TABLE DES MATIERES

NOTATIONS

1. Minuscules latines

d	Distance entre l'émetteur et le récepteur
m	Moyenne d'une variable aléatoire
n	Nombre de ressources occupées par les trafics réseaux
t_m	Durée moyenne d'occupation de la ressource par chaque demande
v	Valuation d'un graphe
x, y	Couple de sommets x et y formant un arc d'un graphe
q	Taille courante d'une file d'attente
w_q	Poids de modification de la taille d'une file d'attente
p_b	Probabilité de rejet de paquets

2. Majuscules latines

D	Délai
G	Graphe
H	Paramètre de Hurst
M	Matrice représentant un graphe
P	Probabilité
Rx	Représente un routeur de numéro x
S	Ensemble de sommets d'un graphe
T	Période d'observation du trafic
T_A	Temps d'attente
T_P	Temps de propagation
T_T	Durée de transmission
V_p	Vitesse de propagation
B_k	Constante de normalisation
N_s	Nombre de clients dans le système
T_s	Temps de séjour dans le système
N_a	Nombre de clients dans la file d'attente
T_a	Temps de séjour dans la file d'attente

3. Minuscule grecque

σ^2	Variance d'une variable aléatoire
γ	Fonction d'auto-covariance
λ	Taux d'arrivée de trafics
μ	Taux de service
ρ	Taux d'utilisation d'une file d'attente

4. Abréviations

ACL	Access Control List
AR	Auto Regressive
ARIMA	Auto-Regressive Integrated Moving Average
ARMA	Auto-Regressive Moving Average
BCMP	F.Baskett, K.M.Chandy,R.R Muntz et F.G Palacios
BP	Bande Passante
cwnd	Congestion window
DiffServ	Differentiated Services
DoD	Department of defense
DSCP	Differentiated Services Code Point
EGP	Exterior Gateway Protocol
EIGRP	Enhanced IGRP
FARIMA	Fractional Auto-Regressive Integrated Moving Average
FBM	Fractional Brownian Motion
FDN	Fractional Differencing Noise
FGN	Fractional Gaussian Noise
FIFO	First In First Out
FTP	File Transfer Protocol
GI	Générale Indépendante
i.i.d	Indépendant identiquement distribué
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IntServ	Integrated Services

IP	Internet Protocol
KPI	Key Performance Indicator
LIFO	Last In First Out
MA	Moving Average
MTU	Maximum Transmission Unit
NIC	Network Interface Card
OPNET	Optimum Network Performance
OSI	Open System Interconnection
OSPF	Open Shortest Path First
PM	Peak to Mean
PQ	Priority Queuing
PS	Processor Sharing
QoS	Quality Of Service
RED	Random Early Detection
RIP	Routing Information Protocol
RSVP	Resource ReSerVation Protocol
RTT	Round Trip Time
SJF	Shortest Job First
SRO	Service in Random Order
sssi	Self-similar stationary increments
TCP	Transmission Control Protocol
TOS	Type Of Service
UIT-T	Union Internationale des Télécommunications-Télécommunication
VoIP	Voice over IP
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Detection

INTRODUCTION GENERALE

La tendance actuelle dans le domaine des réseaux est d'implémenter la qualité de service dans l'Internet qui est aujourd'hui dominé par le modèle de service de type « best effort». De nouvelles demandes et applications en temps réel ont aussi augmenté depuis le développement de l'Internet ces dernières années.

Problématiques :

La majeure partie des entreprises, des individuels utilisant Internet souffre des problèmes de performances réseaux : temps de réponse applicatif détérioré, mauvaise qualité de communication audio, des services interrompus.

En effet, le transport des flux à contraintes temporelles fortes, comme ceux générés par la téléphonie ou la visioconférence, pose toujours problème. Ces applications requièrent du réseau des garanties de qualité de service que les protocoles classiques de l'Internet n'offrent pas.

Un des principaux facteurs de ces défaillances est la congestion au niveau des nœuds d'interconnexion tels que les routeurs.

Objectifs de recherche dans le mémoire:

Ce présent mémoire consiste alors à étudier plus en profondeur le constituant principal des routeurs qui est la file d'attente ainsi que son état en fonction de l'évolution des trafics qui la traverse. La mise en place des fonctions de la qualité de service (QoS) au niveau des files d'attentes est l'approche proposée afin de palier à ce problème de congestion. Trouver la meilleure solution parmi ces fonctions est l'objectif à atteindre.

Notre travail s'intitule alors « Analyse de performance des gestions de file d'attente et des mécanismes d'ordonnancement dans les réseaux IP » et il est divisé en quatre chapitres. Le premier chapitre est consacré aux concepts du télétrafic : les notions fondamentales du télétrafic, la théorie des files d'attente jusqu'aux différents modèles de trafic. Dans le deuxième chapitre, nous allons nous intéresser à la modélisation du trafic IP: nous allons parler des réseaux IP, nous allons déterminer les caractéristiques du trafic IP pour décrire ensuite les modélisations mathématiques qui lui correspondent. Dans le troisième chapitre, nous allons parler des performances réseaux: voir les indicateurs de performance, introduire la QoS et l'impact du routage sur la performance réseau. Le dernier chapitre est dédié à l'analyse des mécanismes de gestion de la QoS: présentation des problématiques, élaboration du principe de gestion de la QoS, étude des algorithmes de gestion de file d'attente (RED et WRED) et des mécanismes d'ordonnancement (FIFO, PQ, WFQ). Cette analyse sera illustrée à partir des divers scénarii de simulation sous l'outil OPNET Modeler.

CHAPITRE 1
CONCEPTS DU TELETRAFIC

1.1 Introduction

Le télétrafic est un outil utilisé et nécessaire pour le dimensionnement, la gestion et l'optimisation des réseaux. Dans ce premier chapitre, nous allons introduire les théories de base du télétrafic en passant par la notion de trafic, celle des files d'attente jusqu'aux réseaux de files d'attente. En dernier lieu nous allons voir d'autres modèles pouvant représenter les trafics.

