1$
   Entrées n (nombre de couleur) et S (Seuil d'évitement de congestion)
   Horizon de simulation [t_0, t_f]
   Tempo: Valeur de la temporisation,
   dtempo: temps restant de la temporisation. Initialement, dtempo = Tempo,
   Initialisation de l'indice du vecteur vitesse $i = 1$ (i.e., $V_3 = v_3(1)$),

2. *BOUCLE PRINCIPALE* Tant que $t_{final} \leq t_f$

Boucle secondaire: Dans un même pas de discrétisation: Tant que $t_0 \leq t_{final}$
et qu'aucune temporisation n'a expiré,

Alors: - Calculer Les vitesses instantanées $v_i(k)$ (en tenant compte des priorités si $m_5 = 0$),
Les bilans de marquages de $m_j(k)$
L'instant de changement de dynamique t_0,
i.e., Pour toute les places m_i^D telles que $B_i < 0$:
calculer $t(i)$, instant de passage à zéro de m_i^D
Faire $t_0 = min_{i:B_i<0}(t(i))$
- Mise à jour du temps de boucle et de Tempo:
$M = M + (t_0 - t_{start})$, ($\forall k$)
$tempo = tempo - (t_0 - t_{start})$, ($\forall k$)
- Calculer m^C à l'instant t_0
- Calculer m^D
- Effectuer le franchissement des transitions discrètes
A chaque franchissement de T_8, faire: $R = R + m$,
Si $Mem(k) \leq S/2$:
Faire $autss = autss + 1$: T_7 validée
A chaque franchissement de T_7, faire: $U = U - 2m$.
Fin Si
Si $Mem(k) > S$
Faire $autca = autca + 1$: T_9 validée
$acq = acq + 1$;
A chaque franchissement de T_9, faire: $U = U - m$.
Fin Si
Si $Mem(k) < S$ et $Mem(k) > S/2$
Mettre la taille de la fenêtre à la valeur du seuil S,
A chaque franchissement de T_9, faire: $U = U - m$.
Fin Si
Si $Mem(k) = 64Ko$
Maintenir la taille de la fenêtre à la valeur courante (64Ko),
A chaque franchissement de T_9, faire: $U = U - m$.
Fin Si
Si $M_i(k) = 0$: fenêtre k transmise
Sauvegarder la valeur du temps de boucle $M_{final} \leftarrow M(k)$
Désactiver tempo(k) et remettre $M(k) = 0$.
Mettre à jour la nouvelle valeur de Tempo (Mise à jour de D,
RTT, puis $Tempo = RTT + 4D$),
Fin Si
Sauvegarde de: $m^C(k)$, $m^D(k)$, $v(k)$
Tempo, RTT, M_{final}
t_0.
Fin boucle secondaire

Si Tempo a expiré
Faire procédure d'initialisation.
Fin Si
Si la fenêtre k est envoyée, Faire $k = k + 1$ (modulo n);
Fin Si
Si $t_0 = t_{final}$:
Changement d'intervalle $t_{start} \leftarrow t_0$ et $t_{final} \leftarrow t_{start} + \delta$.
Faire $i = i + 1$ et $V_3 = v_3(i)$;
Fin Si

Fin Boucle principale

4.1.2.3 Calcul des pertes et procédure de réinitialisation

Si une des temporisations actives expire, une procédure de réinitialisation est enclenchée. Celle-ci consiste à réinitialiser les marquages de la ligne d'émission, la valeur de la temporisation et à remettre à jour la valeur du seuil d'évitement de congestion S (à savoir, à la moitié de la valeur courante de la fenêtre de congestion avant l'occurrence des pertes, voir section 2.2.1). Durant cette phase, nous calculons aussi les pertes survenues sur la ligne afin de la réémettre. Pour calculer la quantité de pertes survenue, on se sert de la quantité initiale de donnée que nous devons transmettre (appelée U_1), de la valeur de la quantité de données restant à réémettre (appelée U) et enfin de la quantité de données déjà reçues par le récepteur (appelée R). Nous disposons en effet, à chaque instant, de ces trois informations. Ainsi, la différence $(U_1 - U)$ nous indique ce qui été envoyé de l'émetteur et $((U_1 - U) - R)$ nous indique ce qui a été envoyé et pas encore reçu. Nous avons ainsi la quantité de données à réémettre. Quantité que nous ajoutons donc au buffer de l'émetteur. La nouvelle valeur de la quantité de donnée à envoyer devient: $U = U_1 - R$.
La quantité de pertes exacte peut être calculée grâce à la transitions $T"_1$ (Figure 3.37). Nous pouvons imaginer une place à la sortie de la transition $T"_1$ qui emmagasine les pertes (m_{pertes}, Figure 4.2).

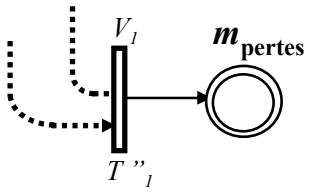


FIG. 4.2 – *Ajout de la place m_{pertes} qui compte les pertes.*

Ainsi, on peut écrire:

$$\dot{m}_{pertes} = v"_1, \tag{4.1}$$

$$m_{pertes} = m_{pertes} + v"_1(t_0 - tstart_0). \tag{4.2}$$

Avec, à l'instant initial ($t = 0$), $m_{pertes} = 0$. Les pertes s'accumulent tout au long de la transmission et la quantité exacte perdue est récupérée grâce à m_{pertes}.

⇛ ***Procédure d'initialisation***

Calcul des pertes: pertes = $U1 - U - R$

$$tot_{pertes} = tot_{pertes} + pertes$$

Faire $U = U1 - R$: pour la réemission des données perdues
Réinitialiser le marquage de la ligne d'émission,
Réinitialiser les vitesses instantanées,
Sauvegarder les nouvelles valeurs des marquages, Tempo, vitesses,...etc,
Mettre à jour la valeur du seuil S,
Remettre le numéro de fenêtre k à 1.

4.2 Hypothèses de travail

Nous allons présenter les hypothèses de travail les plus importantes concernant les simulations qui vont suivre.

Hypothèses sur l'émetteur et le récepteur Nous avons considéré dans les simulations que la capacité au niveau du récepteur n'est pas limitée, celui-ci peut donc recevoir toutes les données qui arrivent. Cette hypothèse est raisonnable vu la capacité des tampons des ordinateurs actuels. Nous avons aussi considéré que le récepteur possède un temps de traitement nul, i.e., il peut envoyer des acquittements dès lors qu'il reçoit les données. De même ce temps dépend de la vitesse de traitement des ordinateurs qui généralement est très grande. Si nous voulons considérer des temps de traitement non nul (cas de file de données à traiter très longue par exemple), il suffit de considérer une temporisation sur la transition T_5 de la figure 3.37, celle-ci représente en effet le temps de traitement du récepteur.
Au niveau de l'émetteur, celui-ci possède une quantité de données fixe à envoyer au récepteur (cas d'un envoi de fichier ou d'un envoi de page web), i.e., il n'y a pas d'arrêt et de reprise de flux (comme ce serait le cas par exemple dans le cas d'une commande à distance). Dans le cas d'arrêts et de reprise du flux, des dispositions sont nécessaires afin de remettre à jour les valeurs des temporisations et de la fenêtre de congestion (en cas de période d'arrêt, les dernières valeurs de $Tempo$ et de $cwnd$ risquent en effet de ne plus refléter l'état de la ligne). Les détails de ce type de mise à jour sont traités dans [HPF00].

Capacité de stockage Rappelons aussi que le stockage des données se fait dans le routeur, de capacité maximale C et que les canaux de transmission ne stockent pas les données (capacité 0^+).

Débits, environnement extérieur et priorités Tout d'abord, afin de mettre en évidence le phénomène de congestion sur la ligne, nous considérons les débits d'entrées dans le buffer supérieurs aux débits de sorties. Ainsi, $v_1 + v_3 > v_2 + v_4$, avec en cas de remplissage du buffer, une priorité des flux extérieurs sur le flux d'émission ($T_3 < T_1$, Figure3.37).
En réalité, certains flux extérieurs sont de priorité inférieure et d'autres de priorité supérieure au flux d'émission. Cependant, ceux qui sont de priorité inférieure ne contraignent pas le flux d'émission (en cas de charge du buffer) et nous pouvons ne considérer que ceux dont la priorité

est supérieure. Notons aussi que nous pouvons programmer une priorité variable, qui change par intervalle de temps par exemple. Le cas considéré ici d'une priorité constamment supérieure est en effet le pire cas de figure possible.
Ainsi, un profil est choisi en ce qui concerne v_3. Il s'agit d'une fonction sinusoïdale bruitée de moyenne supérieure à la vitesse de sortie extérieure v_4 garantissant ainsi qu'en moyenne les flux extérieurs remplissent le buffer. Le choix d'un sinus bruité vient du fait qu'un tel signal est un signal aléatoire tout en ayant une valeur moyenne connue et une forme qui peut être considérée comme des flux qui augmentent puis diminuent tout au long d'une transmission.

Arrivées des acquittements L'hypothèse faite sur les acquittements est que ces derniers arrivent à l'émetteur sans être retardés par la ligne. En effet, les acquittements sont envoyés à travers le réseau (pas automatiquement la ligne que l'on utilise) et sont donc soumis aux mêmes contraintes temporelles et de congestion auxquelles sont soumises les autres données. Cependant, nous pouvons considérer que cette hypothèse n'est pas forte car, dans les réseaux, et afin de ne pas perdre les acquittements, ceux-ci sont un flux prioritaires aux autres flux, de sorte que, même en cas de congestion, les acquittements arrivent quand même à destination. De même que pour le temps de traitement du récepteur, nous pouvons considérer un retard dans l'arrivée des acquittements en mettant une temporisation sur les transitions T_6 et T_8 (Figure 3.37).

4.3 Validation de modèle

La validation du modèle se fera par rapport à un comportement prévu par la théorie, à savoir, les évolutions des tailles des fenêtres et le comportement des différents flux lors de l'occurrence des pertes. Cette partie des simulations a aussi été comparée à des résultats obtenus en utilisant le simulateur de réseaux Network Simulator (NS) [FF97]. Cependant, ne pouvant supposer un profil aléatoire inconnu avec NS, nous nous sommes limité à un environnement extérieur fixe et connu. Avec des conditions de débits identiques (débits extérieurs connus), nous obtenons les mêmes résultats en utilisant notre modèle ou NS (([Jul04]), Annexe 3).

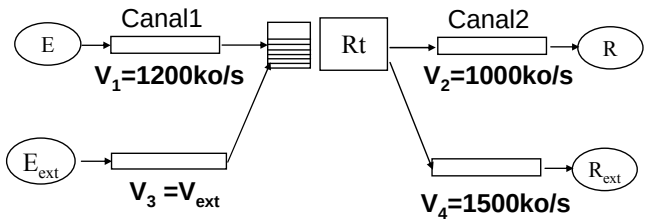


FIG. 4.3 – *Valeurs des débits de la ligne de transmission.*

Les simulations indiquent deux phases: une première phase lorsque le buffer n'est pas encore plein, les transitions sont fortement validées et les différents débits ont leur valeurs maximales. Une deuxième phase lorsque le buffer est plein: toutes les données entrantes ne trouvent pas automatiquement place dans le buffer. Ainsi, nous devons établir des règles de franchissement. Celle que nous avons adoptée ici est la priorité du flux extérieur sur le flux d'émission. La raison de notre choix est de bien voir le phénomènes de pertes sur la ligne d'émission et la

réaction des protocoles à ces pertes (évidemment, même en cas de priorité du flux d'émission, les pertes auront lieu mais avec des proportions plus faibles).

4.3.1 Avant le remplissage du buffer

Les figures 4.5, 4.6 et 4.7 représentent respectivement le débit d'entrée de la ligne d'émission (dite vitesse d'émission), le débit extérieur et l'état du buffer (v_1', v_3 et le marquage m_4+m_9 respectivement, Figure 4.4).

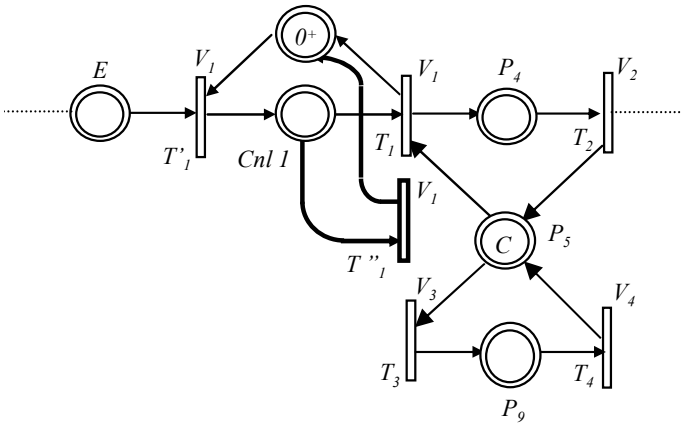


FIG. 4.4 – *Modèle RdP continu du buffer*

On remarque que la vitesse d'émission (v_1', Figures 4.5) est sous forme de créneaux. Cette forme est due à l'envoi du flux par taille de fenêtre. La première fenêtre est longue d'un segment, une fois le segment envoyé, la vitesse v_1' se remet à zéro jusqu'à l'envoi de la deuxième fenêtre. À l'envoi de celle-ci, on a à nouveau $v_1' = V_1$, puis elle est remise encore à zéro, etc. Plus la taille de la fenêtre est grande, plus v_1' est positive longtemps jusqu'à ce qu'elle forme un niveau constant à V_1.
La vitesse extérieure (v_3, Figure 4.6) est maximale.
On voit sur la figure 4.7 que la charge du buffer augmente; les fluctuations sont dues aux sinusoïdes de la vitesse extérieure.

