# Evaluation de performances de réseaux de communication à l'aide de chaînes de Markov hybrides

Alexandre Royer

**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

*N° attribué par la bibliothèque*

***T H È S E***

pour obtenir le grade de

**DOCTEUR DE L'INPG**

***Spécialité :*** **≪Automatique-Productique≫**

préparée au Laboratoire d'Automatique de Grenoble
dans le cadre de l'École Doctorale **EEATS**
**≪Électronique, Électrotechnique, Automatique, Télécommunication et Signal≫**

présentée et soutenue publiquement

par

**Alexandre ROYER**

le 6 janvier 2006

***Titre :***

# ÉVALUATION DE PERFORMANCES DE RÉSEAUX DE COMMUNICATION À L'AIDE DE CHAÎNES DE MARKOV HYBRIDES

---

***Directeur de thèse :***
M. Christian COMMAULT

---

**JURY**

| | |
|---|---|
| Luc DUGARD | , Président |
| Thierry DIVOUX | , Rapporteur |
| Guy JUANOLE | , Rapporteur |
| Christian COMMAULT | , Directeur de thèse |
| Stéphane MOCANU | , Co-encadrant |

# Remerciements

Les travaux présentés ont été réalisés au Laboratoire d'Automatique de Grenoble au sein de l'axe Systèmes à Événements Discrets, Évaluation de Performances.

Je tiens à remercier M. Stéphane Mocanu ainsi que M. Christian Commault qui ont co-dirigé ma thèse. Leur aide, leurs encouragements et leurs conseils m'ont permis de mener à bien ce travail.

Je remercie M. Luc Dugard, M. Thierry Divoux et M. Guy Juanole pour avoir accepté de faire partie du jury de ma thèse. Je remercie particulièrement les rapporteurs de ma thèse, M. Thierry Divoux et M. Guy Juanole, pour l'intérêt qu'ils ont porté à mon travail et les conseils qu'ils m'ont prodigués lors de la rédaction de cette thèse.

Je tiens à remercier tous les membres de l'équipe SEDEP que j'ai côtoyés pendant ces trois années.

Je remercie le personnel de l'IUT1 de l'UJF avec qui j'ai travaillé pendant mes années de monitorat et tout particulièrement Mlle. Alexia Gouin qui m'a beaucoup aidé dans cette tâche et Mme. Edith Clavel qui m'a encadré durant ces trois années.

Je remercie également toute l'équipe administrative du LAG pour son aide lors de ces trois années au laboratoire.

Je tiens à remercier mes parents qui m'ont soutenu pendant ces longues années d'études et qui ont toujours été là pour m'apporter leur aide.

Je remercie tous mes amis du LAG avec qui j'ai partagé de bons moments, en particulier Adib, Benito, Ahmad, Fethi, Alessandro, Ahmed, Bassam, Vincent.

Évidemment, je n'oublie pas non plus tous mes proches : Nadia, Guillaume, Hoang, Marion, Nico, Véro, Christophe, Alice.

# Table des matières

# Introduction

Les réseaux de communication sont utilisés pour diverses applications. Lorsqu'on conçoit un réseau, il est nécessaire de connaître précisément les performances de celui-ci afin de le dimensionner correctement. En effet, l'évaluation de ces performances permet de comparer différentes topologies de réseaux en fonction de leur utilisation. Elle permet également de prévoir le comportement du réseau dans certaines situations, et en particulier de prévoir les éventuels problèmes et de leur trouver des solutions. Il est donc nécessaire d'effectuer de l'évaluation de performances. On peut obtenir ses performances de différentes manières, soit en effectuant des mesures sur un réseau existant, soit en utilisant des simulateurs de réseau, ou bien encore grâce à des méthodes analytiques et numériques.

En ce qui concerne les mesures effectuées sur un réseau existant, c'est une méthode qui permet d'avoir un représentation exacte du trafic dans le réseau à un instant donné. Cependant, elle nécessite l'existence du réseau ou au moins d'une maquette. De plus elle requiert un grand nombre de mesures sur une longue période pour avoir des valeurs représentatives du comportement du réseau. C'est pourquoi cette méthode est plus utilisée pour la surveillance de réseaux que pour leur élaboration.

Les performances de réseaux de communication peuvent être déterminées à l'aide de simulateurs [SF94, KM98, LGK+99, YG99]. Les simulateurs présentent l'avantage de ne pas nécessiter l'existence d'un réseau. De plus, ils permettent de modéliser un grand nombre de types de réseaux différents. Il existe un grand nombre de simulateurs dédiés aux réseaux, comme nous le verrons dans le chapitre 2. Cependant, le principal inconvénient de ces simulateurs est leurs besoins en ressources et en temps de calcul. En effet, on peut modéliser des réseaux d'une très grande complexité, mais le temps et les ressources nécessaires seront très importants.

On peut également évaluer les performances d'un réseau grâce à des méthodes analytiques. On modélise alors le réseau et son comportement à l'aide d'outils mathématiques et on le résout de manière analytique. Il existe plusieurs manières de modéliser analytiquement un réseau comme les réseaux de Petri ou encore les modèles fluides. L'avantage de ces méthodes est qu'elles permettent de connaître la dynamique exacte du système lorsque des solutions existent. De plus elles nécessitent souvent un temps de calcul beaucoup moins important que pour les simulations. Cependant, lorsque le réseau devient trop complexe, il n'y a pas de solution exacte, on doit avoir recours à des approximations afin de déterminer les performances du réseau. Dans cette thèse, nous présentons une méthode

d'évaluation de performances basée sur ce principe.

Dans le chapitre 1, nous présentons les notions nécessaires à la compréhension du travail effectué sur les réseaux de communication. La topologie de ces réseaux ainsi que certains principes de fonctionnement sont exposés. Les principaux indices de performance sont également présentés afin d'introduire la notion de qualité de service dans un réseau. Nous nous intéressons plus particulièrement aux réseaux IP (Internet Protocol), très répandus grâce à internet, et aux réseaux ATM (Asynchronous transfer mode) également très intéressants car possédant des mécanismes de qualité de service. Nous nous intéressons ensuite à la modélisation de ces réseaux. Deux types de modèle sont évoqués : le modèle discret et le modèle fluide. Nous expliquons pourquoi ce dernier est préféré au modèle discret et nous détaillons les différents éléments utilisés dans la modélisation fluide des réseaux [KR92, KSCK96, CL99, WM94, CSP01]. Nous comparons également la complexité des deux modèles, discret et fluide, envisagés au début pour expliquer notre choix d'utiliser une représentation fluide.

Le chapitre 2 présente différents simulateurs employés pour l'évaluation de performance de réseaux de communication. Ces simulateurs s'appuient sur les deux types de modèle exposés dans le chapitre 1 : discret et fluide. Le principe de ces simulateurs est expliqué, puis on présente des simulateurs discrets et l'algorithme du simulateur fluide employé dans la thèse.

Nous nous intéressons ensuite dans le chapitre 3 à la résolution analytique d'un réseau simple : le système « monobuffer » qui n'est constitué que d'une source et d'un routeur associé à son buffer. Pour un tel système, il est possible d'obtenir une solution exacte [Mit88, ST99, DF82, AMS82]. Les résultats obtenus dans ce chapitre sont utilisés plus loin pour la résolution des réseaux de plus grande taille. On présente les outils employés pour résoudre ce système et en obtenir les indices de performance. On différencie, à la fin, deux cas : le cas « homogène » caractérisé par une source et un routeur de même vitesse de transmission. On oppose le cas « homogène » au cas « non-homogène » dans lequel les vitesses de transmission de la source et du routeur sont différentes.

Dans le chapitre 4, nous nous intéressons à des réseaux simples : les réseaux à topologie linéaire. En utilisant les résultats préliminaires du chapitre 3, nous évaluons, dans un premier temps, les performances de réseaux linéaires homogènes. Dans le cas de réseaux complexes, des approximations sont nécessaires afin de trouver une solution [DS98, GPW02, Gri94, DLB97]. Pour cela, nous utilisons une méthode de décomposition adaptée de l'algorithme « DDX » que l'on trouve dans [DDX89, Ger94] et largement employée dans les systèmes de production [DDX89, Ger94, ADD94, LMT00, Dal94, DLB98, LBD98]. Ensuite nous étudions les réseaux linéaires non homogènes. Nous envisageons alors une autre méthode que la méthode de décomposition, car l'algorithme « DDX »n'est pas adapté pour ce genre de réseaux. Nous testons les deux algorithmes développés dans ce chapitre sur des exemples.

La plupart des réseaux de communication possèdent des nœuds de convergences. Nous

allons donc traiter dans le chapitre 5 le cas plus complexe des réseaux comportant des convergences. Tout d'abord, nous présentons le cas mono-buffer multi-sources que nous utilisons ensuite afin d'évaluer les performances de réseaux complexes. L'algorithme présenté s'appuie sur celui utilisé pour les réseaux linéaires non homogènes dans le chapitre 5. Il est adapté afin de prendre en compte les nœuds du réseau complexe. Les performances étudiées restent le débit, les pertes et le niveau de remplissage des buffers. Ces performances sont déterminées en tout point du réseau. L'algorithme développé dans ce chapitre est appliqué sur un exemple afin d'en comprendre le mécanisme. On teste également l'algorithme sur plusieurs exemples afin de comparer les résultats obtenus analytiquement avec ceux déterminés par simulation.

# Chapitre 1

# INTRODUCTION AUX RESEAUX DE COMMUNICATION

Nous présentons, dans ce chapitre, les notions nécessaires à la compréhension du travail effectué sur les réseaux de communication [Tan88, Fer05, Puj05b].

## 1.1 Notions

Un réseau est composé de machines (ordinateurs, routeurs, ...) et de liaisons physiques (câble, fibre optique, ...). Ces machines sont reliées entre elles afin de communiquer et de partager des ressources quelle que soit leur localisation physique. Il existe plusieurs types de réseaux informatiques, deux caractéristiques principales permettent de les classifier : la technique de transmission et la taille. Il existe par exemple des réseaux à diffusion : une machine connectée au réseau envoie des messages reçus par toutes les machines. Chaque machine vérifie si le paquet lui est adressé et le traite ou l'ignore suivant le cas. La topologie de ces réseaux peut être par exemple de type "bus" (Figure 1.1) ou "anneau" (Figure 1.2).

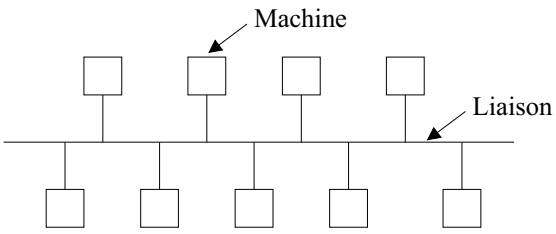


FIG. 1.1 – Topologie de type "bus"

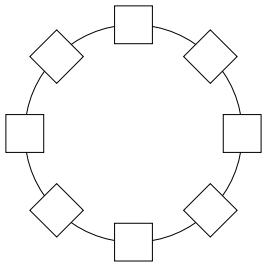

FIG. 1.2 – Topologie de type "anneau"

Au contraire, dans les réseaux de grande taille, on utilise plutôt la technique point à point. Lorsqu'une machine source envoie un paquet, celui-ci est dirigé à travers le réseau jusqu'à sa destination finale. Le paquet peut atteindre sa destination par plusieurs chemins, mais il devra passer par plusieurs machines qui l'orienteront dans la bonne direction. La topologie de ces grands réseaux peut être par exemple de type "étoile" (Figure 1.3) ou bien encore "arbre" (Figure 1.4).

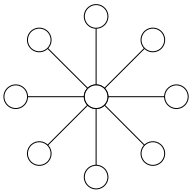

FIG. 1.3 – Topologie de type "étoile"

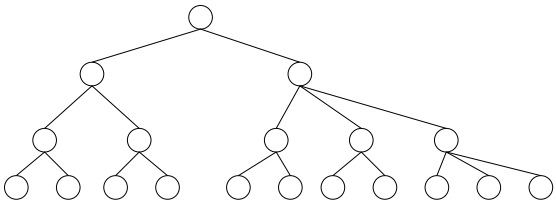

FIG. 1.4 – Topologie de type "arbre"

Quand il s'agit de relier entre eux un très grand nombre de composants, ou quand on veut interconnecter entre eux une multitude de petits réseaux locaux (comme pour Internet), une autre organisation est nécessaire. La topologie la plus adaptée est alors celle du réseau maillé, on la retrouve dans les sous-réseaux d'interconnexion. Comme son nom l'indique, elle se propose de constituer un filet dont les points extrêmes sont les composants à relier et dont les « mailles », les noeuds intermédiaires sont des équipements d'interconnection (routeur, passerelle ou commutateur par exemple) qui permettent aux données d'être transmises sur le réseau. Cette topologie maillée rend les sous-réseaux très fiables pour la transmission de données car il existe plusieurs chemins redondants pour aller d'un point à un autre du réseau.

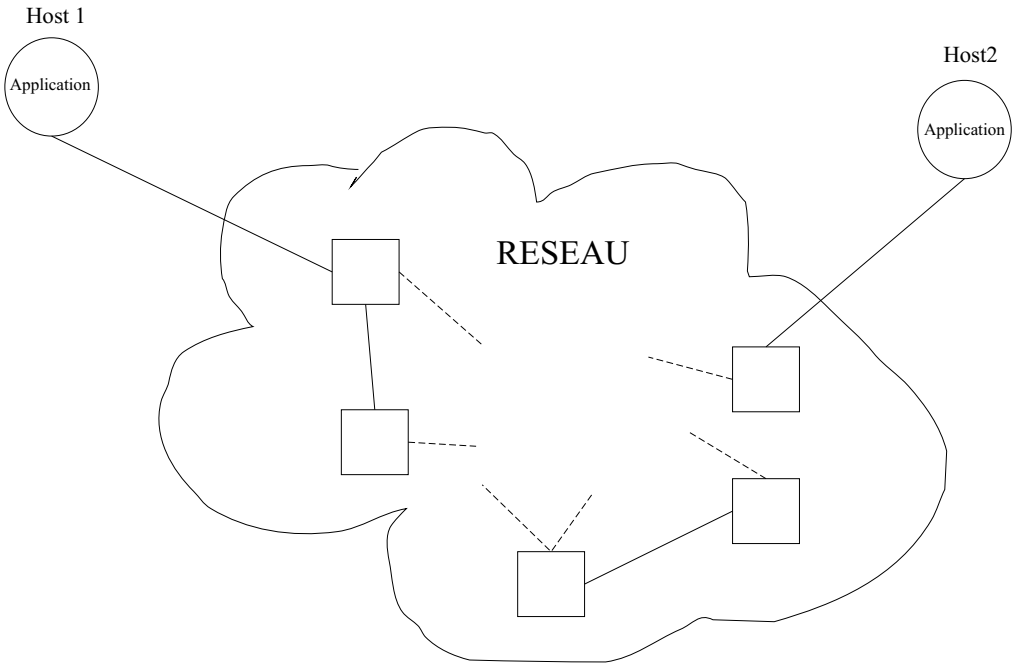


FIG. 1.5 – Connexion à travers un réseau

## 1.2 La qualité de service

### 1.2.1 Introduction à la qualité de service

Les utilisateurs du réseau ont des exigences différentes selon le type d'application qu'ils souhaitent utiliser. La diversité de ces application et l'augmentation de la taille des réseaux conduisent les opérateurs à devoir répondre à des besoins particuliers en termes de disponibilité, de bande passante, de latence, de gigue, de taux de pertes (SLA : service level agreement). La qualité de service, ou QoS (Quality of Service) se définit donc comme la garantie de performances sur les principaux paramètres de la connexion. Nous détaillons ces paramètres un peu plus loin dans cette section [Mél01, Puj05a, Abd02]. Ce besoin de QoS provient des nombreuses et diverses applications utilisant les réseaux. Dans le cas de réseaux IP, on peut par exemple citer :

- téléphonie et visioconférence : de nombreux utilisateurs dialoguent sur Internet et utilisent des « webcams » ;
- jeux en réseau : l'accès de plus en plus facile à Internet et l'augmentation de la puissance des ordinateurs ont permis à ce genre de jeu de se développer rapidement. La plupart des consoles de jeux permettent également maintenant de jouer en réseau par Internet.

On peut également citer des applications plus anciennes qui utilisent maintenant IP :

- applications bancaires ;
- applications médicales.

L'exemple des applications médicales met en évidence la nécessité d'une communication de très grande qualité dans le cas de la téléchirurgie.

Un utilisateur aura donc des exigences différentes selon l'application. Dans le cas de transfert de fichiers, les principales performances réclamées seront la vitesse et l'intégrité des données. Pour une visioconférence, on fera attention à avoir les performances suffisantes pour rendre la conversation audible et une image correcte. Les jeux en réseaux nécessitent, quant à eux, une bonne synchronisation. Enfin dans le cas de transfert de données confidentielles, comme des données bancaires, on attachera une grande importance à la fiabilité et à la sécurité des transactions.

A travers ces exemples, on comprend la nécessité de mettre en place différents paramètres servant à la QoS.

### 1.2.2 Paramètres de la QoS

La mise en place des contrats en terme de QoS s'appuie sur différents critères caractérisant les performances du réseau. Nous allons présenter brièvement les principaux paramètres associés aux réseaux : la disponibilité, la bande passante, la latence, la gigue et le taux de pertes.

#### Disponibilité

La disponibilité d'un réseau est le rapport entre le temps de bon fonctionnement du service vu par l'utilisateur et le temps total d'ouverture du service. Le pourcentage obtenu

donne une idée de la disponibilité du réseau. Cependant ce paramètre ne prend en compte que la connexion réseau, il n'y a pas d'autre mesure de qualité.

Les répercussions de cette disponibilité, ou plutôt de la non-disponibilité, sur les applications vont dépendre de la durée de la rupture de connexion. Si la coupure est courte, les couches inférieures à la couche application masqueront celle-ci, mais si elle est trop longue, le service sera interrompu.

Ces ruptures de connexions sont généralement dues à des pannes ou à des opérations de maintenance, de la part des fournisseurs d'accès notamment.

### Bande passante

La bande passante est le nombre de bits pouvant être transmis par seconde. Elle s'exprime donc en bit/s ou plutôt en Kbit/s et Mbit/s. C'est le débit possible entre deux points.

Certaines applications peuvent s'accommoder d'un faible bande passante, comme le transfert de fichier par exemple ; alors que d'autres, nécessitant un débit constant ou minimal ne pourront pas s'adapter à une réduction de la bande passante : téléphonie, audio, vidéo.

La bande passante dépend du matériel sur lequel s'appuie le réseau : supports physiques et équipements du réseau, ainsi que la politique de gestion de ceux-ci. La diminution de la bande passante est souvent due à une congestion du réseau. Lorsqu'aucun mécanisme de QoS n'intervient, le réseau fonctionne en « best-effort » : plusieurs applications utilisant les mêmes ressources se partageront la bande passante disponible. Dans le cas où on a négocié au préalable une garantie sur la bande passante, sauf panne, aucune congestion ne doit réduire cette bande passante.

### Latence

La latence est le temps que met un paquet pour traverser le réseau du point d'entrée au point de sortie. Cette latence dépend :

- du support physique : une communication par fibre optique est plus rapide qu'une communication satellite.
- du nombre d'équipements traversés : chaque équipement du réseau va ajouter son temps de traitement qui peut être plus ou moins long suivant le matériel. Par exemple un commutateur traite plus rapidement un paquet qu'un routeur classique.
- de la taille des paquets : on considère le temps mis pour émettre le paquet bit par bit sur le lien réseau, de l'émission du premier bit à la réception du dernier bit du paquet. Donc plus le paquet est grand, plus les temps de sérialisation et donc de latence augmentent.

On comprend ainsi qu'il ne suffit pas d'augmenter la bande passante pour avoir des meilleurs délais. L'augmentation de la capacité du lien ne diminue pas systématiquement le temps de latence.

Les applications nécessitant une latence très faible sont les applications interactives, nécessitant une synchronisation. Par exemple, la téléphonie fait partie de ce type d'applications. Si la latence est trop grande, la conversation devient difficile.

### Gigue

La gigue est la variation de la latence. Elle représente l'écart entre les délais d'acheminement des paquets d'un même flux.

Les applications les plus sensibles à la gigue sont les applications multimédia temps réel nécessitant une synchronisation. Par exemple les applications permettant d'écouter un flux audio ou de regarder une vidéo en « streaming », sont sensibles à la gigue. Ce genre d'applications peut s'accommoder des autres paramètres du réseau, mais c'est la gigue qui est le principal souci. En effet, souvent, plusieurs qualités de codage sont disponibles afin de s'adapter à la bande passante disponible. De même pour la latence, parfois, il n'est pas gênant de recevoir le flux avec quelques secondes de décalage. Mais dans ce cas, il est nécessaire que le flux soit constant, que les délais d'acheminement soient stables. Dans le cas contraire, on se retrouvera avec une vidéo ralentie ou accélérée par moments. Des mémoires tampons viennent compenser ces phénomènes de gigue inévitables, mais si la gigue devient trop importante en cas de forte congestion, ces mémoires deviennent insuffisantes.

Les éléments qui interviennent dans la gigue sont les mêmes que ceux qui modifient la latence : un équipement du réseau met plus de temps à traiter les paquets si le réseaux est fortement chargé. Plus la taille du réseau est importante, plus on augmente les risques de gigue car on a de nombreux équipements réseau à traverser.

### Taux de pertes

Le taux de pertes est le rapport entre le nombre d'octets émis et le nombre d'octets effectivement reçus. Ce paramètre renseigne sur la capacité utile de la transmission.

On trouve des mécanismes capables de détecter et corriger les pertes, comme TCP par exemple qui réémet les paquets perdus. Le taux de transfert est un paramètre qui influe directement sur les paramètre précédemment présentés : un taux de perte très élevé entraîne la retransmission d'un grand nombre de paquets. On note alors une baisse de la bande passante ou même une indisponibilité du réseau. Si les paquets doivent être retransmis, leur durée d'acheminement est allongée.

La principale cause de ces pertes est la congestion du réseau : lorsque la capacité des files d'attentes des équipements du réseau est dépassée, les paquets sont supprimés, donc perdus. On peut également observer des pertes au niveau des liens physiques, mais ceux-ci sont de plus en plus fiables, notamment avec l'utilisation de la fibre optique.

## 1.3 Architecture des réseaux

Dans cette section, nous présentons l'architecture des réseaux. Les réseaux sont organisés en couches (Figure 1.6) qui sont décrites par le modèle OSI (Open Systems Interconnection).

Chaque machine possède plusieurs couches qui communiquent entre elles. Chaque couche a un langage que l'on appelle protocole. Pour que deux couches de même niveau sur deux machines différentes se comprennent il faut qu'elles emploient le même protocole. Il existe plusieurs protocoles pour des couches de même niveau, que l'on privilégie suivant


FIG. 1.6 – Modèle OSI

le type de réseau et de service voulus. Le modèle OSI comporte sept couches que nous décrivons ci-après.

*La couche physique*

La couche physique s'occupe de la transmission des bits de façon brute sur un canal de communication. Cette couche doit garantir la parfaite transmission des données (un bit 1 envoyé doit bien être reçu comme bit valant 1). Concrètement, cette couche doit normaliser les caractéristiques électriques (un bit 1 doit être représenté par une tension de 5 V, par exemple), les caractéristiques mécaniques (forme des connecteurs, de la topologie...), les caractéristiques fonctionnelles des circuits de données et les procédures d'établissement, de maintien et de libération du circuit de données.

*La couche liaison de données*

Son rôle est un rôle de "liant" : elle va transformer la couche physique en une liaison a priori exempte d'erreurs de transmission pour la couche réseau. Elle encapsule dans des trames de transmission les paquets récupérés de la couche réseau.Elle transmet ces trames en séquence et gère les trames d'acquittement renvoyées par le récepteur. Rappelons que pour la couche physique, les données n'ont aucune signification particulière. La couche liaison de données doit donc être capable de reconnaître les frontières des trames.

La couche liaison de données doit être capable de renvoyer une trame lorsqu'il y a eu un problème sur la ligne de transmission. De manière générale, un rôle important de cette couche est la détection et la correction d'erreurs intervenues sur la couche physique. Cette couche intègre également une fonction de contrôle de flux pour éviter l'engorgement du récepteur.

*La couche réseau*

C'est la couche qui permet de gérer le sous-réseau d'interconnexion, c'est-à-dire le routage des paquets sur ce sous-réseau et l'interconnexion des différents sous-réseaux entre eux.

La couche réseau contrôle également l'engorgement du sous-réseau.

*La couche transport*

Cette couche est responsable du bon acheminement des messages complets au destinataire. Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux.

Cette couche est également responsable de l'optimisation des ressources du réseau : la couche transport crée une connexion réseau requise par la couche session, mais cette couche est capable de créer plusieurs connexions réseau par processus de la couche session pour répartir les données, par exemple pour améliorer le débit. A l'inverse, cette couche est capable d'utiliser une seule connexion réseau pour transporter plusieurs messages à la fois grâce au multiplexage. Dans tous les cas, tout ceci doit être transparent pour la couche session.

Cette couche est également responsable du type de service à fournir à la couche session, et finalement aux utilisateurs du réseau : service en mode connecté ou non, avec ou sans garantie d'ordre de délivrance, diffusion du message à plusieurs destinataires à la fois... Cette couche est donc également responsable de l'établissement et du relâchement des connexions sur le réseau.

Un autre rôle de la couche de transport est le contrôle de flux.

C'est l'une des couches les plus importantes, car c'est elle qui fournit le service de base à l'utilisateur, et c'est par ailleurs elle qui gère l'ensemble du processus de connexion, avec toutes les contraintes qui y sont liées.

*La couche session*

Cette couche organise et synchronise les échanges entre tâches distantes. Elle réalise le lien entre les adresses logiques et les adresses physiques des tâches réparties. Elle établit également une liaison entre deux programmes d'application devant coopérer et commande leur dialogue.

*La couche présentation*

Cette couche s'intéresse à la syntaxe et à la sémantique des données transmises : c'est elle qui traite l'information de manière à la rendre compatible entre tâches communicantes. Elle va assurer l'indépendance entre l'utilisateur et le transport de l'information.

Typiquement, cette couche peut convertir les données, les reformater, les crypter et les compresser.

*La couche application*

Cette couche est le point de contact entre l'utilisateur et le réseau. C'est donc elle qui va apporter à l'utilisateur les services de base offerts par le réseau, comme par exemple le transfert de fichier, la messagerie...

Le modèle OSI ne précise pas réellement les protocoles à utiliser pour chaque couche. Il décrit plutôt les services : ce que doivent faire les couches. Un autre modèle s'est progressivement imposé à OSI : le modèle TCP/IP. Cela tient tout simplement à son histoire. En effet, contrairement au modèle OSI, le modèle TCP/IP est né d'une implémentation ; la normalisation est venue ensuite. Nous allons maintenant présenter les réseaux IP (Internet Protocol) quasiment généralisés dans tous les réseaux informatiques.

## 1.4 Les réseaux IP

Nous avons parlé précédemment du modèle TCP/IP pour décrire l'architecture des réseaux. Ce modèle peut être mis en correspondance avec le modèle OSI, il s'organise également en couches (Figure 1.7).

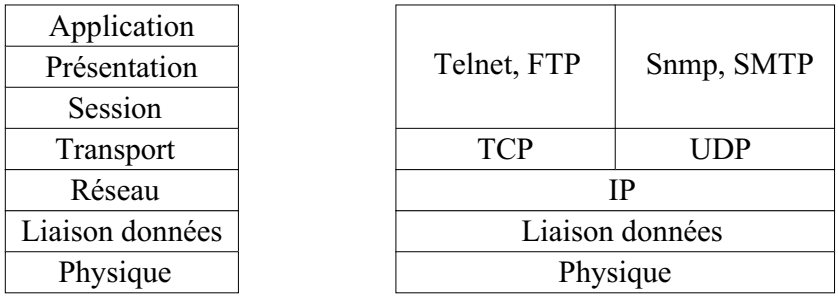


FIG. 1.7 – Modèles OSI et IP

### 1.4.1 Couche IP

IP est un protocole réseau qui offre un service de livraison de paquets (datagrammes). Il est non fiable, c'est à dire qu'il n'a aucune garantie sur le délai de transmission, la réception effective des paquets et leur intégrité : ceci est laissé à la couche supérieure (typiquement TCP). Il est orienté sans connexion : les paquets sont traités indépendamment les uns des autres, leur en-tête contient les informations suffisantes à leur acheminement. Il offre un service « best effort ». IP gère l'adressage des entités (host) du réseau, chaque entité possède une adresse IP permettant ainsi l'acheminement de paquets au travers de réseaux interconnectés. Il s'occupe également de la fragmentation et du réassemblage des paquets si les technologies employées pour la transmission des paquets nécessite des paquets de plus petite taille.

### 1.4.2 Couche transport

Comme indiqué sur la figure 1.7, TCP et UDP sont des protocoles de transport utilisés dans les réseaux IP.

*TCP*

TCP (Transport Control Protocol) fournit un service orienté connexion, c'est-à-dire que les deux extrémités qui souhaitent communiquer établissent au préalable une connexion logique entre elles. Il est fiable : des numéros sont assignés à chaque octet transmis et un acquittement de bonne réception est attendu dans un délai maximum (la connexion est donc bidirectionnelle) ; si l'acquittement n'est pas reçu, les données sont retransmises. De plus ces numéros permettent au destinataire de réordonner les paquets. TCP a également la possibilité de gérer le flux : l'acquittement reçu du destinataire renseigne l'émetteur sur sa capacité à recevoir des données supplémentaires. TCP permet de multiplexer les données, c'est-à-dire de faire circuler simultanément des informations provenant de sources (applications par exemple) distinctes sur une même ligne.

Ce protocole est donc utilisé partout sur Internet, dès lors que la conservation de l'intégrité des données à transmettre est plus important que la donnée elle-même. Par exemple, les transferts de fichiers sur Internet se font par FTP et s'appuient sur TCP. De la même façon, les protocoles peer-to-peer s'appuient sur TCP.

*UDP*

UDP (User Data Protocol) est, contrairement à TCP, sans connexion et non fiable : le protocole UDP ne numérote pas les données envoyées, n'acquitte pas les données reçues et rien ne garantit l'ordre d'arrivée des données.

Ce protocole peut donc être a priori jugé moins intéressant que TCP. Cependant, ses champs d'applications sont larges :

- Lorsque les couches sous-jacentes peuvent pallier son manque de fiabilité. C'est le cas notamment avec TFTP (Trivial File Transfer Protocol) sur les réseaux locaux : en effet, la qualité des réseaux locaux est suffisamment bonne pour compenser les défauts d'UDP et permettre ainsi le transfert de fichiers à très haut débit.
- Pour transmettre de très faibles quantités de données lorsque le coût d'établissement et de maintien d'une connexion fiable est jugé trop important par rapport à ce qu'il y a à transmettre.
- Pour les applications satisfaisant à un modèle de type "interrogation réponse", la réponse étant utilisée comme un accusé de réception à l'interrogation. On y trouve classiquement SNMP (Simple Network Management Protocol) et DNS (Domain Name Server).
- Lorsque la vitesse de transmission a plus d'importance que la perte d'un paquet de données. C'est le cas d'un service tel que la voix sur IP où en effet, l'envoi en temps réel est primordial, et la perte d'une trame d'information ne déformera pas notoirement le message transmis (l'oreille humaine sera capable d'extrapoler la perte de cette information).

- Le protocole de système de fichiers distants NFS (Network Files System) utilise le protocole UDP.

### 1.4.3 Couche liaison de données

IP s'appuie sur le réseau existant, notamment au niveau de la couche de niveau 2 (liaison de données) où on retrouve des technologies telles que Ethernet, Frame Relay ou ATM que nous présenterons plus longuement dans la section 1.5. Nous présentons brièvement ces technologies afin de comprendre les problèmes que l'on peut rencontrer pour assurer une QoS au niveau IP.

*Ethernet*

Lorsqu'une trame est émise, le noeud émetteur vérifie qu'il n'y a pas d'autre émission de trame sur le média. Si ce dernier est libre, il l'émet. Si deux émissions ont lieu simultanément, il y a collision et les deux nœuds concernés par la collision ré-émettent leur trame après un temps aléatoire. Cette méthode de partage du support physique est connue sous le nom de CSMA/CD (Carrier-Sense Multiple Access with Collision Detection). Ce fonctionnement limite la bande passante.

*Frame Relay*

Le modèle « Frame Relay » ne gère pas les erreurs ou pertes de paquets, il utilise le multiplexage afin d'optimiser la bande passante disponible. En cas de congestion, les commutateurs utilisent les bits FECN ou BECN ( Forward & Backward Explicit Congestion Notification) pour propager aux extrémités responsables l'information de congestion.

*ATM*

ATM est une technologie plus évoluée que les deux précédentes en ce qui concerne la qualité de service. Dans la section 1.5, une large présentation en est faite.

### 1.4.4 Qualité de service sur IP

IP pour garantir une QoS à ses utilisateurs s'appuie sur les services proposés par les couches inférieures, notamment ceux de la couche de niveau 2. Cependant, pour assurer une QoS de bout en bout, il est nécessaire qu'il existe des modèles de gestion de QoS fondés sur IP : IntServ et DiffServ.

- IntServ : ce modèle s'appuie sur la réservation de ressources dans le réseau. Il est difficile à mettre en oeuvre sur les réseaux de grande taille car chaque routeur doit mémoriser des informations concernant les flux transitant dans le réseau.
- DiffServ : ce modèle repose sur l'assignation de niveaux de priorité à un ensemble de flux IP. Une valeur de priorité est affectée dans l'en-tête du paquet IP. Les noeuds du réseau prennent en compte la priorité du paquet pour l'acheminer. Avec ce modèle, on ne mémorise pas la QoS requise par chaque flux, c'est la priorité indiquée dans l'en-tête du paquet qui permet d'offrir le service nécessaire.

## 1.5 Les réseaux ATM

Le protocole ATM (Asynchronous Transfer Mode) a été développé pour permettre de transmettre de la voix, de la vidéo et des données. Afin de réaliser cela, il doit offrir :

- un **débit** suffisant pour des applications multimédia qui ont besoin de liens avec des débits de l'ordre du Gigabits/s.
- une **qualité de service** adaptée à chaque type de trafic véhiculé par le réseau. Comme expliqué dans la partie présentant les différents paramètres de la qualité de service, des applications temps réel n'ont pas les mêmes besoins en termes de délai ou de bande passante que le transfert de fichiers.

### 1.5.1 Caractéristiques d'ATM

ATM utilise des paquets de petite taille fixe de 53 octets (5 d'en tête et 48 de données) : les cellules. L'utilisation de ces cellules procure divers avantages :

- lors de pertes de cellules dans le cas de congestion, les données perdues sont peu importantes et des méthodes de corrections peuvent être appliquées plus facilement.
- la longueur fixe rend plus facile l'implémentation au niveau matériel et l'allocation de bande passante.

ATM est orienté connexion, c'est à dire qu'avant de communiquer entre elles, deux machines doivent établir une connexion. On peut distinguer trois phases :

- établissement de la connexion
- transfert des données à travers le canal virtuel établi
- fermeture de la connexion

La phase d'établissement de la connexion permet de réserver un VCI (Virtual Channel Identifier) et un VPI (Virtual Path Identifier) (voir la figure 1.8 ainsi que les ressources nécessaires à la garantie du service demandé. Lors de cette phase, le routage est effectué : ainsi les délais de transmission sont réduits pour le transfert des données. Chaque cellule d'un même appel utilise le même chemin. Ce chemin est composé des liaisons (câbles, fibre optique) et de nœuds : les routeurs, aussi appelés commutateurs (switches) dans les réseaux ATM. Chaque connexion est identifiée par un numéro qui correspond au couple VPI/VCI. Chaque commutateur s'occupe de gérer la correspondance entre le VCI entrant et le VCI sortant d'une connexion. Le VPI correspond, lui, à un groupe de VCI utilisant le même chemin virtuel. Lors du routage, chaque routeur du réseau ATM qui compose le circuit virtuel crée une table de routage qui permet de faire transiter les cellules arrivant vers le conduit virtuel adéquat. Avec un tel type de transfert de données, on atteint des vitesses de l'ordre de 25Mb/s à 622Mb/s (on peut même atteindre 10Gb/s).

Différentes classes de services sont prévues pour permettre de répondre aux exigences des applications en termes de qualité de service. Nous nous intéresserons à ces différentes classes dans la partie consacrée à la QoS sur ATM.