1.2 Bases du télétrafic

1.2.1 Définition d'un trafic

Le trafic d'un réseau correspond au volume d'informations transportées ou traitées par ce réseau. Il pourra s'agir de données relatives aux échanges d'informations entre usagers (voix, images, e-mails, fichiers...), mais aussi des données relatives aux échanges d'informations entre machines de commande du réseau (données de signalisation dans un réseau de circuits, informations de routage dans un réseau IP, données d'exploitation…).
Plus les échanges entre usagers ou machines sont fréquents et de longues durées, plus les ressources nécessaires à l'écoulement de ce trafic seront importantes.

1.2.2 Classes de trafic

Il existe deux grandes classes de trafic : le trafic de type « streaming » et le trafic de type « élastique». [1]

1.2.2.1 Trafic de type « streaming »

Le trafic de type « streaming », appelé aussi trafic « temps réel », a une caractéristique de débit et de durée incompressibles. Il est associé à la notion de service orienté connexion avec une conservation de son intégrité temporelle. Le délai de transfert des données doit être contrôlable, mais les pertes de paquets à certain degré peuvent être tolérables. Les flux de trafic streaming sont typiquement produits par les services téléphoniques et vidéo.

1.2.2.2 Trafic de type élastique

Le trafic de type élastique est caractérisé par un débit qui peut s'adapter à des contraintes extérieures (bande passante insuffisante par exemple) sans pour autant remettre en cause la viabilité du service. Cette classe de trafic est essentiellement engendrée par le transfert d'objets numériques tels que les pages web, les courriers électroniques et les fichiers de données. Le respect de leur intégrité est indispensable mais les contraintes de délai de transfert sont moins fortes.

1.2.3 Différents niveaux de flux de trafic

On distingue plusieurs niveaux de flux de trafic représentés par la figure (1.01).[1]

1.2.3.1 Paquet

Les paquets forment l'entité de trafic la plus fine que l'on considère dans les réseaux de données. Ils constituent l'élément élémentaire traité par la couche réseau. Ils sont de longueur très variable et leur processus d'apparition est très complexe en raison de la superposition de services de nature très diverses.

1.2.3.2 Flot

Les flots constituent l'entité de trafic intermédiaire que l'on pense être la mieux adaptée pour effectuer les études d'ingénierie de trafic de données. Ils correspondent à des transferts plus ou moins continus de séries de paquets associés à une même instance d'une application donnée.

1.2.3.3 Session

Une session est une sorte de transposition de la notion d'appels considérés en téléphonie à commutation de circuits. Une session est une succession de flots correspondant à une certaine période d'activité d'un utilisateur.

Ces trois échelles de décomposition du trafic sont indispensables. Pour les applications temps réel, c'est le délai de traversée de paquets qui mesure la qualité de service et pour les applications à transfert de données, c'est le temps de transfert global de flots qu'on prend en considération. L'intégrité d'une session est aussi importante dans le cas où on souhaite que certaines sessions ne soient pas interrompues.

Il a été constaté qu'une analyse au niveau paquet s'avère plus complexe que celle au niveau des flots ou de sessions. [2]

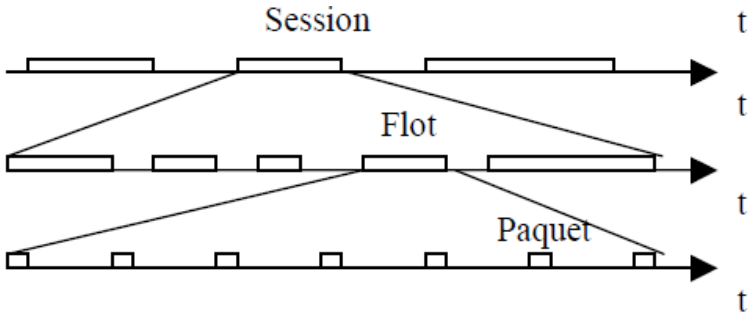


Figure 1.01 : *Décomposition de trafic*

1.3 Théorie de file d'attente

La théorie des files d'attente a pour objectif de considérer les phénomènes d'attente et d'engorgement liés au caractère aléatoire et imprévisible des événements rencontrés telles les requêtes d'établissement de connexions, les paquets traversant les routeurs.
Dans notre travail, cette théorie est d'une réelle importance puisqu'elle a pour finalité l'évaluation des performances des systèmes et des réseaux.

1.3.1 Définition d'une file d'attente

Une file d'attente est un système utilisé pour modéliser des phénomènes de partage de ressources. On y trouve les notions de clients et de serveurs : les clients sont les entités qui se déplacent dans un réseau de serveurs, où ils reçoivent un traitement. [3] Lorsque plusieurs clients tentent simultanément d'obtenir un service, certains doivent patienter et attendre dans des files d'attente, ou des tampons.
La file d'attente est caractérisée, d'après la figure (1.02), par :

- le mécanisme ou la loi d'arrivées des clients : c'est un processus aléatoire nommé « processus des arrivées » ou « processus des naissances » ;
- le temps de service ou la loi de la durée de service, c'est-à-dire la distribution de probabilité de sa durée ;
- la discipline de service qui détermine le type d'ordonnancement des clients de la file (c'est-à-dire la façon de choisir un client dans la file lorsqu'un serveur se libère).