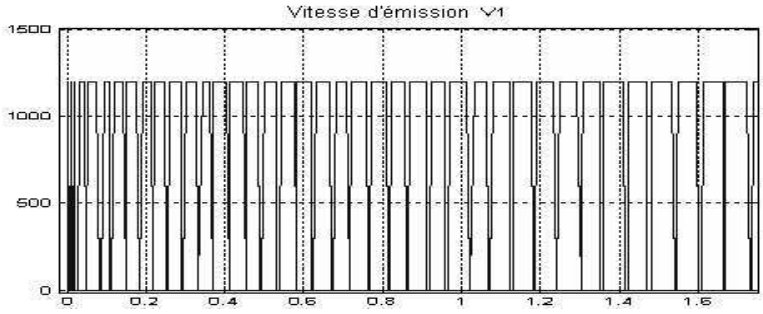


FIG. 4.5 – *Vitesse d'émission*

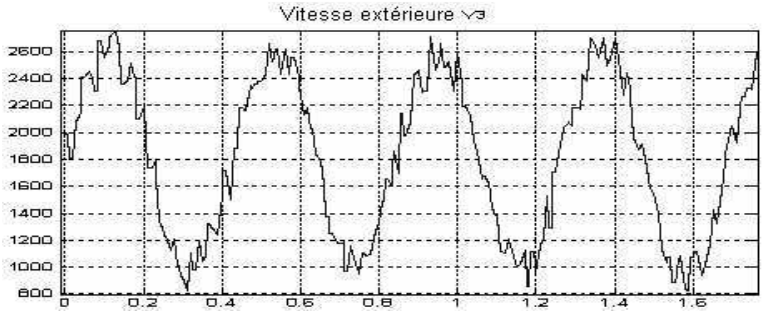


FIG. 4.6 – *Vitesse extérieure*

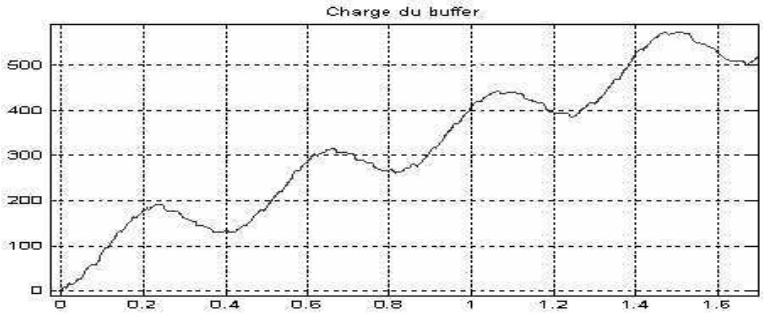


FIG. 4.7 – *Charge du buffer*

4.3.2 Après le remplissage du buffer

4.3.2.1 État du buffer

Lorsque la charge du buffer atteint sa capacité maximale C, certaines données d'émissions et extérieures commencent à se perdre. La figure 4.8 indique l'état du buffer après le début des pertes. Périodiquement, le niveau du buffer se maintient à C puis se vide légèrement avant d'atteindre C à nouveau. Si l'on observe les moments où le buffer se remplit et se vide, l'on voit que ces moments correspondent à ceux où les débits extérieurs augmentent puis diminuent (v_3, Figure 4.9). Si toutes les vitesses sont constantes, le système se stabilise dès que le buffer est plein (cas des dynamiques vues dans les graphes d'évolution des Figures 3.25 et 3.26). Les fluctuations de la vitesse extérieure font que la charge du buffer fluctue aussi. Réellement, les charge et décharge du buffer ne font plus que suivre les fluctuations du débit extérieur. On peut d'ores et déjà conclure que l'état du buffer dépend non seulement de l'intensité des débits d'entrées et de sorties mais aussi des variation de ses débits dans le temps (des conclusions similaires sont faites dans les travaux de F. Baccelli [BB98]).

4.3.2.2 La vitesse d'émission et la vitesse extérieure

Les vitesses d'émission et extérieure baissent à leur tour lorsque le buffer est plein. Ainsi, on voit sur les figures 4.10 et 4.9 qu'aux périodes où le buffer se vide un peu, ces vitesses redeviennent maximales, par contre lorsque le buffer est plein, la vitesse d'émission va jusqu'à zéro

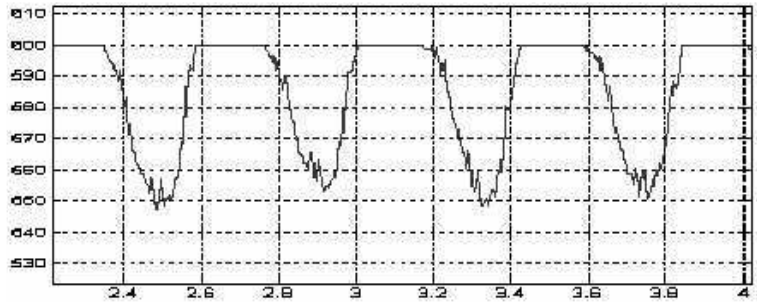


FIG. 4.8 – *L'état du buffer après le début des pertes*

tandis que la vitesse extérieure ne fait que diminuer (elle baisse jusqu'à la valeur V_4). Ceci est évidemment du à la priorité des flux extérieurs sur le flux d'émission. Ainsi, la vitesse extérieure s'aligne aux débits de sortie du buffer et la vitesse d'émission s'annule.

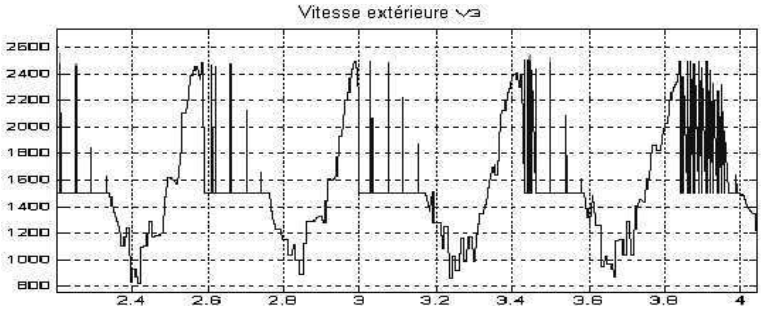


FIG. 4.9 – *Vitesse extérieure après le début des pertes.*

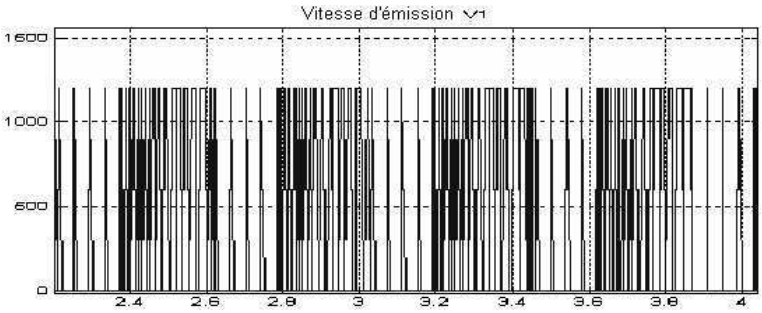
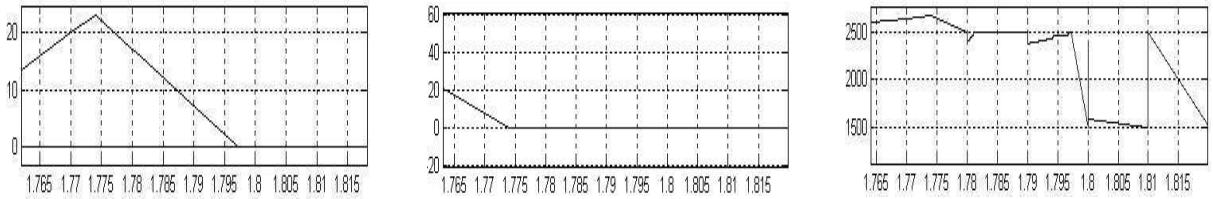


FIG. 4.10 – *Vitesse d'émission après le début des pertes.*

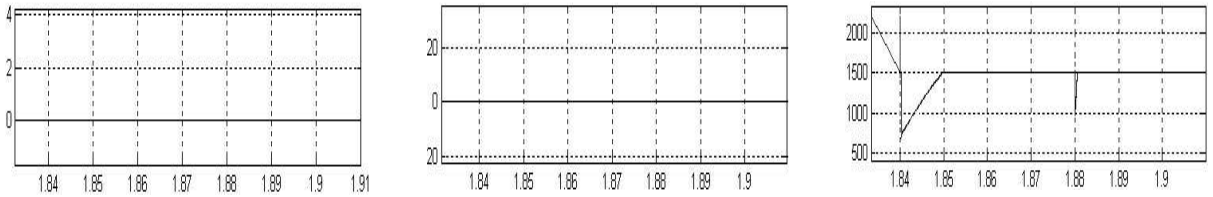
4.3.2.3 Baisse des débits d'entrée du buffer; Exemple: la vitesse extérieure v_3

Les figures 4.11.a et 4.11.b sont un agrandissement des courbes de m_4, m_5 et v_3 (Figure 4.4) respectivement pendant une période de pertes. Sur la figure 4.11.a, les marquages de m_4 et de

m_5 sont dans un premier temps positif. Durant cette période, la vitesse v_3 est maximale. Dans un second temps, le marquage de m_5 s'annule. La vitesse v_3 est inférieure ou égale à $v_2 + v_4$ (2500Koct/s). En fin de période, les marquages de m_4 et de m_5 sont nuls. Le marquage nul de m_4 entraîne l'annulation aussi de la vitesse v_2. Ainsi, la vitesse v_3 s'aligne maintenant à v_4. Cette partie est plus visible sur la figure 4.11.b où m_4 et m_5 sont nuls et v_3 est au plus égale à v_4. Ces résultats sont à comparés avec ceux obtenus dans les graphes d'évolution obtenu dans la section 3.2.2.6. Cependant, V_3 y était considérée constante, on retrouve en simulation les phases de ces graphes d'évolution, mais on passe constamment d'un IB à un autre (ou d'un graphe à un autre d'ailleurs - Figure 3.25 ou Figure 3.26) car v_3 est en constant changement. On remarque aussi (assez rarement), qu'il arrive à v_3 de dépasser la valeur $v_2 + v_4$. Ceci est purement du à des petits aléas de la simulation.



-a-



-b-

FIG. 4.11 – *Agrandissement des marquages m_4, m_5 et v_3 pendant les périodes de pertes.*

4.3.2.4 Périodes de "reprise" et de "tentative de reprise"

Lorsqu'il y a pertes de données, il y a réinitialisation du système (après expiration d'un temporisateur), i.e., la transmission reprend à partir de l'algorithme slow start avec une taille de fenêtre initiale égale à un segment (chapitre 2). Cette "reprise" est visible dans la vitesse d'émission qui a la forme de créneaux irréguliers comme c'est le cas dans tout début d'émission (Figure 4.5). Les pertes reprennent avant que v_1' ait eu le temps de se stabiliser (niveau constant à V_1), et v_1'' varie alors entre zéro et sa valeur maximale V_1. En effet, pendant les périodes de charge du buffer, l'émetteur tente d'envoyer des segments mais ceux-ci sont immédiatement perdus, le temporisateur correspondant expire, le système est réinitialisé, un nouveau segment est envoyé, perdu, etc., jusqu'à ce que le buffer se vide un peu et puisse emmagasiner à nouveau les données d'émission. Ces vains essais successifs sont visibles sur la figure 4.10 et sont appelés "tentatives de reprise", elles représentent des périodes de forte congestion. Ces deux périodes "reprise" et

"tentatives de reprise" se répètent jusqu'à ce que toutes les données aient été transmises de l'émetteur vers le récepteur.

4.3.2.5 Les pertes

La figure 4.12 représente les instants et la vitesse de franchissement de la transition $T"_1$ de la figure 4.4. Elle nous indique donc l'occurrence exacte des pertes sur la ligne. L'on peut remarquer les vitesse des pertes est nulle pendant les périodes "reprise" tandis qu'elle est positive sur les périodes "tentatives de reprises". Sachant les périodes où la vitesse des pertes est positives, nous pouvons calculer la quantité exacte de pertes survenu tout au long de la transmission.