### 1.5.2 Modèle en couches des réseaux ATM

Nous avons vu que l'architecture des réseaux était organisée en couches selon le modèle OSI. Dans ce modèle, on peut placer ATM au niveau de la couche liaison de données

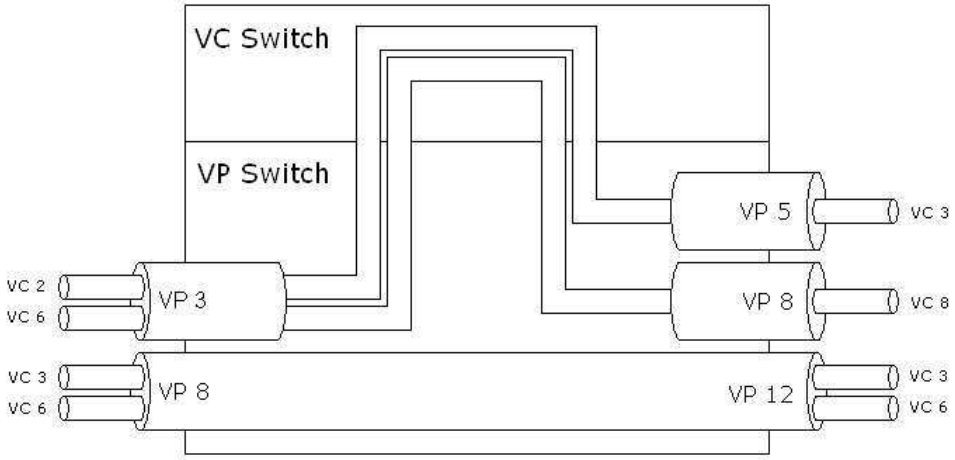


FIG. 1.8 – Chemin virtuel dans un commutateur ATM

(niveau 2). Cependant, ATM ne suit pas exactement la decomposition en 7 couches du modèle OSI. Le modèle ATM se répartit sur 3 niveaux (figure 1.9) :

- la couche physique : utilisation des couches standard existantes comme la fibre optique (réseau SDH (Synchronous Digital Hierarchy)) ou la paire torsadée ;
- la couche ATM qui s'occupe de la construction des cellules ATM, du multiplexage et du démultiplexage, et de la commutation des cellules ;
- la couche AAL (ATM Adaptation Layer) qui adapte le flux de données à la structure des cellules. Il existe plusieurs AAL suivant le type de trafic considéré.

Il existe également trois plans : utilisateur, contrôle et gestion.

### 1.5.3 Qualité de service sur ATM

Avant de communiquer, une connexion doit être établie. L'initialisation de la connexion s'effectue grâce à l'envoi d'un message comportant les informations suivantes :

- l'adresse ATM du destinataire ;
- l'adresse ATM de l'émetteur ;
- la bande passante désirée en cellules par seconde ;
- la qualité de service requise pour la connexion ;
- le type d'AAL utilisé.

L'une des caractéristiques d'ATM est qu'il possède des mécanismes de QoS (spécifiés lors de l'établissement de la connexion). Différentes classes de services proposées par ATM :

- CBR (Constant Bit Rate) : utilisé par les applications nécessitant une bande passante fixe, sensibles au délai et à la gigue. CBR est donc utilisé pour le transport de la voix et de la vidéo par exemple.
- VBR-RT (Variable Bit Rate-Real Time) : destiné aux applications possédant un flux irrégulier avec des contraintes temporelles comme la voix et la vidéo compressées.
- VBR-NRT (Variable Bit Rate-Non Real Time) : pour les applications possédant un flux irrégulier sans contraintes temporelles

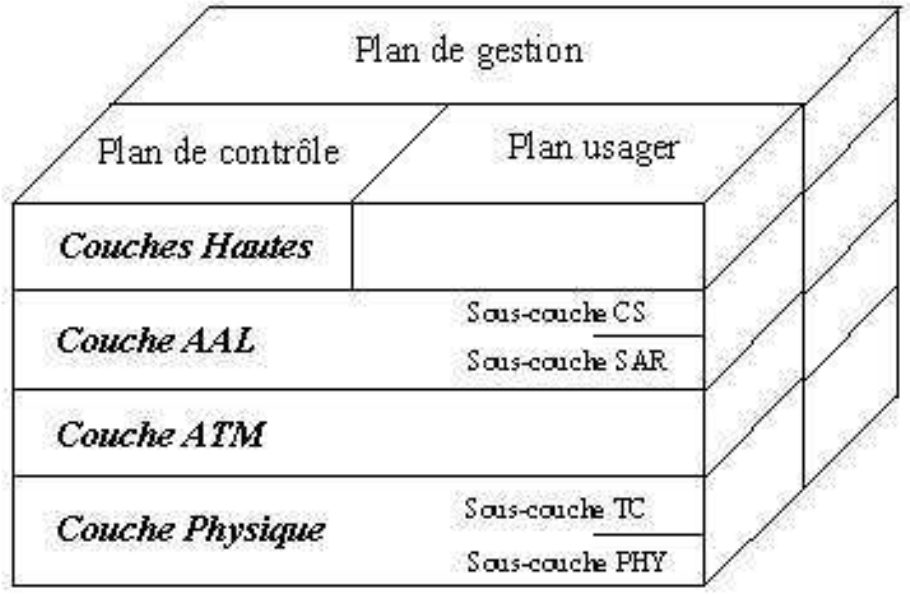


FIG. 1.9 – Modèle ATM

- ABR (Available Bit Rate) : prévu pour les applications capables de réduire ou augmenter leur trafic selon le souhait du réseau. Dans ce service, un mécanisme de contrôle de flux est mis en place : le réseau envoie à l'émetteur des trames contenant les informations sur une éventuelle congestion du réseau afin qu'il réduise son trafic.
- GFR (Guaranteed Frame Rate) : utilisé pour les applications exigeant un débit minimal garanti.
- UBR (Unspecified Bit Rate) : utilisé par des applications de type « best-effort ».

Dans le tableau 1.1 sont regroupés les services proposés par ATM.

## 1.6 Routeurs du réseau

Les performances du réseau dépendent des traitements effectués dans les équipements du réseau. Pour que les protocoles fonctionnent, les éléments du réseau doivent posséder des mécanismes que nous allons présenter. Nous étudierons les mécanismes de base sur les routeurs classiques, puis nous introduirons des mécanismes de QoS présents sur les routeurs supportant la QoS.

### 1.6.1 Routeur classique

Un routeur standard possède les composantes suivantes :
- interfaces d'entrée/sortie, chargées de recevoir et d'émettre des paquets ;
- fonction d'acheminement, qui a pour but de trouver l'interface de destination d'un paquet, à partir de la table de routage ;

| *Service* | *QoS* | *Applications* | *Classe AAL* |
|---|---|---|---|
| CBR | Débit constant | Audio/vidéo non compressé | AAL1 |
| VBR-RT | Débit variable, trafic temps réel | Audio/vidéo compressé | AAL2 |
| VBR-NRT | Débit variable, trafic non temps réel | Transactions | AAL5 |
| ABR | Débit disponible | Transfert de fichiers, messagerie | AAL5 |
| GFR | Débit trame garanti | Application à débit mini garanti | AAL5 |
| UBR | Débit non défini | Applications Internet « best-effort » | AAL5 |

TAB. 1.1 – Services ATM

- fonction de contrôle de gestion qui intègre les fonctions d'administration de l'équipement et la gestion des protocoles de routage pour mettre à jour sa table.

Son principe de fonctionnement est relativement simple : il reçoit un paquet sur une interface. Il consulte sa table de routage afin de déterminer l'interface de sortie à utiliser. Il ajoute le paquet à transmettre dans la file d'attente de l'interface en sortie. Si la file est pleine, il supprime le paquet.

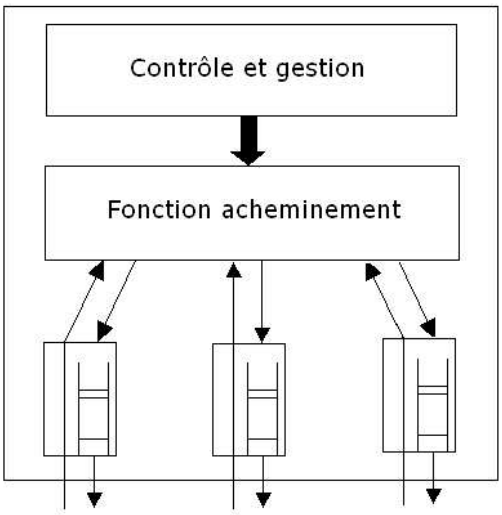


FIG. 1.10 – Routeur standard

Ce fonctionnement est très rapide. Le facteur limitant le nombre de paquets traités se situe au niveau de la fonction acheminement qui gère plusieurs interfaces. Certains routeurs possèdent plusieurs processeurs dédiés aux interfaces afin d'améliorer les performances. Au niveau de la sortie, on trouve une file d'attente, ou « buffer ». La politique de gestion la plus simple de ce buffer reste FIFO (First In First Out). Lorsqu'on rencontre

des problèmes de congestion, ce buffer se remplit rapidement et les paquets supplémentaires sont perdus. Ceci illustre ce que l'on appelle le fonctionnement en « best-effort ». Le réseau fonctionne tant qu'il n'y a pas de problème.

### 1.6.2 Routeur QoS

Lorsqu'on veut faire de la QoS, il faut être capable de prendre en compte les besoins de chaque application et de traiter les flux provenant de ces applications différemment selon le service requis. Un certain nombre de mécanismes sont donc intégrés au routeur :

- classification : analyse des paquets pour déterminer le traitement adéquat ;
- contrôle et marquage : on contrôle si le trafic correspond aux caractéristiques annoncées. Si ce n'est pas le cas, on peut marquer les paquets pour les traiter d'une autre manière
- gestion des files d'attente : on peut utiliser diverses politiques de gestion pour améliorer le traitement des paquets ;
- ordonnancement : on émet les paquets en fonction de leur classification.

*Classification*

La classification permet de déterminer le traitement à appliquer à un paquet reçu et de l'associer à un profil de trafic prévu.

*Contrôle et marquage*

Le contrôle vérifie si le trafic est conforme aux prévisions. On se base sur la définition du flux effectuée au préalable par l'équipement.
On vérifie si l'émetteur respecte son contrat afin qu'il ne dégrade pas le service proposé aux autres flux. Lorsque le contrat n'est pas respecté, il y a plusieurs solutions : on met les paquets excédentaires de côté et on les transmet lorsque le trafic sera repassé dans les conditions de son contrat. On peut également transmettre les paquets excédentaires mais les supprimer en priorité en cas de congestion. Pour déterminer si un trafic est conforme aux prévisions, on utilise deux principaux mécanismes : le « leaky bucket » et le « token bucket ».

- Le **leaky bucket** est principalement utilisé dans les commutateurs ATM. Ce mécanisme fonctionne sur le principe d'un seau percé : on peut le remplir à la vitesse que l'on souhaite, tant qu'il n'est pas plein, mais son débit en sortie sera limité par la taille du trou. La taille du trou permet de limiter la bande passante. Si le trafic d'entrée est variable, on a un trafic plus régulier en sortie. La principale limite de ce modèle est la taille du trou : lorsque le réseau n'est pas chargé, on ne peut pas en profiter pour vider le seau et prévenir ainsi une congestion qui pourrait survenir.
- Le **token bucket** propose lui un mécanisme s'appuyant sur un système de jetons. Ceux-ci sont donnés de manière régulière et indiquent le nombre de paquets pouvant être émis en sortie. Cette méthode permet de mieux profiter de la bande passante que le « leaky bucket »([Mél01]). En limitant le nombre de jetons maximum disponible, on évite les pics de trafic trop élevés.

*Gestion des files d'attente*

Les mécanismes de gestion de files d'attentes permettent d'optimiser le traitement des trafics en essayant, par exemple, de minimiser l'occupation des files d'attente. Pour cela il faut gérer les flux afin d'anticiper les congestions. Dans le cas du réseau IP, le protocole TCP de la couche transport s'occupe du contrôle de flux. Ce mécanisme peut être complété par des méthodes plus complexes d'élimination de paquets. On peut citer la méthode RED (Random Early Detection) qui supprime des paquets aléatoirement de la file d'attente lorsque le niveau de remplissage est trop élevé. Il existe d'autres techniques telles que WRED (Weighted RED) ou encore RIO (RED with In/Out).

*Ordonnancement*

Il a pour but d'émettre les paquets en sortie dans le bon ordre. Il existe plusieurs techniques d'ordonnancement.

- FIFO (First In First Out) : c'est la plus simple, on émet les paquets en sortie dans l'ordre où ils arrivent.
- Priorités : On dispose de plusieurs files d'attentes correspondant à un niveau de priorité différent. On émet les paquets de la file la plus prioritaire et ainsi de suite.
- Pondérations : on associe à chaque file d'attente un poids proportionnel à la bande passante souhaitée.

Les mécanismes présentés permettent de fournir la qualité de service entre l'émetteur et le récepteur. La structure en couches et l'interconnexions de réseaux de technologies différentes rend compliquée la mise en place de ces mécanisme à grande échelle.

## 1.7 Introduction à la modélisation de réseaux

Nous nous intéressons à l'évaluation des principaux indices de performance de manière analytique. Pour cela, on s'appuie sur un modèle du réseau. On peut utiliser un modèle discret ou un modèle fluide. Il existe plusieurs techniques pour l'évaluation de performances de systèmes discrets [Cas93, DG92].

### 1.7.1 Modèle discret

Les modèles discrets permettent de représenter tous les états que peut prendre le système étudié. Le réseau est constitué de noeuds au travers desquels circulent des cellules. Chaque cellule est considérée comme une entité. Un déplacement d'une cellule dans le réseau entraîne systématiquement un changement d'état discret. Comme nous le verrons par la suite, le principal inconvénient de ce type de modélisation est la taille de l'espace d'état qui devient très grande si on considère des réseaux à fort trafic. L'emploi d'un simulateur devient nécessaire dans ce genre de cas, et l'obtention de résultats peut nécessiter un temps de calcul important.

### 1.7.2 Modèle fluide

Dans le cas du modèle fluide, les données circulant dans le réseau ne sont plus considérées comme des entités mais comme un fluide. L'approche fluide des réseaux nous amène à nous placer à une échelle de temps supérieure. En discret, on se situe au niveau paquet, alors qu'en continu, on se place au niveau rafale (voir figure 1.11).

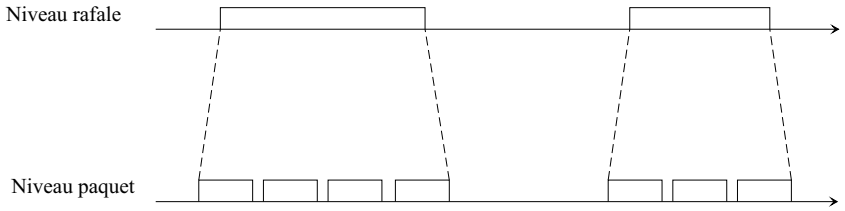


FIG. 1.11 – Échelles temporelles

Les données circulant en quantité importante, on considère le trafic comme un flot continu en se plaçant au niveau rafale. La complexité de l'étude analytique dépendant du nombre d'états considérés, l'utilisation du modèle fluide va réduire la complexité du problème et donc faciliter la résolution analytique.

Les modèles fluides (aussi appelés dans la littérature modèles continus) ont trouvé deux champs principaux d'application : les systèmes de production [SF94] et les systèmes de communication, en particulier les réseaux numériques à intégration de services (RNIS) basés sur la technique ATM [KR92, KSCK96, KWC93]. Par l'utilisation des modèles fluides on arrive à réduire de manière significative l'espace d'état du modèle. Plusieurs travaux ont mis en évidence les avantages de l'utilisation de modèles fluides à la place des modèles discrets [SF94, LGK+99, YG99]. Plusieurs travaux ont été dédiés à la simulation de modèles fluides, par exemple [SF94, KSCK96, KM98, LGK+99, YG99]. Dans cette thèse, on s'intéresse à la résolution analytique du modèle.

#### Modélisation des routeurs

Dans la section 1.6, nous avons décrit les routeurs : leur rôle et leurs caractéristiques de fonctionnement. Notre modèle du routeur sera composé de deux parties : une partie « buffer », file d'attente, qui contiendra les données à envoyer au routeur suivant et une partie qui représentera le résultat du calculateur de notre routeur, c'est à dire que cette partie sera en quelque sorte le « robinet » qui libérera les données contenues dans le buffer vers la routeur suivant. Ainsi, un routeur peut être représenté comme sur la figure 1.12. On notera $B$ le buffer ; il aura une politique de type FIFO et une capacité $C$. Dans les chapitres suivants, on représentera un routeur comme sur la figure 1.13.

#### Modélisation du trafic

La modélisation du trafic prend une part importante dans l'évaluation de performances d'un réseau. Il existe un grand nombre de modèles représentant des flux de données

particuliers. On peut citer, par exemple, le modèle de trafic à arrivées poissonniennes pour les données informatiques. Dans notre cas, nous nous intéressons au flux généré, dans les réseaux ATM, par des applications temps réel, des applications audio/vidéo, nous allons considérer le modèle ON/OFF. Notre routeur va donc fonctionner de la manière suivante : il est alternativement ON (disponible) ou OFF (indisponible) ; dans le cas où il est ON, si le buffer contient des données (sauf cas particulier que nous étudierons en temps voulu), alors la source transmet des données au buffer du routeur suivant. On retrouve, par exemple, ce type de modélisation pour les routeurs dans [MR99, CKR+95]. Les périodes de $ON$ et OFF suivent des lois exponentielles de moyennes $1/\lambda$ et respectivement $1/\mu$ (voir figure 1.13).

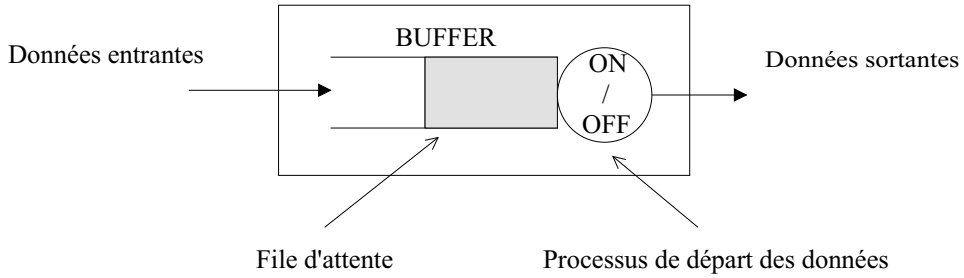


FIG. 1.12 – Modèle de routeur

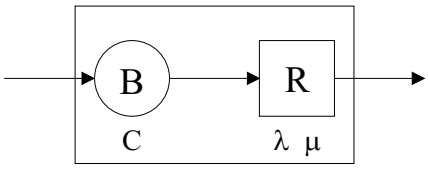


FIG. 1.13 – Représentation d'un routeur

### Complexité du modèle fluide

Nous allons illustrer la complexité du système grâce à un exemple. Considérons deux routeurs. Le premier, source du trafic, envoie des données dans le buffer du second. Ce buffer a une capacité de 10 cellules. Nous allons calculer la taille de l'espace d'état dans le cas du modèle discret et dans le cas du modèle fluide. C'est-à-dire que nous allons déterminer le nombre d'états que l'on va devoir envisager.

**Modélisation discrète** Chaque routeur peut prendre 2 états. Ce qui fait qu'en ne considérant que les états discrets des buffers, on a un espace d'état de dimension $2^2 = 4$. Il faut ensuite prendre en compte le niveau du buffer. Comme on est dans le cas discret, chaque cellule est considérée comme une entité, le buffer peut donc prendre 11 états (niveau de remplissage de 0 à 10 cellules). L'espace d'état final est donc de dimension $2^2$ x 11 = 44. Plus généralement, considérons un réseau de $K$ routeurs et $K-1$ buffers associés de capacité respective $C_i$ (le buffer du premier routeur n'est pas comptabilisé

puisque c'est un routeur source, on considère que son buffer est toujours plein). La taille de l'espace d'état à considérer sera de $2^K \prod_{i=1}^{K-1}(C_i + 1)$.

**Modélisation fluide** Tout comme dans le cas d'une modélisation discrète, chaque routeur peut prendre deux états distincts, on a donc un espace d'état discret de dimension 4. Le niveau du buffer étant représenté par une variable d'état continue, l'espace d'état discret reste de dimension 4. Cette variable d'état continue intervient dans le comportement du système comme nous l'étudions dans le chapitre 3. Dans le cas général faisant intervenir $K$ routeurs, on aura donc à traiter un problème faisant intervenir un espace d'état discret de dimension $2^K$.

Dans ce chapitre, nous avons exposé quelques notions sur les réseaux afin de faciliter la compréhension du travail effectué. Nous avons présenté la topologie de certains réseaux ainsi que certains principes de fonctionnement. Nous avons également défini les indices permettant de caractériser les performances d'un réseau : le débit moyen, les pertes moyennes, ainsi que le niveau de remplissage moyen des buffers. Ensuite, deux types de modèle pour les réseaux ont été présentés : le modèle discret et le modèle fluide. Après une comparaison, nous avons expliqué les raisons de notre choix d'utiliser le modèle fluide.

La difficulté et la plupart du temps l'impossibilité de résoudre analytiquement conduit à utiliser des simulateurs afin d'obtenir des résultats sur le comportement du réseau. Il existe deux grands types de simulation : les simulations discrètes et continues. Dans le chapitre 2, nous présentons quelques-uns de ces simulateurs.

# Chapitre 2

# MÉTHODES DE SIMULATION

La simulation est une méthode très employée pour l'analyse du comportement des réseaux de communication [KW93, KM98]. Elle permet en effet d'étudier un grand nombre de types de réseaux et s'adapte facilement aux conditions réelles du réseau. Elle est efficace lorsqu'il s'agit de réseaux relativement simples. Cependant lorsqu'on modélise des réseaux de taille plus grande, l'utilisation du simulateur devient plus lourde : elle nécessite de grosses ressources informatiques et demande souvent un temps d'exécution très long. Lorsqu'on désire effectuer l'analyse d'un réseau par simulation, on procède de la manière suivante. Il faut tout d'abord définir le système à étudier : ses composants (routeurs, buffers, liens,...) et le comportement de ceux-ci. On simule ensuite le scénario et on extrait les données qui nous intéressent. Il existe des simulateurs à événements discrets dans lesquels chaque entité du système est définie par plusieurs variables. L'état du système est décrit par la valeur de ces variables à un instant donné. Un événement est lié à un changement d'état du système. Lorsque ces variables peuvent prendre un nombre dénombrable d'états on parle d'états discrets et lorsqu'elles peuvent prendre un nombre non dénombrable de valeurs, on parle d'états continus. On a alors deux types de modèles : les modèles à états discrets et le modèles à états continus. Lorsqu'on retrouve ces deux types de variables dans un même modèle, on parle de modèle hybride. Nous exposons les particularités, les avantages et les inconvénients de ces deux représentations dans ce chapitre. Nous présentons également quelques simulateurs de réseaux.

## 2.1 Simulation discrète

Les simulateurs présentés ci-dessous sont dits « à événements discrets ». À chaque événement de la simulation est associé un temps $t$ auquel il se produit. Le système change d'état à ces instants ; entre deux événements, l'état du système reste le même. On comprend donc que le simulateur possède une horloge qui lui indique le temps ainsi qu'un échéancier dans lequel se trouvent les événements avec leur date d'exécution. On trouvera par exemple dans le cas des réseaux l'événement « émettre un paquet ». A noter qu'un changement d'état du système peut engendrer le déclenchement d'un autre événement. Lorsqu'un événement se produit, une routine lui est associée, chargée d'effectuer une action permettant le changement d'état. Ainsi, avec des techniques plus ou moins

différentes, le simulateur exécute tous les événements de l'échéancier. On peut remarquer dans les modèles discrets, le nombre d'événements à gérer est très important du fait du type de trafic sur le réseau. En effet, on a affaire à de nombreux paquets, ainsi la position de chaque paquet sur le réseau intervient dans l'état de celui-ci. Chaque paquet étant considéré comme une entité, les événements associés à toutes ces entités sont très nombreux.

### 2.1.1 Ns2

Il existe plusieurs simulateurs dédiés aux réseaux, ns2 (network simulator version 2) en est un. Il s'agit d'un simulateur à événements discrets orienté objet développé dans le cadre du projet VINT (Virtual InterNetwork Testbed) à l'université de Berkeley (http ://www.isi.edu/nsnam/ns/). Il est « open source » et a été créé dans le but de permettre le développement de nouveaux protocoles, d'en observer les effets sur le trafic dans les réseaux. Il permet ainsi de comparer différentes politiques de contrôle de flux entre elles (par exemple les différents types de TCP). Tous les événements simulés par Ns2 sont définis à l'avance. Le noyau du simulateur est implémenté en C++ et il utilise une interface basée sur le OTcl (Object Tool Command Language) permettant de créer un fichier décrivant le modèle à simuler. L'utilisateur peut définir une topologie en utilisant des nœuds, des routeurs et des liens. De nombreux protocoles peuvent être attachés aux nœuds du réseau, ils sont nommés agents. Ces trois éléments, nœuds, liens et agents sont les blocs de base pour définir la topologie du réseau. Les nœuds sont les points de connexion du réseau, une unique adresse est assignée à chaque nœud. Les liens sont créés entre les nœuds afin de créer la topologie du réseau ; on trouve deux types de liens, « simplex-link » et « double-link » qui sont respectivement des liens unidirectionnels et bidirectionnels. Les agents sont les objets qui gouvernent la simulation. Ils fixent le rôle de chaque nœud : routeur, source, destinataire. En plus de la topologie, l'utilisateur doit définir le scénario de la simulation : des événements sont programmés à des instants précis dans le fichier de configuration. Par exemple, sur la figure 2.1 sont représentés une source, un buffer et un destinataire connectés par des liens.

En plus de simuler, Ns2 propose une interface graphique, Nam (Network Animator), permettant de visualiser les résultats de la simulation.

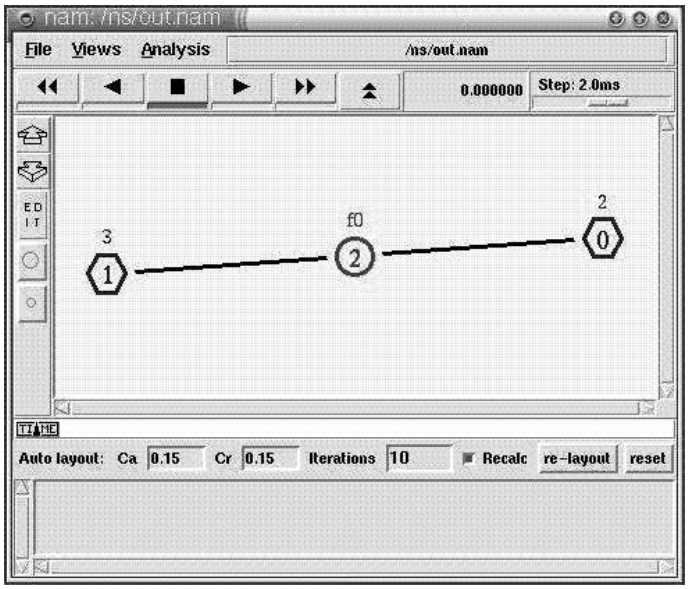


FIG. 2.1 – Interface graphique Ns2

Ce simulateur est complet du fait de la possibilité de programmer soi-même de nouveaux agents en C++. De plus l'emploi du language C++ facilite son implémentation. Cependant il est encore en développement et son utilisation en ligne de commande le rend peu convivial. Ns2 est plus un outil de développement logiciel de réseaux qu'un outil d'évaluation de performances. Il est possible d'obtenir des indices de performances pour les réseaux, cependant ces résultats sont plus qualitatifs que quantitatifs, notamment en ce qui concerne les débits aux différents endroits du réseau. Il permet d'avoir une vue d'ensemble de la répartition du trafic dans le réseau, mais pas d'avoir des résultats numériques précis en termes d'evaluation de performances.

### 2.1.2 Cnet

Cnet est un simulateur de réseaux permettant de définir une topologie de réseau et d'utiliser différents protocoles au niveau des couches réseau. C'est un simulateur « open source » (http ://www.csse.uwa.edu.au/cnet/install.html) très utilisé pour illustrer les cours de réseau, notamment à l'université Western Australia. Il permet principalement de développer des protocoles et de les tester sur des réseaux allant de deux à des centaines de nœuds. Cnet propose une interface graphique qui permet de visualiser la topologie du réseau et de modifier certains paramètres de la simulation pendant qu'elle s'exécute. Il est également possible d'observer quelques statistiques des éléments du réseau comme par exemple, pour les nœuds, la fréquence et la taille des messages envoyés.

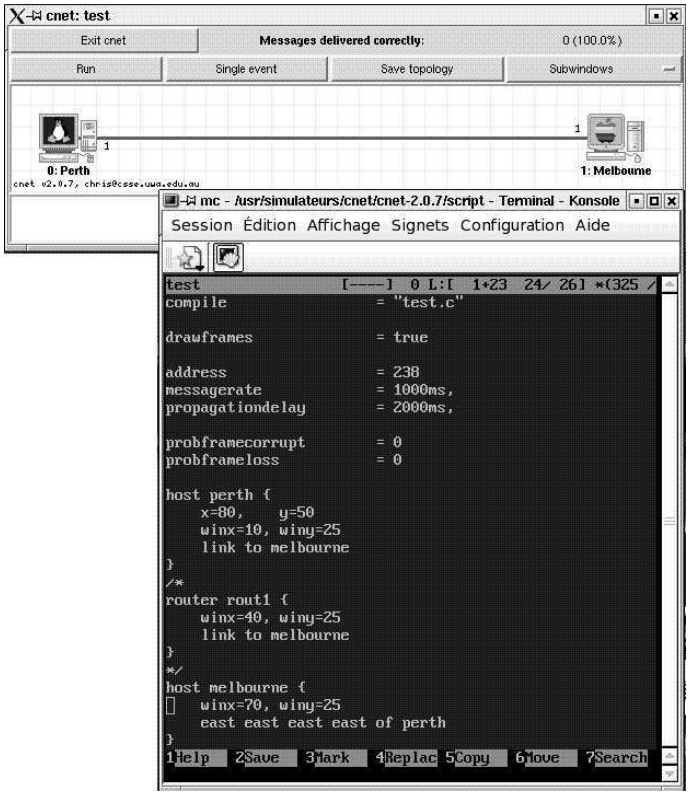


FIG. 2.2 – Topologie et fichier de configuration d'un réseau sous Cnet

Tout comme Ns2, Cnet est un simulateur permettant d'implémenter et de tester de nouveaux protocoles. C'est un outil éducatif très utile pour les réseaux et leur fonctionnement, mais ce n'est pas un simulateur permettant d'obtenir des résultats numériques précis pour l'évaluation de performances.

### 2.1.3 Opnet

Il s'agit également d'un simulateur à événements discrets. OPNET permet de modéliser et de simuler le fonctionnement de réseaux de communication. OPNET est structuré en trois domaines : le domaine réseau, le domaine nœud et le domaine processus. Le domaine réseau permet de définir la topologie du réseau. Il est composé de nœuds auxquels on peut affecter différentes fonctions par l'intermédiaire de paramètres. On peut ainsi par exemple déterminer le type d'application (Web, email, ...) qui va être utilisée à cet endroit du réseau. Les nœuds sont connectés entre eux afin de constituer le réseau. L'interface permettant de créer le réseau se présente comme sur la figure 2.3.

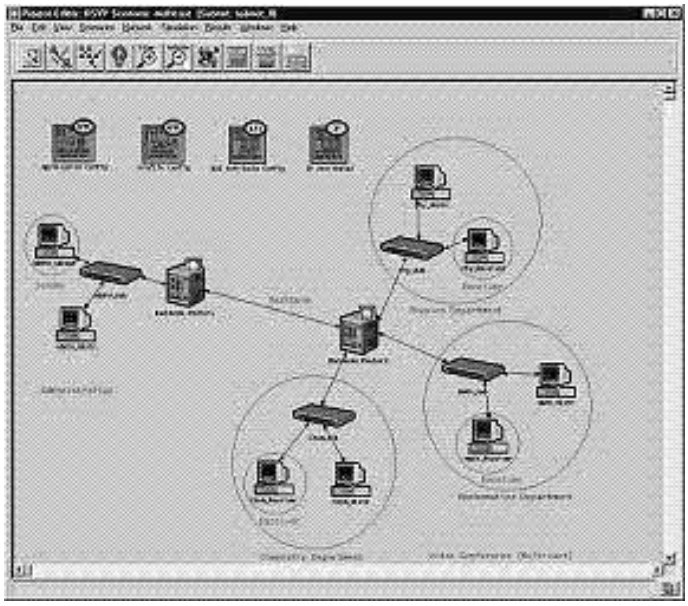

FIG. 2.3 – Création d'une topologie avec OPNET

Le domaine nœud permet, lui, de déterminer le comportement de chaque nœud en précisant les événements du scénario de simulation : envoi et réception de données par exemple. Selon la fonction qu'on attribue à chaque nœud, on parlera de routeur, d'ordinateur ou encore de commutateur .... Sur la figure 2.4 est présenté l'éditeur de noeud permettant de définir le rôle de chaque nœud.

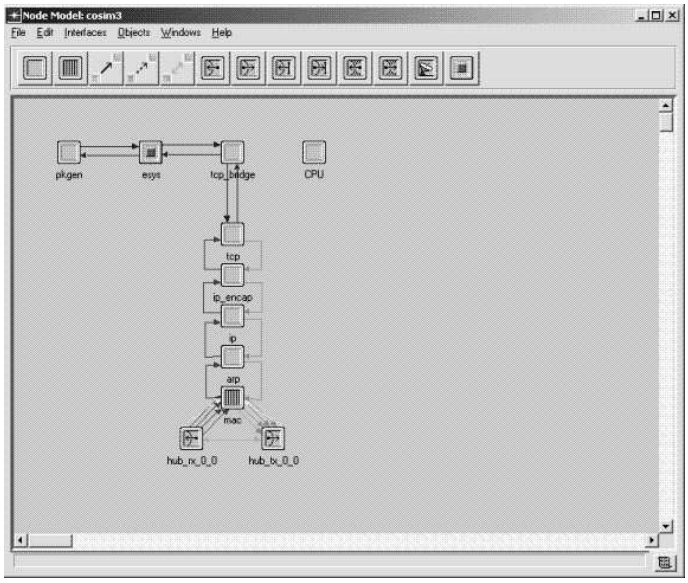


FIG. 2.4 – Exemple d'édition d'un nœud

Certains modules du nœud nécessitent d'être programmés par l'utilisateur. C'est ce qu'on réalise dans le domaine processus. On définit les actions à exécuter lorsqu'on se trouve dans un état particulier et les conditions de passage à un autre état (figure 2.5).

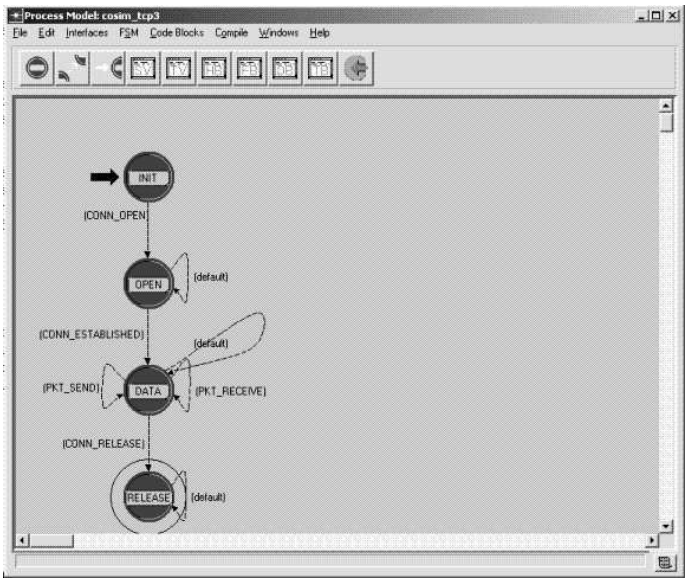


FIG. 2.5 – Domaine processus

OPNET est très complet et permet de simuler un large éventail de réseaux de communication. Ses interfaces graphiques permettent une prise en main rapide, cependant la multitude d'options proposées rend parfois compliquée l'élaboration d'une simulation. De plus, seule la version complète, payante, permet d'implémenter des modules supplémentaires en C++ (http ://www.opnet.com).

### 2.1.4 real

REAL est un simulateur dédié à l'étude de la dynamique des flux de données dans les réseaux. Il permet de définir un réseau et d'observer son comportement. C'est un simulateur « open source », qui peut donc être modifié de manière à correspondre aux attentes de l'utilisateur. Comme pour tous les simulateurs, il est nécessaire de décrire le scénario de la simulation : dans un fichier, on décrit la topologie du réseau, les protocoles employés ainsi que les différents événements qui peuvent se produire. On obtient en sortie un fichier de statistiques contenant le nombre de paquets envoyés par chaque source, les temps d'attente dans chaque queue, le nombre de paquets perdus ...Le language utilisé pour décrire le scénario de la simulation est le NetLanguage, une représentation texte simple du réseau. Le réseau est composé de nœuds représentant des sources de données, des routeurs, des commutateurs, ...Les noeuds sont connectés entre eux par des liens. On définit les protocoles de transport (politique de contrôle de flux), la taille des paquets transmis, la taille des buffers, ...On trouve de nombreux fichiers écrits en C permettant de définir un large éventail de type de réseaux.