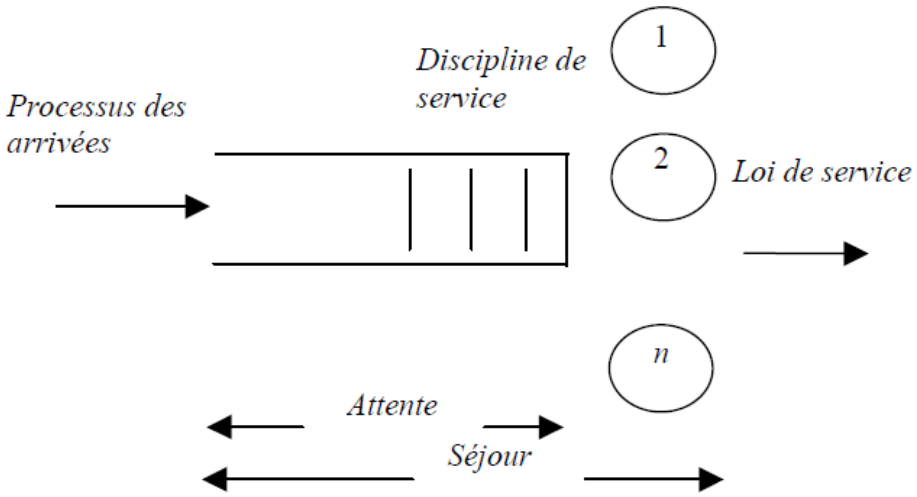


Figure 1.02 : *Caractérisation d'une file d'attente à n serveurs*

1.3.1.1 Processus des arrivées

La description de la loi d'arrivée des clients à l'entrée du système peut être faite en utilisant l'intervalle du temps entre arrivées successives ou bien le nombre des arrivées dans un intervalle donné.

Pendant la durée T, n(T) arrivées se produisent. On chiffre l'intensité du flux arrivant par le taux d'arrivée dont la définition intuitive est :

$$\lambda = \lim_{t\to\infty} \frac{n(T)}{T} \qquad (1.01)$$

Un des processus de naissances le plus utilisé est le processus de Poisson [ANNEXE 1] du fait que celui-ci est général et raisonnable pour de nombreux systèmes.

1.3.1.2 Processus des services

Le processus de service est complexe. Mais pour simplifier on suppose que chaque durée de service est indépendante des autres, et qu'elles obéissent toutes à une même loi de distribution. [3] On parle ainsi de variables indépendantes et identiquement distribuées (i.i.d.). On décrira cette loi, la loi de service, par sa distribution de probabilité :

$$B(x) = P\{temps\,de\,service \le x\} \qquad (1.02)$$

La loi de service la plus populaire est la loi exponentielle, il est traditionnel d'écrire en utilisant comme taux de service (ou paramètre) la lettre μ :

$$B(x) = P\{temps\,de\,service \leq x\} = 1 - e^{-\mu x} \qquad (1.03)$$

La loi exponentielle doit une bonne partie de son prestige à la propriété sans mémoire. L'application de cette absence de mémoire montre que la probabilité d'une fin de service dans l'instant qui vient (dans l'intervalle[t, t + dt [ ) est μdt, quel que soit l'âge du service.

La loi exponentielle est caractérisée par:

- la moyenne de la variable :

$$m = \frac{1}{\mu} \qquad (1.04)$$

- la variance de la variable :

$$\sigma^2 = \frac{1}{\mu^2} \qquad (1.05)$$

1.3.1.3 Processus de naissance et de mort

Il nous vient de considérer la notion *d'état du système* qui permet de donner la liste des caractéristiques qu'il possède, mais également de donner les éléments qui permettent de prévoir son évolution. Ainsi dans notre cas d'un système stochastique, la prévision du futur prend un caractère probabiliste : on ne saura prévoir l'état exact dans quelques secondes, mais seulement sa distribution de probabilité.

Le processus de naissance et de mort est basé sur le processus de Markov dans le cas où celui-ci ne peut faire de « sauts » que vers ses voisins les plus proches ; c'est-à-dire que l'état du système évolue depuis E_k vers E_{k-1} ou E_{k+1}. [Annexe 1] Le taux de passage de l'état E_k vers E_{k+1} est traditionnellement noté λ_k : c'est le taux de naissance, le taux de passage de l'état E_k vers E_{k-1} est le taux de mort noté μ_k comme le montre la figure (1.03)

Le problème à résoudre est de chercher à dériver la distribution de probabilité de l'état, c'est-à-dire la fonction $P_k(t) = P\,[X(t)= k]$, l'état initial étant supposé connu.

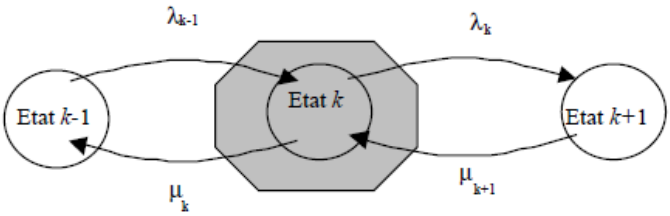


Figure 1.03 : *Evolution d'état d'un processus de naissance et de mort*

1.3.2 Notation de Kendall

Un système de files d'attentes est identifié par la notation de Kendall qui est un formalisme unanimement adopté notée A/B/n/K/N.

- A représente le processus des arrivées des clients et peut prendre les valeurs de conventions suivantes :
  - M (comme Markov) quand le processus suit une loi de Poisson,
  - D quand le processus suit une loi constante (arrivée Déterministe ou synchrone),
  - GI quand le processus suit une loi Générale Indépendante (processus de renouvellement),
  - G quand le processus suit une loi Générale, sans hypothèse d'indépendance.
- B représente le processus de service et est noté :
  - M (comme Markov) quand le processus suit une de loi exponentielle,
  - D quand le processus suit une loi constante (service déterministe),
  - GI quand le processus suit une loi générale indépendante,
  - G quand le processus suit une loi Générale, sans hypothèse d'indépendance.
- n représente le nombre de serveurs dans la file d'attente. Ce nombre entier varie entre 1 et l'infini.
- K représente la capacité mémoire du système d'attente, c'est donc le nombre maximum de clients pouvant être présents simultanément dans le système. Ce nombre entier varie entre 1 et l'infini. Quand la capacité mémoire de la file est considérée comme illimitée (infinie), ce paramètre est omis.
- N représente la discipline de service, par exemple :
  - FIFO : premier arrivé premier servi (appelée aussi First Come First Serve FCFS),
  - LIFO : dernier arrivé premier servi (appelée aussi Last Come First Serve LCFS),
  - SJF : Shortest Job First ou encore servir le travail le plus court d'abord,

- SRO : Service in Random Order ou encore servir en ordre aléatoire.