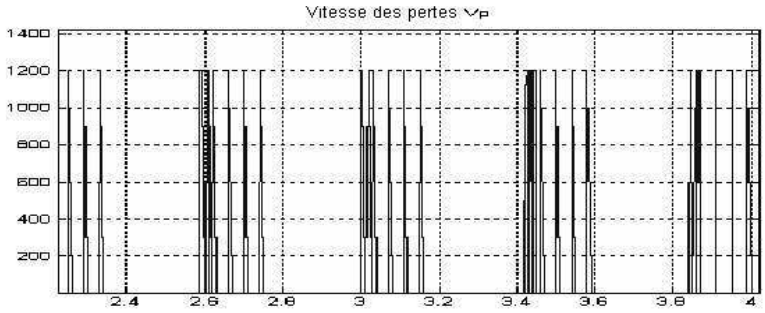


FIG. 4.12 – *Vitesse des pertes*

4.3.3 Évolution de la taille de la fenêtre de congestion

La figure 4.13 montre l'évolution de la taille de la fenêtre de congestion $cwnd$. Tant qu'il n'y a pas de pertes sur la ligne, la taille de la fenêtre évolue d'abord de façon exponentielle (agrandissement de cette partie sur la Figure 4.14.a), puis de façon linéaire (agrandissement de cette seconde phase sur la Figure4.14.b). Au moment où les pertes commencent, le système est réinitialisé et la fenêtre est remise à un (un segment), sa valeur initiale. La transmission reprend à slow start. Après le début des pertes (Figure 4.13), il y a des périodes de reprise de transmission et d'autres où la fenêtre ne dépasse pas la taille d'un segment. Ces deux phases correspondent à celles de pertes et de reprises successives dues aux variations de v_3. Ce sont les phases "reprise" et "tentative de reprise" expliquées précédemment. Ces vains essais successifs de la phase "tentative de reprise" sont visibles sur la figure 4.13 (phase "Après le début des pertes") et sur la figure 4.14.c où l'on voit la présence de petit niveaux successifs (un segment). Lorsque le buffer est à même de recevoir des données (autre que le flux extérieur), le flux reprend mais la transmission se fait selon congestion avoidance sans passer par slow start. En effet, slow start doit être appliqué jusqu'à atteindre le seuil d'évitement de congestion. Or, à chaque réinitialisation de système, ce seuil est mis à jour selon l'équation: $S = max(cwnd/2,\ 2segments)$. Après une tentative de reprise, la fenêtre maximale est d'un segment et ainsi le seuil prend la valeur 2 et l'évolution de la fenêtre de congestion ne peut être que linéaire.

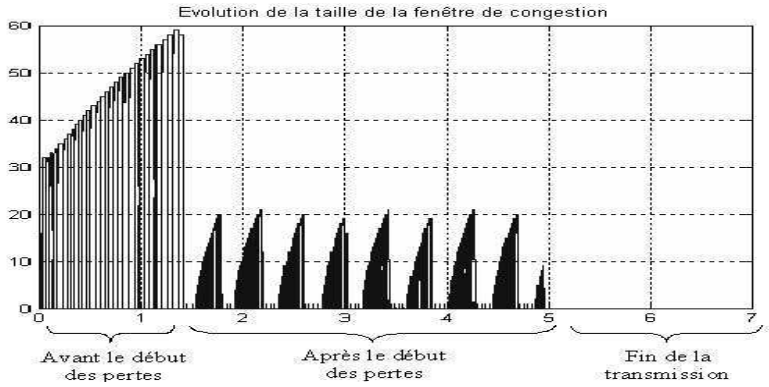
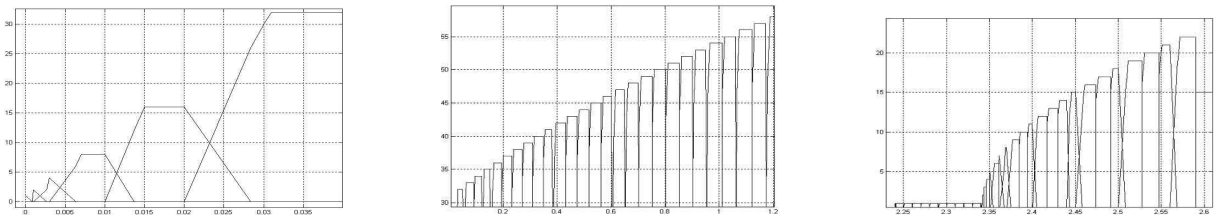


FIG. 4.13 – *Evolution de la taille de la fenêtre de congestion:* $cwnd$*.*



-a- -b- -c-

FIG. 4.14 – *Agrandissement des phases slow start, congestion avoidance et reprise après pertes.*

4.4 Critères de performances

Afin d'évaluer le comportement de la ligne, certains critères sont pris en compte tels que les délais de transmission, les retards et les pertes. Ces critères semblent être les plus pertinents pour mesurer les performances d'une ligne de communication et indiquer de façon globale le comportement du système.

Pour cela, nous allons commencer par analyser les temps de transmission, sachant qu'il s'agit du temps global mis par les données pour arriver jusqu'au récepteur.

4.4.1 Temps de transmission

Le temps de transmission global est directement lisible dans les simulations. En effet, c'est l'instant où la fenêtre de congestion (ou la vitesse d'émission ou la vitesse à la réception) est constante à zéro. Notons cependant que l'instant le plus précis est celui où la vitesse à la réception s'annule, mais s'il existe un blocage sur la ligne, cette non arrivée de données à la réception risque de ne pas être significative de la fin d'une transmission. Ainsi, l'instant exact est celui où la vitesse à la réception et la fenêtre de congestion sont nulles. La figure 4.15 représente le délai de transmission de la quantité totale de données $U = 3Moct$. Ce paramètre est mesuré plusieurs fois en faisant varier la taille du buffer C; tous les autres paramètres restent constants. Ce calcul est effectué pour voir l'influence de la capacité des buffers sur les délais de transmission.

Lorsque la taille du buffer est grande, les pertes se produisent de façon tardive car le buffer met du temps pour se remplir. Ainsi, peu de pertes se produisent et le système perd moins de temps dans les retransmissions. Lorsqu'au contraire, C est faible, beaucoup de pertes se produisent et beaucoup de données doivent être retransmises, les délais de transmission se voient retardés. En conclusion, les délais de transmission sont faibles lorsque C augmente.

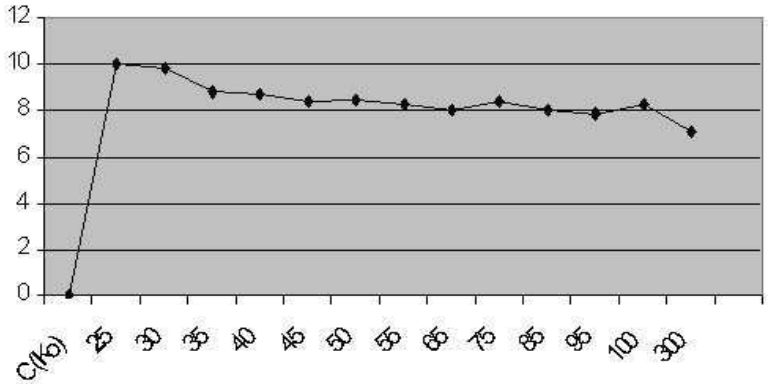


FIG. 4.15 – *Variation des délais de transmission en fonction de la capacité du buffer C.*

4.4.2 Taux de pertes

Le taux de pertes représente le rapport entre la quantité de pertes (vues par les protocoles) sur la quantité de données totale envoyée. Les pertes vues par les protocoles sont celles détectées après expiration des temporisateurs. Le nombre de pertes est alors calculé comme expliqué précédemment dans la section 4.1.2. En faisant varier le paramètre C, ce taux de pertes nous donne une appréciation de la quantité de données que le système a du perdre et donc retransmettre. Ainsi, le taux de pertes est calculé de la façon suivante:

$$Taux\ de\ pertes \quad = \quad 100 \frac{total\ des\ pertes\ durant\ la\ transmission}{total\ des\ donnes\ envoyes}.$$

La figure 4.16 nous indique la variation du taux de pertes en fonction de la capacité C du routeur. On voit que ce taux augmente au fur et à mesure que la capacité diminue. Ce résultat rejoint celui relatif aux délais de transmission et les deux sont liés car les délais dépendent aussi de la quantité de pertes sur la ligne.

4.4.3 comparaison entre les pertes réelles et les pertes vues par le protocole

Nous avons décris précédemment les pertes vues par les protocoles comme étant celles détectées suite à l'expiration des temporisateurs. La quantité de pertes est alors calculée comme étant la quantité encore présente au niveau de l'émetteur (données restant à envoyées) à laquelle on soustrait la quantité déjà arrivée au niveau du récepteur ($U - R$, paragraphe 4.1.2). L'on peut se douter que cette quantité est la quantité maximale de pertes possible. En effet, elle inclut

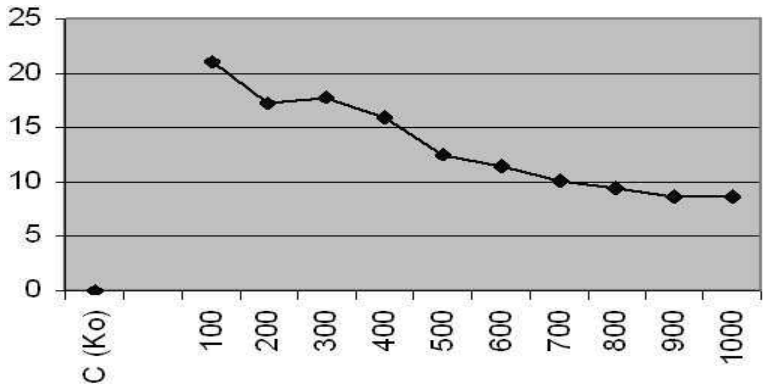


FIG. 4.16 – *Variation du taux de pertes en fonction de la capacité du buffer* C.

les pertes réelles et les données encore présente dans le buffer mais qui n'ont pas eu le temps d'arriver au récepteur.
Du point de vue modèle, la transition $T"_1$ (Figure 3.37) représente la quantité de données réelles qui n'ont pas trouvés place dans le routeur, i.e., les pertes réelles. Une comparaison pertes réelles / pertes vues par les protocoles est effectuée et donnée par la figure 4.17. La première observation est que la courbe qui représente les pertes vues les protocoles est constamment au dessus de la courbe des pertes réelles. Ce résultat est prévisible. D'autre part, en faisant varier la capacité du routeur C, la différence entre les deux courbes croît. En effet, les pertes estimées par les protocoles comprennent aussi les données encore présentes dans le buffer. Or ces dernières sont d'autant plus nombreuses que la taille du buffer est grande. Il apparaît cependant que cette quantité de pertes du protocole qui augmente avec la taille du buffer n'est tout de même pas assez grande pour altérer les performances en termes de délais de transmission (délais plus faibles lorsque C est grand).
Cette comparaison pertes réelles / pertes vues par les protocoles peut aussi servir de point d'évaluation des performances des protocoles mêmes. En effet, plus les protocoles sont bien adaptés à l'état du réseau, plus les deux courbes (pertes réelles, pertes protocoles) seront proches.

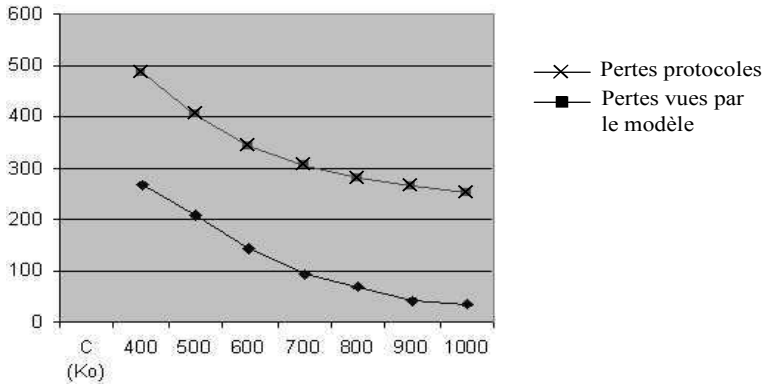
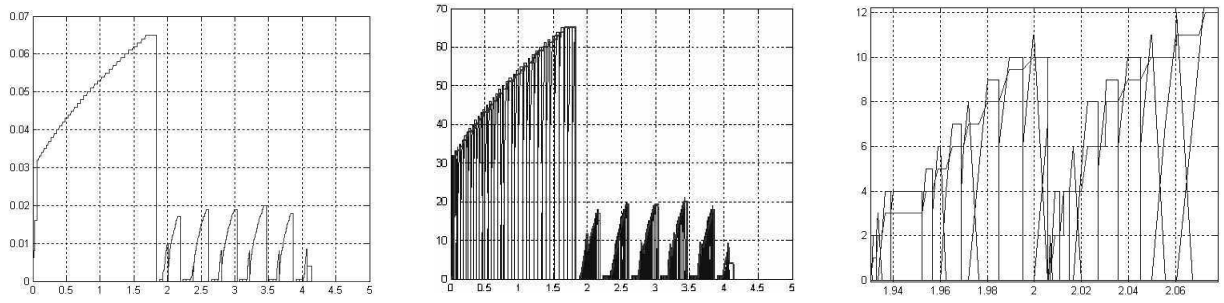


FIG. 4.17 – *Comparaison entre les pertes réelles et celles vues par les protocoles.*

4.5 Contrôle de congestion

Les différents protocoles de congestion décris jusque là se servent de temporisateurs pour déduire ce qui paraît être des pertes sur une ligne de communication. Cela est dû au fait indéniable que l'on a pas d'informations à priori sur l'état de la ligne (notamment des routeurs). Ces méthodes se basent donc sur les temps de retour des acquittements pour deviner l'état du réseau. L'amélioration possible pour l'instant réside dans le temps de détection des pertes: on doit déduire les pertes le plus vite possible afin de baisser les débits et réduire les pertes. Ces méthodes dites rapides se basent non pas seulement sur l'expiration de temporisateurs mais aussi sur l'ordre d'arrivée des acquittements pour déduire une congestion. Un exemple d'un tel algorithme a été donné dans le chapitre 2 (section 2.2.1): il s'agit de l'algorithme fast recovery. De façon générale, les acquittements envoyés par le récepteur portent le numéro de la prochaine séquence attendue. En appliquant fast recovery, au bout du 3ème acquittement reçu portant le même numéro de séquence, on déduit une perte sur la ligne. Cette méthode est dite plus rapide que d'attendre l'expiration des temporisateurs mais possèdent ses avantages et ses inconvénients (la description de cet algorithme est dans le chapitre 2). Des travaux d'estimation de l'état d'un réseau se base aussi sur les temps de boucle des données sur une ligne (temps séparant l'envoi d'une donnée et la réception de son acquittement). Ce temps de boucle est effectivement significatif de l'état du routeur: plus il est long plus le routeur est chargé. Nous nous sommes ainsi intéressé aux différentes allures des paramètres des protocoles, à savoir, les temporisations, les RTT et le temps de boucle (Annexe 3). Il s'est effectivement avéré que le temps de boucle présente l'allure la plus intéressante et notamment en la comparant avec celle de la fenêtre de congestion déjà décrite plus haut.