## 2.2 Simulation fluide

Dans les modèles discrets, chaque paquet traversant le réseau est considéré comme une entité, alors qu'avec les modèles fluides, les paquets sont considérés comme un flot

de données continu, ou « fluide » qui s'écoule dans le réseau. Le nombre de changements d'état est ainsi réduit. Ce type de représentation permet d'améliorer la simulation en la rendant moins coûteuse en nombre d'événements, donc en temps d'exécution et en espace mémoire. Ce type de simulation s'appuie sur le modèle présenté dans la section 1.7.2. L'algorithme utilisé pour la simulation fluide est présenté ci-dessous, il est dérivé de celui trouvé dans [SF91]. Dans [SF91], cet algorithme est utilisé pour l'évaluation de performances de lignes de production ; mais il existe une très forte ressemblance entre l'étude de lignes de production et de réseaux. Nous adaptons donc cet algorithme aux caractéristiques des réseaux de communication étudiés, en supprimant par exemple l'hypothèse de « blocage »qui n'a pas lieu d'être dans notre cas. Nous utilisons cet algorithme, car les simulateurs employés dans le domaine des réseaux ne permettent pas de modéliser le trafic comme un fluide. Nous allons décrire son fonctionnement.

0.*Initialisation*

Cette phase sert à initialiser les variables (décrites plus avant l'algorithme) telles que le temps, le niveau des buffers, les vitesses de transfert,...

1.*Événement suivant sur les routeurs et les buffers*

Dans cette phase, on détermine les dates des prochains événements à partir de l'état actuel. On s'intéresse aux niveaux des buffers en distinguant trois cas :

- Il se remplit, donc le prochain événement sur ce buffer signalera qu'il est plein ($BF_j$) et on détermine le temps auquel se produira cet événement ($TB_j$) en fonction du niveau actuel du buffer et de la vitesse à laquelle il se remplit.
- Il se vide, donc le prochain événement sur ce buffer signalera qu'il est vide ($BE_j$) et on détermine le temps auquel se produira cet événement ($TB_j$) en fonction du niveau actuel du buffer et de la vitesse à laquelle il se vide.
- Il reste au même niveau, donc pas d'événement pour ce buffer ($TB_j = INF$).

On fait de même pour l'état des routeurs :

- Si le routeur était OFF, alors on le passera à ON à l'instant $TM_i$ qui est généré aléatoirement (contenu dans $G_i$).
- Si le routeur était ON, alors on le passera à OFF à l'instant $TM_i$ qui est généré aléatoirement (contenu dans $W_i$.

Enfin on s'intéresse à l'événement qui met fin à la simulation (la simulation s'arrête lorsque le dernier routeur a transmis une certaine quantité de flux $q$ donnée).

2.*Événement suivant*

Ici on cherche, parmi tous les événements suivants déterminés dans la phase précédente, celui qui aura lieu le premier. On cherche donc le plus petit $TM_i$ ou $TB_j$.

3.*Actualisation du système*

Une fois l'événement suivant déterminé, on actualise le système : temps, quantité de fluide sorti du système, débit...

Deux procédures sont utilisées pour mettre à jour les vitesses de transfert de chaque buffer : $Set.Flow.Rate(n)$ et $Update.Down(n)$.

4.*Test de fin*

On teste ici si la simulation est terminée ou si on doit répéter la séquence en retournant à *Événement suivant sur les routeurs et les buffers.*

5.*Sortie*

À la fin de la simulation, on calcule les paramètres de QoS nous intéressant, c'est à dire le débit pour chaque routeur et le niveau moyen de remplissage des buffers.

Voici à présent l'algorithme dans une version plus proche de son implémentation. Les variables et les notations utilisées sont les suivantes :

$t$ : temps de la simulation
$M$ : nombre de buffers
$\Delta t$ : intervalle de temps entre deux événements
$\alpha$ : vecteur contenant l'état des routeurs ($O$ pour ON et $D$ pour OFF)
$x$ : vecteur contenant le niveau des buffers
$L$ : vecteur contenant le niveau moyen des buffers
$q$ : quantité de fluide devant être émise par le dernier routeur $M$ jusqu'à la fin de la simulation (initialisée à une quantité $Q$)
$Deb$ : vecteur contenant la quantité de flux transmise par chaque routeur
$P$ : vecteur contenant les pertes au niveau de chaque buffer
$B$ : vecteur contenant la capacité des buffers
$v$ : vecteur contenant la vitesse de transmission des routeurs
$EB$ : vecteur contenant le prochain événement de chaque buffer, cet événement peut être $BE$ (buffer vide) ou $BF$ (buffer plein)
$TB$ : vecteur contenant la date du prochain événement pour chaque buffer
$TB_M$ : fin de la simulation
$EM$ : vecteur contenant le prochain événement de chaque routeur, cet événement peut être $R$ (passage à l'état ON du routeur) ou $F$ (passage à l'état OFF du routeur)
$TM$ : vecteur contenant la date du prochain événement pour chaque routeur
$G$ : vecteur contenant les temps moyen d'indisponibilité (OFF) des routeurs (généré aléatoirement au début de la simulation par la commande *Generate*)
$W$ : vecteur contenant les temps moyen de disponibilité (ON) des routeurs(généré aléatoirement au début de la simulation par la commande *Generate*)
$e$ : contient le prochain événement (routeurs et buffers confondus)
$K_M$ : contient l'indice du routeur dont l'événement suivant est le plus proche

$K_B$ : contient l'indice du buffer dont l'événement suivant est le plus proche
$K$ : contient l'indice du routeur ou du buffer dont l'événement suivant est le plus proche

0.INITIALISATION

$t = 0; \Delta t = 0; \alpha = O; x = \vec{0}; L = 0; x_0 = INF; q = Q; P = \vec{0}; B_M = INF;$
$v_1 = U_1; TB_M = \tau_f; Generate\ W, G;$
**pour** $i = 2$ **à** $M$ **faire**$\{v_i = min(v_{i-1}, U_i);\}$

1.EVENEMENT SUIVANT SUR LES ROUTEURS ET LES BUFFERS

**pour** $j = 1$ **à** $M - 1$ **faire{**
***cas***
$v_j > v_{j+1}:\ TB_j = (B_j - x_j)/(v_j - v_{j+1}); EB_j = BF_j;$
$v_j < v_{j+1}:\ TB_j = x_j/(v_{j+1} - v_j); EB_j = BE_j;$
$v_j = v_{j+1}:\ TB_j = INF;$
***fin cas***
**}**
**pour** $i = 1$ **à** $M$ **faire{**
***si***$(\alpha_i = D)$ ***alors*** $EM_i = R_i; TM_i = G_i;$
***si***$(\alpha_i = O)$ ***alors*** $EM_i = F_i; TM_i = W_i;$
***si***$(v_M \neq 0)$ ***alors*** $TB_M = q/v_M$ ***sinon*** $TB_M = INF;$
**}**

2.EVENEMENT SUIVANT

$K_M = argmin(TM_i)$ **pour** $i = 1, \ldots, M;$
$K_B = argmin(TB_i)$ **pour** $i = 1, \ldots, M - 1;$
***si*** $(TM_{K_M} > TB_{K_B})$ ***alors*** $K = K_B; \Delta t = TB_K; e = EB_K$
***sinon*** $K = K_M; \Delta t = TM_K; e = EM_K;$

3.ACTUALISATION DU SYSTEME

$t = t + \Delta t; q = q - v_M \Delta t; Deb_j = Deb_j + v_j \Delta t;$
$U = U - \Delta t.\vec{1}; W = W - \Delta t.\vec{1};$
**pour** $j = 1$ **à** $M - 1$ **faire{**
$temp = x_j; x_j = x_j + (v_j - v_{j+1})\Delta t;$
***si*** $temp = B_j$ ***et*** $v_j - v_{j+1} > 0$ ***alors***$\{P_j = (v_j - v_{j+1})\Delta t; x_j = B_j;\}$
$L_j = L_j + (temp + x_j)\Delta t/2;$

**}**
***cas***
$e = F_K : \alpha_K = D; v_K = 0; G_i = 1/r_i$;Update.Down(K);
$e = R_K : \alpha_K = O$;Set.Flow.Rate(K);Update.Down(K);
$e = BE_K$ :Update.Down(K);
***fin cas***

4.TEST DE FIN

***si*** $(e = \tau_f)$ ***alors*** aller à SORTIE
***sinon*** aller à ÉVÉNEMENT SUIVANT SUR LES ROUTEURS ET LES BUFFERS ;

5.SORTIE

Debit=Deb/t ;
Niveau moyen des buffers $= L/t$ ;

PROCÉDURES UTILISÉES

**procedure** Set.Flow.Rate(n)
***cas***
$x_{n-1} = 0 : v_n = min(v_{n-1}, U_n)$;
$x_{n-1} > 0 : v_n = U_n$;
***fin cas***

**procedure** Update.Down(n)
**tant que** $((x_n$ **est vide**$)$ **et** $(n < M - 1))$ **faire{**
$\{n = n + 1$;Set.Flow.Rate(n);
**}**

C'est cet algorithme qui nous servira de base pour obtenir des résultats numériques à comparer aux résultats obtenus analytiquement.

**Conclusion**

Dans ce chapitre, différents simulateurs de réseaux sont présentés. L'étude de ces simulateurs nous a permis d'en constater les inconvénients dont le principal est le temps d'exécution lorsque le réseau est de grande taille. On a également présenté l'algorithme du simulateur fluide utilisé dans cette thèse.

Dans le chapitre 3, nous présentons quelques résultats préliminaires nécessaires à l'étude analytique des réseaux.

# Chapitre 3

# RÉSOLUTION ANALYTIQUE DU CAS MONO-BUFFER

Dans ce chapitre, nous nous intéressons au cas mono-buffer, qui constitue, en quelque sorte, une « brique de base » de notre méthode d'évaluation de performances. Nous présentons le fonctionnement du routeur, puis nous nous intéressons au système mono-buffer. Dans un premier temps, on ne considérera que la partie discrète du modèle (section 3.2.3), puis nous introduisons le niveau du buffer, variable fluide, et nous étudions le système hybride ainsi obtenu (section3.2.4).

## 3.1 Fonctionnement des routeurs

Dans cette section nous expliquons le fonctionnement des routeurs ainsi que leur modélisation. Dans le chapitre 2 nous avons expliqué que les routeurs sont des nœuds du réseau par lesquels les cellules transitent. Lorsqu'elle arrive dans un routeur, une cellule est stockée dans le buffer du routeur, puis, lorsque le routeur l'a décidé, elle est réémise vers le routeur suivant. Un routeur peut faire simultanément partie de plusieurs chemins, il est donc amené à transmettre des cellules appartenant à des flux de données différents de celui auquel on s'intéresse. Lorsque l'on est dans ce cas, ou que le routeur effectue des opérations internes, il ne transmet pas de cellules appartenant au flux étudié. On dit alors qu'il est indisponible. Il devient disponible lorsqu'il est disposé à transmettre des cellules du flux étudié. Dans l'étude proposée, un routeur $R_i$ est caractérisé par les paramètres suivants (figure 3.1) :

- $1/\lambda_i$ : Temps moyen de disponibilité du routeur $R_i$
- $1/\mu_i$ : Temps moyen d'indisponibilité du routeur $R_i$
- $U_i$ : Vitesse de transfert du routeur $R_i$

Les temps moyens de disponibilité et d'indisponibilité du routeur pour notre flux sont caractéristiques du type de flux auquel on s'intéresse. Si c'est un flux « prioritaire », comme le transport de voix, alors le routeur sera très disponible pour ce flux, alors que si c'est un flux de basse priorité, le routeur sera moins disponible. De même, si le réseau est chargé, le temps d'indisponibilité sera plus élevé que s'il est peu chargé. Nous fixons donc ces paramètres en fonction du type de flux et de la charge du réseau que nous voulons

représenter.

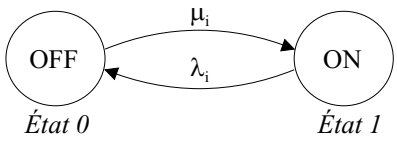


FIG. 3.1 – Fonctionnement ON/OFF d'un routeur $R_i$

Chaque routeur $R_i$ possède un buffer $B_{i-1}$ dans lequel il stocke les données reçues. On note $C_{i-1}$ la capacité du buffer $B_{i-1}$.

Les périodes de disponibilité de $R_i$ , respectivement d'indisponibilité, suivent des lois exponentielles de moyennes $1/\lambda_i$, respectivement $1/\mu_i$.

La matrice $M_i$ suivante est le générateur de la chaîne de Markov représentant l'évolution de l'état d'un routeur $R_i$. Dans la représentation suivante, le premier état est l'état OFF et le second est l'état ON.

$$M_i = \begin{bmatrix} -\mu_i & \mu_i \\ \lambda_i & -\lambda_i \end{bmatrix}$$

Le vecteur $\vec{v_i}$ contient les vitesses de transmission du routeur $R_i$ lorsqu'il est dans l'état OFF ou ON.

$$\vec{v_i} = \begin{bmatrix} 0 & U_i \end{bmatrix}$$

Le vecteur stationnaire de probabilité associé au générateur $M_i$ est :

$$\vec{w_i} = \begin{bmatrix} \frac{\lambda_i}{\lambda_i + \mu_i} & \frac{\mu_i}{\lambda_i + \mu_i} \end{bmatrix} \tag{3.1}$$

La probabilité stationnaire que le routeur $R_i$ soit disponible (dans l'état ON) est $\mu_i/(\mu_i + \lambda_i)$. Un routeur ne transmet des cellules que s'il est disponible et que son buffer est non-vide. Un routeur peut devenir indisponible lorsqu'il transmet des données, mais également lorsqu'il n'en transmet pas. Lorsque son buffer $B_{i-1}$ est plein, les données supplémentaires lui parvenant sont perdues.

## 3.2 Système mono-buffer

### 3.2.1 Présentation du système

Dans cette section nous nous intéressons au cas mono-buffer qui nous permet d'obtenir des résultats que nous réutiliserons pour le réseau linéaire homogène dans le chapitre 4. Les résultats présentés dans cette section sont en grande partie inspirés de travaux que

l'on trouve dans [AMS82, DF82, Mit87, Mit88]. C'est un ensemble composé d'une source ON/OFF et d'un routeur. On peut considérer la source comme un routeur dont le buffer est toujours non-vide (on ne représentera donc pas le buffer de ce dernier, voir figure 3.2). Les états ON/OFF sont indépendants pour chaque routeur.

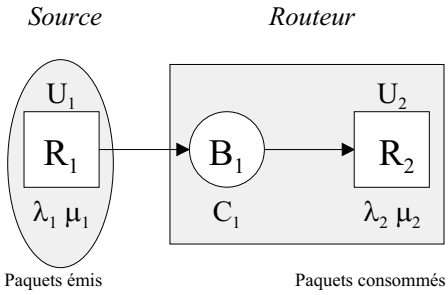


FIG. 3.2 – Topologie du cas « mono-buffer »

Chaque état de ce système peut être caractérisé par le triplet $(a,b,x)$.
- $a$ est l'état du routeur $R_1$ et vaut 1 s'il est disponible (ON) ou 0 s'il est indisponible (OFF)
- $b$ est l'état du routeur $R_2$ et vaut 1 s'il est ON ou 0 s'il est OFF.
- $x$ est le niveau du buffer $B_1$, il est compris entre 0 et $C_1$.

La chaîne de Markov de la figure 3.3 représente les 4 états que peut prendre notre système mono-buffer lorsqu'on s'intéresse uniquement aux états discrets des deux routeurs $R_1$ et $R_2$.

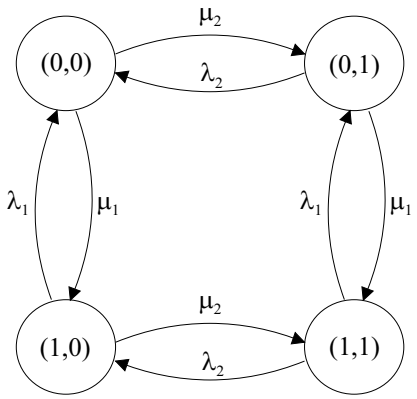


FIG. 3.3 – Chaîne de Markov relative aux états discrets du système mono-buffer

### 3.2.2 Étude analytique du système

Dans la suite, nous allons présenter l'étude analytique du cas mono-buffer. Avec les notations de la section 3.1, les matrices génératrices représentant l'évolution des routeurs

$R_1$ et $R_2$ sont :

$$M_1 = \begin{bmatrix} -\mu_1 & \mu_1 \\ \lambda_1 & -\lambda_1 \end{bmatrix} \quad \text{et} \quad M_2 = \begin{bmatrix} -\mu_2 & \mu_2 \\ \lambda_2 & -\lambda_2 \end{bmatrix}$$

Les vecteurs de probabilités stationnaires associés sont donc :

$$\vec{w_1} = \begin{bmatrix} \frac{\lambda_1}{\lambda_1+\mu_1} & \frac{\mu_1}{\lambda_1+\mu_1} \end{bmatrix} \quad \text{et} \quad \vec{w_2} = \begin{bmatrix} \frac{\lambda_2}{\lambda_2+\mu_2} & \frac{\mu_2}{\lambda_2+\mu_2} \end{bmatrix}$$

Les vecteurs vitesses sont :

$$\vec{v_1} = \begin{bmatrix} 0 & U_1 \end{bmatrix} \quad \text{et} \quad \vec{v_2} = \begin{bmatrix} 0 & U_2 \end{bmatrix}$$

### 3.2.3 Probabilités des états $(a, b)$

Lorsqu'on étudie le cas mono-buffer décrit précédemment, il faut s'intéresser au processus markovien défini sur l'espace produit des deux routeurs. En régime transitoire la probabilité de l'état $(a, b)$ est définie par :

$$p(t; a, b) = Pr[\text{l'état des routeurs à l'instant } t \text{ est } (a, b)]$$

L'espace d'état considéré est de dimension 4 (figure 3.3), on note alors $\vec{p}(t)$ le vecteur probabilité tel que :

$$\vec{p}(t) = \begin{bmatrix} p(t;0,0) & p(t;0,1) & p(t;1,0) & p(t;1,1) \end{bmatrix} \tag{3.2}$$

#### Équations régissant l'évolution des états $(a, b)$

L'évolution des états du système est régie par des équations différentielles. Elles s'obtiennent de la manière suivante. Supposons que nous soyons dans l'état $(0, 0)$ à l'instant $t + dt$. Il existe trois façons de parvenir à cet état présentées sur la figure 3.4.

- À l'instant $t$ on était déjà dans l'état $(0, 0)$ et aucun des deux routeurs n'est devenu disponible durant l'intervalle de temps $dt$ (la probabilité de se trouver dans ce cas est $1 - (\mu_1 + \mu_2)dt$).
- À l'instant $t$ on était dans l'état $(0, 1)$ et le routeur $R_2$ est devenu indisponible durant l'intervalle de temps $dt$ (la probabilité de se trouver dans ce cas est $\lambda_2 dt$).
- À l'instant $t$ on était dans l'état $(1, 0)$ et le routeur $R_1$ est devenu indisponible durant l'intervalle de temps $dt$ (la probabilité de se trouver dans ce cas est $\lambda_1 dt$).

Il est à noter qu'on ne peut pas passer directement de l'état $(1, 1)$ à l'état $(0, 0)$ au premier ordre, une seule variable peut changer d'état pendant un intervalle $dt$.

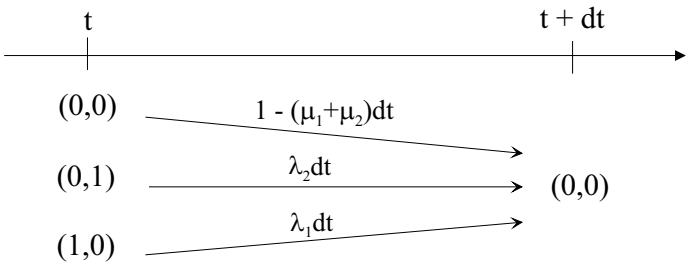


FIG. 3.4 – Évolution du système mono-buffer jusqu'à l'état $(0,0)$

Ceci nous donne une relation entre les probabilités des différents états :

$$p(t+dt;0,0) = p(t;0,0)(1-(\mu_1+\mu_2)dt) + p(t;0,1)\lambda_2 dt + p(t;1,0)\lambda_1 dt$$

À partir de cette relation, on aboutit à l'équation différentielle (3.3) de la manière suivante :

$$\begin{aligned} \frac{p(t+dt;0,0)-p(t;0,0)}{dt} &= -p(t;0,0)(\mu_1+\mu_2) + p(t;0,1)\lambda_2 + p(t;1,0)\lambda_1 \\ \frac{d}{dt}p(t;0,0) &= -p(t;0,0)(\mu_1+\mu_2) + p(t;0,1)\lambda_2 + p(t;1,0)\lambda_1 \end{aligned} \tag{3.3}$$

D'après la définition de $\vec{p}(t)$ (voir (3.2)), on peut écrire (3.3) de la manière suivante :

$$\frac{d}{dt}p(t;0,0) = \vec{p}(t)\begin{bmatrix} -(\mu_1+\mu_2) \\ \lambda_2 \\ \lambda_1 \\ 0 \end{bmatrix} \tag{3.4}$$

Notre espace d'état est de dimension 4, il existe donc trois autres équations différentielles que l'on obtient sur le même principe.

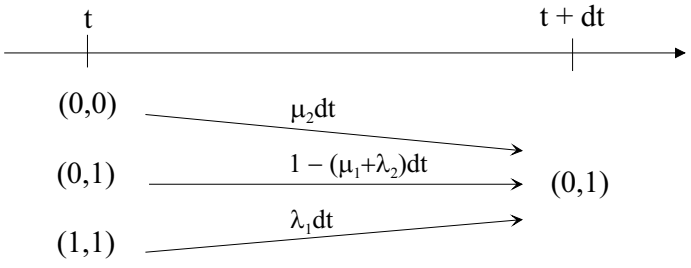


FIG. 3.5 – Evolution du système mono-buffer jusqu'à l'état $(0,1)$

D'après la figure 3.5, on obtient pour l'état $(0,1)$ l'équation différentielle (3.5) :

$$\begin{aligned} p(t+dt;0,1) &= p(t;0,1)(1-(\mu_1+\lambda_2)dt) + p(t;0,0)\mu_2 dt + p(t;1,1)\lambda_1 dt \\ \frac{d}{dt}p(t;0,1) &= \vec{p}(t)\begin{bmatrix} \mu_2 \\ -(\mu_1+\lambda_2) \\ 0 \\ \lambda_1 \end{bmatrix} \end{aligned} \tag{3.5}$$

On procède de la même manière pour les deux derniers états $(1,0)$ et $(1,1)$.

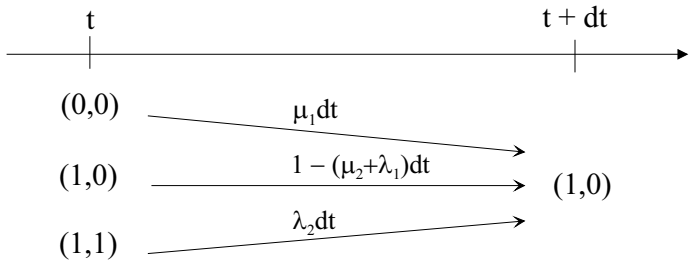


FIG. 3.6 – Évolution du système mono-buffer jusqu'à l'état $(1,0)$

Pour l'état $(1,0)$ (figure 3.6), on obtient l'équation suivante :

$$\begin{aligned} p(t+dt;1,0) &= p(t;1,0)(1-(\mu_2+\lambda_1)dt) + p(t;0,0)\mu_1 dt + p(t;1,1)\lambda_2 dt \\ \frac{d}{dt}p(t;1,0) &= \vec{p}(t)\begin{bmatrix} \mu_1 \\ 0 \\ -(\lambda_1+\mu_2) \\ \lambda_2 \end{bmatrix} \end{aligned} \tag{3.6}$$

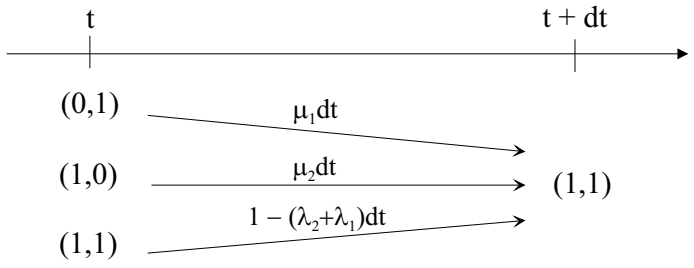


FIG. 3.7 – Évolution du système mono-buffer jusqu'à l'état $(1,1)$

Le dernier état $(1,1)$ (figure 3.7) donne l'équation (3.7).

$$p(t+dt;1,1) = p(t;1,1)(1-(\lambda_2+\lambda_1)dt) + p(t;0,1)\mu_1 dt + p(t;1,0)\mu_2 dt$$

$$\frac{d}{dt}p(t;1,1) = \vec{p}(t)\begin{bmatrix} 0 \\ \mu_1 \\ \mu_2 \\ -(\lambda_1+\lambda_2) \end{bmatrix} \tag{3.7}$$

Finalement les quatre équations (3.4), (3.5), (3.6) et (3.7) régissant l'évolution des états de notre système mono-buffer peuvent être écrites sous forme matricielle. C'est l'équation d'évolution de la chaîne de Markov en régime transitoire. :

$$\frac{d}{dt}\vec{p}(t) = \vec{p}(t)M \tag{3.8}$$

où

$$M = \begin{bmatrix} -(\mu_1+\mu_2) & \mu_2 & \mu_1 & 0 \\ \lambda_2 & -(\mu_1+\lambda_2) & 0 & \mu_1 \\ \lambda_1 & 0 & -(\lambda_1+\mu_2) & \mu_2 \\ 0 & \lambda_1 & \lambda_2 & -(\lambda_1+\lambda_2) \end{bmatrix}$$

M est le générateur de la chaîne de Markov représentée sur la figure 3.3. Il peut être exprimé synthétiquement comme la somme de Kronecker (voir annexe A) des générateurs markoviens des deux routeurs $R_1$ et $R_2$ :

$$\begin{aligned} M = M_1 \oplus M_2 &= \begin{bmatrix} -\mu_1 & \mu_1 \\ \lambda_1 & -\lambda_1 \end{bmatrix} \oplus \begin{bmatrix} -\mu_2 & \mu_2 \\ \lambda_2 & -\lambda_2 \end{bmatrix} \\ &= \begin{bmatrix} -(\mu_1+\mu_2) & \mu_2 & \mu_1 & 0 \\ \lambda_2 & -(\mu_1+\lambda_2) & 0 & \mu_1 \\ \lambda_1 & 0 & -(\lambda_1+\mu_2) & \mu_2 \\ 0 & \lambda_1 & \lambda_2 & -(\lambda_1+\lambda_2) \end{bmatrix} \end{aligned} \tag{3.9}$$

**Probabilités stationnaires des états** $(a,b)$

Le vecteur de probabilité stationnaire $\vec{w}$ associé à $M$ tel que $\vec{w}M = 0$ peut être obtenu directement comme le produit de Kronecker des vecteurs $\vec{w_1}$ et $\vec{w_2}$ puisque $M$ est la somme de Kronecker de $M_1$ et $M_2$. C'est une propriété de la somme de Kronecker : si $\epsilon_1$ et $\epsilon_2$ sont deux valeurs propres respectivement de $A_1$ et $A_2$ avec leurs vecteurs propres associés $\omega_1$ et $\omega_2$, alors $\epsilon_1+\epsilon_2$ est une valeur propre de $A_1 \oplus A_2$ et le vecteur propre associé est $\omega_1 \otimes \omega_2$ (Annexe A). C'est ce que nous faisons pour déterminer $\vec{w}$ vecteur propre de $M$ associé à la valeur propre 0.

$$\begin{aligned} \vec{w} &= \vec{w_1} \otimes \vec{w_2} = \\ &= \begin{bmatrix} w(0,0) & w(0,1) & w(1,0) & w(1,1) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\lambda_1}{\lambda_1+\mu_1}\frac{\lambda_2}{\lambda_2+\mu_2} & \frac{\lambda_1}{\lambda_1+\mu_1}\frac{\mu_2}{\lambda_2+\mu_2} & \frac{\mu_1}{\lambda_1+\mu_1}\frac{\lambda_2}{\lambda_2+\mu_2} & \frac{\mu_1}{\lambda_1+\mu_1}\frac{\mu_2}{\lambda_2+\mu_2} \end{bmatrix} \end{aligned}$$

### 3.2.4 Probabilités des états $(a, b, x)$

Lorsque l'on joint la variable d'état continue $x$, représentant le niveau du buffer, aux deux variables d'état discrètes $a$ et $b$ représentant l'état ON ou OFF des routeurs $R_1$ et $R_2$, on obtient un espace d'état hybride. La probabilité de l'état $(a, b, x)$ est alors :

$$P(t, x; a, b) = Pr[\text{niveau buffer} \leq x \text{ et l'état des routeurs à l'instant } t \text{ est} (a, b)]$$

De la même manière que précédemment, on peut noter $\vec{P}(t, x)$ le vecteur probabilité en régime transitoire :

$$\vec{P}(t, x) = \left[ \begin{array}{cccc} P(t; 0, 0; x) & P(t; 0, 1; x) & P(t; 1, 0; x) & P(t; 1, 1; x) \end{array} \right]$$

**Équations régissant l'évolution des états $(a, b, x)$**

L'équation dynamique peut être obtenue par une analyse infinitésimale des équations d'équilibre probabiliste et une approximation au premier ordre [AMS82]. On procède de la même manière que pour obtenir (3.8), mais en introduisant le niveau du buffer. Sur la figure 3.8, on voit les différentes possibilités de parvenir à l'état $(0, 0, x < X < x + dx)$. À partir de cette figure, on peut extraire une relation liant les probabilités de chaque état. Il est à remarquer que la probabilité à l'instant $t$ de l'état $(a, b, x_1 < X < x_2)$ est égale à $P(t; a, b; x_2) - P(t; a, b; x_1)$.

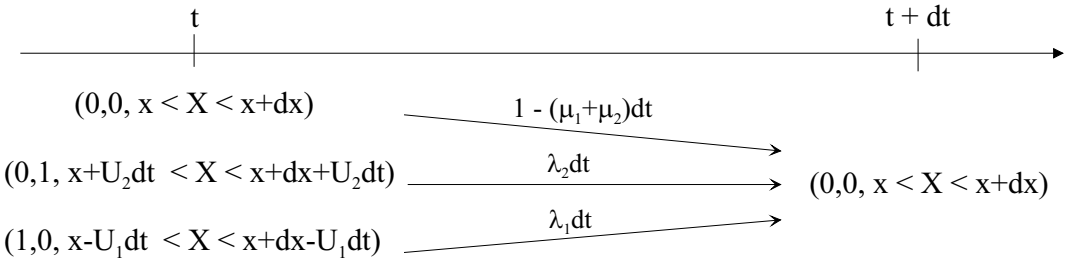


FIG. 3.8 – Évolution du système mono-buffer jusqu'à l'état $(0, 0, x < X < x + dx)$

$$\begin{array}{lll} P & (t + dt; 0, 0; x + dx) - P(t + dt; 0, 0; x) & \\ = & [P(t; 0, 0; x + dx) - P(t; 0, 0; x)] (1 - (\mu_1 + \mu_2))dt & \\ + & [P(t; 0, 1; x + dx + U_2 dt) - P(t; 0, 1; x + U_2 dt)] \lambda_2 dt & (3.10) \\ + & [P(t; 1, 0; x + dx - U_1 dt) - P(t; 1, 0; x - U_1 dt)] \lambda_1 dt & \end{array}$$

En utilisant le fait que :

$$P(t + dt; 0, 0; x + dx) - P(t + dt; 0, 0; x) = \frac{\partial}{\partial x} P(t + dt; 0, 0; x) dx$$

l'expression (3.10) peut s'écrire de la manière suivante :

$$\begin{aligned}\frac{\partial}{\partial x}P(t+dt;0,0;x)dx &= \frac{\partial}{\partial x}P(t;0,0;x)(1-(\mu_1+\mu_2)dt)dx \\ &+ \frac{\partial}{\partial x}P(t;0,1;x+U_2dt)\lambda_2 dtdx \\ &+ \frac{\partial}{\partial x}P(t;1,0;x-U_1dt)\lambda_1 dtdx\end{aligned}$$

$$\begin{aligned}\left[\frac{\partial}{\partial x}P(t+dt;0,0;x)-\frac{\partial}{\partial x}P(t;0,0;x)\right]dx &= -(\mu_1+\mu_2)\frac{\partial}{\partial x}P(t;0,0;x)dtdx \\ &+ \frac{\partial}{\partial x}P(t;0,1;x+U_2dt)\lambda_2 dtdx \\ &+ \frac{\partial}{\partial x}P(t;1,0;x-U_1dt)\lambda_1 dtdx\end{aligned}$$

Comme

$$\begin{aligned}P(t;0,1;x+U_2dt) &= P(t;0,1;x)+\frac{\partial}{\partial x}P(t;0,1;x)U_2dt \\ P(t;1,0;x-U_1dt) &= P(t;1,0;x)-\frac{\partial}{\partial x}P(t;1,0;x)U_1dt\end{aligned}$$

en simplifiant au premier ordre, on obtient comme équation :

$$\frac{\partial^2}{\partial x\partial t}P(t;0,0;x) = -(\mu_1+\mu_2)\frac{\partial}{\partial x}P(t;0,0;x)+\lambda_2\frac{\partial}{\partial x}P(t;0,1;x)+\lambda_1\frac{\partial}{\partial x}P(t;1,0;x)$$

En intégrant l'équation précédente par rapport à $x$ et en utilisant le vecteur $\vec{P}(t,x)$, on peut écrire :

$$\frac{\partial}{\partial t}P(t;0,0;x) = \vec{P}(t,x)\begin{bmatrix}-(\mu_1+\mu_2)\\ \lambda_2\\ \lambda_1\\ 0\end{bmatrix} \tag{3.11}$$

Il reste 3 équations de ce type. La seconde peut être trouvée en s'intéressant à l'état $(0,1,x)$ (figure 3.9).

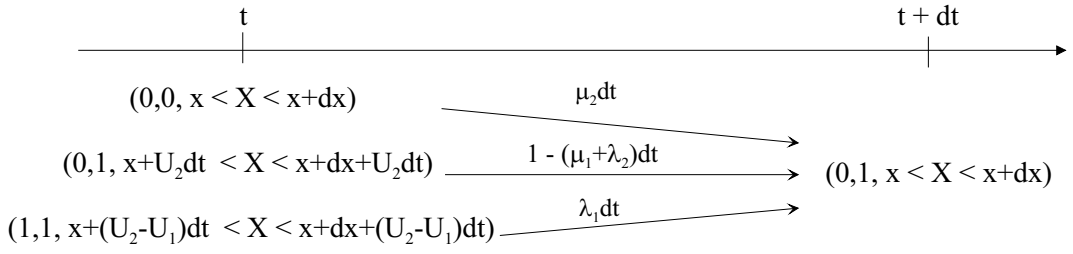


FIG. 3.9 – Évolution du système mono-buffer jusqu'à l'état $(0,1,x<X<x+dx)$

En écrivant la relation liant les probabilités de chaque état, on a :

$$\begin{aligned} P & (t+dt;0,1;x+dx) - P(t+dt;0,1;x) \\ = & \left[P(t;0,1;x+dx) - P(t;0,1;x)\right](1-(\mu_1+\lambda_2))dt \\ + & \left[P(t;0,0;x+dx+U_2dt) - P(t;1,0;x+U_2dt)\right]\mu_2 dt \\ + & \left[P(t;1,1;x+dx-U_1dt) - P(t;1,1;x-U_1dt)\right]\lambda_1 dt \end{aligned}$$

En simplifiant au premier ordre sur le même principe que précédemment, on obtient l'équation suivante :

$$\frac{\partial}{\partial t}P(t;0,1;x) = \frac{\partial}{\partial x}P(t;0,1;x)U_2 + \mu_2 P(t;0,0;x) - (\mu_1+\lambda_2)P(t;0,1;x) + \lambda_1 P(t;1,1;x)$$

Cette équation peut s'écrire sous la forme :

$$\frac{\partial}{\partial t}P(t;0,1;x) + \frac{\partial}{\partial x}\vec{P}(t,x)\begin{bmatrix} 0 \\ -U_2 \\ 0 \\ 0 \end{bmatrix} = \vec{P}(t,x)\begin{bmatrix} -(\mu_1+\mu_2) \\ \lambda_2 \\ \lambda_1 \\ 0 \end{bmatrix} \tag{3.12}$$

En procédant de la même manière pour les deux derniers états $(1,0,x)$ et $(1,1,x)$, on obtient les deux dernières équations différentielles. Sur la figure 3.10, on a représenté les différentes possibilités de parvenir à l'état $(1,0,x<X<x+dx)$ à l'instant $t+dt$.