Ce paramètre est omis si la politique est FIFO.

1.3.3 Objectif du système de files d'attente

Dans une file d'attente, un paramètre de première importance est le taux d'utilisation, noté par ρ. C'est le produit du taux d'arrivée des clients par le temps de service:

$$\rho = \lambda \, \mathrm{E(s)} = \frac{\lambda}{\mu} \qquad (1.06)$$

Avec λ le taux d'arrivée, E(s) le temps de service et μ la vitesse de service.

Démonstration :

♣

Le taux d'utilisation est définie par :

$$\rho = \lambda \, \mathrm{E(s)}$$

Le temps de service E(s) est définie par :

$$\mathrm{E(s)} = \frac{1}{\mu}$$

Nous avons alors :

$$\rho = \frac{\lambda}{\mu}$$

♦

Pour une file de capacité infinie, il faut que $\rho<1$. C'est une condition nécessaire et suffisante de stabilité. Intuitivement, le travail doit entrer dans le système moins vite que celui-ci ne peut l'écouler.

Résoudre un système de files d'attente consiste, en fonction des paramètres du système (taux d'arrivée, taux de service), à écrire la distribution du nombre des clients dans le système et du temps d'attente. On peut ainsi trouver les deux résultats généraux suivants :

- le paramètre ρ est le taux d'activité ou la charge supportée, c'est-à-dire le trafic offert. La probabilité stationnaire que le serveur soit inactif est $P_0 = 1-\rho$.
- les nombres moyens et les temps d'attente sont reliés par la célèbre formule de Little :

$$\mathrm{E(N)} = \lambda \, \mathrm{E(W)} \qquad (1.07)$$

1.4 Réseaux de files d'attente

Dans le cas pratique, nous sommes amenés à étudier non plus une simple file d'attente mais un ensemble de files d'attentes interconnectés qu'on appelle réseaux de files d'attente. Nous allons alors voir deux types de réseaux à forme produit : les réseaux de Jackson et les réseaux BCMP. Un réseau à forme produit est tel que la distribution stationnaire est le produit des distributions stationnaires des files considérant en isolation.

1.4.1 Réseaux de Jackson

Le réseau de Jackson est introduit par J. R. Jackson. Il s'agit du premier développement significatif dans la théorie des réseaux de files d'attente à forme produit. Il appartient à la classe des réseaux Markoviens.

Un réseau de Jackson est un ensemble de N files simples (principalement sur les files d'attente M/M/.). A la file i, les clients arrivent suivant un processus de Poisson de paramètre λ_i et demandent un service de taux μ_i. Après le service à la file i, le client passe à la file j avec une probabilité p_{ij} ou quitte définitivement le réseau avec une probabilité $1-\Sigma\, p_{ij}$.

On distingue deux classes de réseaux de Jackson :

- les réseaux fermés qui ne communiquent pas avec le monde extérieur et contiennent un nombre fini de clients qui tournent en permanence dans le réseau ;
- les réseaux ouverts qui sont largement utilisés pour modéliser le routage. En effet, ces réseaux communiquent avec le monde extérieur, et peuvent potentiellement contenir un nombre illimité de clients.

Les réseaux de Jackson sont caractérisés par [4]:

- le nombre de nœuds du réseau :

  Si n est le nombre de nœuds d'un réseau de Jackson ouvert, le monde extérieur sera considéré comme le (n+1)ième nœud ;
- la description de chaque nœud :

  Chaque noeud i est une file d'attente avec un service exponentiel de moyenne $1/\mu_i$. Le noeud peut contenir m_i serveurs, $m_i = 1, 2, \cdots \infty$, qui seront supposés avoir une capacité de service identique ;
- la topologie du réseau en termes de règles de routage entre les nœuds :

  Elle est définie par une matrice qui donne pour chaque noeud i la probabilité p_{ij} de router une unité de donnée vers chaque noeud j du réseau.

Dans le cas d'un réseau ouvert, le routage vers l'extérieur du réseau sera effectué avec la probabilité p_{io}. Les règles de routage sont données par une matrice stochastique M c'est-à-dire une matrice où, pour un indice de ligne fixé, la somme des colonnes est égale à 1.

$$M = \begin{pmatrix} p_{11} & p_{12} & \cdots & \cdots & p_{1n} & p_{1o} \\ p_{21} & p_{22} & \cdots & \cdots & p_{2n} & p_{2o} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & \cdots & p_{nn} & p_{no} \end{pmatrix} \quad (1.08)$$

Avec :

$$p_{io} + \sum_{j=1}^{n} p_{ij} = 1; avec\ i = 1,...,n \quad (1.09)$$

- la statistique du trafic qui vient de l'extérieur du réseau (cas d'un réseau ouvert):

  Chaque noeud i peut recevoir un trafic extérieur suivant une loi de Poisson de paramètre λ_{oi}.

  Le réseau de files d'attente sera stable si et seulement si chacun des n nœuds du réseau l'est.

  Pour vérifier la stabilité d'un nœud, il est nécessaire de connaître le trafic total qui le traverse venant de l'extérieur (si le réseau est ouvert) et en transit à partir des autres nœuds.

Le trafic sur chaque nœud i (pour un réseau ouvert) est donné par :

$$\lambda_i = \lambda_{io} + \sum_{j=1}^{n} \lambda_{j\,ij}; avec\ i = 1,...,n \quad (1.10)$$

Pour un réseau fermé, λ_{io} est égal à 0. La condition de stabilité du nœud s'énonce comme :

$$\forall\, i = 1,...,n; \frac{\lambda_i}{m_i \mu_i} \prec 1 \quad (1.11)$$

Avec m=1,… si le nœud contient éventuellement plusieurs serveurs.