-a- -b- -c-

FIG. 4.18 – *Observation de l'évolution du temps de boucle et sa comparaison avec la fenêtre de congestion.*

À titre indicatif, des courbes de M (le temps de boucle), de $cwnd$ (la fenêtre de congestion) et la superposition des deux sont données dans la figure 4.18 respectivement. On y voit la parfaite concordance entre le temps de boucle et la taille de la fenêtre de congestion (notamment dans l'agrandissement de la superposition des deux courbes, Figure 4.18.c où la valeur de M est amplifiée pour être visible devant celle de $cwnd$). La courbe de M évolue selon $cwnd$ lorsque la transmission est possible (hors pertes) puis elle est réinitialisée lorsqu'il y a des pertes. Nous allons voir qu'en nous basant sur les temps de boucle, des conclusions sur l'état de la ligne

peuvent faites à chaque réception d'acquittement et qu'il n'est pas nécessaire d'attendre l'arrivées de 3 acquittements.
Attardons-nous un peu plus sur M et observons son évolution exacte.

4.5.1 Détermination analytique du temps de boucle M

Afin de mieux voir l'évolution de M, nous avons choisi de changer les paramètres de notre ligne, i.e., les valeurs des vitesses (v_3 et v_4) et du buffer (C) de telle sorte qu'il n'y ait pratiquement pas de pertes durant la transmission. C'est une façon de ralentir les dynamiques de la ligne et ainsi d'observer les phénomènes de façon plus précise. Nous obtenons les évolutions de M et de $cwnd$ telles que décrite par les figures 4.19 et 4.20. On observe une évolution exponentielle puis linéaire de la fenêtre $cwnd$ et vu qu'il n'y a pas de pertes, la fenêtre atteint 64 Koct qui est le niveau maximal admissible pour un segment TCP (voir chapitre 1). On voit sur la figure 4.19 au niveau des agrandissements des parties exponentielle et linéaire que le temps de boucle M varie lui aussi de façon exponentielle puis linéaire. Ces deux évolutions de M sont bien visibles dans la figure 4.20 ou seulement l'allure de M est mise en évidence.

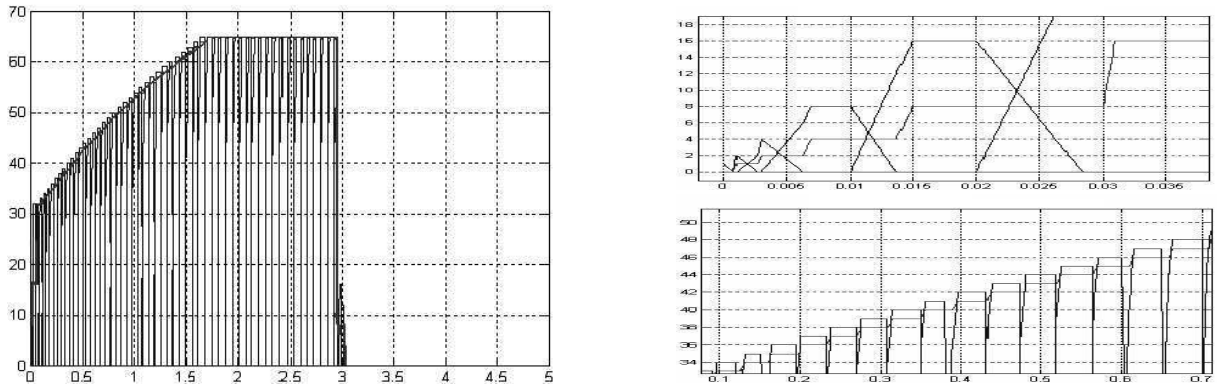


FIG. 4.19 – *Observation de l'évolution du temps de boucle et sa comparaison avec la fenêtre de congestion.*

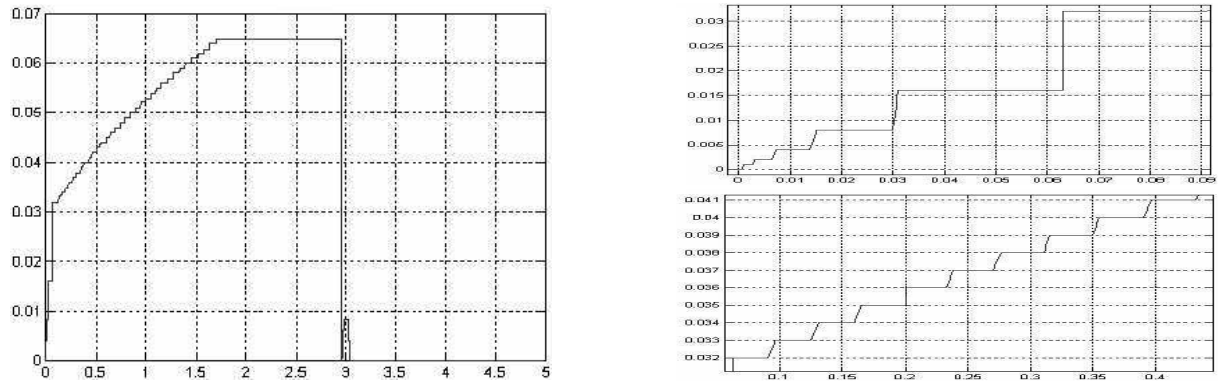


FIG. 4.20 – *Observation de l'évolution du temps de boucle au long d'une transmission.*

La courbe de M est proportionnelle à celle de la fenêtre de congestion $cwnd$. Ainsi, lorsque $cwnd$ double - phase exponentielle -, le temps de boucle M double aussi et lorsque $cwnd$ aug-

mente de l'équivalent d'un segment - phase linéaire -, M augmente aussi d'une valeur constante équivalente comme on l'a observé dans les simulations au temps de boucle d'un segment.
Il s'avère donc que pour ces deux phases, connaissant la première valeur de M, il suffit d'un calcul analytique pour tracer la courbe de M lorsque la transmission est possible (hors pertes).

Détermination des abscises et ordonnées de M Nous reprenons l'évolution de M observée lors des simulations. Les ordonnées de la figure 4.20 représentent la valeur de M tandis que les abscises représentent les instants d'arrivée des acquittements (donc l'instant où l'on a pu calculer M) (l'Annexe 1 présente les détails des résultats ci-dessous).
Notons M_i les valeurs successives de M données par la figure 4.20 et on note t_i les instants respectifs d'occurrence des M_i.

– *Pendant la phase exponentielle*
Observation 1: La **valeur de M** double à chaque transmission. Après calculs (voir Annexe 1), on aboutit à l'expression suivante des valeurs de M:

Résultat 1:

$$M_i = 2^{i-1} M_1. \tag{4.3}$$

avec M_1 la première valeur du temps de boucle M, donc à l'envoi du premier segment et M_i la $i$ème valeur de M, donc à l'envoi de la $i$ème fenêtre.

Observation 2: De même que pour la valeur de M, l'**intervalle qui sépare deux valeurs successives de M** double à chaque transmission. Nous pouvons donc écrire la valeur absolu des instants t_i sous la forme:

Résultat 2:

$$t_i = (2^{i-1} + 2^{i-2} + ... + 2 + 1) M_1. \tag{4.4}$$

Notons cependant que le calcul des M_i et des t_i selon une évolution exponentielle s'arrête lorsqu'on atteint le seuil d'évitement de congestion S que l'on connaît au préalable.

– *Pendant la phase linéaire*
Observation 3: les valeurs successives de M augmente d'une valeur constante à chaque nouvelle transmission. Cette valeur constante est égale à la valeur de M après l'envoi de la première fenêtre (M_1). En effet, la taille de la fenêtre de congestion augmente d'un segment à chaque nouvelle transmission. De ce fait, la durée nécessaire à la réception d'une nouvelle fenêtre est supérieure à la durée précédente du temps nécessaire à réception du segment supplémentaire.

Résultat 3: Après calculs, les valeurs successives de M s'écrivent sous la forme:

$$M_{k+j} = jM_1 + M_k. \tag{4.5}$$

avec M_k la dernière valeur de M calculée pendant la phase exponentielle et l'indice j est tel que $j \geq 1$.

Observation 4: Pour les mêmes raisons que celles citées dans l'observation 3, chaque intervalle de temps qui sépare le calcul de deux valeurs de M est plus grand que le précédent de la valeur M_1.

Résultat 4: Après calculs, la valeur absolu des instants t_i s'écrivent sous la forme:

$$t_{k+j} = j(2^{k-1} + 1)M_1 + [(j-1) + ... + 2 + 1]M_1 + t_k. \tag{4.6}$$

Ainsi, à partir des équations 4.7, 4.8, 4.9 et 4.10, l'on peut dire que si l'on connaît le premier temps de boucle M_1, on peut prédire la courbe d'évolution total de M lorsqu'on a pas de pertes sur la ligne.

4.5.2 Problème de la gestion des files d'attente "FIFO" des messages dans un routeur

Dans notre calcul précédent des temps de boucle, les données d'émission qui entrent dans le buffer en sortent lorsque toutes les données d'émission présentes en sont sorties. Autrement dit, si l'on regarde la figure 4.21.a, une donnée qui entre dans la place P_4 attend la sortie de la quantité de marquage m_4 avant de sortir du buffer. Ce comportement correspond à celui de la figure 4.21.b où la quantité présente dans la file d'attente correspond au marquage m_4 de la figure 4.21.a.

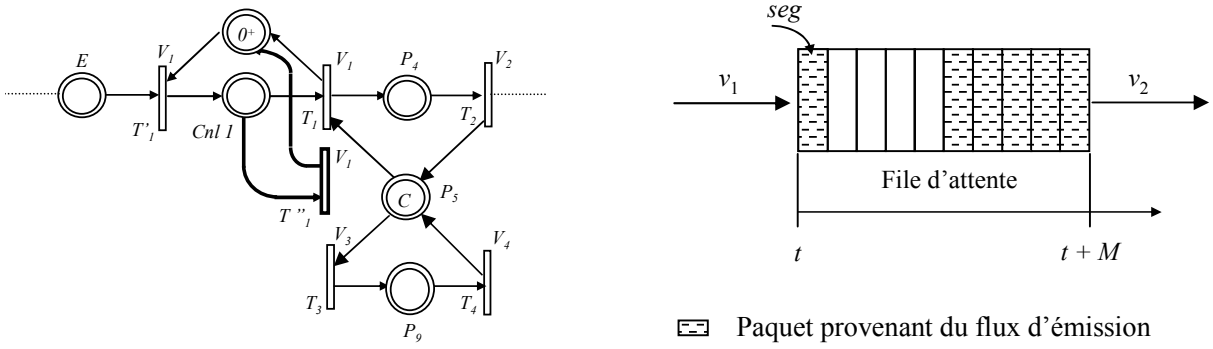


FIG. 4.21 – *a) Modèle RdP continu du buffer, b) File d'attente des données d'émission*

Cette modélisation prend en compte le partage du buffer en terme de capacité.
En réalité, lorsqu'une donnée d'émission entre dans le buffer, elle y trouve aussi bien des données d'émission que des données extérieures. Il existe donc deux types de flux dans le buffer et ces deux flux vont se gêner mutuellement soit à la sortie du buffer soit à son entrée. Il s'ensuit que l'instant de sortie d'une donnée peut dépendre, selon la politique d'organisation des flux

dans le buffer, de la sortie de toutes les données qui l'ont précédé, d'émission soit-elle ou extérieure. Tout dépendra donc de la gestion de file d'attente adopté.