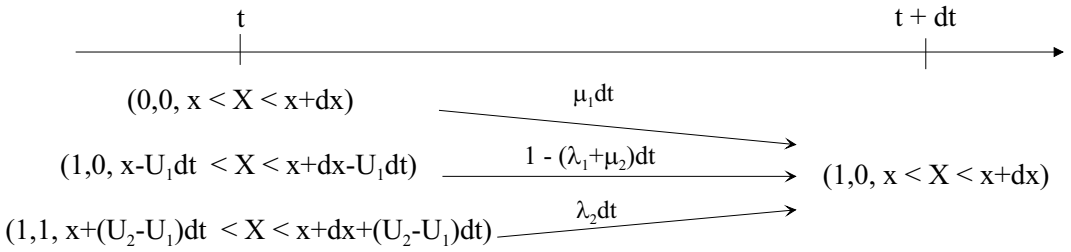


FIG. 3.10 – Évolution du système mono-buffer jusqu'à l'état $(1,0,x<X<x+dx)$

On obtient alors pour l'état $(1,0,x<X<x+dx)$ l'équation suivante qui peut s'écrire sous la forme (3.13).

$$\begin{aligned} P & (t+dt;1,0;x+dx) - P(t+dt;1,0;x) \\ = & \left[P(t;1,0;x+dx) - P(t;1,0;x)\right](1-(\lambda_1+\mu_2))dt \\ + & \left[P(t;0,0;x+dx+U_2dt) - P(t;0,1;x+U_2dt)\right]\mu_1 dt \\ + & \left[P(t;1,1;x+dx-U_1dt) - P(t;1,0;x-U_1dt)\right]\lambda_2 dt \end{aligned}$$

$$\frac{\partial}{\partial t}P(t;1,0;x) + \frac{\partial}{\partial x}\vec{P}(t,x)\begin{bmatrix} 0 \\ 0 \\ U_1 \\ 0 \end{bmatrix} = \vec{P}(t,x)\begin{bmatrix} \mu_1 \\ 0 \\ -(\lambda_1+\mu_2) \\ \lambda_2 \end{bmatrix} \tag{3.13}$$

Les transitions possibles pour atteindre le dernier état, $(1, 1, x < X < x + dx)$, sont représentées sur la figure 3.11.

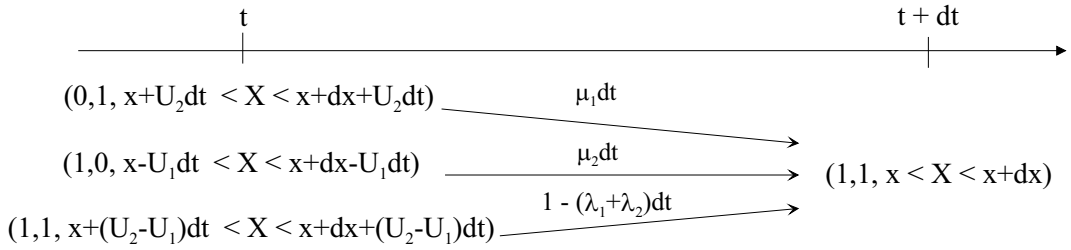


FIG. 3.11 – Évolution du système mono-buffer jusqu'à l'état $(1, 1, x < X < x + dx)$

En écrivant la relation qui lie les probabilités des différents états, on trouve :

$$\begin{aligned} P & (t + dt; 1, 1; x + dx) - P(t + dt; 1, 1; x) \\ = & \left[P(t; 1, 1; x + dx) - P(t; 1, 1; x)\right](1 - (\lambda_1 + \lambda_2))dt \\ + & \left[P(t; 0, 1; x + dx + U_2 dt) - P(t; 0, 1; x + U_2 dt)\right]\mu_1 dt \\ + & \left[P(t; 1, 0; x + dx - U_1 dt) - P(t; 1, 0; x - U_1 dt)\right]\mu_2 dt \end{aligned}$$

$$\frac{\partial}{\partial t}P(t; 1, 1; x) + \frac{\partial}{\partial x}\vec{P}(t, x)\begin{bmatrix} 0 \\ 0 \\ 0 \\ U_1 - U_2 \end{bmatrix} = \vec{P}(t, x)\begin{bmatrix} 0 \\ \mu_1 \\ \mu_2 \\ -(\lambda_1 + \lambda_2) \end{bmatrix} \tag{3.14}$$

Introduisons ici le vecteur $\vec{d}$, contenant les vitesses de variation du niveau du buffer selon l'état dans lequel on se trouve : c'est la dérive (ou « drift »). Le vecteur $\vec{d}$ est donc de dimension 4 et $\vec{P}(t, x)$ est la somme de Kronecker des vecteurs vitesses $\vec{v}_1$ et $\vec{v}_2$ (Annexe A) en tenant compte du signe (pour le routeur qui remplit le buffer, son vecteur vitesse est positif et le vecteur vitesse du routeur qui le vide est de signe négatif). Ici, on a donc :

$$\begin{aligned} \vec{d} &= \vec{v}_1 \oplus (-\vec{v}_2) \\ &= \vec{v}_1 \otimes \vec{1} - \vec{1} \otimes \vec{v}_2 \\ &= \begin{bmatrix} 0 & -U_2 & U_1 & U_1 - U_2 \end{bmatrix} \end{aligned}$$

La matrice diagonale $D$ contenant les éléments de $\vec{d}$ est appelée « matrice de drift ». Dans notre cas :

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -U_2 & 0 & 0 \\ 0 & 0 & U_1 & 0 \\ 0 & 0 & 0 & U_1 - U_2 \end{bmatrix} \tag{3.15}$$

Finalement (3.11),(3.12),(3.13), (3.14) peuvent s'écrire sous la forme d'un seule équation dynamique en utilisant la définition de la matrice de drift :

$$\frac{\partial}{\partial t}\vec{P}(t,x) + \frac{\partial}{\partial x}\vec{P}(t,x)D = \vec{P}(t,x)M \tag{3.16}$$

**Probabilités stationnaires des états** $(a,b,x)$

Soit $\vec{P}(x) = \lim_{t\to+\infty}\vec{P}(t,x)$ le vecteur contenant les probabilités stationnaires des états $(a,b,x)$.

$$\vec{P}(x) = \left[\begin{array}{cccc} P(0,0;x) & P(0,1;x) & P(1,0;x) & P(1,1;x) \end{array}\right]$$

Dans le cas du régime stationnaire, la distribution de probabilité vérifie la partie indépendante du temps de (3.16). Cependant, il existe des discontinuités dans l'expression de $\vec{P}(x)$ au niveau des états frontières ($x = 0$ buffer vide et $x = C_1$ buffer plein). Soit $\vec{\pi}(x)$ la partie absolument continue de $\vec{P}(x)$. Cette distribution est la solution de :

$$\frac{d}{dx}\vec{\pi}(x)D = \vec{\pi}(x)M \text{ pour } 0 \leq x \leq C_1 \tag{3.17}$$

La distribution de probabilité $\vec{P}(x)$ est complètement spécifiée si on rajoute les probabilités des états frontières.

**États de buffer vide** Les probabilités stationnaires des états $P(a,b;0)$ dans lesquels le buffer est vide sont définies par :

$$Pr[\text{buffer vide et état des routeurs } (a,b)] = \pi(a,b;0)$$

Les états de buffer vide dont le « drift » est positif ont une probabilité stationnaire nulle. En effet, un « drift » positif signifie que le buffer se remplit, la probabilité stationnaire de rester dans cet état de buffer vide est donc nulle. Il y a 1 ou 2 états de drift positif suivant le signe de $U_1 - U_2$ : dans l'état $(1,0)$, quelles que soient $U_1$ et $U_2$ (non nulles toutes les deux), le drift vaut $U_1$ et est donc toujours positif ; dans l'état $(1,1)$, le drift vaut $U_1 - U_2$, il est donc positif dans le cas où la vitesse de transfert de $R_1$ est plus grande que celle de $R_2$ ($U_1 > U_2$). On a donc :

$P(1,0;0) = 0$ quelles que soient $U_1$ et $U_2$ différentes de 0
$P(1,1;0) = 0$ si $U_1 > U_2$

**États de buffer plein** En ce qui concerne les états où le buffer est plein, les probabilités aux frontières sont définies de la manière suivante :

$$\begin{aligned} Pr[\text{buffer plein et état des routeurs } (a,b)] &= Pr[\text{état des routeurs } (a,b)] - \pi(a,b;C_1) \\ &= w(a,b) - \pi(a,b;C_1) \end{aligned}$$

Sur le même principe que pour les états de buffer vide, les états de buffer plein dont le drift est négatif ont une probabilité stationnaire nulle. En effet, un drift négatif signifie que le buffer se vide, la probabilité stationnaire de rester dans cet état de buffer plein est donc nulle. Il y a 1 ou 2 états de drift négatif suivant le signe de $U_1 - U_2$ : dans l'état $(0,1)$, quelles que soient $U_1$ et $U_2$ (non nulles toutes les deux), le « drift » vaut $-U_2$ et est donc toujours négatif ; dans l'état $(1,1)$, le drift vaut $U_1 - U_2$, il est donc négatif dans le cas où la vitesse de transfert de $R_2$ est plus grande que celle de $R_1$ ($U_2 > U_1$). On a donc :

$P(0,1;C_1) = 0$ quelles que soient $U_1$ et $U_2$ différentes de 0
$P(1,1;C_1) = 0$ si $U_2 > U_1$

Les équations (3.8) et (3.16) décrivent le comportement du système mono-buffer, nous allons résoudre ces équations afin de déterminer les indices de performance du système mono-buffer. Nous allons envisager deux cas : le cas du système mono-buffer homogène ($U_1 = U_2$) et le cas du système mono-buffer non-homogène ($U_1 \neq U_2$).

## 3.3 Système mono-buffer homogène

### 3.3.1 Présentation du système

Nous allons déterminer les indices de performance du système mono-buffer homogène. Le qualificatif « homogène » est employé pour préciser que les vitesses de transmission des deux routeurs sont identiques ($U_1 = U_2 = U$). Nous utiliserons un exemple pour faciliter la compréhension de l'étude analytique. Nous prendrons pour notre exemple le système présenté figure 3.12.

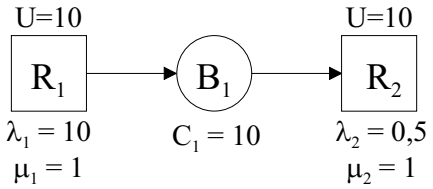


FIG. 3.12 – Exemple de système « mono-buffer homogène »

Les générateurs $M_1$ et $M_2$ des chaînes de Markov représentant l'évolution des routeurs $R_1$ et $R_2$ sont ($1^{er}$état $ON$, $2^{eme}$ état $OFF$) :

$$M_1 = \begin{bmatrix} -\mu_1 & \mu_1 \\ \lambda_1 & -\lambda_1 \end{bmatrix} \quad \text{et} \quad M_2 = \begin{bmatrix} -\mu_2 & \mu_2 \\ \lambda_2 & -\lambda_2 \end{bmatrix}$$
$$= \begin{bmatrix} -1 & 1 \\ 10 & -10 \end{bmatrix} \qquad = \begin{bmatrix} -1 & 1 \\ 0,5 & -0,5 \end{bmatrix}$$

Les vecteurs vitesses sont :

$$\vec{v}_1 = \begin{bmatrix} 0 & U \end{bmatrix} \quad \text{et} \quad \vec{v}_2 = \begin{bmatrix} 0 & U \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 10 \end{bmatrix} \qquad = \begin{bmatrix} 0 & 10 \end{bmatrix}$$

Les vecteurs de probabilités stationnaires sont :

$$\vec{w}_1 = \begin{bmatrix} 0,91 & 0,09 \end{bmatrix} \quad \text{et} \quad \vec{w}_2 = \begin{bmatrix} 0,33 & 0,67 \end{bmatrix}$$

Les matrices $M$ et $D$ intervenant dans les équations (3.8) et (3.16) régissant l'évolution du système mono-buffer sont obtenues à partir de (3.9) et (3.15) :

$$M = \begin{bmatrix} -2 & 1 & 1 & 0 \\ 0,5 & -1,5 & 0 & 1 \\ 10 & 0 & -11 & 1 \\ 0 & 10 & 0,5 & -10,5 \end{bmatrix} \quad \text{et} \quad D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

### 3.3.2 Solution

A cause de la singularité de la matrice $D$ le système d'équations (3.17) est un système d'équations différentielles avec contraintes algébriques linéaires (équivalent à ce qu'on trouve dans [DF82]). On permute les états pour regrouper à la fin les états de drift nul. Après une telle permutation l'ordre des états est, par exemple :

$$\{(1,0),(0,1),(1,1),(0,0)\}$$

Les matrices $D$ et $M$ permutées sont alors :

$$D = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M = \begin{bmatrix} -11 & 0 & 1 & 10 \\ 0 & -1,5 & 1 & 0,5 \\ 0,5 & 10 & -10,5 & 0 \\ 1 & 1 & 0 & -2 \end{bmatrix}$$

On peut considérer la partition suivante des matrices permutées $D$ et $M$ (partition conforme aux états correspondant à des vitesses non nulles de variation du niveau du buffer) :

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & 0 \end{bmatrix} = \left[\begin{array}{cc|cc} 10 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$$M = \begin{bmatrix} M_1 & M_{10} \\ M_{01} & M_0 \end{bmatrix} = \left[\begin{array}{cc|cc} -11 & 0 & 1 & 10 \\ 0 & -1,5 & 1 & 0,5 \\ \hline 0,5 & 10 & -10,5 & 0 \\ 1 & 1 & 0 & -2 \end{array}\right]$$

Supposons une partition correspondante du vecteur $\vec{\pi} = \left[ \begin{array}{cc} \vec{\pi}_1 & \vec{\pi}_0 \end{array} \right]$. On peut alors expliciter le système (3.17) :

$$\frac{d}{dx}\vec{\pi}_1 D_1 = \vec{\pi}_1 M_1 + \vec{\pi}_0 M_{01} \tag{3.18}$$

$$0 = \vec{\pi}_1 M_{10} + \vec{\pi}_0 M_0 \tag{3.19}$$

Grâce à l'irréductibilité de la matrice $M$, générateur markovien d'un système où tous les états communiquent, la sous-matrice $M_0$ est non singulière [BP70]. Il s'ensuit qu'on peut exprimer $\vec{\pi}_0$ en fonction de $\vec{\pi}_1$ à partir de (3.19) :

$$\vec{\pi}_0 = -\vec{\pi}_1 M_{10} M_0^{-1}. \tag{3.20}$$

En utilisant (3.20) dans (3.18) on obtient le système non singulier :

$$\frac{d}{dx}\vec{\pi}_1 D_1 = \vec{\pi}_1 (M_1 - M_{10} M_0^{-1} M_{01}) \tag{3.21}$$

Il a été prouvé [Mit88, ST99] que la matrice $(M_1 - M_{10} M_0^{-1} M_{01})$ est un générateur markovien.

En appliquant la transformation (3.21) à notre système, on obtient :

$$\frac{d}{dx}\vec{\pi}_1 \left[ \begin{array}{cc} U & 0 \\ 0 & -U \end{array} \right] = \beta \vec{\pi}_1 \left[ \begin{array}{cc} -\lambda_1\mu_2 & \lambda_1\mu_2 \\ \mu_1\lambda_2 & -\mu_1\lambda_2 \end{array} \right]$$

avec

$$\beta = \frac{\lambda_1 + \lambda_2 + \mu_1 + \mu_2}{(\lambda_1 + \lambda_2)(\mu_1 + \mu_2)}$$

On peut donc formuler le système d'équations différentielles ordinaires :

$$\frac{d}{dx}\vec{\pi}_1 = \frac{\beta}{U}\vec{\pi}_1 \left[ \begin{array}{cc} -\lambda_1\mu_2 & -\lambda_1\mu_2 \\ \mu_1\lambda_2 & \mu_1\lambda_2 \end{array} \right] \tag{3.22}$$

Ce qui donne pour notre exemple :

$$\frac{d}{dx}\vec{\pi}_1 = 0,06\vec{\pi}_1 \left[ \begin{array}{cc} -10 & -10 \\ 0,5 & 0,5 \end{array} \right] \tag{3.23}$$

Le problème est complètement spécifié par les conditions aux frontières, comme décrit à la fin de la section 3.2.4. Nous ne sommes pas dans le cas où $U_1 > U_2$ ni dans le cas où $U_1 < U_2$ puisque $U_1 = U_2 = U$, nous avons donc pour les probabilités aux frontières :

$$P(1,0;0) = \pi_1(1,0;0) = 0$$

$$P(0,1;C_1) = 0 \text{ , donc } \pi_1(0,1;C_1) = w(0,1) = \frac{\lambda_1}{\lambda_1+\mu_1}\frac{\mu_2}{\lambda_2+\mu_2} = 0,61$$

On peut résoudre le système (3.22) en calculant les valeurs et vecteurs propres du système. Un calcul élémentaire donne les deux valeurs propres suivantes :$\{0, \frac{\beta}{U}(\mu_1\lambda_2 - \lambda_1\mu_2)\}$. On distingue deux cas : le cas équilibré ($\lambda_2/\mu_2 = \lambda_1/\mu_1 = I$) et le cas non équilibré (notre exemple est un cas non équilibré).

**Cas équilibré :** $\lambda_2/\mu_2 = \lambda_1/\mu_1$

Dans le cas équilibré, 0 est une valeur propre double du système (3.22) avec les vecteurs propres :

$$\begin{aligned} \vec{\varphi}_1 &= \begin{bmatrix} 1 & 1 \end{bmatrix} \\ \vec{\varphi}_2 &= \begin{bmatrix} \dfrac{U}{-\beta I \mu_1 \mu_2} & 0 \end{bmatrix} \end{aligned}$$

et la solution générale est

$$\vec{\pi}_1(x) = a_1\vec{\varphi}_1 + a_2(\vec{\varphi}_2 + x\vec{\varphi}_1)$$

On utilise les conditions aux frontières afin d'obtenir les valeurs des constantes $a_1$ et $a_2$ et on obtient :

$$\begin{aligned} \pi(1,0;x) &= \frac{\mu_1\mu_2}{(1+I)(C_1\mu_1\mu_2(1+I)+U(\mu_1+\mu_2)}x \\ \pi(0,1;x) &= \frac{\mu_1\mu_2}{(1+I)(C_1\mu_1\mu_2(1+I)+U(\mu_1+\mu_2)}\left(\frac{U(\mu_1+\mu_2)}{(1+I)\mu_1\mu_2}+x\right) \end{aligned}$$

En appliquant la transformation (3.20) on obtient les distributions de probabilité des deux autres états :

$$\begin{aligned} \pi(1,1;x) &= \frac{\mu_1\mu_2}{I(1+I)(C_1\mu_1\mu_2(1+I)+U(\mu_1+\mu_2)}\left(\frac{U}{(1+I)\mu_2}+x\right) \\ \pi(0,0;x) &= \frac{I\mu_1\mu_2}{(1+I)(C_1\mu_1\mu_2(1+I)+U(\mu_1+\mu_2)}\left(\frac{U}{(1+I)\mu_1}+x\right) \end{aligned}$$

**Cas non équilibré :** $\lambda_2/\mu_2 \neq \lambda_1/\mu_1$

Dans le cas non équilibré, le système (3.22) possède deux valeurs propres distinctes : 0 et $\gamma = \beta/U(\mu_1\lambda_2 - \lambda_1\mu_2)$, avec les vecteurs propres associés :

$$\begin{aligned} \vec{\varphi}_0 &= \begin{bmatrix} \mu_1\lambda_2 & \lambda_1\mu_2 \end{bmatrix} \\ \vec{\varphi}_1 &= \begin{bmatrix} 1 & 1 \end{bmatrix} \end{aligned}$$

et la solution générale est :

$$\vec{\pi}_1(x) = a_1\vec{\varphi}_0 + a_2e^{(\gamma x)}\vec{\varphi}_1$$

Ce qui donne pour notre exemple :

$$\vec{\pi}_1(x) = a_1 \begin{bmatrix} 0,5 & 10 \end{bmatrix} + a_2e^{(-0,57x)} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

En utilisant les conditions aux frontières on détermine $a_1$ et $a_2$ :

$$\begin{aligned}\pi(1,0;x) &= \frac{\lambda_1\lambda_2\mu_1\mu_2}{\lambda_1\mu_2-\mu_1\lambda_2e^{\gamma C_1}}\frac{1-e^{\gamma x}}{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}\\ \pi(0,1;x) &= \frac{\lambda_1\mu_2}{\lambda_1\mu_2-\mu_1\lambda_2e^{(\gamma C_1)}}\frac{\lambda_1\mu_2-\mu_1\lambda_2e^{\gamma x}}{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}\end{aligned}$$

Par la transformation (3.20) :

$$\begin{aligned}\pi(1,1;x) &= \frac{\lambda_1\mu_1\mu_2}{(\lambda_1\mu_2-\mu_1\lambda_2e^{\gamma C_1})(\lambda_1+\mu_1)(\lambda_2+\mu_2)(\lambda_1+\lambda_2)}(\mu_2(\lambda_1+\lambda_2)-\lambda_2e^{\gamma x}(\mu_1+\mu_2))\\ \pi(0,0;x) &= \frac{\lambda_1\lambda_2\mu_2}{(\lambda_1\mu_2-\mu_1\lambda_2e^{\gamma C_1})(\lambda_1+\mu_1)(\lambda_2+\mu_2)(\mu_1+\mu_2)}(\lambda_1(\mu_1+\mu_2)-\mu_1e^{\gamma x}(\lambda_1+\lambda_2))\end{aligned}$$

L'application numérique donne :

$$\begin{aligned}\pi(1,0;x) &= 0,0303(1-e^{-0,57x})\\ \pi(0,1;x) &= 0,0606(10-0,5e^{-0,57x})\end{aligned}$$

Les matrices de la transformation (3.20) sont :

$$M_{10}=\begin{bmatrix}1 & 10\\ 1 & 0,5\end{bmatrix} \quad \text{et} \quad M_0^{-1}=\begin{bmatrix}-\frac{2}{21} & 0\\ 0 & -0,5\end{bmatrix}$$

On obtient alors pour $\vec{\pi_0}$ :

$$\vec{\pi_0}=\begin{bmatrix}0,0058(10.5-e^{-0,57x}) & 0,0152(20-10,5e^{-0,57x})\end{bmatrix}$$

Sur la figure 3.13, sont représentées les probabilités stationnaires déterminées précédemment dans le cas de notre exemple. Sur ce graphe on peut voir la proportion du temps où le routeur aval est non utilisé alors qu'il est disponible ($\pi(0,1;0)$). On constate également que pour $x > 3$, les probabilités stationnaires des 4 états sont pratiquement constantes, ce qui signifie qu'en régime stationnaire, on se retrouvera rarement avec un buffer dont le niveau dépassera 3.

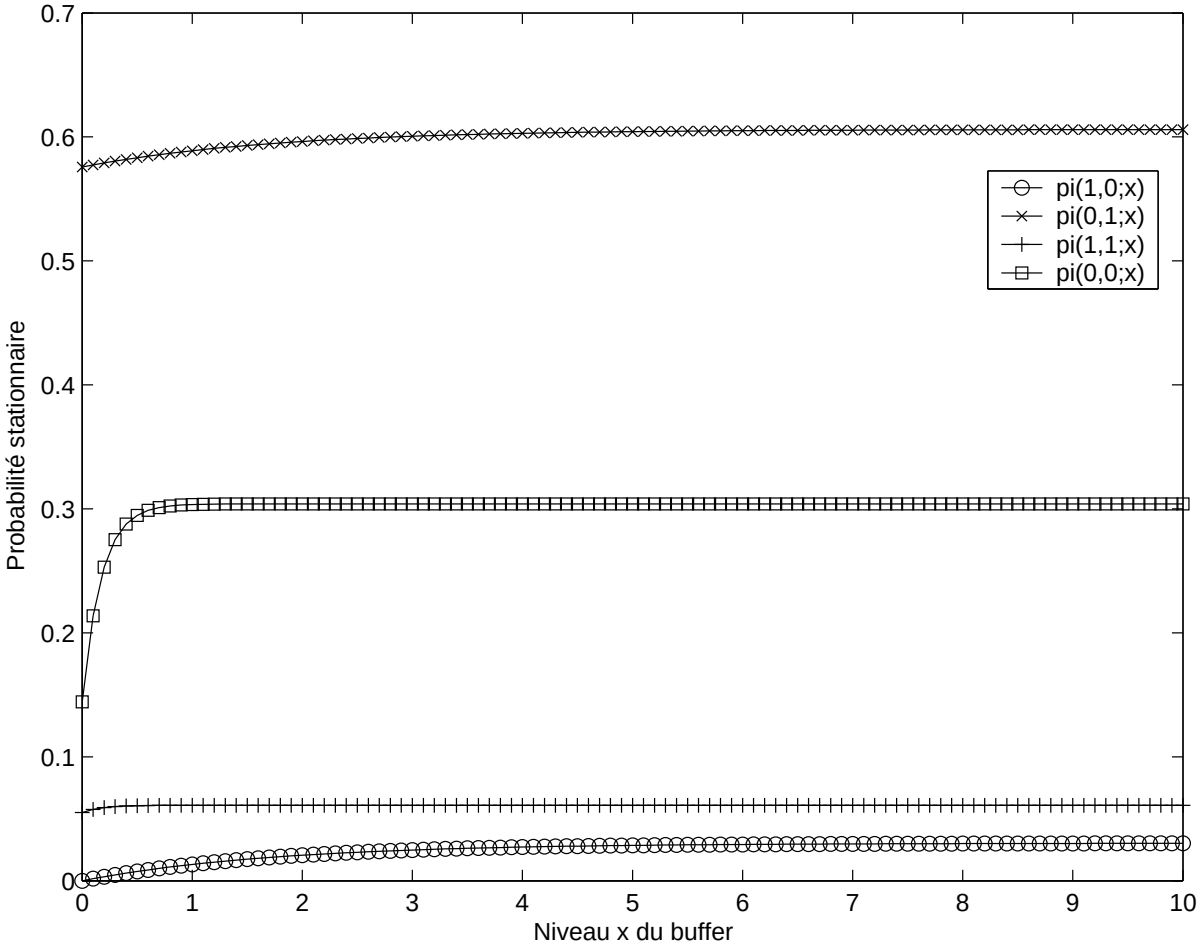


FIG. 3.13 – Probabilités stationnaires dans le cas de notre exemple

### 3.3.3 Performances du système mono-buffer

Les paramètres auxquels on s'intéresse sont le débit moyen de transmission, le taux de pertes et le niveau de remplissage moyen des buffers.

#### Débit moyen

On calcule le débit moyen, $X$, en multipliant la vitesse du routeur $R_2$ par la proportion de temps passée dans les états où des données sont transmises, c'est à dire quand le buffer est non-vide et que $R_2$ est ON ou bien quand $B_1$ est vide mais que $R_1$ et $R_2$ sont tous les deux ON.

$$\begin{aligned} X &= U(w(1,1) + w(0,1) - \pi(0,1;0)) \\ &= \begin{cases} U\left(\dfrac{\mu_2}{\lambda_2+\mu_2} - \dfrac{U(\mu_1+\mu_2)}{(1+I)^2(C_1\mu_1\mu_2(1+I)+U(\mu_1+\mu_2)}\right) & \text{équilibré} \\ U\left(\dfrac{\mu_2}{\lambda_2+\mu_2} - \dfrac{\lambda_1\mu_2}{\lambda_1\mu_2-\mu_1\lambda_2 e^{(\gamma C_1)}}\dfrac{\lambda_1\mu_2-\mu_1\lambda_2}{(\lambda_1+\mu_1)(\lambda_2+\mu_2)}\right) & \text{non équilibré} \end{cases} \end{aligned}$$

L'application numérique sur notre exemple, qui est un cas non équilibré, nous donne pour le débit moyen $X = 0,91$.

D'après l'expression analytique de $X$ (cas équilibré), on constate que si on augmente $\lambda_2$ , c'est à dire que l'on réduit la disponibilité du routeur aval, alors le débit moyen sera diminué. De même, si on augmente la capacité $C_1$ du buffer, on augmente également le débit moyen. Ceci confirme ce qu'on pouvait penser de manière intuitive.

### Pourcentage des pertes

Le seul état dans lequel le système enregistre des pertes est l'état $(1, 0, C_1)$, donc le pourcentage des pertes $Prt$ est égal à la probabilité stationnaire de cet état :

$$\begin{aligned} Prt &= w(1,0) - \pi(1,0;C_1) \\ &= \begin{cases} \dfrac{IC_1\mu_1\mu_2 + U(\mu_1+\mu_2)}{(1+I)(C_1\mu_1\mu_2(1+I) + U(\mu_1+\mu_2)} & \text{équilibré} \\ \dfrac{\mu_1\lambda_2}{(\lambda_1+\mu_1)(\lambda_2+\mu_2)} \dfrac{e^{(\gamma C_1)}(\lambda_1\mu_2 - \mu_1\lambda_2)}{\lambda_1\mu_2 - \mu_1\lambda_2 e^{(\gamma C_1)}} & \text{non équilibré} \end{cases} \end{aligned}$$

L'application numérique sur notre exemple qui est un cas non équilibré, nous donne pour la probabilité de pertes $P = 9,6.10^{-5}$. On constate dans l'expression analytique des pertes $Prt$ (cas non équilibré) que si $\mu_1\lambda_2$ est grand, alors les pertes sont élevées. $\mu_1$ grand correspond à un routeur amont qui ne reste pas longtemps indisponible et $\lambda_2$ grand correspond à un routeur aval très peu disponible, on comprend donc que les pertes soient élevées dans ce cas.

### Niveau moyen du buffer

On obtient le niveau moyen $\tilde{C_1}$ du buffer $B_1$ en tant que somme des niveaux moyens sur tous les états discrets et les états frontières.

$$\tilde{C_1} = \sum_{a,b} \left( C_1(w(a,b) - \pi(a,b;C_1)) + \int_0^{C_1} x f(a,b;x) dx \right)$$

où $f(x;a,b)$ est la densité de probabilité de l'état $(a,b,x)$.

L'application numérique donne $\tilde{C_1} = 0,40$. Le buffer n'est donc pas beaucoup occupé en moyenne, c'est ce qu'on avait déjà pu constater avec la figure 3.13.

On est ainsi capable de déterminer les indices de performance du cas mono-buffer avec les probabilités stationnaires de chaque état.

## 3.4 Cas mono-buffer non homogène

Dans la section 3.3, nous avons présenté la méthode de résolution du cas mono-buffer homogène, c'est-à-dire pour deux routeurs ayant la même vitesse de transmission $U$. Nous avons été capables de déterminer les probabilités stationnaires de tous les états, états frontières inclus, ainsi que de déterminer les indices de performances qui nous intéressent :

débit moyen de chaque routeur, taux de perte et occupation moyenne de chaque buffer. Nous allons dans cette section présenter brièvement la résolution du cas mono-buffer non homogène.

### 3.4.1 Présentation du système mono-buffer non homogène

Le système se compose de deux routeurs $R_1$ et $R_2$ séparés par le buffer $B_1$. Ils possèdent les caractéristiques décrites dans la section 3.4 et ont des vitesses de transfert $U_1$ et $U_2$ différentes, sinon on se rapporte au cas traité dans la section 3.3. La description du système reste valable. $M_1$ et $M_2$ sont les générateurs de la chaîne de Markov modélisant le fonctionnement des routeurs $R_1$ et $R_2$. Rappelons les différents éléments utilisés pour modéliser les routeurs $R_1$ et $R_2$ d'un système mono-buffer.

$$M_1 = \begin{bmatrix} -\lambda_1 & \lambda_1 \\ \mu_1 & -\mu_1 \end{bmatrix} \quad \text{et} \quad M_2 = \begin{bmatrix} -\lambda_2 & \lambda_2 \\ \mu_2 & -\mu_2 \end{bmatrix}$$

$$\vec{v_1} = \begin{bmatrix} U_1 & 0 \end{bmatrix} \quad \text{et} \quad \vec{v_2} = \begin{bmatrix} U_2 & 0 \end{bmatrix}$$

$$\vec{w_1} = \begin{bmatrix} \frac{\mu_1}{\lambda_1+\mu_1} & \frac{\lambda_1}{\lambda_1+\mu_1} \end{bmatrix} \quad \text{et} \quad \vec{w_2} = \begin{bmatrix} \frac{\mu_2}{\lambda_2+\mu_2} & \frac{\lambda_2}{\lambda_2+\mu_2} \end{bmatrix}$$

Le vecteur probabilité stationnaire associé à $M$ est :

$$\begin{aligned} \vec{w} &= \vec{w_1} \otimes \vec{w_2} = \\ &= \left[ \frac{\mu_1}{\lambda_1+\mu_1}\frac{\mu_2}{\lambda_2+\mu_2}, \frac{\mu_1}{\lambda_1+\mu_1}\frac{\lambda_2}{\lambda_2+\mu_2}, \frac{\lambda_1}{\lambda_1+\mu_1}\frac{\mu_2}{\lambda_2+\mu_2}, \frac{\lambda_1}{\lambda_1+\mu_1}\frac{\lambda_2}{\lambda_2+\mu_2} \right] \end{aligned}$$

Les matrices $M$ et $D$ intervenant dans les équations dynamiques (3.8) et (3.16) définissant les probabilités des états du système mono-buffer sont :

$$M = M_1 \oplus M_2 = \begin{bmatrix} -(\lambda_1+\lambda_2) & \lambda_2 & \lambda_1 & 0 \\ \mu_2 & -(\lambda_1+\mu_2) & 0 & \lambda_1 \\ \mu_1 & 0 & -(\mu_1+\lambda_2) & \lambda_2 \\ 0 & \mu_1 & \mu_2 & -(\mu_1+\mu_2) \end{bmatrix}$$

$$D = \begin{bmatrix} U_1-U_2 & 0 & 0 & 0 \\ 0 & -U_2 & 0 & 0 \\ 0 & 0 & U_1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

La distribution de probabilité $\vec{P}(x)$ est toujours obtenue à partir du système (3.17). Dans ce cas la permutation effectuée dans le cas homogène qui consistait à regrouper les états de drift nul n'a pas lieu d'être, car pour se trouver dans notre cas non homogène, on a $U_1$ et $U_2$ non égaux. Pour obtenir les probabilités stationnaires du système, on résout un

système du même type que (3.22) avec les conditions aux frontières correspondant au cas mono-buffer non homogène. On devra faire en particulier attention au signe de $U_1 - U_2$, afin de savoir si dans le cas où $U_1$ et $U_2$ sont $ON$ $B_1$ se remplit. On se reporte à la fin de la section 3.2.4 pour l'étude des probabilités stationnaires des états frontières. Dans le cas où $(U_1 - U_2 > 0)$ :

$$\begin{aligned} P(1,1;0) = \pi(1,1;0) = 0 \\ P(1,0;0) = \pi(1,0;0) = 0 \\ P(0,1;C_1) = 0 \end{aligned} \tag{3.24}$$

Sinon, dans le cas où $B_1$ se vide $(U_1 - U_2 < 0)$ :

$$\begin{array}{lcl} P(1,1;0) = \pi_1(1,1;0) = 0 & & \pi(1,1;0) = 0 \\ P(1,1;C_1) = 0 & \Leftrightarrow & \pi(1,1;C_1) = w(1,1) \\ P(0,1;C_1) = 0 & & \pi(0,1;C_1) = w(0,1) \end{array}$$

La résolution du système se fait sur le même principe que dans la section 3.3.2. Elle permet d'obtenir les probabilités stationnaires de chaque état. Nous ne donnerons pas ici de solution explicite comme nous l'avons fait pour le cas mono-buffer homogène, car elle fait intervenir une étude de vecteurs et de valeurs propres plus facilement soluble numériquement.

### 3.4.2 Indices de performance

On peut ensuite déterminer les indices de performance du système en triant les états qui nous intéressent de la même manière que dans la section 3.3.3.

#### Débit moyen

On calcule le débit moyen $X$ en multipliant la proportion de temps passée dans les états où des données sont transmises (états actifs sur la figure 4.11) par la vitesse de transmission de $R_2$ ou $R_1$ dans le cas où le buffer est vide, $R_1$ et $R_2$ sont ON et $U_1 < U_2$.

$$\begin{array}{ll} X = U_2(w(1,1) + w(0,1) - \pi(0,1;0)) & \text{pour } U_1 > U_2 \\ X = U_2(w(1,1) - \pi(1,1;0) + w(0,1) - \pi(0,1;0)) + U_1\pi(1,1;0) & \text{pour } U_1 < U_2 \end{array}$$

#### Pourcentage des pertes

À nouveau on différencie les cas où $U_1 > U_2$ et $U_1 < U_2$

**Cas $U_1 > U_2$.** On enregistre des pertes dans les états $(1,0,C_1)$ et $(1,1,C_1)$, donc les pertes s'expriment de la manière suivante :

$$Prt = U_1(w(1,0) - \pi(1,0;C_1)) + (U_1 - U_2)(w(1,1) - \pi(1,1;C_1)) \tag{3.25}$$

**Cas $U_1 < U_2$.** On enregistre des pertes dans un seul état $(1, 0, C_1)$, les pertes sont alors :

$$Prt = U_1(w(1,0) - \pi(1,0;C_1))$$

### Niveau moyen du buffer

On obtient le niveau moyen $\tilde{C_1}$ du buffer $B_1$ en tant que somme des niveaux moyens sur tous les états discrets et les états frontières.

$$\tilde{C_1} = \sum_{a,b} \left( C_1(w(a,b) - \pi(a,b;C_1)) + \int_0^{C_1} x f(a,b;x) dx \right)$$

On détermine ainsi toutes les performances du système mono-buffer non-homogène. Connaissant les temps moyens de disponibilité et d'indisponibilité des routeurs, leur vitesse de transmission, et la taille du buffer les séparant, on est capable de trouver les débits moyens des routeurs, le pourcentage de données perdues et le niveau moyen de remplissage du buffer.