La distribution stationnaire du nombre de clients dans le système est explicitée dans le théorème de Jackson suivant.

Théorème : *Sous la condition de stabilité du réseau de file d'attente :* $\rho_i = \lambda_i/\mu_i < 1\ \forall i$,

- Pour les réseaux ouverts :

  La distribution de probabilité stationnaire du nombre de clients dans le réseau se factorise en un produit de probabilités marginales

$$P\{N_1 = k_1,..., N_n = k_n\} = P\{N_1 = k_1\}...P\{N_n = k_n\} \quad (1.12)$$

  Chaque probabilité marginales P $\{N_i=k_i\}$ est définie par :

$$P\{N_i = k_i\} = \frac{b_i \lambda_i^{k_i}}{\mu_i(1)...\mu_i(k_i)} \quad (1.13)$$

  Avec b_i une constante de normalisation satisfaisant la relation:

$$\frac{1}{b_i} = \sum_{k=0}^{\infty} \frac{\lambda_i^{k_i}}{\mu_i(1)...\mu_i(k_i)} \quad (1.14)$$

  tel que $\mu_i(l)$ est la capacité de service offerte par le noeud i quand le nombre de clients présents dans le noeud est l, le trafic entrant λ_i dans les nœuds i est donné par les équations de trafic de la formule (1.10).

- Pour les réseaux fermés :

  La distribution de probabilité stationnaire se met sous la forme :

$$P\{N_1 = k_1,..., N_n = k_n\} = B_K \frac{\lambda_i^{k_i}}{\mu_i(1)...\mu_i(k_i)} ... \frac{\lambda_n^{k_n}}{\mu_n(1)...\mu_n(k_n)} \quad (1.15)$$

  Où B_k est une constante de normalisation définie par :

$$B_K = \left[\sum_K \frac{\lambda_i^{k_i}}{\mu_i(1)...\mu_i(k_i)} ... \frac{\lambda_n^{k_n}}{\mu_n(1)...\mu_n(k_n)} \right]^{-1} \quad (1.16)$$

  Avec $K = k_1 + \cdots + k_n$ représentant le nombre fini de clients dans le réseau fermé.

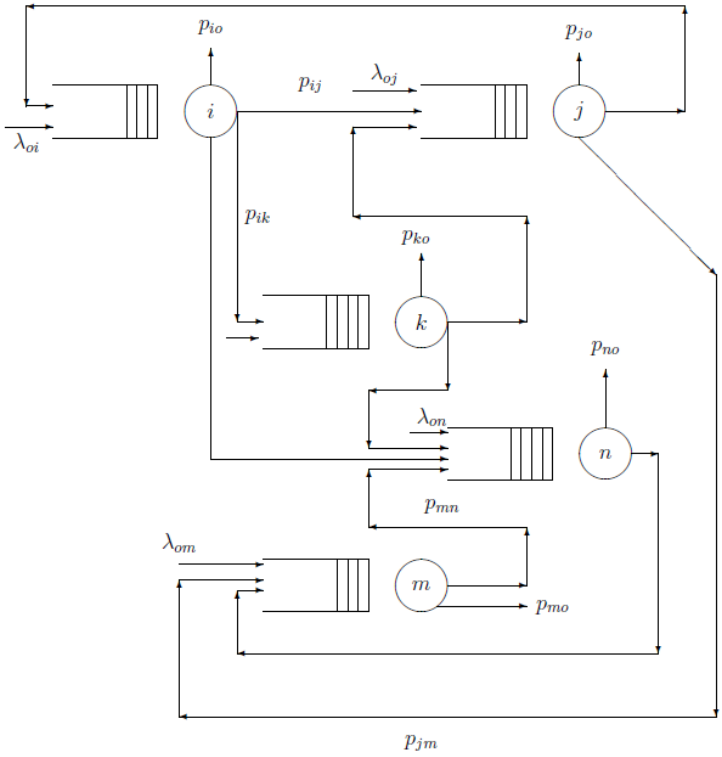


Figure 1.04 : *Réseaux de Jackson*

1.4.2 Réseaux BCMP

Les réseaux BCMP, issus des initiales des quatre mathématiciens F.Baskett, K.M.Chandy, R.R.Muntz et F.G.Palacios, sont les généralisations des réseaux de Jackson dont la distribution stationnaire a encore une forme produit mais en présentant différentes classes de client et de nouvelles disciplines de service. [5]

En effet, ces spécialistes des files d'attente ont déterminé les conditions pour ces réseaux BCMP :

- Il y a un nombre fini de classes k de paquets, chaque classe suit un service exponentiel de paramètres $\{\mu_i^j\}$; i = 1,…, n et j = 1,…, k pour un réseau de n nœuds et k classes ;
- Pour une seule classe de paquets on a les files d'attentes suivantes :
    - M/GI/1 avec la politique "processor sharing" ;
    - M/GI/∞
    - M/GI/m avec la politique LIFO et priorité préemptive resume

1.4.2.1 Politique de processor sharing

On appelle politique « Processor Sharing » (PS) une politique de partage du serveur entre les clients présents dans le système. Si x clients sont présents dans le système à un instant t, alors sur une

seconde de temps chaque client va recevoir un service de 1/x secondes. Les clients sont servis cycliquement. Ainsi, cette politique est du type ”partage du temps”, et le temps du processeur alloué à un client par unité de temps est variable en fonction du nombre de clients présents dans le système.

1.4.2.2 Politique LIFO et priorité « preemptive resume »

On appelle politique LIFO ”preemptive resume” une politique où la priorité est donnée au dernier client qui arrive. Le client le plus prioritaire va interrompre le client en cours de service. Mais le service déjà alloué à un client n’est pas perdu. Le service du client interrompu reprend au point où l’interruption a eu lieu.