Une modélisation plus fine du routeur consisterait donc à intégrer et le partage de sa capacité globale et cette gestion de file d'attente. Regardons de plus près la gestion des files d'attente dans les routeurs.

4.5.2.1 Fonctionnement interne d'un routeur

Le fonctionnement interne d'un routeur est globalement décrit par la figure 4.22. Celle-ci représente un noeud de transfert (dans notre cas un routeur) ayant 3 entrées et 3 sorties. Les différentes entrées sont d'abord dirigées vers une première file d'attente qui a pour fonction de décider du choix de la ligne de sortie [Puj05]. Une fois la décision prise, la donnée est dirigée vers une des 3 files de sorties.

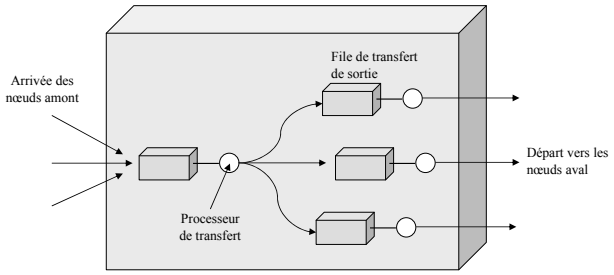


FIG. 4.22 – *Fonctionnement interne d'un noeud de transfert.*

Gestion de file d'attente de type FIFO Si la gestion de la file d'attente est de type FIFO (First In First Out, appelée "premier arrivé, premier servi"), les données en sortie sont sélectionnées selon leur ordre d'arrivée.
Si l'espace mémoire devient insuffisant, on choisit une *règle de rejet de paquet* (*packet discarding policy*, [Puj05]). La file d'attente dans le routeur peut être modélisée comme sur la figure 4.23.

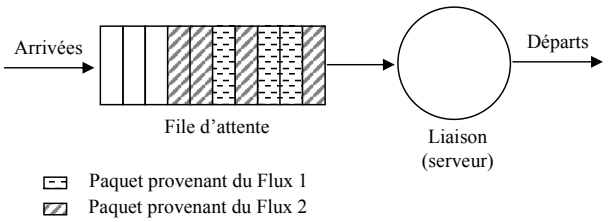


FIG. 4.23 – *Système de mise en attente FIFO.*

Gestion de file d'attente selon les priorités Si la règle de mise en attente des données se fait selon un ordre de priorité entre les flux, les données sont classées dans une catégorie de priorité définie pour la liaison (cette priorité est prise en compte dans le classement des données en sorties). Généralement, chaque catégorie de priorité dispose de sa file d'attente (figure 4.24). Au moment de sélectionner la prochaine donnée à transmettre, la règle de priorité choisit la donnée appartenant à la classe de priorité la plus haute. Si toutes les données sont de même priorité, la sélection se fait selon la règle FIFO. Un modèle de mise en file d'attente avec respect des priorités est illustré dans la figure 4.24. Finalement, l'on peut dire que le flux de la plus haute priorité n'est pratiquement limité que par les débits de sortie du routeur tandis que les autres flux seront aussi limités par la quantité de données présente dans les files de priorité supérieure. Certaines politiques de *rejet de paquets avec priorité* rejettent des paquets de moindre priorité déjà présents dans la file afin de les remplacer par des paquets de plus forte priorité.

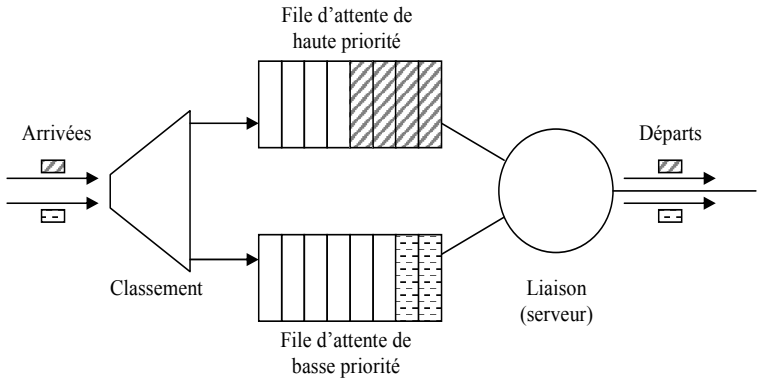


FIG. 4.24 – *Mise en file d'attente avec respect des priorités.*

Le modèle adopté dans la modélisation du buffer Dans le cas du modèle du routeur tel que donné sur la Figure 4.21.a, nous avons considéré que tant que le buffer peut contenir les données qui arrivent, la gestion de file d'attente considérée est de type FIFO. En cas de remplissage du buffer, une priorité est donnée au flux extérieur: *La priorité dans ce cas sert de politique de rejet de paquets*. Autrement dit, lorsque le buffer ne peut recueillir toutes les données qui arrivent, celles provenant du flux extérieur sont prises (à condition de pouvoir les contenir) tandis que celles provenant de la ligne d'émission sont rejetées.

4.5.3 Comparaison entre le temps de boucle réel et le temps de boucle calculé

Revenons au temps de boucle sur la ligne. Auparavant, les calculs étaient faits sur la base que la durée de séjour d'un paquet dans le buffer dépend de la quantité de donnée présente de ce même flux (le flux d'émission dépend du marquage m_4, celui extérieur dépend du marquage m_9, Figure 4.21.a). La présence des deux flux dans le routeur intervient en terme de partage de la capacité totale du routeur et de la quantité globale de données. Si on modélise le routeur en

intégrant la gestion de file d'attente fifo, le dernier segment d'émission entré dans le buffer va attendre le temps nécessaire à toutes les données présentes dans le buffer ($m_4 + m_9$) avant d'en sortir. Ce temps est appelé **temps d'attente** d (chapitre 1) et est directement dépendant de la charge du buffer et des débits en sortie. En effet, si la liaison est libre et la file d'attente vide, ce temps d'attente sera nul. Si à l'inverse, le trafic est dense et de nombreux paquets sont en attente, ce temps va augmenter.
Au temps de boucle calculé jusque là va donc s'ajouter un délai supplémentaire dû au marquage m_9 (Figure 4.21.a).
Supposons que cette gestion de file d'attente est prise en compte dans le système, le temps de boucle résultant ne peut être que supérieur (cas de présence du flux extérieur dans le buffer) ou égal (cas de données extérieures nulles) au temps de boucle calculé précédemment. Il peut être observé, donc connu, à chaque retour d'acquittement (mais pas avant). Considérons la figure 4.25 où sont illustrés d'une part le temps de boucle tel que calculé dans la section 4.5, et d'autre part un temps de boucle *réel* que l'on appelle M_f observé. L'écart δ entre les deux valeurs M et M_f représente la différence entre les temps d'attente dans le buffer dans chacun des deux cas.

D'une part, nous pouvons tracer la courbe de M dès la réception du premier acquittement (toutes les valeurs de M dépendent en effet de la toute première M_1) (voir la section 4.5.1). D'autre part, nous observons la valeur de M_f à la réception de chaque acquittement. Il s'ensuit qu'à chaque acquittement reçu, nous avons la différence exacte entre la valeur calculée de M et celle non prévisible de M_f. Or, cette différence est directement en relation avec la charge du buffer. Nous pouvons conclure que nous avons un indice important relatif à la charge du buffer.

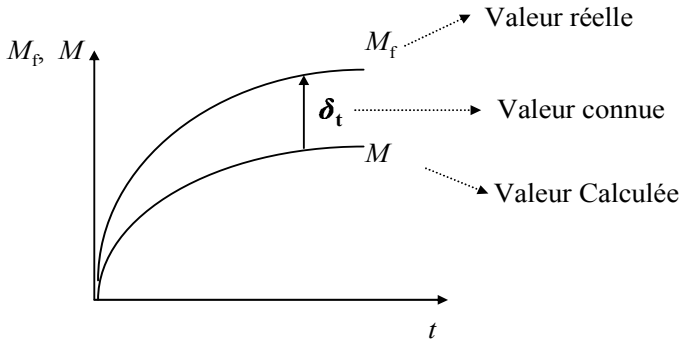


FIG. 4.25 – *Comparaison entre les temps de boucle réel calculé.*

Une étude de l'exploitation de la différence entre les deux temps de boucle est entamée et les premiers résultats sont donnés en Annexe 2. Parmi ces résultats, la déduction à priori à chaque intervalle de temps fixe du taux d'occupation du buffer par les données extérieures. Un début de formalisation des résultats montre la pertinence des conclusions qui peuvent être faites en comparant les deux modèles. Ce premier pas nous indique l'importance de pouvoir intégrer dans notre modèle la gestion de données FIFO. Pour cela, nos efforts futurs vont se concentrer sur ce dernier point en vue d'une part de donner avec précision le comportement des flux à l'intérieur des routeurs et d'autre part, d'exploiter les nouvelles informations qui en

découleront pour le contrôle de congestion.

Conclusions et perspectives

Nous avons traité tout au long de ce travail de thèse aussi bien de réseaux de communication que de contrôle de congestion en prenant en compte la modélisation de phénomènes complexes de différentes natures présents dans les réseaux. Ce travail s'insère dans le cadre du projet NECS (Nework Controlled Systems, projet CNRS) dont la thématique globale est le contrôle par les réseaux.

Les réseaux et plus particulièrement Internet sont devenus un outil de travail et d'échange présent dans le quotidien de tous. Cette infrastructure de niveau mondiale requiert l'intégration d'une organisation pointue et hiérarchique, d'un contrôle puissant et d'études de performances et de qualité de service à tous les niveaux des réseaux.
Pour notre part, nous nous sommes penchés durant ce travail de thèse sur les réseaux TCP/IP et plus particulièrement sur le contrôle des pertes et l'évaluation de performances d'une ligne de transmission. Avant d'entamer un tel travail, nous avons d'abord modélisé le système considéré.

Les travaux trouvés dans la littérature qui traitent de performances des réseaux se basent dans l'établissement de leur modèle sur une connexion dite de *bout en bout* (les informations dont on se sert proviennent de l'émetteur ou du récepteur), ou sur une étude des liens présents sur la ligne (comme les routeurs, méthodes dites *router based*). Cet état de fait nous a poussé à apporter un modèle qui représente une ligne complète de l'émetteur au récepteur en passant par les liens. Afin de tenir compte de l'influence des autres émetteurs qui peuvent se partager les ressources de cette ligne (notamment le routeur), un environnement extérieur est ajouté au modèle. Une entrée supplémentaire est ajoutée au routeur qui représente la somme de toutes les entrées possibles. Cette entrée possède une allure aléatoire à l'image de ce qui peut se produire sur Internet comme connexion extérieures. Le modèle complet représente une *ligne entière* soumise à l'*environnement extérieur* Internet.
Deux parties distinctes apparaissent dans la modélisation d'une telle ligne. D'une part, les données (succession de bits) sont transmises à des débits généralement de l'ordre du mega bits / secondes. Ce rapport débit/taille de données nous a fait considérer qu'un flux continu peut représenter une très bonne approximation du passage discret de bits (ou d'octets). D'autre part, la transmission est régie par les protocoles de contrôle de flux du TCP/IP qui sont constitués d'un ensemble de consignes discrètes (arrivées d'acquittements, expiration de temporisateurs, etc.). Ce modèle comprend donc des dynamiques continues de par le passage de flux de messages sur les lignes et des dynamiques discrètes de par les consignes discrètes des protocoles de contrôle. Ces deux parties interagissant entre elles pour donner une dynamique continue ponctuellement modifiée par une dynamique discrète. Les dynamiques des réseaux sont donc hybrides.
D'une part, des travaux passés de modélisation de systèmes à flux continus par RdP continus

montrent que ces derniers modélisent de manière satisfaisante de tels systèmes. Nous avons effectué une analogie entre l'écoulement d'un flux à travers un système à réservoirs et une transmission de message via un réseau et nous avons montré que ces deux systèmes ont des comportements semblables. Nous pouvons donc conclure qu'un flux de messages dans un réseau peut être modélisé par des RdP continus. D'autre part, la partie discrète des réseaux peut être aisément modélisée par les RdP discrets. Finalement, notre choix d'outil de modélisation s'est porté sur les RdP hybrides et nous avons pu mettre en évidence des phénomènes complexe tels que la charge des buffers, la baisse des débits à l'entrée des routeurs, la séparation des flux, la gestion des priorités ou encore les pertes. Afin d'évaluer les performances de la ligne et d'étudier son comportement face aux changement des conditions de transmission, un large panel de simulations a été effectué. Nous avons pu visualiser des performances telles que les débits, les délais de transmission, les taux de pertes. Les différents paramètres qui gèrent l'évolution d'une transmission tels que les temps de boucle et les temporisateurs ont aussi été visualisés. Une comparaison entre les pertes réelles produites sur la ligne et celles calculées par les protocoles est possible grâce au modèle RdP hybride qui nous donne la quantité exacte de données perdues. Cette comparaison constitue un bon indice d'évaluation de performances des protocoles.
Enfin, et grâce aux résultats de simulations de notre modèle, nous avons pu observer les temps de boucle des données sur la ligne et arriver à une modélisation analytique de ces derniers. Tant qu'il n'y pas de pertes sur une ligne, ce temps de boucle suit une allure fixe que l'on peut calculer dès lors que nous possédons la toute première valeur du temps de boucle.