Dans le cas non homogène ($U_1 \neq U_2$), en plus des temps de disponibilité et d'indisponibilité des routeurs amont et aval, on doit tenir compte du signe de $U_1 - U_2$ dans la détermination des performances du réseau, car il aura une influence sur le taux d'occupation des buffers.

### Conclusion

Les résultats obtenus dans ce chapitre sont des résultats préliminaires en vue de l'étude de réseaux plus complexes. Ce chapitre a permis de présenter les outils employés pour la modélisation des réseaux et la détermination de ses performances.

Dans le chapitre 4, nous développons un méthode d'évaluation de performance pour les réseaux linéaires homogènes puis non-homogènes en nous appuyant sur les résultats de ce chapitre.

# Chapitre 4

# RÉSOLUTION ANALYTIQUE DE TOPOLOGIES LINÉAIRES

On s'intéresse dans un premier temps à des réseaux simples : les réseaux à topologie linéaire. Nous nous intéresserons d'abord aux réseaux linéaires homogènes, puis aux réseaux linéaires non homogènes. Nous utiliserons, dans un premier temps, une méthode de décomposition adaptée de l'algorithme « DDX » que l'on trouve dans [DDX89]. Puis pour les réseaux non homogènes, nous envisagerons une autre méthode car l'algorithme « DDX »n'est pas adapté pour ce genre de réseaux.

## 4.1 Réseau linéaire homogène

Beaucoup de réseaux possèdent une « épine dorsale » (« backbone ») autour de laquelle est organisé le réseau. C'est en quelque sorte une « ligne » de routeurs. Avant d'envisager l'étude d'un réseau complexe, nous allons donc d'abord nous intéresser à un réseau linéaire de routeurs possédant la même vitesse de transmission $U$ (figure 4.1). C'est ce que nous appelons « réseau linéaire homogène ».

### 4.1.1 Modèle

Nous utilisons le modèle de routeur décrit dans la section 3.1. Les notations restent les mêmes. Comme nous sommes dans le cas d'une ligne de routeurs homogène en ce qui concerne la vitesse de transmission, nous notons $U_1 = U_2 = \ldots = U_K = U$. Le routeur $R_1$ émet des cellules à la vitesse $U$ à destination de $R_K$. Nous allons déterminer les débits moyens en sortie de chaque routeur de la ligne. Nous nous intéresserons également aux pertes ainsi qu'au niveau de remplissage moyen des buffers.

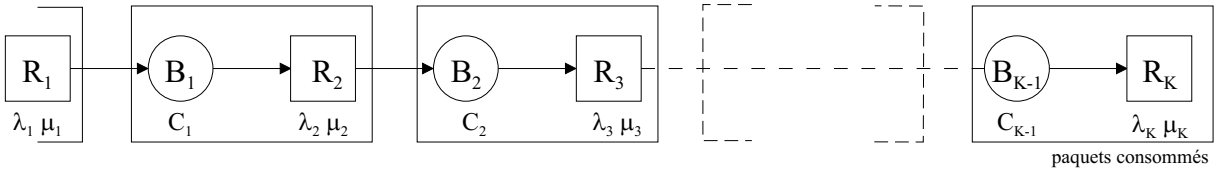


FIG. 4.1 – Réseau linéaire homogène

### 4.1.2 Présentation du réseau

Dans la description du fonctionnement des routeurs, nous avons fait l'hypothèse de non-contrôle du flux de données. C'est-à-dire qu'il n'y a pas de remontée d'information sur l'état du réseau : lorsqu'un buffer est plein, ceci n'a pas d'influence sur les routeurs situés en amont. Ces derniers continuent à transmettre des données à la vitesse $U$ tant que leur buffer est non-vide. De même, les routeurs de la partie aval transmettent à la vitesse $U$ quel que soit le niveau de remplissage non-vide de leur buffer. Cette hypothèse nous amène à remarquer que pour l'évaluation de performances du réseau, notamment du débit, 2 états sont à envisager pour le buffer : lorsqu'il est vide et lorsqu'il est non vide. Le fonctionnement des routeurs de la ligne complète peut donc être représenté par le graphe de la figure 4.2, où l'on distingue 4 états possibles pour un routeur $R_i$ :

- Transfert : le routeur $R_i$ transfère des données de $B_{i-1}$ vers $B_i$ à la vitesse $U$.
- Attente : le routeur $R_i$ est disponible, mais $B_{i-1}$ est vide.
- OFF 1 : le routeur $R_i$ est devenu indisponible alors qu'il transférait des données.
- OFF 2 : le routeur $R_i$ est devenu indisponible alors qu'il était en attente de données à transférer.

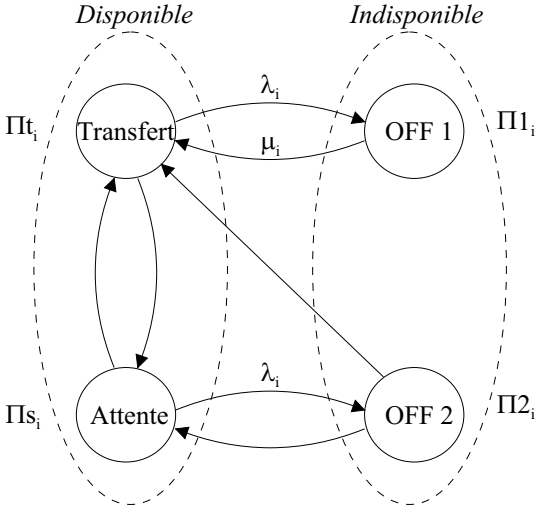


FIG. 4.2 – Évolution de l'état du routeur $R_i$

Aux états décrits précédemment, on associe les probabilités stationnaires suivantes :

- $\Pi t_i$ : probabilité stationnaire que $R_i$ transfère des données.
- $\Pi s_i$ : probabilité que le routeur $R_i$ soit en attente de données.

- $\Pi 1_i$ : probabilité que le routeur $R_i$ soit devenu indisponible alors qu'il transférait des données.
- $\Pi 2_i$ : probabilité que le routeur $R_i$ soit devenu indisponible alors qu'il était en attente de données.

Le graphe de la figure 4.2 ne peut pas être décrit comme une chaîne de Markov exacte modélisant le fonctionnement d'un routeur dans le réseau. En effet, on peut noter que certaines transitions entre les différents états ne sont pas markoviennes (taux non inscrits sur la figure). Ces transitions dépendent, en particulier, du niveau de remplissage des buffers et de la vitesse de transfert des routeurs. Nous verrons par la suite comment résoudre ce problème, mais établissons d'abord quelques relations caractéristiques du réseau.

### 4.1.3 Relations caractéristiques du réseau linéaire homogène

Le taux de transfert $X_i$ du routeur $R_i$ est lié à la vitesse du routeur et à la proportion du temps pendant lequel il transfère des données :

$$X_i = U\Pi \mathrm{t}_i \text{ pour } i \text{ de } 1 \text{ à } K \tag{4.1}$$

Si on note $\Pi \mathrm{f}_i$ la probabilité de pertes du buffer $B_i$, alors entre deux routeurs consécutifs $R_i$ et $R_{i+1}$, le taux de transfert évolue de la manière suivante :

$$X_{i+1} = X_i(1 - \Pi \mathrm{f}_i) \text{ pour } i \text{ de } 1 \text{ à } K-1 \tag{4.2}$$

De (4.1) et (4.2), nous obtenons :

$$\Pi \mathrm{t}_{i+1} = \Pi \mathrm{t}_i(1 - \Pi \mathrm{f}_i) \text{ pour } i \text{ de } 1 \text{ à } K-1 \tag{4.3}$$

On peut établir des relations à partir du graphe de fonctionnement des routeurs de la figure 4.2. Cependant, nous allons d'abord appliquer une méthode de décomposition afin de simplifier ce graphe. Nous appelons cette méthode « décomposition » afin de garder la même désignation que l'algorithme dont elle dérive ([DDX89]).

### 4.1.4 Décomposition

On peut facilement déterminer de manière analytique les performances d'un réseau composé de deux routeurs séparés par un buffer de la manière explicitée dans la section 3.3 sur le cas mono-buffer. Cela devient plus difficile quand la taille du réseau augmente. Nous allons donc utiliser une méthode de décomposition représentée sur la figure 4.3 : le réseau complet $L$ est décomposé en une série de $K-1$ réseaux simples ($L_1, \ldots, L_{K-1}$). Dans ce réseau simple, le premier routeur $R_i^u$ représente la partie du réseau qui se trouve en amont du buffer $B_i$ et le second routeur $R_i^d$ modélise la partie du réseau qui se situe en aval de $B_i$.

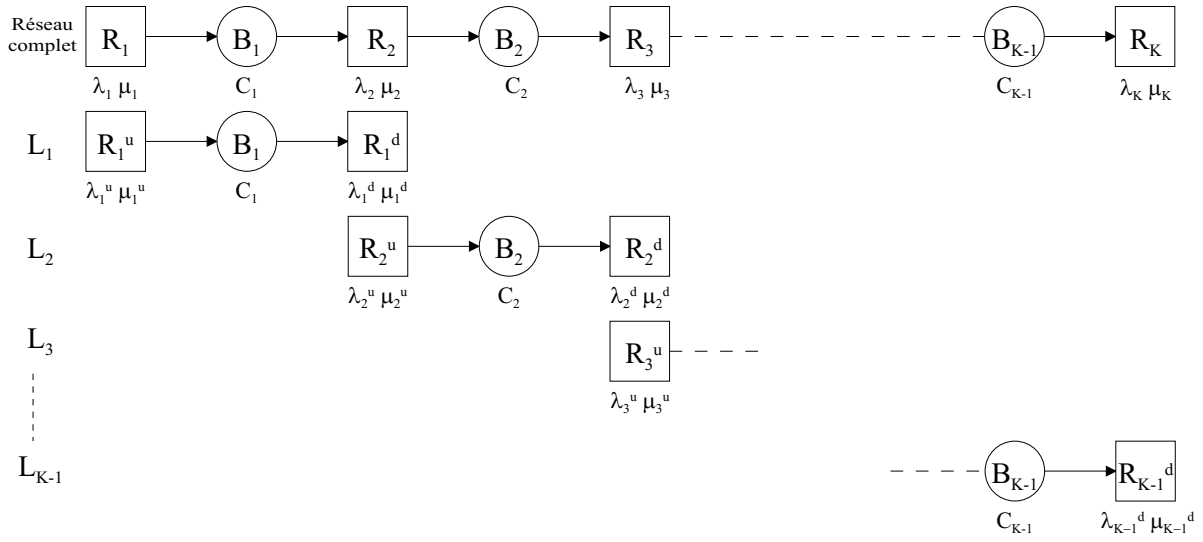


FIG. 4.3 – Principe de la décomposition

Cette décomposition permet de diminuer la complexité du système à étudier. En effet, le routeur amont $R_i^u$ peut prendre 2 états : ON pour signifier qu'il transmet des données et OFF lorsqu'il n'en transmet pas. Ce routeur ne peut pas être en attente de données puisqu'il n'est précédé par aucun buffer dans le sous-réseau $L_i$, c'est le routeur « source ». En ce qui concerne le routeur aval $R_i^d$, il peut prendre 4 états, ceux représentés sur la figure 4.2.

Donc, le principal objectif poursuivi dans ce qui suit est de déterminer les caractéristiques de $R_i^u$ et $R_i^d$, c'est à dire $\lambda_i^u$, $\mu_i^u$, $\lambda_i^d$ et $\mu_i^d$ pour $i$ de 1 à $K-1$.

### Caractéristiques de $R_i^u$ et $R_i^d$

Tout d'abord, puisqu'un routeur peut se trouver en attente de données mais jamais bloqué, on peut écrire :

$$\begin{aligned} &\text{Pour } i \text{ de 1 à } K-1 \\ &\lambda_i^d = \lambda_{i+1} \\ &\mu_i^d = \mu_{i+1} \end{aligned} \tag{4.4}$$

En effet, la partie aval de la ligne n'influence pas la partie amont, l'échange de données est unidirectionnel, les routeurs $R_i^d$ sont donc identiques aux routeurs $R_{i+1}$ pour i de 1 à $K-1$. Donc seules les caractéristiques $\lambda_i^u$ et $\mu_i^u$ de $R_i^u$ restent à déterminer.

Même si nous avons remarqué que les routeurs $R_i^d$ et $R_{i+1}$ étaient identiques et qu'ainsi $\lambda_i^d$ et $\mu_i^d$ étaient connus, dans la suite nous utiliserons tout de même des notations faisant intervenir l'exposant $^d$ pour plus de clarté dans le déroulement des calculs.

Intéressons nous aux relations dans le cas mono-buffer :

Taux de transfert :

$$X_i^u = U\Pi t_i^u \text{ pour } i \text{ de 1 à } K-1 \tag{4.5}$$

$$X_i^d = U\Pi t_i^d \text{ pour } i \text{ de 1 à } K-1 \tag{4.6}$$

L'équation (4.3) précédemment établie devient :

$$\Pi t_i^d = \Pi t_i^u (1 - \Pi f_i) \text{ pour } i \text{ de 1 à } K-1 \tag{4.7}$$

Introduisons à présent $w_i(1)$, la probabilité stationnaire que le routeur $R_i$ soit disponible (voir le chapitre 3.1 sur le cas mono-buffer ) :

$$w_i(1) = \frac{1/\lambda_i}{1/\mu_i + 1/\lambda_i} = \frac{1}{1 + \frac{\lambda_i}{\mu_i}} = \frac{1}{1 + I_i} \tag{4.8}$$

où $I_i = \frac{\lambda_i}{\mu_i}$ est le coefficient d'indisponibilité de $R_i$

$w_i(1)$ étant la probabilité stationnaire que le routeur $R_i$ soit disponible, on peut l'exprimer de la manière suivante (voir figure 4.2) :

$$w_i(1) = \Pi t_i + \Pi s_i \tag{4.9}$$

Effectivement, $R_i$ est disponible lorsqu'il transfère des données ou bien lorsqu'il est en attente. C'est ce que l'on a représenté sur la figure 4.2. On peut adapter cette équation au cas mono-buffer. Pour le routeur aval $R_i^d$, l'équation (4.9) reste la même, mais pour le routeur amont $R_i^u$ elle est simplifiée puisqu'il n'est jamais en attente de données. L'équation (4.9) devient :

$$\begin{aligned} w_i^d(1) &= \Pi t_i^d + \Pi s_i^d \text{ pour } i \text{ de 1 à } K-1 \\ w_i^u(1) &= \Pi t_i^u \text{ pour } i \text{ de 1 à } K-1 \end{aligned} \tag{4.10}$$

Le fonctionnement du routeur amont $R_i^u$ peut être modélisé de la manière représentée sur la figure 4.4. En effet les états « Attente » et « OFF2 » de la figure 4.2 représentant le fonctionnement d'un routeur dans le réseau complet ne sont pas accessibles pour un routeur « source ». Le graphe de fonctionnement s'en trouve simplifié.

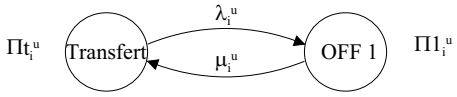


FIG. 4.4 – Chaîne de Markov représentant le fonctionnement du routeur $R_i^u$

On construit une chaîne de Markov « simple » décrivant le fonctionnement d'un routeur « équivalent » remplaçant un dipole. Lorsqu'il sera ON il transmettra des données et lorsqu'il sera OFF il n'en transmettra pas. On effectue, ici, une approximation en disant que les temps de séjour dans chacun des deux états (ON et OFF) suivent des lois exponentielles. En effet, on constate sur le graphe figure 4.2 que certaines transitions sont non markoviennes, elles dépendent notamment de la vitesse de transfert. Cependant étant donné que ces temps de séjour sont positifs, de moyenne bornée, on pourra les approximer par une loi exponentielle. Il nous suffit maintenant de déterminer les temps moyens de séjour.

A partir de la chaîne de Markov de la figure 4.4, nous pouvons écrire l'équation d'équilibre $\lambda_i^u \Pi t_i^u = \mu_i^u \Pi 1_i^u$ qui, en introduisant le coefficient d'indisponibilité de $R_i^u$, devient :

$$\Pi 1_i^u = I_i^u \Pi t_i^u \tag{4.11}$$

Si notre décomposition est correcte, les taux de transfert de $R_i^u$ et $R_i^d$ devraient être, respectivement, les mêmes que ceux de $R_i$ et $R_{i+1}$ :

$$X_i^u = X_i \text{ pour } i \text{ de 1 à } K-1 \tag{4.12}$$

Grâce à (4.1), (4.5) et (4.12), on peut écrire :

$$\Pi t_i^u = \Pi t_i \text{ pour } i \text{ de 1 à } K-1 \tag{4.13}$$

Maintenant, en ce qui concerne l'état « Attente », la probabilité pour le routeur $R_i^d$ d'être dans cet état est la même que pour le routeur $R_{i+1}$, car comme déjà expliqué précédemment, ces routeurs ont le même comportement :

$$\Pi s_i^d = \Pi s_{i+1} \text{ pour } i \text{ de 1 à } K-1 \tag{4.14}$$

**Relations basées sur le fonctionnement ON/OFF des routeurs**

A présent, intéressons-nous au mécanisme ON/OFF de $R_i^u$ : $R_i^u$ se retrouve dans l'état OFF pour représenter le fait que $R_i$ est indisponible ou en attente de données. $R_i$ est en attente de données si $R_{i-1}^u$ est OFF et $B_{i-1}$ est vide.

Posons les notations suivantes :

- $t_i^u$ et $t_i$ sont les temps moyens d'indisponibilité des routeurs $R_i^u$ et $R_i$,
- $r_{i-1}^u$ est le temps moyen résiduel d'indisponibilité du routeur $R_{i-1}^u$ quand $R_i$ est en attente de données,
- $\alpha_i$ est la probabilité que la cause de l'état OFF de $R_i^u$ soit l'état OFF de $R_{i-1}^u$ et $B_{i-1}$ vide.

Concernant le mécanisme ON/OFF de $R_i^u$, on peut écrire :

$$t_i^u = \alpha_i r_{i-1}^u + (1-\alpha_i) t_i \text{ pour } i \text{ de 1 à } K-1 \tag{4.15}$$

L'équation (4.15) traduit le fait que l'état OFF du routeur équivalent $R_i^u$ représente les attentes de données (famines) de $R_i$ survenant dans des proportions $\alpha_i$ et qu'il représente également les états OFF de $R_i$ dans des proportions $(1-\alpha_i)$.

D'après la définition de $\alpha_i$, c'est le nombre de fois où l'état OFF de $R_i^u$ représente un état d'attente de données de $R_i$ divisé par le nombre de fois où $R_i^u$ est OFF, donc :

$$\alpha_i = \frac{\Pi s_{i-1}^d}{r_{i-1}^u} \frac{t_i^u}{\Pi 1_i^u} \text{ pour } i \text{ de 1 à } K-1 \tag{4.16}$$

avec :

$\Pi s_{i-1}^d / r_{i-1}^u$ est le nombre de fois où l'état OFF est dû à un état d'attente de données de $R_i$, ceci se produit lorsque $R_{i-1}^u$ est en état OFF alors que $B_{i-1}$ est vide.

$\Pi 1_i^u / t_i^u$ est le nombre de fois où le routeur $R_i^u$ est dans l'état OFF. Nous pouvons remarquer, en observant la figure 4.2, que $\Pi 2_i^u = 0$ puisque $R_i^u$ ne peut jamais être en attente de données, donc $\Pi 1_i^u$ est la probabilité que $R_i^u$ soit OFF.

**Obtention d'une relation de récurrence sur les caractéristiques du routeur $R_i^u$**

En remplaçant $\Pi 1_i^u$ dans (4.16) grâce à (4.11), nous obtenons :

$$\alpha_i = \frac{\Pi s_{i-1}^d}{r_{i-1}^u} \frac{t_i^u}{I_i^u \Pi t_i^u} \text{ pour } i \text{ de 1 à } K-1 \tag{4.17}$$

Définissons $\gamma_i$ :

$$\gamma_i = \frac{\Pi s_{i-1}^d}{I_i^u \Pi t_i^u}$$

Avec cette notation, (4.17) devient :

$$\alpha_i = \gamma_i \frac{t_i^u}{r_{i-1}^u} \tag{4.18}$$

Grâce à (4.18), la relation (4.15) qui provient du mécanisme ON/OFF peut être écrite :

$$\frac{1}{t_i^u} = \frac{\gamma_i}{r_{i-1}^u} + \frac{1-\gamma_i}{t_i} \text{ pour } i \text{ de 1 à } K-1 \tag{4.19}$$

On remarque que $t_i^u = 1/\mu_i^u$, $t_i = 1/\mu_i$. Comme les routeurs fonctionnent selon un processus sans mémoire, $r_{i-1}^u$, temps moyen résiduel d'indisponibilité de $R_{i-1}^u$, suit également une loi exponentielle de taux $\mu_{i-1}^u$, donc : $r_{i-1}^u = 1/\mu_{i-1}^u$. Ainsi (4.19) devient l'équation récurrente :

$$\mu_i^u = \gamma_i \mu_{i-1}^u + (1-\gamma_i)\mu_i \text{ pour } i \text{ de 2 à } K-1 \tag{4.20}$$

L'équation (4.20) nous donne donc la caractéristique $\mu_i^u$ du routeur $R_i^u$, grâce à différents éléments :

- $\mu_{i-1}^u$ : cette caractéristique est connue puisqu'elle a été déterminée par la même équation à l'itération précédente.
- $\mu_i$ : c'est le taux d'indisponibilité du routeur $R_i$, c'est une caractéristique connue.
- $\gamma_i$ : $\gamma_i = \Pi s_{i-1}^d / (I_i^u \Pi t_i^u)$, nous allons expliciter dans la suite les différents éléments de cette équation afin de déterminer $\gamma_i$.

Nous allons donc maintenant déterminer les différents éléments permettant de calculer $\gamma_i$. La probabilité stationnaire $\Pi s_{i-1}^d$ peut être obtenue en analysant le réseau $L_i$ et en utilisant les résultats connus pour les systèmes simples de deux routeurs séparés par un buffer [Mit88]. Il nous reste donc à déterminer $I_i^u \Pi t_i^u$.

Grâce à (4.8), nous avons :

$$I_i^u \Pi t_i^u = 1 - \Pi t_i^u \tag{4.21}$$

En utilisant (4.9), (4.13), (4.14) nous obtenons pour $\Pi t_i^u$ :

$$\Pi t_i^u = w_i(1) - \Pi s_{i-1}^d \tag{4.22}$$

Dans l'équation (4.22), $w_i(1)$ est une caractéristique du routeur $R_i$ et $\Pi s_{i-1}^d$ peut être obtenue comme expliqué précédemment. Donc grâce à (4.21) et (4.22), on peut obtenir $\gamma_i$ :

$$\gamma_i = \frac{\Pi s_{i-1}^d}{1 + \Pi s_{i-1}^d - w_i(1)}$$

Ensuite on obtient $\mu_i^u$ grâce à (4.20).

Le dernier taux à déterminer pour caractériser le routeur $R_i^u$ est $\lambda_i^u$. Avec (4.21) et (4.22), on peut déterminer $I_i^u$, donc $\lambda_i^u$ puisqu'on connaît $\mu_i^u$ :

$$\lambda_i^u = \mu_i^u \left( \frac{1}{w_i(1) \Pi s_{i-1}^d} - 1 \right)$$

Une fois le réseau décomposé en une série de dipoles, il est possible d'obtenir les indices de performance tels que l'occupation moyenne des buffers, le débit à la sortie de chaque routeur et le taux de pertes. Il suffit de résoudre chaque dipole $L_i$ successivement pour $i$ de 1 à $K-1$. Pour la résolution du dipole, on procède de la manière présentée dans la section traitant de ce sujet comme on le fait dans [Mit88] ou [ST99]. On résout le dipole du sous-réseau $L_i$ pour i de 1 à $K-1$, il possède les caractéristiques déterminées précédemment :

$\lambda_i^u$ et $\mu_i^u$ pour le routeur amont $R_i^u$
$\lambda_i^d$ et $\mu_i^d$ pour le routeur aval $R_i^d$
$C_i$ la capacité du buffer $B_i$

### 4.1.5 Algorithme de décomposition

L'algorithme proposé permet de déterminer par récurrence les caractéristiques des routeurs $R_i^u$ et $R_i^d$ des sous-réseaux $L_i$. En ce qui concerne les routeurs $R_i^d$ il suffit de leur assigner les caractéristiques des routeurs du réseau complet $R_{i+1}$ pour les raisons expliquées dans les sections précédentes :

$$\lambda_i^d = \lambda_{i+1} \text{ pour } i \text{ de 1 à } K-1$$
$$\mu_i^d = \mu_{i+1} \text{ pour } i \text{ de 1 à } K-1$$

Les relations de récurrence portent donc sur les routeurs $R_i^u$. On initialise l'algorithme de la manière suivante :

$$\lambda_1^u = \lambda_1$$
$$\mu_1^u = \mu_1$$

On a ainsi défini le premier sous-réseau $L_1$ que l'on peut étudier. Avec des relations analytiques connues et assez simples, on trouve les performances qui nous intéressent comme le taux de transfert ou le taux d'occupation du buffer, de la manière évoquée à la fin du paragraphe précédent. On détermine également les caractéristiques de $L_1$

qui vont permettre ensuite de construire $L_2$ grâce aux relations de récurrence définies précédemment :

$$\begin{aligned}\mu_2^u &= \frac{\Pi s_1^d}{1+\Pi s_1^d - w_2(1)}\mu_1^u + \left(1 - \frac{\Pi s_1^d}{1+\Pi s_1^d - w_2(1)}\right)\mu_2 \\ \lambda_2^u &= \mu_2^u\left(\frac{1}{w_2(1)\Pi s_1^d} - 1\right)\end{aligned}$$

où $\Pi s_1^d$ provient de l'étude du dipole $L_1$.

On poursuit ainsi jusqu'au sous-réseau $L_{N-1}$ avec les relations suivantes pour le sous-réseau $L_i$ :

$$\begin{aligned}\mu_i^u &= \frac{\Pi s_{i-1}^d}{1+\Pi s_{i-1}^d - w_i(1)}\mu_{i-1}^u + \left(1 - \frac{\Pi s_{i-1}^d}{1+\Pi s_{i-1}^d - w_i(1)}\right)\mu_i \\ \lambda_i^u &= \mu_i^u\left(\frac{1}{w_i(1)\Pi s_{i-1}^d} - 1\right)\end{aligned}$$

où $\Pi s_{i-1}^d$ est toujours déterminé à partir de l'étude analytique du dipole précédent $L_{i-1}$

### 4.1.6 Résultats numériques

Dans cette section, nous appliquons notre algorithme sur un réseau linéaire homogène de cinq routeurs dont on fait varier les périodes de disponibilité et d'indisponibilité. Chaque routeur a une vitesse de transfert constante $U = 10$ unités par seconde. Les buffers de chaque routeur du réseau ont chacun une capacité $C_i = 10$ unités.

#### Exemple 1

Dans un premier temps on prend des caractéristiques identiques pour ces routeurs (voir réseau figure 4.5) : $\lambda_i = 1$ et $\mu_i = 1$ pour $i$ de 1 à 5. Les résultats concernant les débits moyens, les pertes et le niveau d'occupation des buffers sont regroupés dans les tableaux 4.1, 4.3, 4.2.

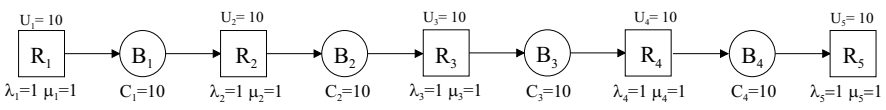


FIG. 4.5 – Exemple d'application numérique 1

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 5,0000 | 5,0073 | 0,15 |
| Routeur 2 | 3,7500 | 3,7568 | 0,18 |
| Routeur 3 | 3,1208 | 3,1186 | 0,07 |
| Routeur 4 | 2,7343 | 2,7245 | 0,36 |
| Routeur 5 | 2,4700 | 2,4514 | 0,76 |

TAB. 4.1 – Débits moyens des routeurs du réseau pour $I_i = 1$

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 5,0000 | 5,0030 | 0,06 |
| Buffer 2 | 3,6871 | 3,7108 | 0,64 |
| Buffer 3 | 3,0151 | 3,0415 | 0,88 |
| Buffer 4 | 2,5985 | 2,6263 | 1,07 |

TAB. 4.2 – Niveau moyen des buffers du réseau pour $I_i = 1$

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 1,2500 | 1,2519 | 0,15 |
| Routeur 2 -> Routeur 3 | 0,6292 | 0,6337 | 0,72 |
| Routeur 3 -> Routeur 4 | 0,3865 | 0,3925 | 1,55 |
| Routeur 4 -> Routeur 5 | 0,2643 | 0,2733 | 3,41 |

TAB. 4.3 – Débits moyens de pertes pour $I_i = 1$

**Exemple 2**

Dans ce deuxième exemple, on modifie les caractéristiques suivantes : $\lambda_i = 10$ et $\mu_i = 1$ pour $i$ de 1 à 5 (voir figure 4.6). Les résultats sont regroupés dans les tableaux 4.4, 4.6 et 4.5.

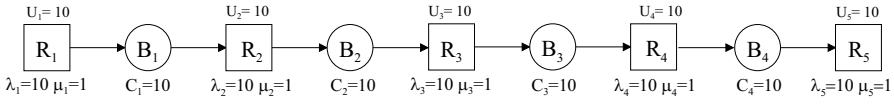


FIG. 4.6 – Exemple d'application numérique 2

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 0,9091 | 0,9092 | 0,01 |
| Routeur 2 | 0,7819 | 0,7818 | 0,01 |
| Routeur 3 | 0,7222 | 0,7274 | 0,72 |
| Routeur 4 | 0,6851 | 0,6938 | 1,26 |
| Routeur 5 | 0,6590 | 0,6697 | 1,62 |

TAB. 4.4 – Débits moyens des routeurs du réseau pour $I_i = 10$

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 5,0000 | 4,9984 | 0,03 |
| Buffer 2 | 3,9650 | 4,0138 | 1,23 |
| Buffer 3 | 3,4233 | 3,5157 | 2,70 |
| Buffer 4 | 3,0791 | 3,2053 | 4,10 |

TAB. 4.5 – Niveau moyen des buffers du réseau pour $I_i = 10$

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 0,1272 | 0,1271 | 0,08 |
| Routeur 2 -> Routeur 3 | 0,0597 | 0,0542 | 10,15 |
| Routeur 3 -> Routeur 4 | 0,0371 | 0,0340 | 9,12 |
| Routeur 4 -> Routeur 5 | 0,0261 | 0,0243 | 7,41 |

TAB. 4.6 – Débits moyens de pertes pour $I_i = 10$

**Exemple 3**

Dans cet exemple, on prend $\lambda_i = 0,1$ et $\mu_i = 1$ pour $i$ de 1 à 5 (figure 4.7). Les résultats sont présentés dans les tableaux 4.7, 4.9 et4.8.

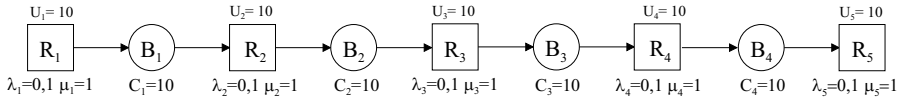


FIG. 4.7 – Exemple d'application numérique 3

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 9,0909 | 9,0926 | 0,02 |
| Routeur 2 | 8,5577 | 8,5579 | 0,00 |
| Routeur 3 | 8,1316 | 8,1299 | 0,02 |
| Routeur 4 | 7,7674 | 7,7636 | 0,05 |
| Routeur 5 | 7,4462 | 7,4393 | 0,09 |

TAB. 4.7 – Débits moyens des routeurs du réseau pour $I_i = 0,1$

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 5,0000 | 5,0080 | 0,16 |
| Buffer 2 | 3,6534 | 3,6680 | 0,40 |
| Buffer 3 | 2,9361 | 2,9568 | 0,71 |
| Buffer 4 | 2,4766 | 2,5139 | 1,51 |

TAB. 4.8 – Niveau moyen des buffers du réseau pour $I_i = 0,1$

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 0,5332 | 0,5328 | 0,08 |
| Routeur 2 -> Routeur 3 | 0,4261 | 0,4260 | 0,02 |
| Routeur 3 -> Routeur 4 | 0,3642 | 0,3658 | 0,44 |
| Routeur 4 -> Routeur 5 | 0,3212 | 0,3246 | 1,06 |

TAB. 4.9 – Débits moyens de pertes pour $I_i = 0,1$

**Exemple 4**

On peut également effectuer une comparaison lorsque les routeurs sont de moins en moins disponibles à mesure que l'on avance dans le réseau (figure 4.8) : $\lambda_i = 2$ x $i$ et $\mu_i = 1$ pour $i$ de 1 à 5 (tableaux 4.7, 4.9 et 4.8).

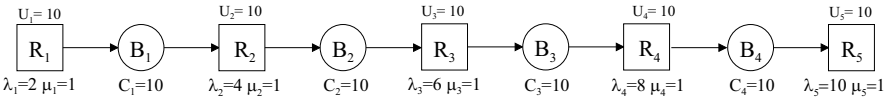


FIG. 4.8 – Exemple d'application numérique 4

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 3,3333 | 3,3390 | 0,17 |
| Routeur 2 | 1,7976 | 1,7976 | 0,00 |
| Routeur 3 | 1,2520 | 1,2612 | 0,73 |
| Routeur 4 | 0,9747 | 0,9868 | 1,24 |
| Routeur 5 | 0,8041 | 0,8149 | 1,34 |

TAB. 4.10 – Taux de transfert des routeurs du réseau pour des routeurs de moins en moins disponibles

| | Décomposition | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 6,9944 | 7,0047 | 0,14 |
| Buffer 2 | 6,0827 | 6,1499 | 1,10 |
| Buffer 3 | 5,6696 | 5,8081 | 2,44 |
| Buffer 4 | 5,4520 | 5,6393 | 3,44 |

TAB. 4.11 – Niveau moyen des buffers du réseau pour des routeurs de moins en moins disponibles

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 1,5357 | 1,5348 | 0,06 |
| Routeur 2 -> Routeur 3 | 0,5456 | 0,5366 | 1,68 |
| Routeur 3 -> Routeur 4 | 0,2773 | 0,2739 | 1,24 |
| Routeur 4 -> Routeur 5 | 0,1706 | 0,1724 | 1,06 |

TAB. 4.12 – Débits moyens de pertes pour des routeurs de moins en moins disponibles

A travers ces comparaisons, on constate que la méthode de décomposition développée est relativement efficace. L'utilisation de la méthode analytique permet, en comparaison à la simulation, d'obtenir rapidement des résultats qui décrivent assez bien le comportement du système réel. Dans les cas étudiés précédemment, les simulations nécessitent un temps de l'ordre de la dizaine de secondes, alors que la méthode analytique fournit des résultats quasi instantanément comme le montre le tableau 4.13 indiquant les temps mis pour obtenir les résultats par simulation et par la méthode analytique (sur un Pentium 4 à 2GHz).

| | Méthode analytique | Simulation |
|---|---|---|
| Exemple 1 | $\sim$ 1s | 80s |
| Exemple 2 | $\sim$ 1s | 80s |
| Exemple 3 | $\sim$ 1s | 60s |
| Exemple 4 | $\sim$ 1s | 40s |

TAB. 4.13 – Temps d'obtention des résultats

## 4.2 Réseau linéaire non homogène

Les réseaux étudiés dans la section précédentes avaient comme particularité que tous les routeurs avaient la même vitesse de transfert $U$. Cette caractéristique avait l'avantage de simplifier des équations dans la résolution du réseau. Cependant dans les réseaux réels on se retrouve rarement dans ce genre de situation. L'algorithme développé dans la section 4.1 ne peut donc pas être appliqué aux réseaux non homogènes. Nous allons donc à présent introduire le fait que chaque routeur peut avoir une vitesse de transfert différente et présenter un algorithme de manière à pouvoir extraire du réseau les caractéristiques qui nous intéressent.

### 4.2.1 Modèle

On conserve le même modèle de routeur que décrit dans la section 3.1. Le réseau étudié est linéaire, on considère $K$ routeurs ($R_1$, ..., $R_K$), où $R_1$ est la source, avec $K-1$ buffers ($B_1$, ..., $B_{K-1}$) de capacités finies ($C_1$, ..., $C_{K-1}$). Il est représenté figure 4.9. On suppose la ligne non homogène, c'est-à-dire que les vitesses de transmission des routeurs ne sont pas toutes égales entre elles.