1.5 Autres modèles de trafics

L’hypothèse de Poisson, qui constitue le processus de base du télétrafic, n’est plus bonne dès que le trafic devient sporadique. Il en est de même pour le cas des données (trafics IP) circulant sur Internet qui a beaucoup plus de dépendance. En effet, le trafic de données sur Internet est largement affecté par les protocoles utilisés dans différentes couches ainsi que par les dépendances entre les différentes couches. La description des paramètres simples, comme le temps inter-arrivée de paquets IP, est d'un usage limité et ne peut pas être considérée isolément pour les modèles.

D’autres recherches ont donc montré d’autres modèles mathématiques représentant au mieux ces trafics sporadiques dont voici quelques exemples.

1.5.1 Processus ON/OFF

Un processus ON/OFF X_t est un processus de Markov $\{X_t, t \in R_+\}$ à temps continue et à 2 états discrets 0 et 1. Le temps de séjour de la trajectoire X_t dans l’état 0 est distribué suivant une loi exponentielle de paramètre λ. Le temps de séjour de la trajectoire dans l’´état 1 est exponentiellement distribué de paramètre μ.

Une source ON/OFF est un générateur de trafic qui émet à débit constant pendant un temps aléatoire D1 et qui cesse toute activité pendant un temps aléatoire D2. La distribution de ces temps aléatoires suit des lois exponentielles de paramètres λ et μ. Le temps D1 est appelé la durée d’une ”rafale”. Ce processus modélise une source de voix à la sortie du décodeur.

Des études ont montré que la superposition de sources ON/OFF peut être utilisée pour représenter le trafic de données à faibles contraintes temporelles (TELNET, FTP, WEB…) entrant dans un multiplexeur statistique d’accès à un réseau à commutation de paquets. [4]

1.5.2 Processus MMPP

Un processus MMPP est un processus de Poisson modulé $\{N_{J_t}; t \in \mathbb{R}_+\}$ par un processus de Markov à états finis $\{J_t, ; t \in \mathbb{R}_+\}$ appelé processus de phase du MMPP.

Soit n le nombre de phases. Quand $J_t = i; i = 1, \dots, n$ le processus de Poisson N_i est d'intensité λ_i. Les sauts d'intensité du Poisson sont commandés par les sauts du processus de Markov J_t.

Soit $\{A_{(ij)},\ i,j \in \{1, \dots, n\}\}$ le générateur infinitésimal du processus J_t. Nous avons :

- A (i, i) = - q_i, où q_i est le paramètre de la durée exponentielle de séjour de la trajectoire J_t dans l'état i.
- A (i, j) = q_i Q (i, j) où Q (i, j) est la probabilité de transition de J_t entre la phase i à l'instant 0 et la phase j à l'instant T1.

  $Q(i,j) = P\{J_{T_1} = j | J_o = i\}$, T_1 est le premier instant de saut de J_t après t = 0.

Des études ont montré que ce modèle mathématique peut représenter la superposition de flux voix/données à l'entrée d'un multiplexeur statistique d'un réseau à intégration de services. [4]

1.6 Conclusion

Dans ce premier chapitre, nous nous sommes familiarisés avec les fondamentaux du télétrafic tels que les trafics qui circulent au travers d'un réseau, les théories de files d'attente pour la gestion de ces trafics et la mise en réseaux des files d'attentes. Nous y avons introduit un certain nombre de formules à connaître.

Dans le chapitre suivant, nous allons nous intéresser à la modélisation du trafic IP qui se différencie des trafics classiques.

CHAPITRE 2
MODELISATION DE TRAFIC IP

2.1 Introduction

La diversité des applications déployées sur Internet ainsi que la nature hétérogène des réseaux d'accès (filaire et sans fil) font évoluer les caractéristiques du trafic Internet.
Suite à la révélation de Paxon et Floyd de l'inadéquation du processus de Poisson pour la modélisation du trafic IP, des études ont mis en évidence l'existence de différents types de corrélations dans le trafic Internet et plus particulièrement des corrélations à long terme.
Dans ce chapitre nous allons donc étudier principalement les caractéristiques des trafics IP pour pouvoir mieux les modéliser.

2.2 Concepts de réseaux et Internet

2.2.1 Modèles d'architecture des réseaux

2.2.1.1 Modèle de référence OSI

L'OSI ou Open System Interconnection est un système modèle de référence pour interconnecter des systèmes ouverts. Il définit une architecture en couches normalisées adoptées conjointement par l'ISO et l'UIT-T pour les réseaux informatiques, téléinformatiques et télématiques. Cette architecture est constituée de sept couches dont la liaison réelle entre couches adjacentes se fait à partir des «services » et pour deux systèmes en communication, la relation logique entre les couches se fait à partir des « protocoles » comme le montra la figure (2.01) [6] [7]

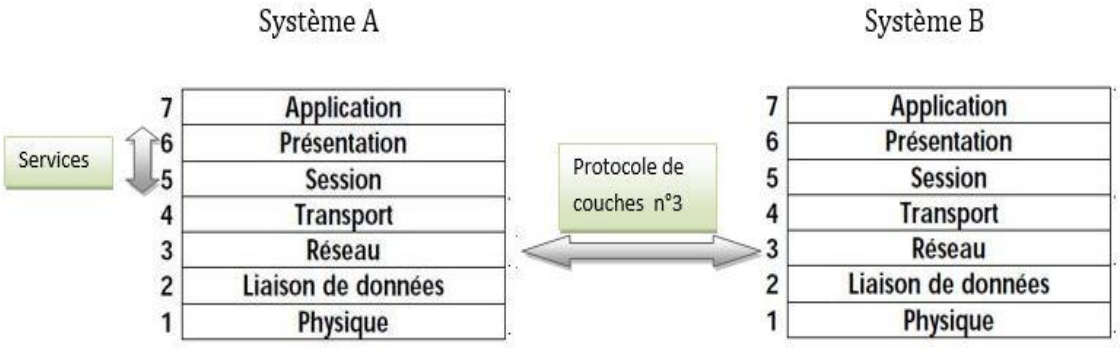


Figure 2.01 : *Liaisons entre deux couches adjacentes et deux couches de même niveau*