Dans les routeurs tels que modélisés dans ce présent travail, la présence de flux différents intervient en terme de partage de la capacité totale du routeur. Une modélisation plus précise des routeurs doit tenir compte de la gestion de file d'attente des données en sortie. Un premier travail de prise en compte de file d'attente FIFO dans les routeurs a été effectué avec comme objectif d'intégrer les délais supplémentaires générer par la file dans le modèle RdP. Une comparaison entre les résultats obtenus avec ou sans prise en compte la file d'attente permet d'arriver à une connaissance à priori des taux d'occupation des buffers sur une ligne de transmission. Une formalisation des ces travaux constitue une perspective immédiate de la continuité de ces travaux de recherche.

Tout les flux présents sur Internet ne sont pas soumis aux mêmes règles et protocoles. Le protocole UDP (User Datagram Protocol) par exemple ne gère pas la retransmission des données perdues. Les flux UDP ne sont donc pas soumis aux baisses de débits imposées aux flux TCP en cas de pertes. Un contrôle sans pertes peut donc engendrer des temps de transmission long et une baisse des performances sans pour autant empêcher les buffers de se remplir par d'autres flux. Des travaux en perspectives consistent à nous pencher sur la détection des flux qui causent la congestion et d'agir directement sur ces derniers.

Annexe 1:
Présentation détaillée du calcul des temps de boucle M

– *Pendant la phase exponentielle*, la **valeur de** M double à chaque transmission. Après calculs, on aboutit à l'expression suivantes des valeurs de M:

$$\begin{aligned} &M_2 = 2M_1, \\ &M_3 = 2M_2 = 4M_1, \\ &M_4 = 2M_3 = 8M_1, \\ &... \\ &M_i = 2^{i-1}M_1. \end{aligned} \tag{4.7}$$

avec M_1 la première valeur du temps de boucle M, donc à l'envoi du premier segment et M_i la $i$ème valeur de M, donc à l'envoi de la $i$ème fenêtre.
De même, les instants d'occurrence de M sont sous la forme:

$$\begin{aligned} &t_1 = M_1, \\ &t_2 - t_1 = 2t_1 \Rightarrow t_2 = 3t_1, \\ &t_3 - t_2 = 2(t_2 - t_1) \Rightarrow t_3 = 7t_1, \\ &... \\ &t_i = (2^{i-1} + 2^{i-2} + ... + 2 + 1)t_1. \end{aligned} \tag{4.8}$$

– *Pendant la phase linéaire* et à partir des résultats de simulation, les valeurs successives de M s'écrivent:

$$\begin{aligned} &M_{k+1} = M_k + M_1, \\ &M_{k+2} = M_{k+1} + M_1 \Rightarrow M_{k+2} = 2M_1 + M_k, \\ &... \\ &M_j = jM_1 + M_k. \end{aligned} \tag{4.9}$$

avec M_k la dernière valeur de M calculée pendant la phase exponentielle. Ainsi, M_{k+1} est la première valeur de M calculée pendant la phase linéaire.

Supposons que α est le premier intervalle de temps qui sépare deux instants successifs de calcul de M pendant la phase linéaire, alors:
$\alpha = (t_k - t_{k-1}) + t_1 = (2^{k-1} + 1)t_1$, dans la phase linéaire, chaque intervalle est plus grand que le précédent de la valeur t_1.

Ainsi,

$$
\begin{aligned}
&t_{k+1} = \alpha + t_k, \\
&(t_{k+2} - t_{k+1}) - (t_{k+1} - t_k) = t_1, \Rightarrow t_{k+2} = 2\alpha + t_1 + t_k, \\
&(t_{k+3} - t_{k+2}) - (t_{k+2} - t_{k+1}) = t_1, \Rightarrow t_{k+3} = 3\alpha + 3t_1 + t_k, \\
&(t_{k+4} - t_{k+3}) - (t_{k+3} - t_{k+2}) = t_1, \Rightarrow t_{k+4} = 4\alpha + 6t_1 + t_k, \\
&... \\
&t_{k+j} = j\alpha + [(j-1) + ... + 2 + 1]t_1 + t_k. \qquad (4.10)
\end{aligned}
$$

Ainsi, à partir des équations 4.7, 4.8, 4.9 et 4.10, l'on peut dire que si l'on connaît le premier temps de boucle t_1 (ou M_1), on peut prédire la courbe d'évolution total de M lorsqu'on a pas de pertes sur la ligne.

Annexe 2
Gestion de la file d'attente FIFO des messages dans un routeur

Cette Annexe représente un début d'étude de la différence entre le temps de boucle calculé de façon analytique lorsque la gestion de file n'est pas intégrer dans le routeur et le temps de boucle réel qui ne peut qu'être observé à l'arrivée des acquittements.

L'objectif de cette partie est d'exploiter la différence δ entre M et M_f (Figure 4.26) sachant qu'elle est est due essentiellement à la présence d'un second flux (ici le flux extérieur) dans le buffer.

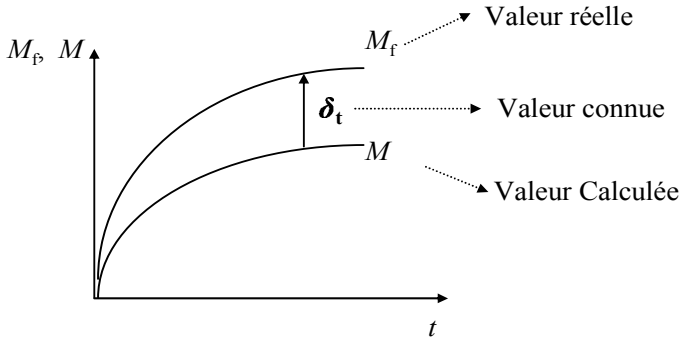


FIG. 4.26 – *Comparaison entre les temps de boucle réel calculé.*

4.6 Étude du temps d'attente δ et reconstruction progressive de l'état de la ligne

Nous désirons évaluer la différence δ entre les deux temps de boucle M et M_f. La principale différence entre ces temps de boucle réside dans le temps d'attente dans le buffer.

Commençons par calculer le temps d'attente dans le buffer lorsque seul les données d'émission sont prises en compte. Appelons seg le dernier segment entré dans le buffer à l'instant t. À

cet instant, il se trouve une quantité q_1 de donnée d'émission dans le buffer (Figure 4.27). Entre l'instant où seg est complètement dans le buffer et celui où il en sort complètement, il s'écoule un délai d_1 égal à:

$$d_1 = q_1/v_2. \tag{4.11}$$

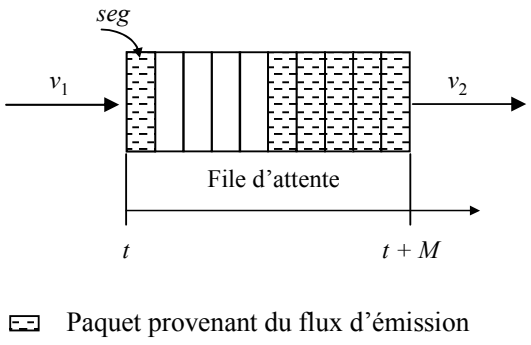


FIG. 4.27 – *File d'attente des données d'émission.*

Considérons maintenant le cas où le dernier segment d'émission seg qui entre dans le buffer attend la sortie des données d'émission et des données extérieures (Figure 4.28.a). Le délai passé par seg dans le buffer est appelé d_2 et peut être calculé de la façon suivante:

$$\begin{aligned} d_2 &= \beta/v_4 + \alpha/v_2 + \alpha/v_2 + \beta/v_4 + ... + \alpha/v_2, \\ d_2 &= q_1/v_2 + q_2/v_4, \end{aligned} \tag{4.12}$$

avec α et β les données d'émission et extérieures respectivement (Figure 4.28.b),

q_1 et q_2 l'ensemble des données d'émission et l'ensemble des données extérieures respectivement.

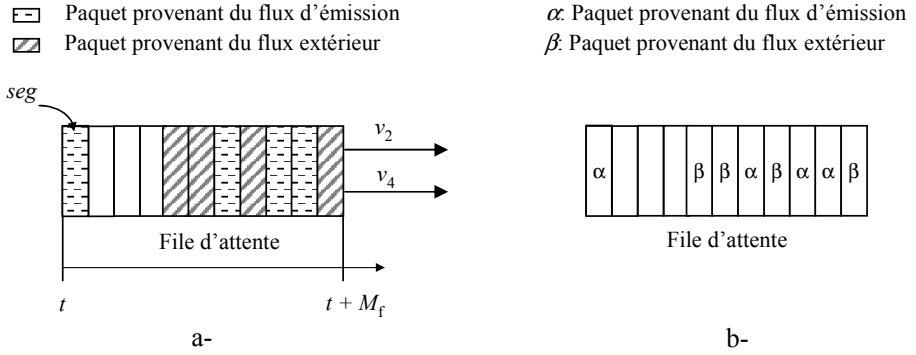


FIG. 4.28 – *File d'attente des données d'émission et des données extérieures.*

À partir des équations 4.11 et 4.12, la différence entre d_1 et d_2 s'écrit:

$$d = (q_1/v_2 + q_2/v_4) - q_1/v_2,$$
$$\Rightarrow d = q_2/v_4. \tag{4.13}$$

Or, la différence entre ces deux délais passés dans le buffer représente l'écart δ entre M et M_f. Ainsi,

$$\delta = q_2/v_4. \tag{4.14}$$

À chaque réception d'un nouvel acquittement, nous observons M_f, donc nous calculons la différence δ. La vitesse v_4 est connue, nous connaissons donc l'occupation du buffer des données extérieures notée ici q_2 : $q_2 = v_4\delta$ (marquage de P_9 dans la Figure 3.37). Ainsi, **la quantité "données extérieures dans le buffer" est désormais connue à chaque réception d'acquittement**.
Nous désirons, à partir de cette information, déduire si oui ou non nous pouvons continuer à émettre des données sans risquer de causer des pertes sur la ligne. Pour cela, nous allons nous servir d'autres données connues telles que la fenêtre de congestion $cwnd$ ou encore le seuil d'évitement de congestion S.

4.6.0.1 Évaluation des charges du buffer

La figure 4.29.a représente une évolution quelconque (mais probable) de la fenêtre de congestion $cwnd$. La figure 4.29.b représente les temps de boucle M_f associés aux différentes fenêtres lancées. Les instants t_i (t_1, t_2, etc., Figure 4.29.a) sont les instants d'envoi d'une nouvelle fenêtre de congestion. Ces derniers sont connus car les données sont envoyées de l'émetteur. Les instants r_i (r_1, r_2, etc., Figure 4.29.b) sont les instants de réception des acquittements de ces même fenêtres (par exemple, r_1 est l'instant de réception des acquittements de la fenêtre 1, lancée à t_1). Lorsque nous lançons une fenêtre, nous ne savons pas à quel moment nous allons recevoir l'ensemble de ses acquittements, les r_i ne sont donc pas connus à l'avance.

Lorsque tous les acquittements d'une fenêtre i sont arrivés, donc à l'instant r_i nous possédons l'information marquage extérieur du buffer (que l'on va noté dorénavant m_{9i} en rapport avec l'instant r_i, i.e., à chaque instant r_i nous possédons l'information m_{9i}).

Procédons par étape et regardons ce qui se passe pour la première fenêtre lancée:

À $t = t_1 = 0$ (Figure 4.29.a), nous avons: $m_5 = C$,
$m_4 = 0$,
$m_9 = 0$.
Et à $t = r_1$ nous avons : $m_9 = m_{91} = v_4\delta_1$, ($\delta_1$ valeur de δ à l'instant r_1)
$m_4 =?$,
$m_5 =?$.