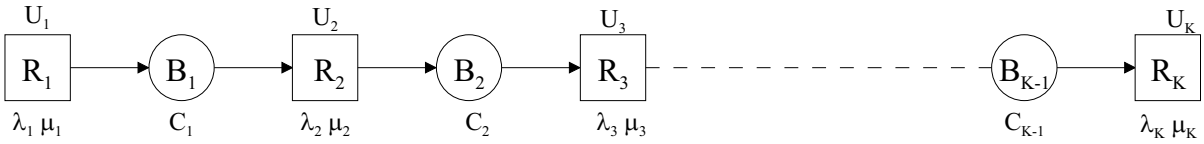


FIG. 4.9 – Réseau linéaire non homogène

Le fonctionnement du réseau reste le même que précédemment. Lorsque $B_i$ est plein, si des données sont transmises par $R_i$, elles sont perdues. Quand $B_i$ est vide, $R_{i+1}$ transmet des données à la vitesse $U_i$ seulement si $R_i$ est en train de transmettre des données (on remarque que ceci se produit seulement si $U_i < U_{i+1}$).

### 4.2.2 Méthode d'agrégation

Nous allons étudier une méthode d'agrégation qui va nous permettre de caractériser un système complexe en utilisant les résultats obtenus sur le système mono-buffer.

#### Principe

Le principe est présenté figure 4.10.

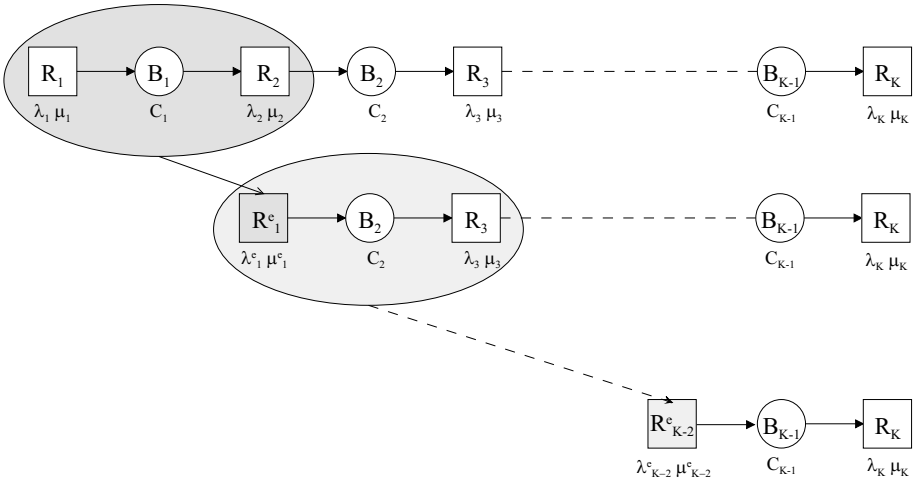


FIG. 4.10 – Principe de la méthode d'agrégation

On agrège les deux premiers routeurs $R_1$, $R_2$ et le premier buffer $B_1$ du réseau en un unique routeur équivalent $R_1^e$ qui aura le même comportement que le système simple constitué par $R_1$,$R_2$ et $B_1$. Nous devons donc déterminer les caractéristiques suivantes pour $R_1^e$ :

$1/\lambda_1^e$ : Temps moyen de disponibilité du routeur $R_1^e$
$1/\mu_1^e$ : Temps moyen d'indisponibilité du routeur $R_1^e$
$U_1^e$ : Vitesse de transfert du routeur $R_1^e$

Dans la suite nous présentons comment déterminer ces caractéristiques. Une fois $R_1^e$ caractérisé, nous étudions le système composé de $R_1^e$, $B_2$, $R_3$, ..., $R_K$ puisque $R_1^e$ a le même comportement que $R_1$,$R_2$ et $B_1$. Nous appliquons ensuite le même procédé sur ce nouveau système afin de déterminer $R_2^e$, le routeur équivalent à $R_1^e$,$R_3$ et $B_2$. On poursuit l'opération d'agrégation jusqu'à déterminer $R_{K-2}^e$ qui avec $B_{K-1}$ et $R_K$ compose le dernier système simple à étudier.

### Chaîne de Markov approchée du système mono-buffer

En réduisant l'étude du système complet à l'agrégation de deux routeurs consécutifs, nous réduisons la complexité du système à étudier, ce qui permet de trouver une solution de manière analytique. Dans le chapitre 3, on présente la manière d'obtenir les probabilités stationnaires de tous les états accessibles pour un système monobuffer. La figure 4.11 représente certains de ces états ainsi que les transitions entre ces états. Ces états sont divisés en deux catégories : les états de buffer vide $(a, b, 0)$ et les états de buffer non-vide $(a, b, x)$, dans les état de buffer non-vide on inclut les états frontières de buffer plein. Les états grisés sont dits « actifs », c'est à dire que le routeur aval transmet des données. Les autres états sont dits « inactifs ».

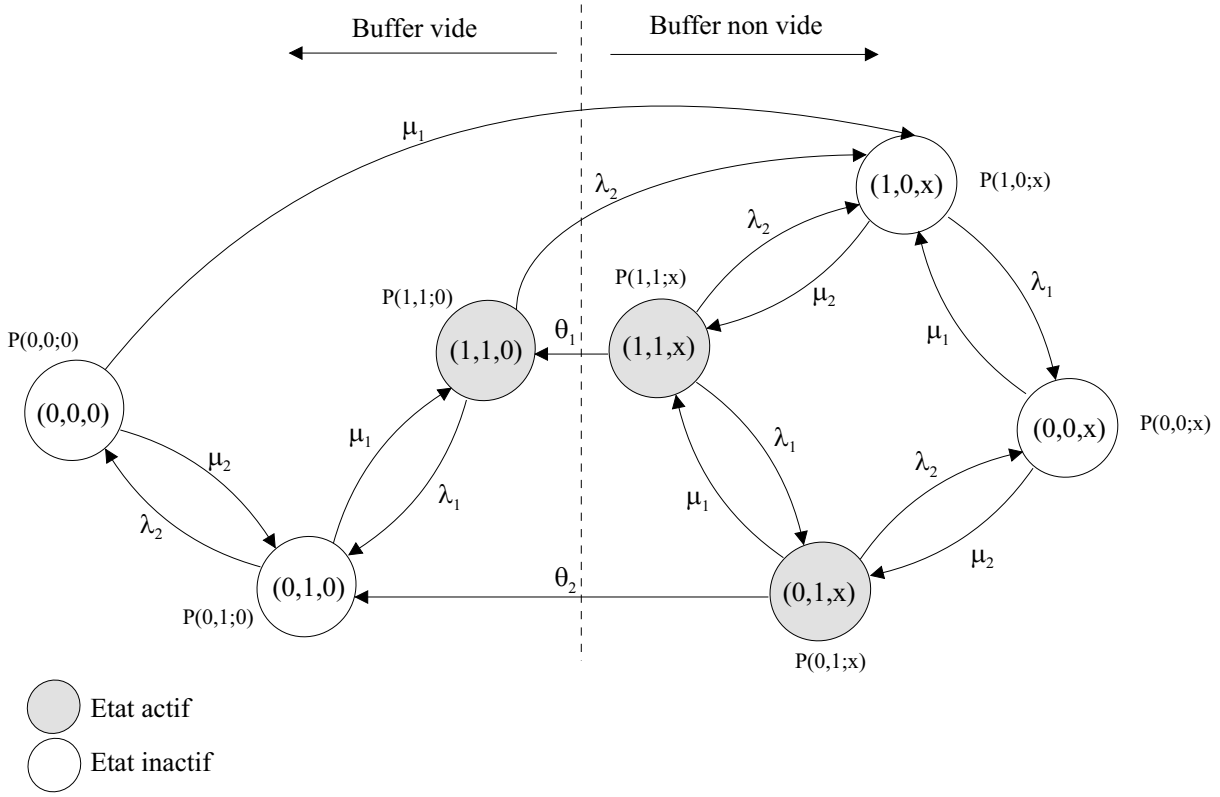


FIG. 4.11 – États du système « mono-buffer »

Cette chaîne n'est pas une chaîne de Markov exacte. Les transitions entre les états $(1,1,x)$, $(1,1,0)$ et $(0,1,x)$, $(0,1,0)$ dépendent de la vitesse de transfert de chaque routeur, donc les temps de séjour dans chaque état ne suivent pas exactement une loi de distribution exponentielle. Cependant puisque ces temps sont positifs et possèdent une moyenne finie (liée à la vitesse de transfert), on peut approcher les distributions de probabilité de ces temps par des lois exponentielles de moyenne $1/\theta_1$ et $1/\theta_2$. Nous avons donc en figure 4.11 une chaîne de Markov approchée du processus de départ :

$$M = \begin{bmatrix} -(\lambda_1+\lambda_2+\theta_1) & \lambda_2 & \lambda_1 & 0 & \theta_1 & 0 & 0 \\ \mu_2 & -(\lambda_1+\mu_2) & 0 & \lambda_1 & 0 & 0 & 0 \\ \mu_1 & 0 & -(\mu_1+\theta_2+\lambda_2) & \lambda_2 & 0 & \theta_2 & 0 \\ 0 & \mu_1 & \mu_2 & -(\mu_1+\mu_2) & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & -(\lambda_1+\lambda_2) & \lambda_1 & 0 \\ 0 & 0 & 0 & 0 & \mu_1 & -(\mu_1+\lambda_2) & \lambda_2 \\ 0 & \mu_1 & 0 & 0 & 0 & \mu_2 & -(\mu_1+\mu_2) \end{bmatrix}$$

Dans cette matrice, on a ordonné, arbitrairement, les états de la manière suivante : $(1,1,x)$, $(1,0,x)$, $(0,1,x)$, $(0,0,x)$, $(1,1,0)$, $(0,1,0)$, $(0,0,0)$. Les probabilités stationnaires de tous ces états sont connues, elles sont déterminées de la manière exposée dans le chapitre 3. $\lambda_1$, $\mu_1$, $\lambda_2$ et $\mu_2$ sont les caractéristiques connues des routeurs $R_1$ et $R_2$. Les seules inconnues de notre système sont donc $\theta_1$ et $\theta_2$. Celles-ci sont obtenues à partir des

équations stationnaires venant de notre chaîne figure 4.11.

$$(\theta_1 + \lambda_1 + \lambda_2)P(1,1;x) = \mu_2 P(1,0;x) + \mu_1 P(0,1;x)$$
$$(\theta_2 + \mu_1 + \lambda_2)P(0,1;x) = \lambda_1 P(1,1;x) + \mu_2 P(0,0;x)$$

Ces deux équations donnent les expressions de $\theta_1$ et $\theta_2$ :

$$\theta_1 = \frac{\mu_2 P(1,0;x) + \mu_1 P(0,1;x)}{P(1,1;x)} - (\lambda_1 + \lambda_2)$$
$$\theta_2 = \frac{\lambda_1 P(x;1,1) + \mu_2 P(0,0;x)}{P(0,1;x)} - (\mu_1 + \lambda_2)$$

On a ainsi une chaîne de Markov différenciant les états actifs et inactifs. Nous allons à présent réduire la taille de cet espace d'états, c'est ce qui est décrit dans la section suivante.

### Calcul du routeur équivalent

Le calcul d'un routeur équivalent va nous permettre de réduire cet espace à 7 états en un espace à 2 états seulement : le premier état sera actif et le second inactif (voir la figure 4.12.

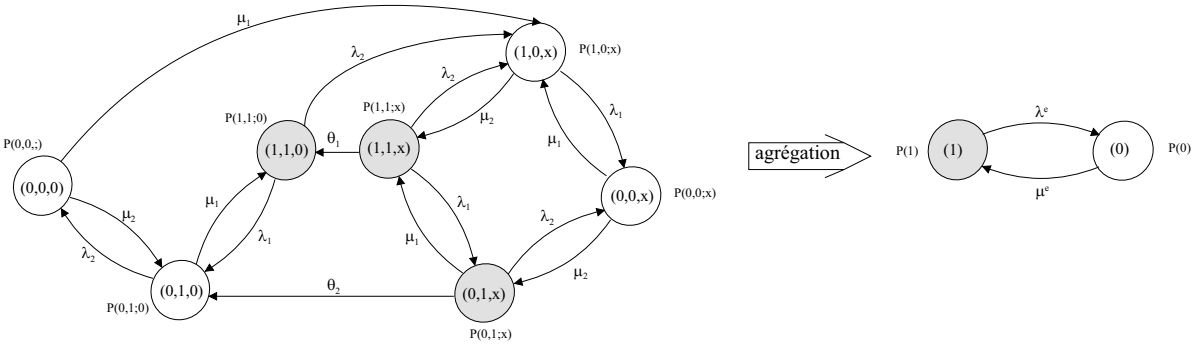


FIG. 4.12 – États actifs et inactifs dans le système constitué d'un buffer et de deux routeurs

Il nous faut donc déterminer les temps moyens de séjour dans les états actifs et dans les états inactifs. Cela revient à déterminer $\lambda_1^e$ et $\mu_1^e$. On utilise pour cela la méthode décrite dans ([RS89]). On veut regrouper l'ensemble des états actifs en un unique état actif et de même pour les états inactifs, on divise donc l'espace de 7 états en deux sous espaces :

$B$, de dimension 4, est la partie qui contient les états non-actifs.

$B^c$, de dimension 3, est la partie qui contient les états actifs.

On peut réorganiser la matrice $M$ de manière à regrouper les états non-actifs au début et les actifs à la fin. Par exemple, en ordonnant les états de cette manière : $(0,0,x)$, $(1,0,x)$, $(0,0,0)$, $(0,1,0)$, $(0,1,x)$, $(1,1,0)$, $(1,1,x)$, on obtient alors la matrice $M^{part}$ que l'on peut partitionner de cette manière :

$$M^{part} = \begin{bmatrix} M^B & M^{BB^c} \\ M^{B^cB} & M^{B^c} \end{bmatrix} \tag{4.23}$$

$$= \left[\begin{array}{cccc|ccc} -(\mu_1+\mu_2) & \mu_1 & 0 & 0 & \mu_2 & 0 & 0 \\ \lambda_1 & -(\lambda_1+\mu_2) & 0 & 0 & 0 & 0 & \mu_2 \\ 0 & \mu_1 & -(\mu_1+\mu_2+) & \mu_2 & 0 & 0 & 0 \\ 0 & 0 & \lambda_2 & -(\mu_1+\lambda_2) & 0 & \mu_1 & 0 \\ \hline \lambda_2 & 0 & 0 & \theta_2 & -(\mu_1+\lambda_2+\theta_2) & 0 & \mu_1 \\ 0 & \lambda_2 & 0 & \lambda_1 & 0 & -(\lambda_1+\lambda_2) & 0 \\ 0 & \lambda_2 & 0 & 0 & \lambda_1 & \theta_1 & -(\theta_1+\lambda_1+\lambda_2) \end{array}\right]$$

On réorganise en conséquence le vecteur probabilités stationnaires $\vec{\pi}$ afin d'obtenir $\vec{\pi}^{part}$.

$$\vec{\pi}^{part} = \left[\begin{array}{cccc|ccc} \pi(0,0;x) & \pi(1,0;x) & \pi(0,0;0) & \pi(1,0;0) & \pi(0,1;x) & \pi(1,1;0) & \pi(1,1;x) \end{array}\right]$$

Soit $\Gamma$ la matrice diagonale contenant les même éléments diagonaux que $-M^{part}$. On appelle $P^{part}$ la matrice définie par $N^{part} = I + \Gamma M^{part}$. Soit $z_1^{part}$ la distribution stationnaire associée ($z^{part}N^{part} = z^{part}$). Il est à noter que cette transformation est en fait la discrétisation du système continu ($M^{part}$ et $\vec{\pi}^{part}$ en continu deviennent en discret $N^{part}$ et $\vec{z}^{part}$). $\vec{z^{part}}$ peut se calculer, mais se trouve plus rapidement à partir de $\vec{\pi}^{part}$, car $\vec{z^{part}} = \frac{\vec{\pi}^{part}\Gamma}{\vec{\pi}^{part}\Gamma\vec{1}^T}$. On a donc finalement pour notre système discrétisé :

$$N^{part} = \left[\begin{array}{cc} N^B & N^{BB^c} \\ N^{B^cB} & N^{B^c} \end{array}\right]$$

$$= \left[\begin{array}{cccc|ccc} 0 & \frac{\mu_1}{\mu_1+\mu_2} & 0 & 0 & \frac{\mu_2}{\mu_1+\mu_2} & 0 & 0 \\ \frac{\lambda_1}{\lambda_1+\mu_2} & 0 & 0 & 0 & 0 & 0 & \frac{\mu_2}{\lambda_1+\mu_2} \\ 0 & \frac{\mu_1}{\lambda_1+\mu_2} & 0 & \frac{\mu_2}{\lambda_1+\mu_2} & 0 & 0 & 0 \\ 0 & 0 & \frac{\lambda_2}{\mu_1+\lambda_2} & 0 & 0 & \frac{\mu_1}{\mu_1+\lambda_2} & 0 \\ \hline \frac{\lambda_2}{\mu_1+\lambda_2+\theta_2} & 0 & 0 & \frac{\theta_2}{\mu_1+\lambda_2+\theta_2} & 0 & 0 & \frac{\mu_1}{\mu_1+\lambda_2+\theta_2} \\ 0 & \frac{\lambda_2}{\lambda_1+\lambda_2} & 0 & \frac{\lambda_1}{\lambda_1+\lambda_2} & 0 & 0 & 0 \\ 0 & \frac{\lambda_2}{\theta_1+\lambda_1+\lambda_2} & 0 & 0 & \frac{\lambda_1}{\theta_1+\lambda_1+\lambda_2} & \frac{\theta_1}{\theta_1+\lambda_1+\lambda_2} & 0 \end{array}\right]$$

$$\vec{z}^{part} = \left[\begin{array}{cc} \vec{z}^B & \vec{z}^{B^c} \end{array}\right]$$

Lorsque $M^{part}$, $N^{part}$ et $z^{part}$ sont déterminés , on trouve les temps moyens $T_{on} = 1/\lambda_1^e$ et $T_{off} = 1/\mu_1^e$ grâce aux équations suivantes qui proviennent de [RS89] :

$$1/\lambda_1^e = -v^B(M^B)^{-1}\vec{1}^T \text{ où } v^B = \frac{z^{B^c}N^{B^cB}}{z^{B^c}N^{B^cB}\vec{1}^T}$$

$$1/\mu_1^e = -v^{B^c}(M^{B^c})^{-1}\vec{1}^T \text{ où } v^{B^c} = \frac{z^{B}N^{BB^c}}{z^{B}N^{BB^c}\vec{1}^T}$$

La dernière caractéristique inconnue du routeur $R_1^e$ est sa vitesse de transfert $U_1^e$ qui peut être trouvée en utilisant la propriété de conservation du débit. Le routeur équivalent $R_1^e$ doit avoir, en moyenne, le même débit que le routeur $R_2$. Lorsque $B_1$ est non vide, la vitesse de transfert de $R_2$ est $U_2$ sinon, si $B_1$ est vide, $R_2$ transfert des données à la vitesse $U_1$ qui est inférieure à $U_2$ (c'est une condition évidente pour être dans l'état $(1,1,0)$, si $U_1$ est supérieure à $U_2$, alors $P(1,1;0)$ est nulle car le buffer ne peut rester vide).

$$U_1^e = \frac{\mu_1^e + \lambda_1^e}{\mu_1^e}(U_2(P(0,1;x) + P(1,1;x)) + U_1P(1,1;0))$$

Finalement, on peut remplacer le système mono-buffer par un simple routeur source équivalent au début du réseau comme décrit sur la figure 4.10.

### 4.2.3 Résultats numériques

Dans cette section on compare les résultats obtenus par notre méthode d'agrégation à ceux de la simulation sur plusieurs exemples.

#### Exemple 1

D'abord, nous devons fixer les paramètres du réseau étudié : temps moyens de disponibilité et d'indisponibilité, vitesse de transfert et capacité du buffer pour chaque routeur.

Nous allons d'abord tester notre algorithme sur une ligne de routeurs identiques avec $\lambda_i = 1$, $\mu_i = 1$ et $U_i = 1$ pour $i$ de 1 à 5, et $C_i = 5$ pour $i$ de 1 à 4 (figure 4.13 et tableaux 4.14, 4.15 et 4.16).

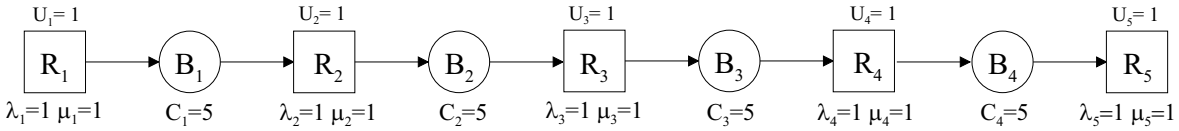


FIG. 4.13 – Routeurs identiques

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 0,5000 | 0,4992 | 0,16 |
| Routeur 2 | 0,4583 | 0,4575 | 0,17 |
| Routeur 3 | 0,4322 | 0,4360 | 0,88 |
| Routeur 4 | 0,4142 | 0,4218 | 1,83 |
| Routeur 5 | 0,4009 | 0,4111 | 2,54 |

TAB. 4.14 – Débits pour des routeurs identiques

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 2,5000 | 2,4929 | 0,28 |
| Buffer 2 | 2,0318 | 1,9993 | 1,63 |
| Buffer 3 | 1,7589 | 1,7392 | 1,13 |
| Buffer 4 | 1,5756 | 1,5872 | 0,74 |

TAB. 4.15 – Niveaux des buffers pour des routeurs identiques

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 0,0417 | 0,0417 | 0,00 |
| Routeur 2 -> Routeur 3 | 0,0261 | 0,0215 | 21,40 |
| Routeur 3 -> Routeur 4 | 0,0180 | 0,0141 | 27,66 |
| Routeur 4 -> Routeur 5 | 0,0133 | 0,0108 | 23,15 |

TAB. 4.16 – Débits moyens de pertes pour des routeurs identiques

**Exemple 2**

Ensuite nous allons étudier un réseau de routeurs moins disponibles $\lambda_i = 10$, $\mu_i = 1$ et $U_i = 1$ pour $i$ de 1 à 5, et$C_i = 5$ pour $i$ de 1 à 4 (figure 4.14 et tableaux 4.17, 4.18 et 4.19).

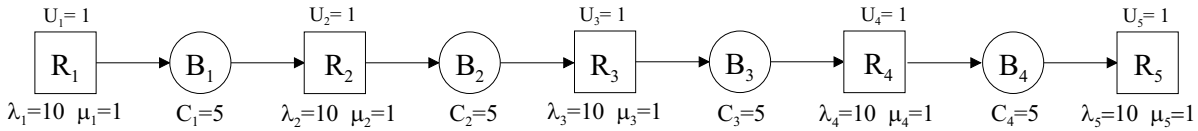


FIG. 4.14 – Routeurs peu disponibles

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 0,0909 | 0,0910 | 0,11 |
| Routeur 2 | 0,0880 | 0,0881 | 0,11 |
| Routeur 3 | 0,0861 | 0,0866 | 0,58 |
| Routeur 4 | 0,0847 | 0,0856 | 1,06 |
| Routeur 5 | 0,0837 | 0,0849 | 1,43 |

TAB. 4.17 – Débits pour des routeurs peu disponibles

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 2,5000 | 2,5014 | 0,06 |
| Buffer 2 | 2,0984 | 2,0814 | 0,82 |
| Buffer 3 | 1,8640 | 1,8368 | 1,48 |
| Buffer 4 | 1,6989 | 1,6837 | 0,90 |

TAB. 4.18 – Niveaux des buffers pour des routeurs peu disponibles

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 0,0029 | 0,0029 | 0,00 |
| Routeur 2 -> Routeur 3 | 0,0019 | 0,0015 | 26,67 |
| Routeur 3 -> Routeur 4 | 0,0014 | 0,0010 | 40,00 |
| Routeur 4 -> Routeur 5 | 0,0010 | 0,0007 | 42,86 |

TAB. 4.19 – Débits moyens de pertes pour des routeurs peu disponibles

**Exemple 3**

Finalement nous étudions le cas où les routeurs sont autant disponibles qu'indisponibles mais possèdent des vitesses de transfert différentes, $\lambda_i = 1$, $\mu_i = 1$ pour $i$ de 1 à 5, et $C_i = 20$ pout $i$ de 1 à 4.Ensuite $U_1 = 10$, $U_2 = 50$, $U_3 = 100$, $U_4 = 200$ et $U_5 = 200$ (figure 4.15 et tableaux 4.20, 4.21 et 4.22).

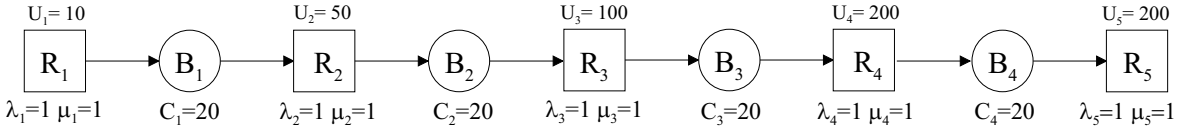


FIG. 4.15 – Routeurs de vitesses différentes

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 | 5,0000 | 5,0041 | 0,08 |
| Routeur 2 | 4,8389 | 4,8424 | 0,07 |
| Routeur 3 | 4,5971 | 4,6482 | 1,11 |
| Routeur 4 | 4,2972 | 4,4460 | 3,46 |
| Routeur 5 | 3,9573 | 4,2453 | 7,28 |

TAB. 4.20 – Débits avec des routeurs de vitesses différentes

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Buffer 1 | 2,9935 | 2,9901 | 0,11 |
| Buffer 2 | 2,4889 | 2,5454 | 2,27 |
| Buffer 3 | 2,1428 | 2,2801 | 6,41 |
| Buffer 4 | 1,9456 | 2,1775 | 11,92 |

TAB. 4.21 – Niveaux des buffers avec des routeurs de vitesses différentes

| | Méthode analytique | Simulation | Erreur relative (%) |
|---|---|---|---|
| Routeur 1 -> Routeur 2 | 0,1611 | 0,1617 | 0,37 |
| Routeur 2 -> Routeur 3 | 0,2418 | 0,1942 | 24,51 |
| Routeur 3 -> Routeur 4 | 0,2999 | 0,2022 | 48,32 |
| Routeur 4 -> Routeur 5 | 0,3399 | 0,2008 | 69,27 |

TAB. 4.22 – Débits moyens de pertes pour des routeurs identiques

**Analyse des résultats**

Cette méthode d'agrégation permet d'évaluer les performances de réseaux linéaires. La comparaison des résultats analytiques aux résultats de la simulation donne de bons renseignements sur la fiabilité de notre méthode. Par rapport à la méthode de décomposition développée en première partie de ce chapitre, elle est moins précise, mais permet

d'étudier des réseaux de routeurs non homogènes en vitesse de transmission. En ce qui concerne les débits, on peut remarquer que les résultats sont très proches. Il en est de même pour le niveau des buffers, performance pourtant plus difficile à évaluer. Pour les débits moyens de pertes, les écarts relatifs sont plus importants.

Contrairement aux simulateurs, le temps mis avec la méthode analytique pour obtenir ces résultats est très court, comme on peut le constater d'après le tableau 4.23 indiquant les temps d'obtention des résultats par le simulation et par la méthode analytique (sur un Pentium 4 à 2GHz).

| | Méthode analytique | Simulation |
|---|---|---|
| Exemple 1 | $\sim$ 1s | 45s |
| Exemple 2 | $\sim$ 1s | 35s |
| Exemple 3 | $\sim$ 1s | 50s |

TAB. 4.23 – Temps d'obtention des résultats

### Conclusion

Dans ce chapitre, nous avons étudié des réseaux simples : les réseaux à topologie linéaire. Nous avons évalué les performances de réseaux linéaires homogènes grâce aux résultats du chapitre 3 et à une méthode de décomposition. Ensuite l'étude de réseaux linéaires non homogènes nous a conduit à développer un autre méthode que la décomposition, car cette dernière n'était plus adaptée pour ce genre de réseaux.

Dans le chapitre 5 nous traitons le cas de réseaux plus complexes. La méthode que nous employons est similaire à celle développée pour les réseaux linéaires non homogènes, mais on l'élargit pour permettre de prendre en compte des convergences au niveau des nœuds du réseau.

# Chapitre 5

# ÉVALUATION DE PERFORMANCES D'UN RÉSEAU COMPLEXE

Dans le chapitre précédent, nous nous sommes intéressés aux réseaux linéaires, c'est à dire à une succession de routeurs sans nœuds de convergence. Ce type de modèle permet d'évaluer les performances de l'épine dorsale d'un réseau qui, lui, comporte certainement des nœuds de convergence. Dans ce chapitre, nous allons traiter du cas plus complexe des réseaux comportant des convergences, dans ce cas les buffers des routeurs sont partagés par plusieurs flux. Nous allons d'abord présenter le cas mono-buffer multi-sources que nous utiliserons ensuite afin d'évaluer les performances de réseaux complexes.

## 5.1 Présentation du problème

La méthode d'évaluation de performances que nous allons présenter peut être appliquée sur les réseaux comportant des nœuds de convergence comme représenté sur la figure 5.1. Précisons que ce réseau est un exemple de ce qu'il peut résulter comme connexions entre les routeurs une fois les chemins virtuels établis. On peut remarquer qu'il est possible d'avoir d'autres nœuds dans les parties du réseau convergeant vers l'épine dorsale. Dans l'algorithme présenté dans la suite, on a supposé que les convergences étaient composées d'un unique routeur source, mais ce n'est pas une restriction, puisqu'en appliquant l'algorithme sur chaque branche convergente, on peut la réduire à un routeur source équivalent.

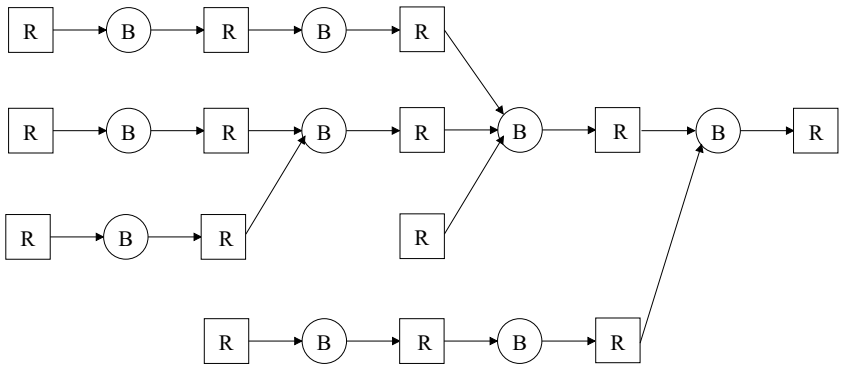


FIG. 5.1 – Topologie générale d'un réseau comportant des nœuds de convergence

## 5.2 Cas mono-buffer multi-sources

### 5.2.1 Présentation du système

Tout d'abord, nous allons considérer le réseau simple composé de routeurs sources $R_{11}$, ..., $R_{1N_1}$, du buffer $B_1$ et du routeur $R_{21}$ (Figure 5.2). Lorsqu'une source $R_{1j}$ est ON, elle transmet des données à la vitesse constante $U_{1j}$ vers $B_1$. Ensuite, quand $R_{21}$ est ON et si $B_1$ est non-vide, les données sont vidées de $B_1$ à la vitesse $U_{21}$.

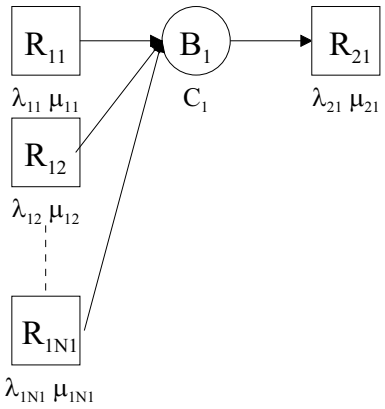


FIG. 5.2 – Cas mono-buffer multi-sources

Dans les sections précédentes, on a caractérisé les différents états du système que l'on a nommé dipole par un triplet $(a, b, x)$. Dans le cas qui nous intéresse à présent, nous avons $N_1$ routeurs sources, un buffer, et un routeur destinataire. Donc, sur le même principe, nous utilisons un $(N_1 + 2)$-uplet $(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}, x_1)$ pour caractériser les différents états du système.

- $\alpha_{11}, \ldots, \alpha_{1N_1}$ et $\alpha_{21}$ sont les états des routeurs $R_{11}, \ldots, R_{1N_1}$ et $R_{21}$ (ON ou OFF).
- $x_1$ est le niveau du buffer $B_1$ ($x_1 \in [0, C_1[$).

Afin d'évaluer les performances d'un tel système (débit moyen de $R_{21}$, niveau moyen de $B_1$ et pertes), nous allons procéder de la même manière que pour le cas mono-buffer

traité dans la section 3.3. Nous allons déterminer les probabilités stationnaires de chaque état.

### 5.2.2 Détermination des probabilités stationnaires des états du système

A chaque routeur on associe un générateur markovien $M_{ij}$ et son vecteur $\vec{v}_{ij}$ contenant sa vitesse de transmission ($U_{ij}$ lorsqu'il transfert des données et 0 sinon) comme présenté dans [Moc03] :

$$M_{ij} = \begin{bmatrix} -\lambda_{ij} & \lambda_{ij} \\ \mu_{ij} & -\mu_{ij} \end{bmatrix}, \vec{v}_{ij} = (U_{ij}, 0)$$

Avec ces notations, le générateur markovien $M_1$ du système composé de $R_{11}$, ..., $R_{1N_1}$, $B_1$ et $R_{21}$ est une matrice carrée de dimension $2^{N_1+1}$ donc :

$$M_1 = M_{11} \oplus M_{12} \oplus \ldots \oplus M_{1N_1} \oplus M_{21}$$

Le vecteur $\vec{d_1}$ qui contient les $2^{N_1+1}$ vitesses de transfert des données dans $B_1$ est obtenu de la même manière en tenant compte du signe. Cependant dans la somme de Kronecker on prendra soin de modifier le signe des vecteurs vitesses de chaque routeur : positif si le routeur remplit le buffer aval et négatif s'il le vide. Ici les routeurs $R_{11}$, ..., $R_{1N_1}$ remplissent le buffer (signe positif $+\vec{v}_{1i}$) et le routeur $R_{21}$ le vide (signe négatif $-\vec{v}_{21}$).

$$\vec{d_1} = \vec{v}_{11} \oplus \vec{v}_{12} \oplus \ldots \oplus \vec{v}_{1N_1} \oplus (-\vec{v}_{21})$$

L'espace d'état discret ainsi engendré est de dimension $2^{N_1+1}$. Les probabilités $p_1(t; \alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$ de chaque état ($\alpha_{11}$, ..., $\alpha_{1N_1}$, $\alpha_{21}$) sont définies de cette manière :

$$p_1(t; \alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}) = Pr[\text{L'état des routeurs au temps } t \text{ est } (\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})]$$

On note $\vec{p_1}(t)$ le vecteur probabilité à l'instant $t$. Il contient les probabilités des $2^{N_1+1}$ états $(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$.

Et c'est l'équation suivante qui régit l'évolution de cet espace d'état discret :

$$\frac{d}{dt}\vec{p_1}(t) = \vec{p_1}(t)M_1$$

Le vecteur stationnaire correspondant $\vec{w}_1$ contient les probabilités stationnaires de chaque état $(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$.

$$\vec{w}_1 = \begin{bmatrix} w_1(1, \ldots, 1) & \ldots & w_1(0, \ldots, 0) \end{bmatrix}$$

En introduisant le niveau du buffer $B_1$ dans la représentation de notre système, nous créons un espace d'état hybride. Les probabilités de ces états sont :

$$P_1 \quad (t; \alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}; x_1) =$$
$$Pr[\text{niveau du buffer} \leq \quad x_1 \quad \text{et l'état des routeurs au temps } t \text{ est } (\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$$

On note $\vec{P_1}(t, x_1)$ le vecteur probabilité à l'instant $t$. Il contient les probabilités $P_1(t; \alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}; x_1)$.