Le tableau (2.01) décrit cette architecture :

Tableau 2.01: *Modèle OSI*

Couche	Unité de données	Fonctions	Equipements utilisés
7-Application	Donnée	Services réseaux fournis aux processus d'application : synchronisation ; contrôle d'intégrité de données.	hôte
6-Présentation	Donnée	Représentation de données : Lisibilité, format, structure des données (utilisation de format commun).	hôte
5-Session	Donnée	Communication entre les hôtes : Etablissement, gestion et fermeture de sessions entre applications.	hôte
4-Transport	Segment	Communication de bout en bout : Transport des données, fiabilité du transport ; établissement, maintien et fermeture des circuits virtuels ; détection des pannes et reprise ; contrôle de flux d'informations.	Hôte, nuage (possibilité de se connecter à un autre réseau ou internet en entier)
3-Réseau	Paquet	Adressage et sélection du meilleur chemin : connectivité et sélection du meilleur chemin, domaine de routage	Routeur
2-Liaison de données	Trame	Accès au média : Transfert fiable de données par le média, adressage physique et topologie de réseau, notification des erreurs et contrôle de flux	Pont, commutateur, carte réseau(NIC)
1-Physique	Bits	Transmission binaire : Spécifications électriques et mécaniques pour maintenir la liaison physique des systèmes d'extrémité : fils, connecteurs, tension, débit	Emetteur-récepteur, câble (média réseau), répéteur, concentrateur

2.2.1.2 Modèle TCP/IP

Le ministère de la défense Américaine (Departement of defense ou DoD) a développé le modèle de référence TCP/IP dont le but d'avoir un réseau qui résiste à toutes les situations. Ce modèle est

inspiré du modèle OSI, il reprend l'approche modulaire (utilisation de modules ou couches) mais il est seulement constitué de quatre couches :

- Couche Application
- Couche Transport
- Couche Internet
- Couche Accès réseau

Depuis lors, ce modèle s'est imposé comme la norme Internet.

Le tableau (2.02) décrit l'architecture de ce modèle :

Tableau 2.02: *Modèle TCP/IP*

Couche	**Unités de données**	**Fonctions**
Application	Donnée	Gestion des protocoles de haut niveau, des questions de présentation, assure le code et le contrôle de dialogue.
Transport	Segment	Fiabilité des communications réseaux ; établissement d'un dialogue entre ordinateur source et ordinateur destination ; contrôle de flux et correction des erreurs.
Internet	Paquet	Identification de meilleur chemin pour l'envoi des paquets sources ; commutation de paquets.
Accès au réseau	Trame - bits	Etablissement de liaison physique pour un paquet physique. Elle comprend les détails dans les couches physiques et liaison de donnée du modèle OSI

2.2.2 Réseau à commutation de paquets

Une commutation de paquets est une commutation logique qui consiste à découper l'information initiale en plusieurs parties appelées paquets. Ces paquets vont ensuite transiter dans le réseau en empruntant des différents chemins selon la disponibilité des nœuds du réseau. Ces paquets sont libellés afin que le récepteur puisse les restituer. La taille des paquets dépend de la capacité d'acheminement des nœuds. Cette capacité est appelée MTU (Maximum Transmission Unit) ou Unité de transfert maximal. C'est donc la taille maximale d'un paquet pouvant être transmis en une seule fois sur une interface. La bande passante est partagée entre les différents trafics de façon permanente.

Internet est basé sur cette commutation de paquets.

2.3 Caractérisation du trafic IP

Le trafic IP se caractérise par de fréquents changements de phases (passives/actives) à différentes échelles de temps. Une modélisation adéquate avec les processus de Poisson et les distributions exponentielles n'est pas possible, puisque les variations conduiraient à de grandes phases actives ou passives en raison de diminution rapide de la probabilité. [8] En effet le processus de Poisson ne peut pas:

- couvrir tous les paramètres de trafic IP ;
- refléter la dynamique du trafic IP ;
- représenter la complexité multicouche dans le trafic IP

Le trafic IP présente un caractère auto-similaire. Pour traiter ce trafic, on utilise le processus de dépendance à long terme (ou à longue mémoire). Le phénomène d'auto-similarité implique fortement le processus à longue mémoire.

2.3.1 Auto-similarité

L'auto-similarité représente la propriété d'un processus d'être statistiquement proche de lui-même lorsqu'on le regarde à différentes échelles.

L'auto-similarité du trafic IP a été introduite du fait des résultats des observations effectués par des chercheurs telles que [8]:

- la variance et la corrélation de la quantité de trafic ne peuvent être expliquées par le modèle poissonnien.
- les différentes valeurs du trafic IP comme la quantité de données ou le temps inter-arrivées ne présentent aucun changement crucial sur différentes échelles.

2.3.1.1 Définition 2.01

Un processus $X_{t\epsilon I}$ I $\subset \mathbb{R}$ est appelé exactement auto similaire si, pour tous les choix de facteurs $a > 0$ et $b(a) > 0$, des vecteurs aléatoires

$$(X_{at_0},..., X_{at_n}) et\, b(a)(X_{at_0},..., X_{at_n}) \qquad (2.01)$$

coïncident sur la même distribution pour tous les temps $t_0,\ldots, t_n$.

En d'autres termes, un processus $X_{t\epsilon I}$ I $\subset \mathbb{R}$ est auto similaire s'il existe H > 0, tel que pour tout $a > 0$ et tous les temps $t_0, \ldots, t_n$, $(X_{at_0}, \ldots, X_{at_n})$ et $a^H (X_{at_0}, \ldots, X_{at_n})$ coïncident sur la même distribution pour tous les temps $t_0,\ldots, t_n$.