Nous pouvons conclure à partir de là qu'à l'instant r_1 les marquages m_4 et m_5 satisfont l'équation $m_4 + m_5 = C - m_{91}$. En nous mettant dans le pire des cas où toute cette quantité est dans

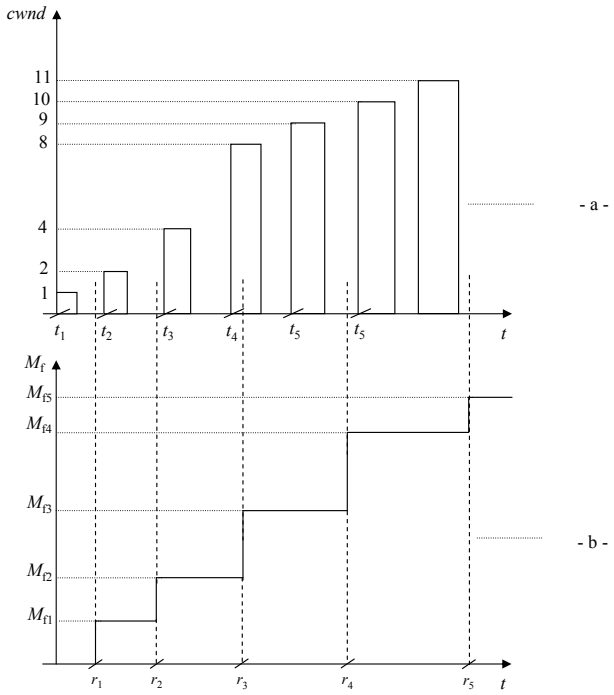


FIG. 4.29 – *Méthode de contrôle basée sur* M_f.

le buffer, i.e., $m_4 = C - m_{91}$ et $m_5 = 0$, l'on pourrait faire des déductions sur les quantités de données maximales que l'on peut envoyer sans causer des pertes. Cependant, cette méthode revient à considérer que le buffer est constamment plein (à chaque instant r_i). Nous allons donc constamment baisser les débits sur la ligne. Une telle méthode se focalise uniquement sur une transmission sans pertes sans s'occuper d'exploiter au maximum la bande passante de la ligne de transmission. Or, un contrôle sans pertes doit aller de paire avec une bonne exploitation de la bande passante pour rendre la transmission optimale.

Regardons donc le problème autrement.

Nous savons que durant l'intervalle $[0, r_1]$, l'environnement extérieur a occupé une place égale à m_{91} dans le buffer. Nous connaissons donc l'influence de l'environnement sur la ligne durant cet intervalle. De ce fait, nous pouvons soustraire, dès le début de l'intervalle (à $t_1 = 0$) cette quantité m_91 du buffer (i.e., faire $m_5 = C - m_{91}$) et considérer que l'environnement n'influence plus la ligne durant le reste de l'intervalle. En somme, cela revient à considérer non plus une ligne dans un environnement extérieur et une capacité de buffer égale à C, mais une ligne seule avec une capacité de buffer égale à $C - m_{91}$ (Figure 4.30). N'ayant plus qu'une ligne isolée durant l'intervalle $[0, r_1]$, nous pouvons aisément calculer ce que peut occuper l'émetteur comme place dans le buffer sachant que l'on a la valeur de v_1, celle de la fenêtre $cwnd$ à $t_1 = 0$ et la valeur du seuil qui nous indique si l'on est dans la phase exponentielle de l'évolution ou linéaire. Ce dernier point nous sert à respecter, dans les calculs, les règles d'évolutions "normales", donc selon slow start ou congestion avoidance. On arrive à la fin de l'intervalle r_1 avec une valeur connue de m_4 et de m_5. Finalement, la valeur de m_{91}, de m_4 et de m_5 étant calculées, on aboutit

à $t = r_1$, avec la connaissance exacte de l'occupation du buffer des données d'émission et des données extérieures.
Lorsqu'arrive les acquittements relatifs à la deuxième fenêtre, donc à $t = r_2$, on calcule comme

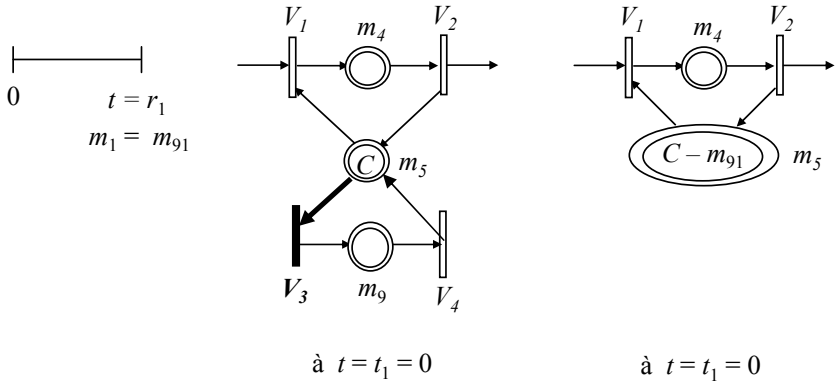


FIG. 4.30 – *Calcul de l'influence de l'environnement extérieur sur l'intervalle* $[0, r_1]$.

précédemment la valeur de m_{92} : $m_{92} = v_4\delta_2$, puis on revient à $t = r_1$ et on soustrait à m_5 la quantité m_{92} : $m_5 = m_5(\ t = r_1) - m_{92}$. L'influence de l'environnement durant l'intervalle $[r_1, r_2]$ est maintenant considérée dans la nouvelle quantité m_5 dès l'instant r_1. Nous pouvons de nouveau considérer une ligne isolée sur l'intervalle $[r_1, r_2]$. La valeur de m_4 à l'instant r_1 est connue (calculée précédemment), la nouvelle valeur de m_5 est de même connue ainsi que celle de $cwnd$; on calcule les valeurs que peuvent atteindre chacune des places m_4 et m_5 à $t = r_2$. On obtient de même que précédemment, les valeurs de m_{92}, de m_4 et de m_5 pour $t = r_2$.

Nous poursuivons ainsi les calculs à chaque obtention d'une nouvelle valeur de M_{fi}. Ainsi, à chaque réception d'acquittement, nous avons l'état exacte de la ligne de transmission.
Ceci constitue un premier résultat très important car nous savons déjà, à chaque instant r_i (les instants r_i sont de l'ordre de la nano seconde et peuvent atteindre dans les pires des cas la milliseconde) quelle est l'occupation exacte du buffer. Si ce dernier est plein, l'émetteur doit baisser les débits de sorte à être supporté par le buffer. Dans le cas contraire, nous poursuivons la transmission jusqu'au prochain instant r_{i+1}. Ce résultat vient en nette amélioration de la méthode fast recovery (chapitre 2) où il nous faut attendre l'arrivée de 3 acquittements successifs avant de déduire des pertes.

La question qui se pose à ce niveau est que connaissant à l'instant r_i l'état du buffer, comment vais-je calculer ou estimer la quantité de données nécessaire pour ne pas faire déborder le buffer en attendant d'atteindre l'instant r_{i+1}? Cette question nous renvoie vers une autre question: À partir de l'instant r_i et jusqu'à r_{i+1}, comment va influencer l'environnement extérieur sur le buffer? Pour répondre à cette question, nous allons nous attarder sur les évolutions passées des m_{9i} tout au long de la transmission et faire des déductions sur les évolutions correspondantes de la vitesse extérieure v_3.

Évaluation de la vitesse extérieure v_3 La prochaine étape est donc d'observer les évolutions de m_9 et d'évaluer à partir de là les évolutions correspondantes du débit extérieur v_3. L'objectif

de cette partie est qu'étant à l'instant r_i (donc connaissant v_{3i}), pouvons-nous évaluer la valeur suivante de la vitesse v_3 (donc v_{3i+1}).

Prenons un intervalle $[r_i, r_{i+1}]$ quelconque. La vitesse v_3 y varie de façon aléatoire (un exemple d'allure possible de v_3 est considérée dans la Figure 4.31). Lorsque des données extérieures se trouvent prêtent à sortir du buffer (i.e., il n'y a pas de données d'émission qui les précèdent dans la file, voir Figure 4.28.a), la vitesse v_4 est maximale (égale à V_4), sinon elle est nulle. Une forme quelconque de la vitesse v_4 est donnée dans la Figure 4.31.

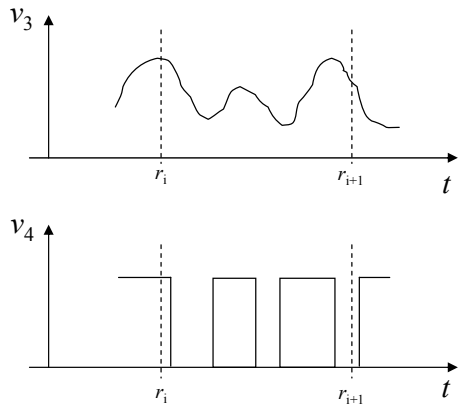


FIG. 4.31 – *Allures des vitesses extérieures d'entrée et de sortie.*

De façon générale, aux instants r_{i+1}, nous avons toutes les valeurs m_{9j} pour j allant de 1 à $i+1$. Appelons $\triangle m_{9i+1}$ l'écart $m_{9i+1} - m_{9i}$. Si l'on considère des valeurs moyennes des deux vitesses v_3 et v_4 sur l'intervalle $[r_i, r_{i+1}]$, l'on peut écrire:

$$\triangle m_{9i+1} = (v_{3moy} - v_{4moy})(r_{i+1} - r_i). \tag{4.15}$$

Avec v_{3moy} et v_{4moy} les valeurs moyennes des vitesses extérieures d'entrée et de sortie respectivement sur l'intervalle $[r_i, r_{i+1}]$.
Nous savons que $v_{4moy} \leq V_4$. Ainsi, l'on peut écrire:

$$(v_{3moy} - v_{4moy}) \geq (v_{3moy} - V_4),$$

Il s'ensuit que:

$$\begin{aligned} &\triangle m_{9i+1} \geq (v_{3moy} - V_4)(r_{i+1} - r_i), \\ &\Rightarrow v_{3moy} \leq V_4 + \triangle m_{9i+1}/(r_{i+1} - r_i). \end{aligned} \tag{4.16}$$

Nous pouvons choisir de majorer l'effet de l'environnement extérieur (v_{3max}) et on écrit:

$$\Rightarrow v_{3max} = V_4 + \triangle m_{9i+1}/(r_{i+1} - r_i). \tag{4.17}$$

Notons que l'on a aussi la valeur minimale de la vitesse v_3 en considérant v_4 constamment nulle (au lieu de la considérer constamment maximale à V_4 comme précédemment).
L'expression minimale de v_3 va nous donner:

$$\Rightarrow v_{3min} = \triangle m_{9i+1}/(r_{i+1} - r_i). \tag{4.18}$$

Ce qui revient à dire qu'à chaque instant r_{i+1}, nous avons la valeur minimale et maximale de v_3 sur l'intervalle $[r_i, r_{i+1}]$.
Notre choix peut se porter sur la valeur maximale (elle ne peut évidemment pas se porter sur celle minimale pour des raisons de fiabilité) ou nous pouvons décider de considérer une valeur milieu entre v_{3max} et v_{3min}. Tant que des tests ne sont pas faits sur la meilleure valeur de v_3 à considérer et par soucis de fiabilité, notre choix va se porter sur la valeur maximale $v_3 = v_{3max}$.

Considérons la figure 4.32. Nous sommes à l'instant r_{i+1} où nous venons de calculer la vi-

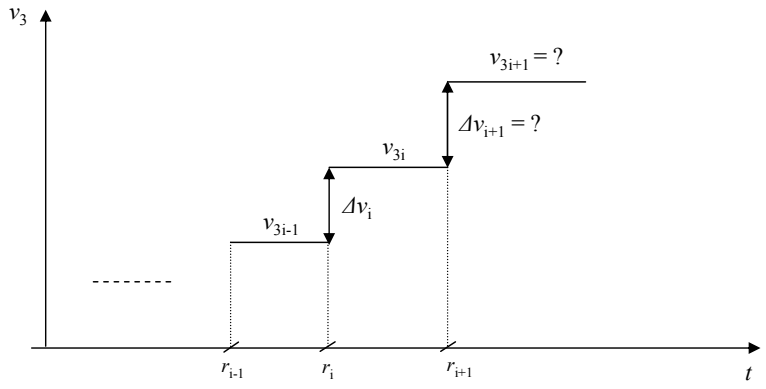


FIG. 4.32 – *Valeurs successives de la vitesse* v_3.

tesse v_{3i} sur l'intervalle $[r_{i+1}, r_i]$. De ce fait,
à $t = r_{i+1}$, v_{3i} est connue et
$\triangle v_i = v_{3i} - v_{3i-1}$ est connue.

Nous appelons $\triangle v_{i+1\ est}$ l'écart estimé $(v_{3i+1} - v_{3i})$ à l'instant r_{i+1}. Nous désirons que cet écart soit dépendant d'une part du précédent écart *vrai* $\triangle v_i$ (dit *vrai* car il est connu à l'instant r_{i+1}) et d'autre part de la différence entre l'écart *vrai* $\triangle v_i$ et celui *estimé* $\triangle v_{i+1\ est}$: $\triangle v_{i+1\ est} = f(\triangle v_i,\ \triangle v_i - \triangle v_{iest})$. Cette procédure est établie dans le but de tenir compte des écarts précédents et de *la variation de ces écarts dans le temps*. Remarquons que cette procédure ressemble à celle du calcul des temporisations (chapitre 2) qui tient compte des précédentes temporisations et de la variations de celles-ci dans le temps. Pour cela, nous prenons comme facteur de pondération la valeur $\alpha = 7/8$. Finalement, on écrit:

$$v_{3i+1} = v_{3i} + \triangle v_{i+1\ est}, \tag{4.19}$$
$$\triangle v_{i+1\ est} = \alpha \triangle v_i + (1 - \alpha)(\triangle v_i - \triangle v_{iest}). \tag{4.20}$$

Nous estimons donc à chaque instant r_i la vitesse v_{3i+1} sur le prochain intervalle de travail. Connaissant maintenant les valeurs de m_4, m_5, m_9 ainsi qu'une estimation de v_3, nous pouvons calculer la quantité de données qui va nous permettre de ne pas atteindre un état de congestion.