L'équation d'évolution de l'espace d'état hybride est la même que (3.16) déterminée dans le chapitre sur la résolution analytique du cas mono-buffer :

$$\frac{\partial}{\partial t}\vec{P_1}(t, x_1) + \frac{\partial}{\partial x_1}\vec{P_1}(t, x_1)D_1 = \vec{P_1}(t, x_1)M_1$$

La matrice drift $D_1$ contient les éléments de $\vec{d_1}$ sur sa diagonale. Comme nous l'avons déjà fait remarquer dans le cas mono-buffer mono-source, le vecteur probabilité stationnaire $\vec{P_1}(x) = \lim_{t\to\infty} \vec{P_1}(t, x_1)$ possède une partie continue $\vec{\pi}_1(x_1)$ et des discontinuités aux frontières (états où le buffer est plein ou vide). L'équation précédente devient alors, pour la partie continue de $\vec{P_1}(x_1)$, en régime stationnaire :

$$\frac{d}{dx_1}\vec{\pi}_1(x_1)D_1 = \vec{\pi}_1(x_1)M_1$$

Cette équation peut être résolue analytiquement afin d'obtenir $\vec{\pi}_1(x_1)$. $\vec{P_1}(x_1)$ est complètement caractérisé avec $\vec{\pi}_1(x_1)$ et les probabilités stationnaires des états frontières. La détermination des probabilités des états frontières a été présentée dans la chapitre mono-buffer où l'on obtenait les relations suivantes :

. Pr[buffer vide et routeurs dans l'état $(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$] = $\pi_1(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}; 0)$

. Pr[buffer plein et routeurs dans l'état $(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$]
= Pr[routeurs dans l'état $(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21})$] - $\pi_1(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}; C_1)$
= $w_1(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}) - \pi_1(\alpha_{11}, \ldots, \alpha_{1N_1}, \alpha_{21}; C_1)$

### 5.2.3 Transitions états frontières - états internes

Les états du système mono-buffer multi-sources peuvent êtres regroupés en deux catégories : les états actifs (ceux où $R_{21}$ transmet des données) et les états inactifs (ceux où $R_{21}$ ne transmet pas de données). Dans ces catégories, on trouve des états frontières et internes. Le passage d'un état interne vers un état frontière peut donc par exemple signifier qu'on passe d'un état actif à un état inactif. Il est alors nécessaire de connaître les taux de transition entre ces deux types d'états. Dans la section 4.2.2 qui traitait du cas mono-buffer non homogène dans les réseaux linéaires, la figure 4.11 représentait tous les états et les transitions entre eux. Ainsi, à l'aide de cette chaîne de Markov approchée, on avait pu déterminer l'équation nous donnant les transitions manquantes entre les états internes et frontières($\theta_1$ et $\theta_2$). Dans le cas multi-sources, il n'est pas aisé de représenter une telle chaîne de Markov approchée, à cause du grand nombre d'états à envisager. Lorsqu'on s'intéresse au débit, les états frontières qui nous intéressent sont les états où le buffer est vide. En effet, lorsque le buffer est plein, le débit en sortie du routeur aval est le même que lorsque le buffer est non-vide (c'est le débit maximum du routeur aval), on ne distinguera donc pas les états de buffer plein de ceux où il est non-vide. En revanche, lorsque le buffer est vide, on peut se trouver dans deux cas de figure : soit les routeurs en amont du buffer sont OFF et donc le débit en sortie du routeur aval est nul. Soit au moins un des routeurs en amont transmet des données et le débit en sortie du routeurs aval est

non-nul si celui-ci est ON. Nous allons présenter un algorithme permettant d'obtenir ces transitions.

Les éléments de $M_1$ et $\vec{d_1}$ peuvent être triés afin d'obtenir $M_1^{tri}$ et $\vec{d_1}^{tri}$ dans lesquels sont regroupés les états où la vitesse de transfert est positive (le buffer $B_1$ se remplit), nulle ou négative (le buffer $B_1$ se vide). Le vecteur $\vec{d_1}$ est donc trié et partitionné de manière à obtenir $\vec{d_1}^{tri}$ :

$$\vec{d_1}^{tri} = \left[ \begin{array}{ccc} \vec{d_1^+} & \vec{d_1^0} & \vec{d_1^-} \end{array} \right]$$

La matrice $M_1$ est réorganisée en conséquence, afin qu'elle regroupe les états de drift positif, nul ou négatif dans le même ordre que $\vec{d_1}$, on la note alors $M_1^{tri}$.

$$M_1^{tri} = \left[ \begin{array}{ccc} M_1^{++} & M_1^{+0} & M_1^{+-} \\ M_1^{0+} & M_1^{00} & M_1^{0-} \\ M_1^{-+} & M_1^{-0} & M_1^{--} \end{array} \right]$$

On peut construire le générateur $M_1^d$ qui différencie les états de buffer vide et non-vide :

$$M_1^d = \left[ \begin{array}{ccccc} M_1^{++} & M_1^{+0} & M_1^{+-} & 0 & 0 \\ M_1^{0+} & M_1^{00} & M_1^{0-} & 0 & 0 \\ M_1^{-+} & M_1^{-0} & M_1^{--} - \Theta_1 & 0 & \Theta_1 \\ M_1^{0+} & 0 & 0 & M_1^{00} & M_1^{0-} \\ M_1^{-+} & 0 & 0 & M_1^{-0} & M_1^{--} \end{array} \right] \tag{5.1}$$

La matrice $\Theta_1$ contient les transitions entre les états de buffer non-vide et les états de buffer vide. La relation qui nous permet de déterminer $\Theta_1$ s'obtient en écrivant l'équation d'équilibre de l'état « buffer vide et drift négatif », c'est la dernière colonne de la matrice $M_1^d$ : à l'équation stationnaire suivante :

$$(\vec{P_1^-}(C_1) - \vec{P_1^-}(0))\Theta_1 + \vec{P_1^0}(0)M_1^{0-} + \vec{P_1^-}(0)M_1^{--} = 0 \tag{5.2}$$

avec :

- $\vec{P_1^-}(C_1)$ contient les probabilités stationnaires des états où la vitesse de remplissage du buffer est négative et le niveau du buffer est $\leq C_1$, c'est à dire buffer non-vide.
- $\vec{P_1^-}(0)$ contient les probabilités stationnaires des états où la vitesse de remplissage du buffer est négative et le buffer est vide.
- $\vec{P_1^0}(0)$ contient les probabilités stationnaires des états où la vitesse de remplissage du buffer nulle et le buffer est vide.

La matrice $M_1^d$ caractérise donc le système composé de $R_{11}$, ..., $R_{1N_1}$, $B_1$ et $R_{21}$. De plus dans cette représentation, on a différencié les états de buffer vide et les états de buffer non-vide. A ces états correspondent des probabilités stationnaires contenues dans le vecteur $\vec{P_1^d}$ et des « drifts » $\vec{d_1^d}$ associés, on peut construire ces vecteurs à partir de $\vec{P_1}^{tri}(0)$, $\vec{w_1}^{tri}$ et $\vec{d_1}^{tri}$.

$$\vec{P_1^d} = \left[ \begin{array}{ccc} \vec{w}^{tri} - \vec{P}^{tri}(0) & \vec{P^0}(0) & \vec{P^-}(0) \end{array} \right] \tag{5.3}$$

$$\vec{d_1^d} = \left[ \begin{array}{cc} \vec{d}^{tri} & \vec{0} \end{array} \right] \tag{5.4}$$

avec $\vec{0}$ le vecteur nul de dimension le nombre d'états dont la vitesse de transfert est nulle ou négative. En effet, dans $\vec{d}_1^d$, $\vec{0}$ représente la vitesse de variation du niveau du buffer (drift) dans les états de buffer vide. Cette vitesse est donc forcément nulle, puisque si elle est positive, alors on sort de l'état dans lequel le buffer est vide et elle ne peut être négative puisque le buffer est vide (niveau du buffer non-négatif).

Le sous-système ainsi modélisé sera utilisé dans la méthode d'agrégation afin de déterminer les caractéristiques du réseau.

## 5.3 Méthode d'agrégation

Les résultats précédents nous permettent d'évaluer de manière analytique les performances d'un système mono-buffer multi-sources. Lorsque le système est trop grand, du type présenté figure 5.3, le problème devient difficile à résoudre analytiquement, nous allons donc réduire le réseau complet en une série de réseaux mono-buffer multi-sources. Dans la section suivante nous présentons la méthode d'agrégation employée dérivée de [RS89].

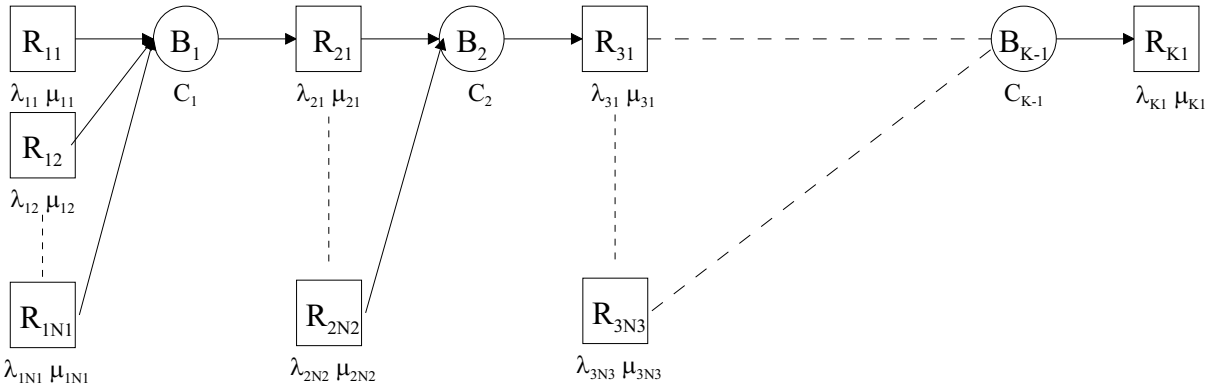


FIG. 5.3 – Topologie d'un réseau complexe

### 5.3.1 Principe de l'agrégation

La figure 5.4, représente le principe de l'agrégation.

La méthode consiste à s'intéresser successivement aux buffers des routeurs. On étudie leur comportement, et en particulier leur niveau de remplissage. Dans la figure 5.4, le premier buffer auquel on s'intéresse est le buffer du routeur $R_{21}$, c'est-à-dire $B_1$. Les routeurs $R_{11}$, ..., $R_{1N_1}$ sont les sources, quant à $R_{21}$ il joue le rôle du destinataire. La partie aval de $R_{21}$ n'a pas d'influence sur son comportement : un buffer plein ne modifie pas la vitesse de transmission des routeurs amont puisque les données supplémentaires sont perdues. Le système à considérer est donc l'ensemble constitué de $R_{11}$, ..., $R_{1N_1}$, de $B_1$ et de $R_{21}$. C'est un système mono-buffer multi-sources du même type que celui que nous avons traité dans la section précédente. Nous sommes donc en mesure de déterminer les performances de ce système. On s'intéresse ensuite au buffer $B_2$ du routeur $R_{31}$, pour les mêmes raisons que précédemment, $R_{31}$ est un routeur destinataire. Les routeurs $R_{22}$,

..., $R_{2N_2}$ sont des routeurs sources (lorsqu'ils sont ON ils transmettent des données), mais $R_{21}$ ne peut pas être considéré comme une routeur source. Son comportement dépend notamment du niveau de remplissage des buffers amont. La méthode d'agrégation présentée dans la suite consiste donc à synthétiser un routeur source $R_1^e$ équivalent à la partie du réseau amont à $B_2$. On se rapporte alors à un cas mono-buffer multi-sources bien connu. On renouvelle l'opération de manière itérative pour les $K-1$ buffers afin de déterminer les performances au niveau de chaque routeur.

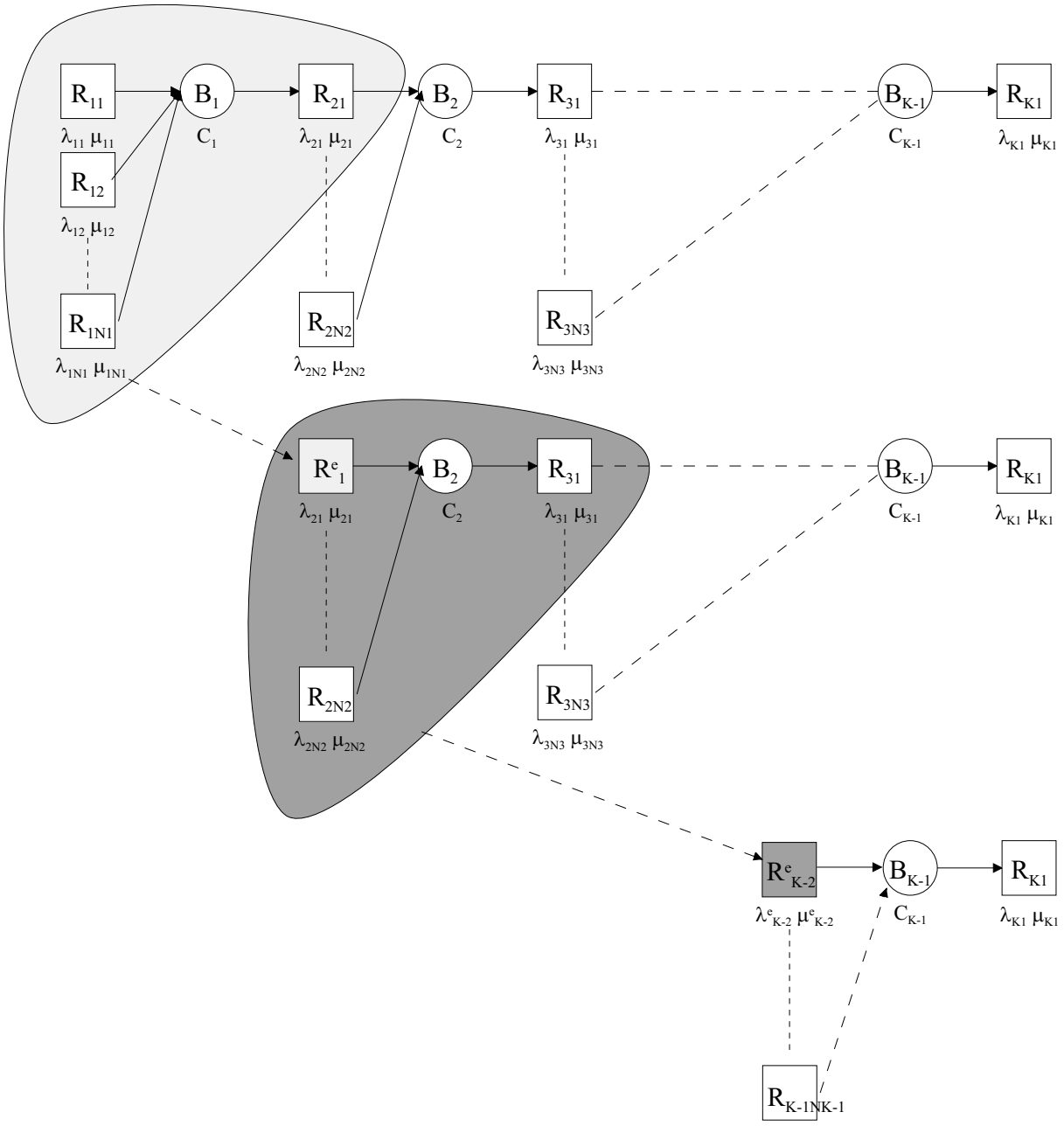


FIG. 5.4 – Principe de l'agrégation

Pour caractériser totalement le routeur équivalent $R_1^e$, il faut déterminer :

- $1/\lambda_1^e$ : Temps moyen de disponibilité du routeur équivalent $R_1^e$
- $1/\mu_1^e$ : Temps moyen d'indisponibilité du routeur équivalent $R_1^e$
- $U_1^e$ : Vitesse de transfert de $R_1^e$

Dans la section suivante, nous expliquons comment obtenir ces caractéristiques par agrégation.

### 5.3.2 Routeur équivalent

Nous allons tout d'abord présenter le manière d'obtenir les temps moyens de disponibilité ($1/\lambda_1^e$) et d'indisponibilité ($1/\mu_1^e$) en utilisant une méthode développée dans [RS89].

On considère le sous-système composé de $R_{11}$, ..., $R_{1N_1}$, $B_1$ et $R_{21}$. Le générateur markovien de ce système est $M_1^d$ comme expliqué dans la section précédente. Cette matrice contient les transitions entre les états du système considéré. Ces états sont caractérisés par la situation ON ou OFF de chaque routeur, le signe du drift et le fait que le buffer est vide ou non. On appelle $N_1$ la matrice définie par $N_1 = I + \Gamma_1^{-1} M_1^d$ (voir chapitre précédent sur cette méthode de détermination des temps moyens de disponibilité et d'indisponibilité). C'est la matrice qui contient les transitions correspondant à la chaîne de Markov discrétisée. $\vec{z}_1$ est son vecteur de distributions stationnaires. Soit $E_1$ l'espace d'état du sous-système considéré. $E_1$ peut être divisé en deux parties complémentaires $B$ et $B^c$.

$B$ est la partie qui contient les états non-actifs.

$B^c$ est la partie qui contient les états actifs.

Donc les matrices $M_1^d$ et $N_1$ peuvent être partitionées en quatre sous-matrices et $\vec{z}_1$ en deux sous-vecteurs :

$$M_1^d = \begin{bmatrix} M_1^B & M_1^{BB^c} \\ M_1^{B^cB} & M_1^{B^c} \end{bmatrix}, N_1 = \begin{bmatrix} N_1^B & N_1^{BB^c} \\ N_1^{B^cB} & N_1^{B^c} \end{bmatrix}, \vec{z}_1 = \begin{bmatrix} \vec{z}_1^B \vec{z}_1^{B^c} \end{bmatrix}.$$

Les états actifs et non-actifs ne sont pas triés correctement dans $M_1^d$, $N_1$ et $\vec{z}_1$. Nous devons les trier afin de regrouper les états actifs et les non-actifs. On obtiendra alors $M_1^{part}$, $N_1^{part}$ et $z_1^{part}$ On peut déterminer tous les états actifs à partir de ces critères :

- Le routeur $R_{21}$ doit être ON.
- Le buffer $B_1$ doit être non-vide, ou s'il l'est, au moins un des routeurs de la partie amont $(R_{11}, \ldots, R_{1N})$ doit être ON.

Une fois $M_1^{part}$, $N_1^{part}$ et $\vec{z}_1^{part}$ déterminés, on trouve les temps moyens $1/\lambda_1^e$ et $1/\mu_1^e$ grâce aux équations de [RS89] :

$$\begin{aligned} 1/\lambda_1^e &= -v_1^B (M_1^B)^{-1} \vec{1}^T \text{ où } v_1^B = \frac{z_1^{B^c} N_1^{B^cB}}{z_1^{B^c} N_1^{B^cB} \vec{1}^T} \\ 1/\mu_1^e &= -v_1^{B^c} (M_1^{B^c})^{-1} \vec{1}^T \text{ où } v_1^{B^c} = \frac{z_1^B N_1^{BB^c}}{z_1^B N_1^{BB^c} \vec{1}^T} \end{aligned} \tag{5.5}$$

Nous connaissons donc à présent $\lambda_1^e$ et $\mu_1^e$ qui caractérisent le routeur équivalent $R_1^e$. Le dernier paramètre à determiner est $U_1^e$. Il peut être trouvé en utilisant le principe de conservation du débit. Le routeur équivalent $R_1^e$ doit avoir, en moyenne, le même débit

que le routeur $R_2$. Quand $B_1$ est non-vide, la vitesse de transfert de $R_{21}$ est $U_{21}$, sinon si $B_1$ est vide, $R_{21}$ transfère des données à la même vitesse qu'elles arrivent dans $B_1$.

$$\frac{\lambda_1^e}{\lambda_1^e + \mu_1^e} U_1^e = \sum_{B^c_{non-vide}} P_1 U_{21} + \sum_{B^c_{vide}} P_1 U_{1j} \tag{5.6}$$

### 5.3.3 Performances

Les performances auxquelles on s'intéresse sont le débit moyen, les pertes, ainsi que le niveau moyen des buffers. Dans le cas de notre système mono-buffer qui est maintenant équivalent au routeur source $R_1^e$, le débit moyen est celui de $R_{21}$ et le niveau moyen du buffer concerne $B_1$.

- Débit moyen : il a déjà été déterminé précédemment pour calculer $U_1^e$ dans l'équation (5.6). Expliquons plus précisément comment on obtient ce résultat : on peut différencier deux types d'états lorsque $R_{21}$ transmet des cellules, soit le buffer $B_1$ est non vide soit il est vide. Lorsqu'il est non-vide, $R_{21}$ transfère des données à la vitesse $U_{21}$ s'il est ON. Quand $B_1$ est vide, il transmet des données seulement s'il est ON est que un ou plusieurs routeurs parmi $R_{11}$, ..., $R_{1N_1}$ est ON. Dans ce cas, la vitesse de transmission de $R_{21}$ est la somme des vitesses de transfert des routeurs de la partie amont qui sont ON. On peut remarquer que cette somme est inférieure à $U_{21}$, car si elle est supérieure, $B_1$ devient non-vide. Alors le débit moyen $X_1$ à la sortie de $R_{21}$ avec les définitions précédentes est :

$$X_1 = \sum_{B^c_{non-vide}} \vec{P}_1(x) U_{21} + \sum_{B^c_{vide}} \vec{P}_1(0) U_{1j}$$

- Pertes $Prt_1$ au niveau du buffer $B_1$. Ces pertes ont lieu lorsque le buffer $B_1$ est plein et que les routeurs amont continuent à transmettre des paquets. Pour les déterminer, il suffit de faire la différence entre le débit moyen des routeurs amont (facile à déterminer puisque ce sont des routeurs sources) et $X_1$.

$$Prt_1 = \sum_{i=1}^{N_1} \frac{\lambda_{1i}}{\lambda_{1i} + \mu_{1i}} U_{1i} - X_1 \tag{5.7}$$

- Niveau moyen du buffer : c'est la somme des niveaux moyens du buffer sur tous les états discrets et les états frontières.

$$\tilde{C_1} = \sum_{\alpha_{11} \ldots \alpha_{21}} \left( C_1(w(\alpha_{11}, \ldots, \alpha_{21}) - \pi(C_1; \alpha_{11}, \ldots, \alpha_{21})) + \int_0^{C_1} x f(x; \alpha_{11}, \ldots, \alpha_{21}) dx \right) \tag{5.8}$$

### 5.3.4 Réseau complet

Dans la section précédente, nous avons déterminé un routeur équivalent $R_1^e$ pour $R_{11}$, ..., $R_{1N_1}$, $R_{21}$ et le buffer $B_1$, et nous avons pu ainsi déterminer les performances de ce sous-système : débit moyen en sortie du routeur $R_{21}$ et niveau moyen du buffer $B_1$. Pour obtenir les performances du réseau complet, on doit exécuter cette opération d'agrégation

itérativement pour tous les routeurs et buffers du réseau. L'étape suivante est donc de déterminer un routeur équivalent $R_2^e$. C'est la même méthode que précédemment avec $R_1^e$ qui sera utilisée pour remplacer la partie du réseau située en amont de $B_2$ puisqu'il en possède les caractéristiques équivalentes. Donc $R_2^e$ devra être équivalent à $R_1^e$, $R_{22}$, ..., $R_{2N_2}$, $B_2$, et $R_{31}$ comme montré sur la figure 5.4. Ce routeur équivalent nous permettra de déterminer le débit moyen en sortie de $R_{31}$ et le niveau moyen du buffer $B_2$. A la fin de l'algorithme, on obtient $K-1$ débits moyens pour les routeurs $R_{21}$, ..., $R_{K1}$ et $K-1$ niveaux moyens pour les buffers $B_1$, ..., $B_{K-1}$.

## 5.4 Exemple

Nous allons tout d'abord appliquer l'algorithme sur un exemple afin de mieux comprendre son utilisation.

### 5.4.1 Topologie

On considère le réseau suivant de la figure 5.5.

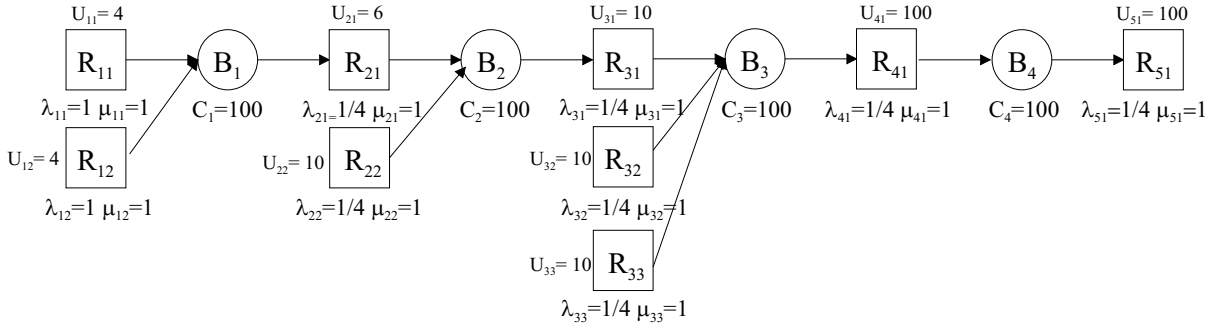


FIG. 5.5 – Exemple de réseau

Avec les notations précédentes, ce réseau possède les caractéristiques suivantes : $K = 5$, $N_1 = 2$, $N_2 = 2$, $N_3 = 3$, $N_4 = 1$.

### 5.4.2 Premier routeur équivalent $R_1^e$

D'après la description de l'algorithme, $R_1^e$ doit être déterminé de manière à ce qu'il ait le même comportement que le sous-système composé de $R_{11}$, $R_{12}$, $B_1$ et $R_{21}$.

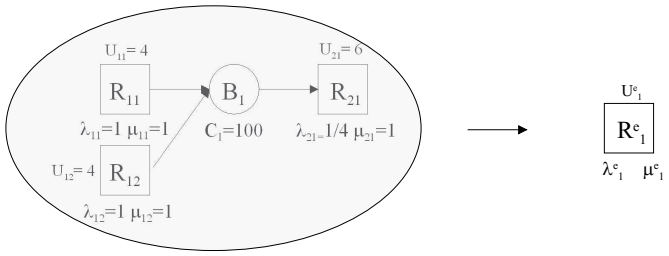


FIG. 5.6 – Premier routeur équivalent

Voici les générateurs $M_{ij}$ et les vecteurs vitesse $\vec{v}_{ij}$ de chaque routeur du sous-système :

$$M_{11} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}, \vec{v}_{11} = \begin{bmatrix} 4 & 0 \end{bmatrix}$$
$$M_{12} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}, \vec{v}_{12} = \begin{bmatrix} 4 & 0 \end{bmatrix}$$
$$M_{21} = \begin{bmatrix} -0.25 & 0.25 \\ 1 & -1 \end{bmatrix}, \vec{v}_{21} = \begin{bmatrix} 6 & 0 \end{bmatrix}$$

Le générateur markovien $M_1$ de ce processus est la somme de Kronecker de $M_{11}$, $M_{12}$ et $M_{21}$ :

$$M_1 = \begin{bmatrix} -2.25 & 0.25 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -3 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -2.25 & 0.25 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & -3 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -2.25 & 0.25 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & -3 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & -2.25 & 0.25 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & -3 \end{bmatrix}$$

Le vecteur des vitesses correspondant est :

$$\vec{d}_1 = \begin{bmatrix} 2 & 8 & -2 & 4 & -2 & 4 & -6 & 0 \end{bmatrix}$$

Et la matrice de drift $D_1$ :

$$D_1 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Le vecteur stationnaire $\vec{w}_1$ correspondant est trouvé à l'aide des résultats présentés dans le chapitre traitant de la résolution du cas mono-buffer (chapitre 3).

$$\vec{w}_1 = \left[\begin{array}{cccccccc} 0.2 & 0.05 & 0.2 & 0.05 & 0.2 & 0.05 & 0.2 & 0.05 \end{array}\right]$$

On trouve également les probabilités des états frontières buffer vide :

$$\vec{P}_1(0) = \left[\begin{array}{cccccccc} 0 & 0 & 0,06 & 0 & 0,06 & 0 & 0,09 & 0,01 \end{array}\right]$$

Ensuite, on manipule $M_1$ et $d_1$ comme expliqué dans la partie précédente pour obtenir le générateur $M_1^{tri}$ et le vecteur $d_1^{tri}$, différenciant les états selon le signe du drift. On remarque qu'il y a 4 états dans lesquels le drift est positif (le buffer $B_1$ se remplit), 1 état où il est nul, et 3 états avec un drift négatif (le buffer $B_1$ se vide).

Dans le tableau ci-dessous, on répertorie les états du sous-système étudié, rappelons qu'il fait intervenir les routeurs $R_{11}$, $R_{12}$ et $R_{21}$. En fonction du signe du drift et de la position des états dans $\vec{d_1}$ on détermine les permutations à effectuer pour trouver les états correctement triés dans $\vec{d}_1^{tri}$.

| position dans $\vec{d_1}$ | état correspondant | drift | position dans $\vec{d}_1^{tri}$ |
|---|---|---|---|
| 1 | (1,1,1) | 2 | 1 |
| 2 | (1,1,0) | 8 | 2 |
| 3 | (1,0,1) | -2 | 8 |
| 4 | (1,0,0) | 4 | 3 |
| 5 | (0,1,1) | -2 | 7 |
| 6 | (0,1,0) | 4 | 4 |
| 7 | (0,0,1) | -6 | 6 |
| 8 | (0,0,0) | 0 | 5 |

On permute les états afin de regrouper dans le bon ordre les états de drift positif, nul et négatif. On obtient :

$$\vec{d}_1^{tri} = \left[\begin{array}{cccccccc} 2 & 8 & 4 & 4 & 0 & -6 & -2 & -2 \end{array}\right]$$

On trie de la même manière le générateur $M_1^{tri}$ et les probabilités stationnaires $w_1^{tri}$ :

$$M_1^{tri} = \left[\begin{array}{cccccccc} -2.25 & 0.25 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -3 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -3 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & -3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & -2.25 & 1 & 1 \\ 1 & 0 & 0 & 0.25 & 0 & 1 & -2.25 & 0 \\ 1 & 0 & 0.25 & 0 & 0 & 1 & 0 & -2.25 \end{array}\right]$$

$$\vec{w}_1^{tri} = \left[\begin{array}{cccccccc} 0.2 & 0.05 & 0.05 & 0.05 & 0.05 & 0.2 & 0.2 & 0.2 \end{array}\right]$$

Une fois la matrice triée, il nous faut introduire les états frontières correspondant au cas où le buffer est vide. Il nous faut donc déterminer la matrice $M_1^d$, et plus précisément les taux de transition entre les états internes et les états frontières (voir section 5.2.3). L'équation (5.2) permet d'obtenir ces transitions. Dans notre exemple, il y a trois états de drift négatif, la dimension de $\Theta_1$ sera donc de 3 :

$$\begin{aligned}
\vec{P}_1^-(C_1) &= \begin{bmatrix} 0.2 & 0.2 & 0.2 \end{bmatrix} \text{ car } \vec{P}(C_1) = \vec{w}_1 \\
\vec{P}_1^-(0) &= \begin{bmatrix} 0.09 & 0.06 & 0.06 \end{bmatrix} \\
\vec{P}_1^0(0) &= 0.0077
\end{aligned}$$

Donc d'après l'équation (5.2) :

$$\Theta_1 = \begin{bmatrix} 0.69 & 0 & 0 \\ 0 & 0.35 & 0 \\ 0 & 0 & 0.35 \end{bmatrix}$$

On détermine à présent à partir de $M_1^{tri}$ les sous-matrices $M_1^{++}$, $M_1^{+0}$, $M_1^{+-}$, $M_1^{0+}$, $M_1^{00}$, $M_1^{0-}$, $M_1^{-+}$, $M_1^{-0}$ et $M_1^{--}$. D'après la section 5.2.3, on a donc :

$$M_1^{tri} = \left[\begin{array}{cccc|c|cccc} -2.25 & 0.25 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -3 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -3 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -3 & 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & -3 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0.25 & -2.25 & 1 & 1 \\ 1 & 0 & 0 & 0.25 & 0 & 1 & -2.25 & 0 \\ 1 & 0 & 0.25 & 0 & 0 & 1 & 0 & -2.25 \end{array}\right]$$

$$\begin{aligned}
M_1^{++} &= \begin{bmatrix} -2.25 & 0.25 & 0 & 0 \\ 1 & -3 & 1 & 1 \\ 0 & 1 & -3 & 0 \\ 0 & 1 & 0 & -3 \end{bmatrix} \quad M_1^{+0} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad M_1^{+-} = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\
M_1^{0+} &= \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \quad M_1^{00} = \begin{bmatrix} -3 \end{bmatrix} \quad M_1^{0-} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\
M_1^{-+} &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0.25 \\ 1 & 0 & 0.25 & 0 \end{bmatrix} \quad M_1^{-0} = \begin{bmatrix} 0.25 \\ 0 \\ 0 \end{bmatrix} \quad M_1^{--} = \begin{bmatrix} -2.25 & 1 & 1 \\ 1 & -2.25 & 0 \\ 1 & 0 & -2.25 \end{bmatrix}
\end{aligned}$$

On peut ainsi construire le générateur $M_1^d$, le vecteur de probabilités stationnaires $\vec{P}_1^d$

est le vecteur vitesse $\vec{d}_1^d$ d'après (5.1), (5.3) et (5.4) :

$$M_1^d = \begin{bmatrix} -2.25 & 0.25 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -3 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & -2.94 & 1 & 1 & 0 & 0.69 & 0 & 0 \\ 1 & 0 & 0 & 0.25 & 0 & 1 & -2.60 & 0 & 0 & 0 & 0.35 & 0 \\ 1 & 0 & 0.25 & 0 & 0 & 1 & 0 & -2.60 & 0 & 0 & 0 & 0.35 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & -2.25 & 1 & 1 \\ 1 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 1 & -2.25 & 0 \\ 1 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -2.25 \end{bmatrix}$$

$$\vec{P}_1^d = \begin{bmatrix} 0.2 & 0.05 & 0.05 & 0.05 & 0.04 & 0.11 & 0.14 & 0.14 & 0.01 & 0.09 & 0.06 & 0.06 \end{bmatrix}$$

$$\vec{d}_1^d = \begin{bmatrix} 2 & 8 & 4 & 4 & 0 & -6 & -2 & -2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

On note que dans $\vec{d}_1^d$ on trouve quatre états de drift nul, ce sont les états de buffer vide. Comme expliqué dans la section 5.2.3, lorsqu'on est dans l'un de ces états, la vitesse de variation du niveau du buffer est forcément nulle sinon on sort de cet état.

Ensuite on trie les états actifs afin de déterminer $\lambda_1^e$ et $\mu_1^e$. On va les regrouper de la même manière dans $M_1^{part}$, $d_1^{part}$, $w_1^{part}$ et $P_1^{part}(0)$.

Dans le tableau ci-dessous, on répertorie les états du sous-système étudié en considérant l'état vide ou non vide du buffer. Pour repérer les états actifs on utilise les deux critères énoncés dans 5.3.2, on détermine les permutations à effectuer pour trouver les états correctement partitionnés dans $\vec{d}_1^{part}$.

| position dans $\vec{d}_1^{tri}$ | état correspondant | type de l'état | position dans $\vec{d}_1^{part}$ |
|---|---|---|---|
| 1 | ($x_1$;1,1,1) | actif | 12 |
| 2 | ($x_1$;1,1,0) | non-actif | 1 |
| 3 | ($x_1$;1,0,0) | non-actif | 2 |
| 4 | ($x_1$;0,1,0) | non-actif | 3 |
| 5 | ($x_1$;0,0,0) | non-actif | 4 |
| 6 | ($x_1$;0,0,1) | actif | 11 |
| 7 | ($x_1$;0,1,1) | actif | 10 |
| 8 | ($x_1$;1,0,1) | actif | 9 |
| 9 | (0;0,0,0) | non-actif | 5 |
| 10 | (0;0,0,1) | non-actif | 6 |
| 11 | (0;0,1,1) | actif | 8 |
| 12 | (0;1,0,1) | actif | 7 |

$$\vec{d}_1^{part} = \left[\begin{array}{cccccc|cccccc} 8 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & -2 & -2 & -6 & 2 \end{array}\right] \quad (5.9)$$

$$\vec{P}_1^{part} = \left[\begin{array}{cccccc|cccccc} 0.05 & 0.05 & 0.05 & 0.04 & 0.01 & 0.09 & 0.06 & 0.06 & 0.14 & 0.14 & 0.11 & 0.20 \end{array}\right]$$

$$M_1^{part} = \left[\begin{array}{cccccc|cccccc} -3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & -3 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -3 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & -3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25 & -2.25 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0.25 & 0 & 0 & 0 & 1 & -2.25 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0.25 & 0 & 0 & 1 & 0 & -2.25 & 0 & 0 & 0 & 1 \\ 0 & 0.25 & 0 & 0 & 0 & 0 & 0.35 & 0 & -2.60 & 0 & 1 & 1 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0.35 & 0 & -2.60 & 1 & 1 \\ 0 & 0 & 0 & 0.25 & 0 & 0.69 & 0 & 0 & 1 & 1 & -2.94 & 0 \\ 0.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -2.25 \end{array}\right]$$

Nous avons à présent besoin de $N_1^{part} = I + \Gamma_1^{-1} M_1^{part}$ et de du vecteur de probabilités stationnaires correspondant $z_1^{part}$. D'après la définition de $\Gamma_1$ et de $N_1^{part}$, on :

$$\Gamma_1 = \left[\begin{array}{cccccccccccc} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.25 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.60 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.94 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.25 \end{array}\right]$$

$$N_1^{part} = \left[\begin{array}{cccccc|cccccc} 0 & 0.33 & 0.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.33 \\ 0.33 & 0 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0.33 & 0 & 0 & 0 \\ 0.33 & 0 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0 & 0.33 & 0 & 0 \\ 0 & 0.33 & 0.33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.33 & 0 \\ 0 & 0.33 & 0.33 & 0 & 0 & 0.33 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.11 & 0 & 0.44 & 0.44 & 0 & 0 & 0 & 0 \\ \hline 0 & 0.11 & 0 & 0 & 0 & 0.44 & 0 & 0 & 0 & 0 & 0 & 0.44 \\ 0 & 0 & 0.11 & 0 & 0 & 0.44 & 0 & 0 & 0 & 0 & 0 & 0.44 \\ 0 & 0.10 & 0 & 0 & 0 & 0 & 0.13 & 0 & 0 & 0 & 0.38 & 0.38 \\ 0 & 0 & 0.10 & 0 & 0 & 0 & 0 & 0.13 & 0 & 0 & 0.38 & 0.38 \\ 0 & 0 & 0 & 0.09 & 0 & 0.24 & 0 & 0 & 0.34 & 0.34 & 0 & 0 \\ 0.11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.44 & 0.44 & 0 & 0 \end{array}\right]$$

Et le vecteur de probabilités stationnaires est :

$$z_1^{part} = \left[\begin{array}{cccccc|cccccc} 0.13 & 0.13 & 0.13 & 0.07 & 0.00 & 0.53 & 0.24 & 0.24 & 0.13 & 0.13 & 0.11 & 0.13 \end{array}\right]$$

Nous déterminons $\lambda_1^e$ grâce à l'équation (5.5) et aux sous-matrices $M_1^B$, $N_1^{B^cB}$ et $z_1^{B^c}$ :

$$M_1^B = \begin{bmatrix} -3 & 1 & 1 & 0 & 0 & 0 \\ 1 & -3 & 0 & 1 & 0 & 0 \\ 1 & 0 & -3 & 1 & 0 & 0 \\ 0 & 1 & 1 & -3 & 0 & 0 \\ 0 & 1 & 1 & 0 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0.25 & -2.25 \end{bmatrix}$$

$$N_1^{B^cB} = \begin{bmatrix} 0 & 0.11 & 0 & 0 & 0 & 0.44 \\ 0 & 0 & 0.11 & 0 & 0 & 0.44 \\ 0 & 0.10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.09 & 0 & 0.24 \\ 0.11 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$z_1^{B^c} = \begin{bmatrix} 0.13 & 0.13 & 0.13 & 0.07 & 0 & 0.53 \end{bmatrix}$$

On obtient alors le temps moyen d'indisponibilité du routeur équivalent $R_1^e$ :

$$1/\lambda_1^e = 0.78$$

On procède de même pour $\mu_1^e$ avec l'équation (5.5) et les sous-matrices $M_1^{B^c}$, $N_1^{BB^c}$ et $z_1^B$.

$$M_1^{B^c} = \begin{bmatrix} -2.25 & 0 & 0 & 0 & 0 & 1 \\ 0 & -2.25 & 0 & 0 & 0 & 1 \\ 0.35 & 0 & -2.60 & 0 & 1 & 1 \\ 0 & 0.35 & 0 & -2.60 & 1 & 1 \\ 0 & 0 & 1 & 1 & -2.94 & 0 \\ 0 & 0 & 1 & 1 & 0 & -2.25 \end{bmatrix}$$

$$N_1^{BB^c} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0.33 \\ 0 & 0 & 0.33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.33 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.44 & 0.44 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$z_1^B = \begin{bmatrix} 0.24 & 0.24 & 0.13 & 0.13 & 0.11 & 0.13 \end{bmatrix}$$

On obtient alors le temps moyen de disponibilité du routeur équivalent $R_1^e$ :

$$1/\mu_1^e = 1.88$$

Pour finir de déterminer le routeur équivalent $R_1^e$, il nous faut trouver sa vitesse de transmission $U_1^e$. Le débit moyen en sortie de $R_{21}$ peut être déterminé puisque nous connaissons les états actifs, leurs probabilités stationnaires et la vitesse de transmission en sortie de $R_{21}$. Nous avons six états actifs : $(0;1,0,1)$, $(0;0,1,1)$, $(x_1;1,0,1)$, $(x_1;0,1,1)$,

$(x_1; 0, 0, 1)$, $(x_1; 1, 1, 1)$ (voir le tableau précédent indiquant les permutations à effectuer). Pour les états de buffer vide, la vitesse de transmission de données est égale à la vitesse de transmission du routeur amont (forcément inférieure à celle du routeur aval puisque le buffer reste vide dans cet état). Dans notre cas, cette vitesse vaut 4. Pour les états de buffer non vide, la vitesse de transmission est celle du routeur aval, c'est à dire 6.

$$X_1 = 4 \text{ x } 0.06 + 4 \text{ x } 0.06 + 6 \text{ x } 0.14 + 6 \text{ x } 0.14 + 6 \text{ x } 0.11 + 6 \text{ x } 0.20 = 3.99$$

Sachant que $R_1^e$ a le même débit que $R_{21}$, que c'est un routeur source et que sa probabilité stationnaire d'être dans l'état ON est $\frac{\lambda_1^e}{\mu_1^e+\lambda_1^e}$, alors :

$$U_1^e = X_1 \frac{\mu_1^e + \lambda_1^e}{\lambda_1^e}$$

Finalement on obtient notre premier routeur équivalent $R_1^e$ :
- $1/\lambda_1^e = 0.78$
- $1/\mu_1^e = 1.88$
- $U_1^e = 5.65$

### 5.4.3 Performances du routeur $R_{21}$

A ce niveau de l'algorithme, nous connaissons les trois indices de performance auxquels nous nous intéressons.

- Débit moyen de $R_{21}$ : Il a été déterminé dans la section précédente et vaut 3.99 paquets par seconde.
- Pertes au niveau du buffer $B_1$ : Il suffit de faire la différence entre le débit moyen des routeurs amont ($R_{11}$ et $R_{12}$) et le débit moyen du routeur aval ($R_{21}$). Les routeurs amont sont des routeurs sources, leur débit moyen est donc $U_{1i}\frac{\lambda_{1i}}{\lambda_{1i}+\mu_{1i}}$. Les pertes $Prt_1$ au niveau de $B_1$ sont donc :

$$Prt_1 = 4\frac{1}{1+1} + 4\frac{1}{1+1} - 3.99 = 0.01$$

- Niveau moyen de $B_1$ : Le niveau moyen du buffer est déterminé lorsqu'on étudie le cas mono-buffer de la manière décrite dans la section 5.3.3 grâce à l'équation (5.8). On obtient $\tilde{C}_1 = 8.4056$.

### 5.4.4 Étapes suivantes

Après cette étape, on connaît toutes les caractéristiques de $R_{21}$. De plus, on est capable de remplacer toute la partie amont à $R_{21}$ par un seul routeur source équivalent, c'est-à-dire $R_{11}$, $R_{12}$, $B_1$ et $R_{21}$ (figure 5.5). C'est ce que l'on fait dans l'étape suivante : on considère le sous-système composé de $R_1^e$, $R_{22}$, $B_2$ et $R_{31}$. Nous allons étudier ce sous-système pour déterminer ses indices de performance. Ensuite, nous réduisons ce sous-système à un routeur équivalent $R_2^e$ comme nous l'avons fait précédemment (figure 5.7).

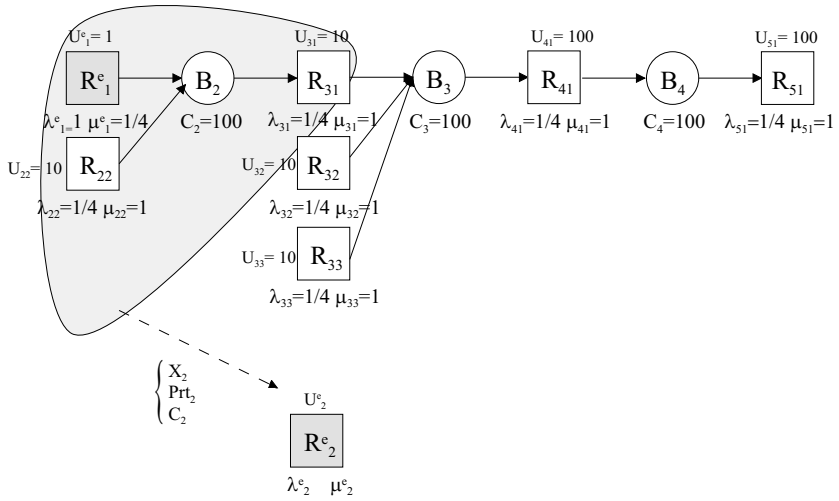


FIG. 5.7 – Second routeur équivalent

Nous procédons ainsi itérativement jusqu'à obtenir les performances de tous les routeurs du système.

## 5.5 Résultats numériques

Dans cette section, nous comparons les résultats donnés par notre méthode analytique et par un simulateur fluide. Le premier réseau étudié est celui de la figure 5.5. Un second réseau représenté sur la figure 5.8 est aussi étudié, ainsi qu'un troisième figure 5.9.

### 5.5.1 Exemple 1

Pour cet exemple, nous avons choisi des vitesses de transmission croissantes pour les routeurs de l'épine dorsale. On risque donc de retrouver peu de pertes dans ce genre de situation. Les résultats concernant les débits moyens, les niveaux moyens et les pertes sont présentés dans les tableaux 5.1, 5.2 et 5.3.

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Router $R_{11}$ | 2,0000 | 2,0002 | 0,01 |
| Router $R_{21}$ | 3,9999 | 4,0014 | 0,04 |
| Router $R_{31}$ | 8,0000 | 8,0005 | 0,01 |
| Router $R_{41}$ | 23,8860 | 23,8838 | 0,01 |
| Router $R_{51}$ | 23,7462 | 23,7188 | 0,12 |
| Router $R_{12}$ | 2,0000 | 2,0013 | 0,07 |
| Router $R_{22}$ | 8,0000 | 7,9997 | 0,00 |
| Router $R_{32}$ | 8,0000 | 8,0003 | 0,00 |
| Router $R_{33}$ | 8,0000 | 7,9990 | 0,01 |

TAB. 5.1 – Débits moyens (1er réseau)

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Buffer $B_1$ | 8,4056 | 8,4522 | 0,55 |
| Buffer $B_2$ | 97,2356 | 97,1373 | 0,10 |
| Buffer $B_3$ | 6,5992 | 6,6163 | 0,26 |
| Buffer $B_4$ | 6,6668 | 6,7452 | 1,18 |

TAB. 5.2 – Niveaux moyens des buffers (1er réseau)

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Buffer $B_1$ | 0,0001 | 0,0001 | 0,00 |
| Buffer $B_2$ | 3,9999 | 4,0006 | 0,02 |
| Buffer $B_3$ | 0,1140 | 0,1160 | 1,75 |
| Buffer $B_4$ | 0,1398 | 0,1650 | 18,03 |

TAB. 5.3 – Pertes (1er réseau)

### 5.5.2 Exemple 2

Dans ce second exemple, les vitesses de transmission des routeurs de l'épine dorsale sont décroissantes. On risque donc d'avoir beaucoup de pertes au niveau des buffers. Les résultats concernant les débits moyens, les niveaux moyens et les pertes sont présentés dans les tableaux 5.4, 5.5 et 5.6.

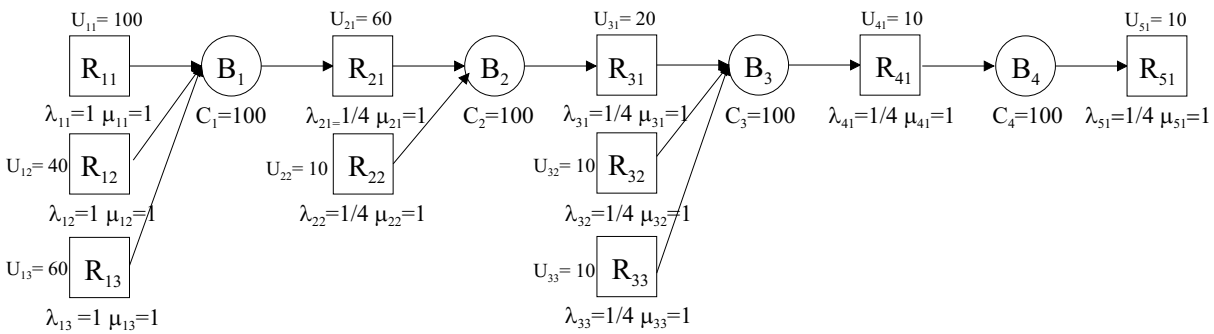


FIG. 5.8 – 2nd réseau

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Router $R_{11}$ | 50,0000 | 50,0747 | 0,15 |
| Router $R_{21}$ | 47,7304 | 47,7304 | 0,00 |
| Router $R_{31}$ | 15,9996 | 16,0083 | 0,05 |
| Router $R_{41}$ | 8,0000 | 8,0083 | 0,10 |
| Router $R_{51}$ | 7,7793 | 7,7769 | 0,03 |
| Router $R_{12}$ | 20,0000 | 20,0118 | 0,03 |
| Router $R_{13}$ | 30,0000 | 30,0087 | 0,03 |
| Router $R_{22}$ | 8,0000 | 7,9933 | 0,08 |
| Router $R_{32}$ | 8,0000 | 8,0072 | 0,09 |
| Router $R_{33}$ | 8,0000 | 7,9987 | 0,02 |

TAB. 5.4 – Débits moyens (2nd réseau)

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Buffer $B_1$ | 90,0818 | 90,1315 | 0,06 |
| Buffer $B_2$ | 97,6850 | 97,6305 | 0,06 |
| Buffer $B_3$ | 99,9420 | 99,9429 | 0,00 |
| Buffer $B_4$ | 50,0000 | 50,5224 | 1,04 |

TAB. 5.5 – Niveaux moyens des buffers (2nd réseau)

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Buffer $B_1$ | 52,2696 | 52,3648 | 0,18 |
| Buffer $B_2$ | 39,7308 | 39,7153 | 0,04 |
| Buffer $B_3$ | 23,9996 | 24,0117 | 0,05 |
| Buffer $B_4$ | 0,2207 | 0,2255 | 2,17 |

TAB. 5.6 – Pertes (2nd réseau)

### 5.5.3 Exemple 3

Nous nous intéressons à un réseau plus complexe puisque les convergences ne sont plus uniquement des routeurs sources. Nous allons déterminer les performances de tous les éléments de ce réseau en utilisant la méthode analytique. Les résultats concernant les débits moyens, les niveaux moyens et les pertes sont présentés dans les tableaux 5.7, 5.8 et 5.9.

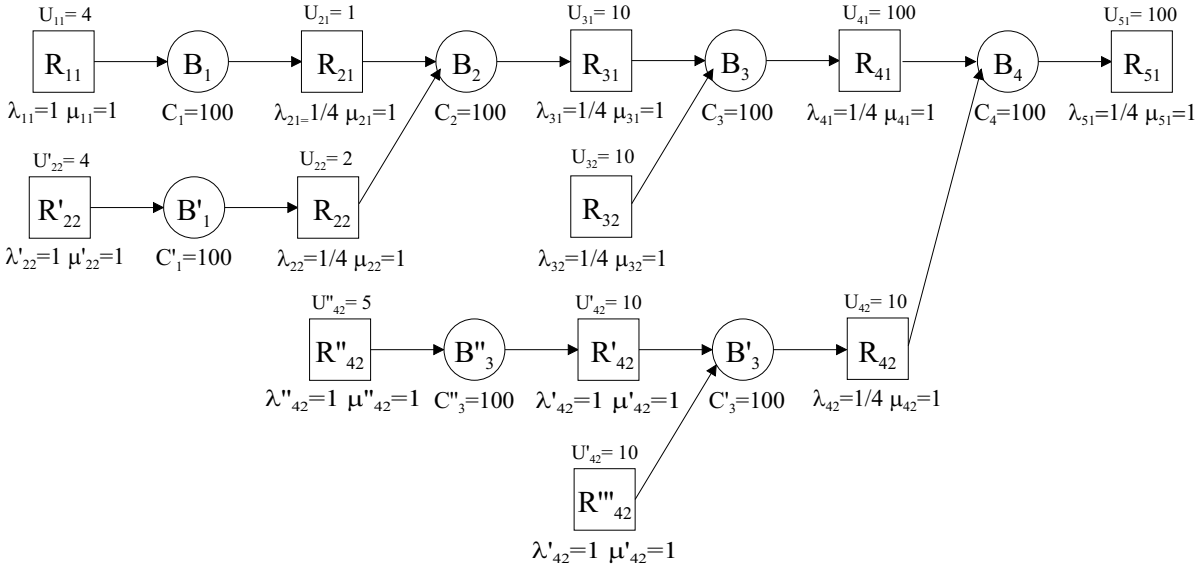


FIG. 5.9 – 3ème réseau

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Router $R_{11}$ | 2,0000 | 1,9992 | 0,04 |
| Router $R_{21}$ | 0,8000 | 0,7997 | 0,04 |
| Router $R_{31}$ | 2,4000 | 2,3986 | 0,06 |
| Router $R_{41}$ | 10,3996 | 10,4079 | 0,08 |
| Router $R_{51}$ | 17,7761 | 17,8010 | 0,14 |
| Router $R'_{22}$ | 2,0000 | 2,0025 | 0,13 |
| Router $R_{22}$ | 1,6000 | 1,5988 | 0,08 |
| Router $R_{32}$ | 8,0000 | 8,0096 | 0,12 |
| Router $R''_{42}$ | 2,5000 | 2,4992 | 0,03 |
| Router $R'_{42}$ | 2,5000 | 2,4993 | 0,03 |
| Router $R_{42}$ | 7,4009 | 7,4194 | 0,25 |
| Router $R'''_{42}$ | 5,0000 | 5,0056 | 0,11 |

TAB. 5.7 – Débits moyens (3ème réseau)

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Buffer $B_1$ | 99,2901 | 99,2906 | 0,00 |
| Buffer $B_2$ | 0,7043 | 0,7063 | 0,28 |
| Buffer $B_3$ | 2,4159 | 2,4083 | 0,32 |
| Buffer $B_4$ | 4,5941 | 4,6186 | 0,53 |
| Buffer $B'_1$ | 95,1015 | 95,1705 | 0,07 |
| Buffer $B''_3$ | 3,2019 | 3,1853 | 0,52 |
| Buffer $B'_3$ | 33,3175 | 32,6846 | 1,94 |

TAB. 5.8 – Niveaux moyens des buffers (3ème réseau)

| | Méthode analytique | Simulation | Erreur (%) |
|---|---|---|---|
| Buffer $B_1$ | 1,2000 | 1,1994 | 0,05 |
| Buffer $B_2$ | 0,0000 | 0,0000 | 0,00 |
| Buffer $B_3$ | 0,0004 | 0.0004 | 0,00 |
| Buffer $B_4$ | 0,0244 | 0,0264 | 8,20 |
| Buffer $B'_1$ | 0,4000 | 0,4036 | 0,9 |
| Buffer $B''_3$ | 0,0000 | 0,0000 | 0,00 |
| Buffer $B'_3$ | 0,0991 | 0,0855 | 15,91 |

TAB. 5.9 – Pertes (3ème réseau)

En comparant les résultats analytiques à ceux de la simulation, on constate des erreurs relativement faibles. De plus ces résultats sont obtenus très rapidement (voir tableau 5.10, résultats obtenus sur un Pentium 4 à 2GHz).

| | Méthode analytique | Simulation |
|---|---|---|
| Exemple 1 | $\sim$ 2s | 50s |
| Exemple 2 | $\sim$ 2s | 50s |
| Exemple 3 | $\sim$ 2s | 80s |

TAB. 5.10 – Temps d'obtention des résultats

Dans ce dernier chapitre , nous avons traité le cas plus complexe des réseaux comportant des nœuds de convergence. Une étude du cas mono-buffer multi-sources nous a permis de mettre en évidence les difficultés liées à la présence de convergences et de trouver les modifications à effectuer sur l'algorithme développé dans le chapitre 5 précédent.

# Conclusion

Dans cette thèse, nous nous sommes intéressés à l'évaluation de performances de réseaux de communication. Nous avons développé une méthode permettant d'obtenir ces indices de performances de manière analytique.

Tout d'abord, nous avons présenté quelques notions sur les réseaux afin de faciliter la compréhension du travail effectué. Nous avons présenté la topologie de certains réseaux ainsi que certains principes de fonctionnement. Nous avons également défini les indices permettant d'évaluer la qualité de service des réseaux. Les réseaux auxquels nous nous sommes intéressés sont les réseaux IP et surtout ATM. Ensuite, deux types de modèle pour les réseaux ont été présentés : le modèle discret et le modèles fluide. Après une comparaison, nous avons expliqué les raisons de notre choix d'utiliser le modèle fluide.

Dans le chapitre 2, différents simulateurs de réseaux sont présentés. L'étude de ces simulateurs nous a permis d'en constater les inconvénients dont le principal est le temps d'exécution lorsque le réseau est de grande taille. On a également présenté l'algorithme du simulateur fluide utilisé dans cette thèse.

Nous nous sommes ensuite intéressés, dans le chapitre 3, à l'étude analytique d'un réseau simple : le système « monobuffer ». Les résultats obtenus dans ce chapitre étaient des résultats préliminaires en vue de l'étude de réseaux plus complexes. Ce chapitre a permis de présenter les outils employés pour la modélisation des réseaux et la détermination de ses performances. On a différencié le cas « homogène » et le cas « non-homogène ».

Dans le chapitre 4, nous avons étudié des réseaux simples : les réseaux à topologie linéaire. Nous avons évalué les performances de réseaux linéaires homogènes grâce à une méthode de décomposition qui nous a permis de nous ramener à une étude de cas « mono-buffer » . Nous avons ensuite développé cette méthode afin de l'étendre au cas de réseaux linéaires non homogènes. Nous avons testé les deux algorithmes développés sur des exemples et avons pu observer une bonne précision des résultats obtenus.

Le dernier chapitre 5 nous a permis de traiter du cas plus complexe des réseaux comportant des nœuds de convergence. Une étude du cas mono-buffer multi-sources nous a permis de mettre en évidence les difficultés liées à la présence de convergences et de trouver les modifications à effectuer sur l'algorithme développé dans le chapitre 5 précédent. Les exemples présentés à la fin de ce chapitre ont permis de mettre en évidence la bonne

précision des résultats obtenus à l'aide de la méthode analytique. Les performances du réseau en termes de débit, de pertes et de niveau de remplissage des buffers sont ainsi obtenues en un minimum de temps de calcul.

Pour développer la méthode d'évaluation de performances présentée dans cette thèse, on a effectué des hypothèses sur le fonctionnement des routeurs et sur le type de trafic. Ces hypothèses ne permettent pas de prendre en compte tous types d'applications. On peut envisager d'étendre la méthode pour différents cas de figures, avec des sources de trafic différentes, ou bien avec un retour d'information en cas de congestion.

# Annexe A

# Produit et somme de Kronecker

Dans cette annexe sont rappelées les propriétés de base de l'algèbre tensorielle. On peut trouver une présentation plus complète de l'algèbre tensorielle dans [Dav81].

## A.1 Produit de Kronecker

Soient deux matrices $A$ et $B$ de dimensions respectives $m$ x $n$ et $p$ x $q$, définies de la manière suivante :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \qquad \text{et} \qquad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{p1} & a_{p2} & \cdots & b_{pq} \end{bmatrix}$$

Le produit de Kronecker, aussi appelé produit tensoriel de $A$ et $B$ est défini par :

$$C = A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}$$

La matrice $C$ obtenue est de dimension $mp$ x $nq$.

**Exemple.** Soient $A$ de dimension 2 x 2 et $B$ de dimension 3 x 3.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad \text{et} \qquad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

Le produit de Kronecker $C = A \otimes B$ est égal à :

$$\begin{aligned} C &= \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix} \\ &= \left[\begin{array}{ccc|ccc} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} \\ a_{11}b_{31} & a_{11}b_{32} & a_{11}b_{33} & a_{12}b_{31} & a_{12}b_{32} & a_{12}b_{33} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} \\ a_{21}b_{31} & a_{21}b_{32} & a_{21}b_{33} & a_{22}b_{31} & a_{22}b_{32} & a_{22}b_{33} \end{array}\right] \end{aligned}$$

Voici quelques propriétés du produit de Kronecker. Les 3 premières justifient le nom de produit.

$P1$ $\quad (\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B)$

$P2$ Soient $A$ et $B$ deux matrices de même dimension, $C$ et $D$ deux autres matrices de même dimension, alors :
$(A + B) \otimes (C + D) = A \otimes C + A \otimes D + B \otimes C + B \otimes D$

$P3$ $\quad (A \otimes B) \otimes C = A \otimes (B \otimes C)$

$P4$ On note $I_{np}$ la matrice identité de dimension $n$x$p$ et $I_n$, $I_p$ les matrices identités de dimensions respectives $n$x$n$ et $p$x$p$. $I_{np} = I_n \otimes I_p = I_p \otimes I_n$

$P5$ $\quad (AC) \otimes (BD) = (A \otimes B)(C \otimes D)$

$P6$ Soient $A$ et $B$ deux matrices carrées respectivement $n$x$n$ et $p$x$p$ avec $spect(A) = \lambda_1, \ldots, \lambda_n$ et $spect(B) = \mu_1, \ldots, \mu_p$, alors :

$$spect(A \otimes B) = \{\lambda_i \mu_j \mid i = 1 \ldots n; j = 1 \ldots p\}$$

De plus $x_i \otimes y_j$ sont vecteurs propres de $A \otimes B$ si $x_i$ et $y_j$ sont les vecteurs propres associés respectivement à $\lambda_i$ et $\mu_j$.

## A.2 Somme de Kronecker

### A.2.1 Somme de matrices carrées

Soient deux matrices carrées $A = [a_{ij}]$ et $B = [b_{ij}]$ de dimensions respectives $n$ x $n$ et $p$ x $p$.

La somme de Kronecker de $A$ et $B$ est définie par :

$$C = A \oplus B = A \otimes I_p + I_n \otimes B$$

**Exemple.** La somme de Kronecker des deux matrices $A$ et $B$ de l'exemple précédent est égale à :

$$\left[\begin{array}{ccc|ccc} a_{11}+b_{11} & b_{12} & b_{13} & a_{12} & 0 & 0 \\ b_{21} & a_{11}+b_{22} & b_{23} & 0 & a_{12} & 0 \\ b_{31} & b_{32} & a_{11}+b_{33} & 0 & 0 & a_{12} \\ \hline a_{21} & 0 & 0 & a_{22}+b_{11} & b_{12} & b_{13} \\ 0 & a_{21} & 0 & b_{21} & a_{22}+b_{22} & b_{23} \\ 0 & 0 & a_{21} & b_{31} & b_{32} & a_{22}+b_{33} \end{array}\right]$$

### A.2.2 Somme de vecteurs

Soient $\vec{x}$ et $\vec{y}$ deux vecteurs lignes de dimensions respectives $n$ et $p$, alors on appelle somme de Kronecker de ces deux vecteurs :

$$\vec{x} \oplus \vec{y} = \vec{x} \otimes \vec{1}_p + \vec{1}_n \otimes \vec{y}$$

où $\vec{1}_p$ et $\vec{1}_n$ sont des vecteurs lignes unitaires de dimensions respectives $p$ et $n$.
**Exemple.** Si $\vec{x} = \left[ \begin{array}{cc} x_1 & x_2 \end{array} \right]$ et $\vec{y} = \left[ \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \right]$, alors :

$$\begin{array}{rcl} \vec{x} \oplus \vec{y} & = & \left[ \begin{array}{cc} x_1 & x_2 \end{array} \right] \otimes \left[ \begin{array}{ccc} 1 & 1 & 1 \end{array} \right] + \left[ \begin{array}{cc} 1 & 1 \end{array} \right] \otimes \left[ \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \right] \\ & = & \left[ \begin{array}{ccc|ccc} x_1 + y_1 & x_1 + y_2 & x_1 + y_3 & x_2 + y_1 & x_2 + y_2 & x_2 + y_3 \end{array} \right] \end{array}$$

# Bibliographie


[Abd02] S. Abdellatif. *Contribution à la modélisation et à l'analyse de la qualité de service dans les réseaux à commutation de paquets.* PhD thesis, Université Paul Sabatier, 2002.

[ADD94] R. Alvarez, Y. Dallery, and R. David. Experimental study of the continuous flow model of production lines with unreliable machines and finite buffers. *Journal of Manufacturing Systems*, 13(3) :221–234, 1994.

[AMS82] D. Anick, D. Mitra, and M. M. Sondhi. Stochastic theory of a data-handling system with multiple sources. *The Bell System Technical Journal*, 61 :1871–1894, 1982.

[BP70] A. Berman and R. J. Plemmons. *Non-negative matrices in the mathematical sciences.* Academic Press, New York, 2nd edition, 1970.

[Cas93] C. G. Cassandras. *Discrete Event Systems : modeling and performance analysis.* Aksen Associates, Irwin, Boston, MA, 1993.

[CKR+95] C. Courcoubetis, G. Kesidis, A. Ridder, J. Walrand, and R. R. Weber. Admission control and routing in ATM networks using inferences from measured buffer occupancy. *IEEE Transactions on Communications*, 43 :1778–1784, 1995.

[CL99] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[CSP01] C. G. Cassandras, G. Sun, and C. G. Panayiotou. Stochastic fluid models for control and optimization of systems with quality of service requirements. In *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001.

[Dal94] Y. Dallery. On modeling failure and repair times in stochastic models of manufacturing systems unsing generalized exponential distributions. *Queueing Systems*, 15 :199–209, 1994.

[Dav81] A. Davio. Kronecker products and shuffle algebra. *IEEE Trans. Comput.*, 30 :116–125, 1981.

[DDX89] Y. Dallery, R. David, and X. L. Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Trans. on Automatic Control*, 34(9) :943–953, 1989.

[DF82] D. Dubois and J. P. Forestier. Productivité et en-cours moyens d'un ensemble de deux machines séparées par une zone de stockage. *RAIRO Automatique/Systems Analysis and Control*, 16(2) :105–132, 1982.

[DG92] Y. Dallery and S. B. Gershwin. Manufacturing flow line systems : a review of models and analytical results. *Queueing Systems*, 12 :3–94, 1992.

[DLB97] Y. Dallery and H. Le Bihan. Homogeneisation techniques for the analysis of production lines with unreliable machines having different speeds. *EJC*, 3 :200–215, 1997.

[DLB98] Y. Dallery and H. Le Bihan. An improved decomposition method for the analysis of production lines with unreliable machines and finite buffers. *Int. J. of Prod. Res.*, 1998.

[DS98] A. Das and R. Srikant. Diffusion approximations for models of congestion control in high-speed networks. In *Proceedings of the 37th IEEE Conference on Decision and Control*, 1998.

[Fer05] A. Ferréro. *Architectures télécoms de l'internet.* Lavoisier, 2005.

[Ger94] S. B. Gershwin. *Manufacturing systems engineering.* Prentice Hall, New Jersey, 1994.

[GPW02] H. C. Gromoll, A. L. Puha, and R. J. Williams. The fluid limit of a heavily loaded processor sharing queue. *Annals of Applied Probability*, 12 :797–859, 2002.

[Gri94] S. Grishechkin. GI/GI/1 processor sharing queue in heavy traffic. *Advances in Applied Probability*, 26 :539–555, 1994.

[KM98] K. Kumaran and D. Mitra. Performance and fluid simulations of a novel shared buffer management system. In *Proceedings of the IEEE INFOCOM*, 1998.

[KR92] H. Kobayashi and Q. Ren. A mathematical theory for transient analysis of communications networks. *IEICE Transactions on Communications*, E75B :1266–1276, 1992.

[KSCK96] G. Kesidis, A. Singh, D. Cheung, and W. W. Kwok. Feasibility of fluid-driven simulation for ATM network. In *Proceedings of the IEEE Globecom*, volume 3, pages 2013–2017, 1996.

[KW93] G. Kesidis and J. Walrand. Quick simulation of ATM buffers with on-off multiclass Markov fluid sources. *Modeling and Computer Simulation*, 3(3) :269–276, 1993.

[KWC93] G. Kesidis, J. Walrand, and C.S Chang. Effective bandwiths for multiclass Markov fluid and other ATM sources. *IEEE/ACM Transactions on Networking*, 1(4) :424–428, 1993.

[LBD98] H. Le Bihan and Y. Dallery. A robust decomposition method for the analysis of production lines with unreliable machines and finite buffers. *Annals of Operations Research*, 1998.

[LGK+99] B. Liu, Y. Guo, J. Kurose, D. Towsley, and W. B. Gong. Fluid simulation of large scale networks :issues and tradeoffs. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 1999.

[LMT00] R. Levantesi, A. Matta, and T. Tolio. Performance evaluation of continuous production lines with deterministic processing times, multiple failure modes and finite buffer capacity. In *Queuing Networks with Finite Capacity*, 2000.

[Mit87] D. Mitra. Stochastic fluid models. *Performance*, pages 39–51, 1987.

[Mit88] D. Mitra. Stochastic theory of a fluid model of producers and consumers coupled by a buffer. *Advances in Applied Probability*, 20 :646–676, 1988.

[Mél01] J.L. Mélin. *Qualité de service sur IP*. Eyrolles, 2001.

[Moc03] S. Mocanu. A versatile decomposition method for continuous flow lines. *Rapport technique*, 2003.

[MR99] L. Massouli and J. Roberts. Arguments in favour of admission control for TCP flows. In *Teletrafic Engineering in a Competitive World - Proceedings of ITC-16*, pages 33–44, Edinburgh, 1999. Elsevier, Amsterdam.

[Puj05a] G. Pujolle. *Contrôle dans les réseaux IP*. Lavoisier, 2005.

[Puj05b] G. Pujolle. *Les réseaux*. Eyrolles, 2005.

[RS89] G. Rubino and B. Sericola. Sojourn times in Markov processes. *Journal of Applied Probability*, 27(4) :744–756, 1989.

[SF91] R. Suri and B. R. Fu. On using continuous flow lines for performance estimation of discrete production lines. In *Proceedings of the 1991 Winter Simulation Conference*, 1991.

[SF94] R Suri and B. R. Fu. On using continuous flow lines to model discrete production lines. *Journal of Discrete Event Dynamic Systems*, 4 :129–169, 1994.

[ST99] B. Sericola and B. Tuffin. A fluid queue driven by a markovian queue. *Queuing System – Theory and Applications*, 31(3), 1999.

[Tan88] A. S. Tanenbaum. *Computer Networks, 2nd Ed.* Prentice Hall, New Jersey, 1988.

[WM94] Y. Wardi and B. Melamed. IPA gradient estimation for the loss volume in continuous flow models. In *Proceedings of the Hong Kong International Workshop on New Directions of Control and Manufacturing*, pages 30–33, Hong Kong, 1994.

[YG99] A. Yan and W. B. Gong. Fluid simulation for high-speed networks with flow-based routing. *IEEE Transactions on Information Theory*, 45 :1588–1599, 1999.

## Evaluation de performances de réseaux de communication à l'aide de chaînes de Markov hybrides

Cette thèse est consacrée à l'évaluation de performances de réseaux de communication. On s'intéresse plus particulièrement à leur modélisation à l'aide de chaînes de Markov hybrides et à la résolution analytique de ces modèles. On caractérise les performances d'un réseau avec différents paramètres comme le débit ou les pertes. On peut les obtenir à l'aide de simulateurs, reposant sur un modèle discret pour la plupart. Mais ceci peut entraîner des temps de simulation très longs. C'est pour cela que nous développons une méthode analytique basée sur un modèle fluide du réseau. L'utilisation d'un modèle fluide associé à une méthode d'agrégation réduit la complexité du problème et permet une résolution analytique plus rapide que la simulation. Nous nous intéressons d'abord à un système simple, dit mono-buffer, afin de déterminer quelques résultats utiles à l'étude de réseaux plus complexes. Ensuite nous présentons une méthode analytique pour le cas des réseaux de routeurs sous certaines hypothèses en utilisant une représentation Markovienne des états du réseau considéré. Les réseaux comportant la plupart du temps des convergences et des divergences, la méthode analytique a été adaptée pour permettre l'évaluation de performances dans ce type de situation. Les résultats obtenus sont comparables à ceux donnés par les simulateurs classiques, mais avec un temps de calcul beaucoup plus court.

**Mots-clés :** Evaluation de performances, réseau, modèle fluide, méthode analytique, décomposition.

## Performance evaluation of communication networks with hybrid Markov chains

This thesis is devoted to study the performance evaluation of communication networks. We are especially interested in their modelling with hybrid Markov chains and analytical solutions of the models. The network performance is characterized with parameters such as the flow or the losses. These parameters can be obtained by means of simulators based on a discrete model. But this needs sometimes long computing times. That is why we have developed an analytical method based on a fluid model of the network. The use of such a model associated with an aggregation method reduces the problem complexity and allows a faster analytical resolution than simulation. We first study the simple system, known as mono-buffer, in order to find some results that can be used in more complex networks. Then, we present an analytical method for complex router networks while using Markovian state representation of the considered network. Since most of the networks have convergences and divergences, this method was adapted for the performance evaluation in this case. The results obtained are comparable with those of the classical simulators but in shorter computing times.

**Keywords :** Performance evaluation, network, fluid model, analytical method, decomposition.