Annexe 3:
Comparaison des résultats de simulation avec ceux obtenus avec NS

Les résultats de simulation sous Network Simulator (NS) ainsi que les travaux de cette annexe ont fait l'objet d'un sujet de stage de licence professionnelle achevée en l'an 2004 et encadrée par moi même ([Jul04]).

NS est un simulateur à événements discrets disponible sur Internet, développé à Lawrence Berkeley National Laboratory (LBNL). Il est devenu aujourd'hui un standard de référence en ce domaine. Ce logiciel est exécutable sous Unix et sous Windows. Il permet à l'utilisateur de définir un réseau et de simuler des communications entre les nIJuds de ce réseau.

Sous NS, chaque équipement est décrit par un nIJud. Les nIJuds sont reliés entre eux par des liens. La simulation doit d'abord être saisie sous forme de fichier texte que NS va utiliser pour produire un fichier trace contenant les résultats.
NS est fourni avec des utilitaires dont un logiciel de tracé : Xgraph. Le logiciel Xgraph permet de tracer des courbes en fonction du temps.

– Exemple d'une ligne isolée : (Figure 4.33)

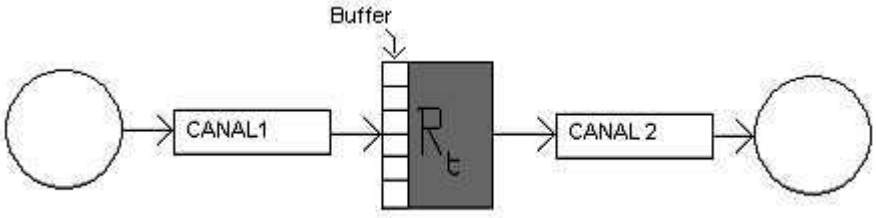


FIG. 4.33 – *Représentation graphique d'une ligne isolée.*

– Exemple d'une ligne complète (Figure 4.34)

Pour le programme de la ligne complète, la structure est la même que pour la ligne isolé, sauf qu'un agent TCP et un receveur ont été ajoutés sur le routeur.

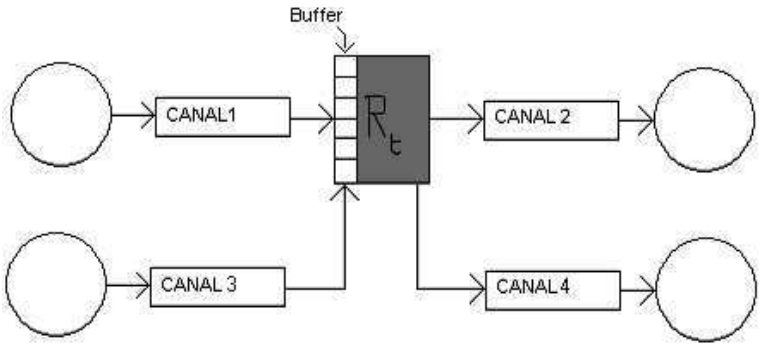


FIG. 4.34 – *Représentation graphique d'une ligne.*

Simulations:

Le logiciel NS ne permet pas de créer un flux aléatoire tel que donné par les figures 3.24 et 4.6 (Chapitre 3 et 4) mais juste des flux TCP (ou autres) connus et fixes. De ce fait, nous n'avons pas considéré un environnement extérieur quelconque pour comparer nos résultats à ceux de NS mais nous nous sommes contenté de considérer un environnement extérieur composé d'un ou deux émetteurs extérieurs pour nous mettre dans les mêmes conditions de travail que NS.

Simulation si la taille du buffer est de 50ko

(V1 = 12Moct ; V2 = 10Moct ; V3 = 12Moct ; V4 = 10 Moct)

– En utilisant le logiciel NS:
  Sur cette simulation (Figure 4.35), nous obtenons un transfert de données sans pertes car la taille du buffer est assez élevée.

– Les résultats de simulations à partir du modèle:

En augmentant la taille du buffer, il ne se produit pas de pertes sur chacune des deux figures 4.35 et 4.36. La ligne envoie les données avec la taille de fenêtre maximum égale à 64ko.

Simulation si la taille du buffer est de 20ko

(V1 = 12Moct ; V2 = 10Moct ; V3 = 12Moct ; V4 = 10 Moct)

– En utilisant le logiciel NS:
  Sur cette simulation (Figure 4.37), nous obtenons un transfert de données sans pertes car la taille du buffer est assez élevée.

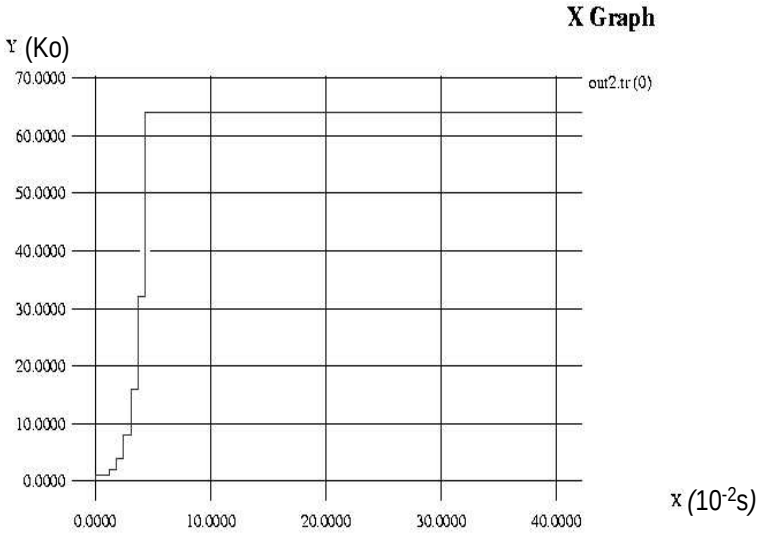


FIG. 4.35 – *Résultats de simulation sous NS.*

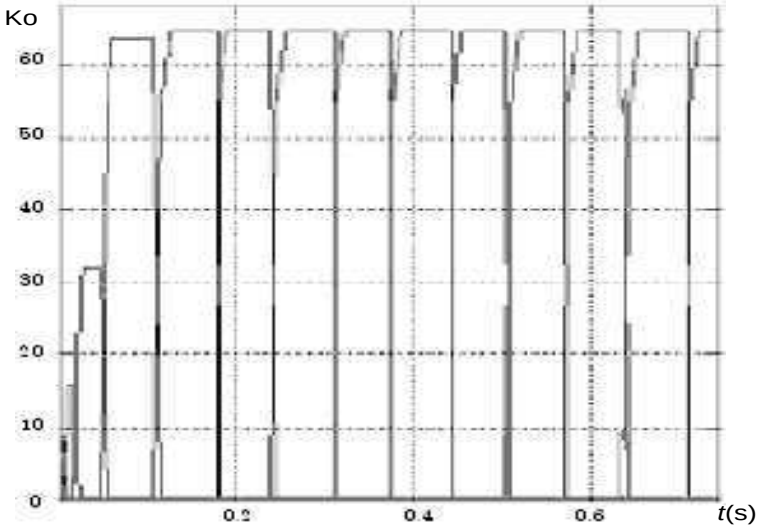


FIG. 4.36 – *Résultats de simulation à partir du modèle.*

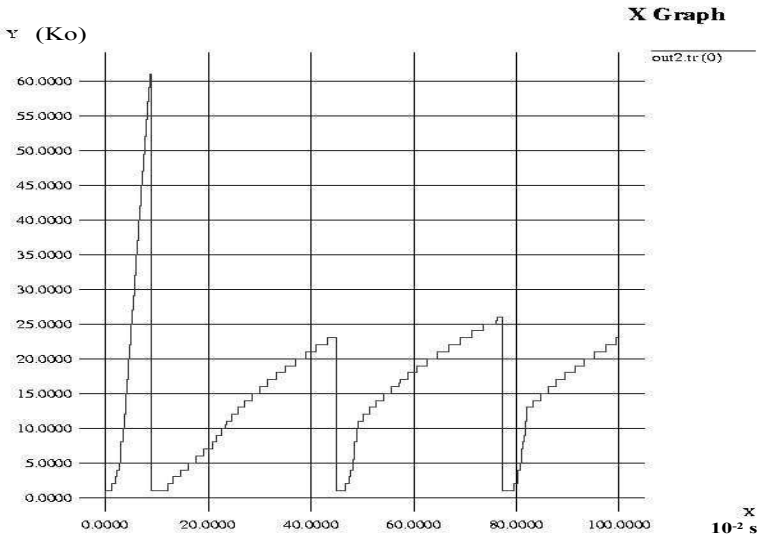


FIG. 4.37 – *Résultats de simulation sous NS.*

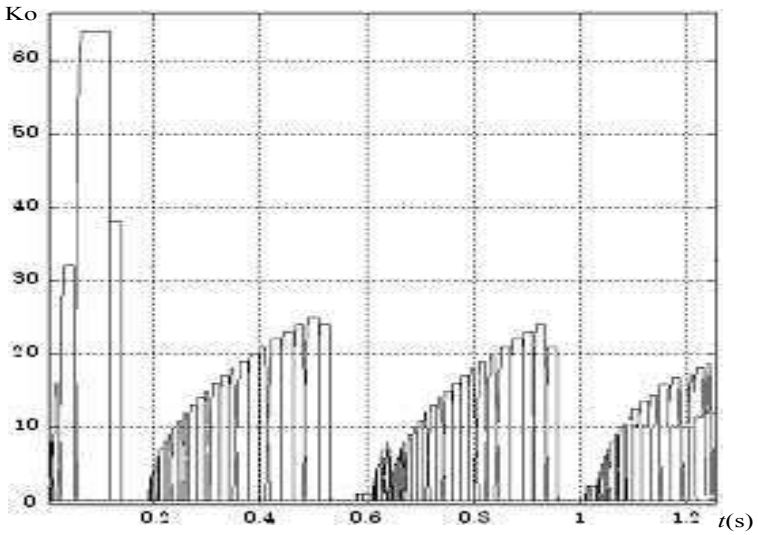


FIG. 4.38 – *Résultats de simulation à partir du modèle.*

– Les résultats de simulations à partir du modèle:

Avec un buffer de taille plus faible, nous obtenons de nombreuses pertes pendant la transmission de données. Si nous comparons les deux figures 4.38 et 4.37, nous pouvons voir que les allures sont très proches, avec de nombreuses pertes pendant toute la durée de la transmission.

Résumé : Les réseaux de communication offrent à ce jour un ensemble de services très diversifié aussi bien pour les industriels, les banques, la recherche, pour des particuliers et récemment dans la médecine et la commande à distance. A cette forte utilisation des réseaux viennent s'ajouter des contraintes de performances, de sécurité et de qualité de service que les fournisseurs de service se doivent de respecter. L'étude des performances des réseaux ainsi que leur modélisation exacte est donc un pas important pour la connaissance des influences des différents liens des lignes de transmission sur les données qui les traversent. Afin d'évaluer l'influence des réseaux de communication dans la transmission des données, nous proposons de modéliser puis d'étudier une ligne de transmission TCP/IP soumise aux protocoles de contrôle de congestion et plongée dans un environnement Internet. L'outil de modélisation utilisé est les RdP hybrides. Un modèle a été établi à base duquel toutes les dynamiques peuvent être observées à tous les niveaux d'une communication (charge des buffers, baisse des vitesses à l'entrée des routeurs, séparation des flux par les routeurs, priorités, stockage des données, retards, pertes, etc.). Ce modèle détaillé nous a aussi permis de visualiser directement l'effet des changements de paramètres de la ligne et des protocoles sur l'évolution de la transmission sur la ligne : Un large panel de simulations a donc été effectué afin de visualiser ces différentes performances. Enfin, un travail d'amélioration du temps de détection des pertes sur une ligne a été effectué.

Mots clés : Modélisation, Evaluation de performances, réseaux de Petri hybrides, réseaux TCP/IP et contrôle de congestion, Simulation.

Modelisation and behavior study of TCP/ IP communication line

Abstract: Networks communication offer a whole of services very diversified as well for industrialists, banks, research and recently in medicine and for remote control. To this large use of networks are added constraints of performances, safety and quality of service which service suppliers must respect. Thus, the study of networks performances and their precise modeling are important steps to evaluate the several links influences on the data which cross transmission lines. In order to evaluate the influence of the communication networks on data transmission, we propose first to model and then to study a TCP/IP transmission line subjected to congestion control protocols and merged in the Internet external environment. The tool of modeling used is hybrid Petri nets. A model was established thanks to which all dynamics can be observed at all the levels of transmission (the buffers load, flows decrease at the entry of the routers, flows separation in routers, priorities, data storage, delays, losses, etc). This detailed model also allowed us to visualize what are the effects of the line and protocols parameters changes on the transmission evolution: a large simulations panel was thus carried out in order to visualize these various performances. At last, a work of losses detection time improvement was carried out.

Keywords: Modelisation, Performances evaluation, Hybrid Petri nets, TCP/IP networks and congestion control, Simulation.