2.3.1.2 Définition 2.02

Soit (X_k), k ∈ ℕ un processus discret stationnaire. On appelle $X^{(m)}(i)$ un processus *ramené à une moyenne*, avec i=1,2,… et m=1,2,… défini par :

$$X^{(m)}(i) = \frac{1}{m}\left(X_{(i-1)m+1} + ... + X_{im}\right) \tag{2.02}$$

$\left(X^{(m)}(i)\right)_{i\in\mathbb{N}}$ est auto similaire s'il existe une constante H ∈] 0, 1[, produisant :

$$\lim_{m\to\infty} \frac{Var\left(X^{(m)}(k)\right)}{\sigma^2 m^{2H-2}} = 1, 0 \prec \sigma^2 \prec \infty \tag{2.03}$$

Le passage de X à $X^{(m)}$ correspond à un changement d'échelle. Un processus auto similaire a donc la même distribution, quelle que soit l'échelle de temps considéré.

2.3.1.3 Propriété 2.01

La valeur H est appelé paramètre de Hurst et H est unique. Alors, on appelle le processus $(X_t)_{t\epsilon I}$ un processus H-ss (ss pour self-similar).

2.3.1.4 Propriété 2.02

La relation (2.01) exprime l'invariance du processus dans toutes les échelles. Cela représente une égalité dans la distribution.

2.3.1.5 Propriété 2.03

L'autosimilarité et son extrême variabilité à toutes les échelles de temps sont une caractéristique du principe de la commutation de paquets qui induit des transmissions en rafales. Ce comportement se caractérise notamment par une décroissance lente (par exemple sous forme de loi de puissance, sous exponentielle ou à queue lourde) de la fonction d'autocorrélation du nombre de paquets transférés par unité de temps. Ce qui est différent de celui du modèle poissonnien qui est caractérisé par une baisse de fonction d'autocorrélation de manière exponentielle.

2.3.1.6 Propriété 2.04

Le processus H-ss avec accroissement stationnaire est appelé processus H-sssi ('sssi' pour self-similar stationary increments).

2.3.1.7 Propriété 2.05

Le processus d'auto-similarité est utile pour la simulation de X_t, puisque en connaissant la trajectoire sur [0,1], le facteur a^H est utilisé pour une échelle *at.*

2.3.1.8 Analyse du paramètre H

- Si H =1/2, alors le processus est à mémoire courte (ARMA : Auto-Regressive Moving Average ou autorégressif à moyenne mobile, i.i.d, processus Markoviens).
- La décroissance de la covariance est d'autant plus lente que H est proche de 1. On dira que la mémoire est d'autant plus longue que H est proche de 1.
- Si on mesure un H supérieur à 1, alors on sort du cadre des processus à longue mémoire.
- La valeur typiquement observée du paramètre Hurst sur le de trafic internet est située entre $\left]\frac{1}{2}, 1\right[$.

2.3.2 Sporadicité

Le deuxième caractère du trafic IP est la sporadicité (burstiness). Elle signifie qu'il existe des périodes concentrées d'activités importantes et des périodes de faibles activités.
Les critères utilisés sont le coefficient pic sur moyenne, le coefficient de variation ainsi que l'indice de dispersion. Dans notre cas, nous allons mesurer la sporadicité avec le ratio pic par moyenne (Peak to Mean ou PM) qui est obtenu par la division du pic de la bande passante par la moyenne de la bande passante.
La formule du PM est donnée par : [9]

$$PM = \frac{m}{n}\sum_{k=1}^{n/m}\frac{\max_{i=(k-1)m+1}^{km}(y(i))}{\sum_{j=(k-1)m+1}^{km}\frac{y(j)}{m}} \qquad (2.04)$$

Avec y(i) est une série temporelle, m le nombre de mesures considérées par intervalle d'étude et n est le nombre d'observations disponibles de la série y(i).

2.3.3 Dépendance à long terme

La dépendance à long terme est l'une des propriétés les plus importantes des processus auto-similaire. Elle est définie par le comportement de l'autocorrélation du processus.
Un processus aléatoire est dit à longue mémoire lorsque la probabilité d'un événement dépend aussi d'événements qui se sont produits relativement loin dans le passé, ce qui exclut toute modélisation usuelle fondée sur les processus markoviens classiques.

Nous allons nous intéresser aux processus stochastiques stationnaires au sens large, c'est à dire aux processus à covariance stationnaire.

2.3.3.1 Définition 2.03

Un processus stochastique $X = X_t$ est à covariance stationnaire si la moyenne et la variance existent et elles sont indépendantes du temps et aussi si l'auto covariance est indépendante par translation dans le temps, on a alors les relations suivantes :

$$E(X_t) = \mu \tag{2.05}$$

$$E((X_t - \mu)^2) = \sigma^2 \tag{2.06}$$

$$E((X_t - \mu)(X_{t+k} - \mu)) = \gamma \tag{2.07}$$

Avec μ représente l'espérance non conditionnelle du processus, σ est l'écart type du processus au temps t, et γ représente la fonction d'auto-covariance de retard k.
La fonction d'autocorrélation est donnée par:

$$\rho(k) = \frac{\text{cov}(X_t, X_{t+k})}{\sqrt{\text{var}(X_t)\,\text{var}(X_{t+k})}} \tag{2.08}$$

Comme notre processus est à covariance stationnaire alors la fonction d'auto corrélation ρ est de la forme :

$$\rho(k) = \frac{\gamma(k)}{\sigma} \tag{2.09}$$

De plus, un processus stationnaire présente un spectre continu avec une fonction de densité spectrale de puissance $S(w)$ pour tout $w \in [-\Pi, \Pi]$. Ce spectre est la série de fourrier de la fonction d'autocorrélation donné par :

$$S(w) = \frac{1}{2}\sum_{-\infty}^{\infty} e^{-jwk}\rho(k) \tag{2.10}$$

2.3.3.2 Définition 2.04

Soit X_t un processus caractérisé par sa densité spectrale. On dit que X_t est un processus à mémoire longue si :

$$S(w) \approx b|w|^{1-2H} \tag{2.11}$$

Avec $S(w)$ la densité spectrale de X_t et b une constante.

2.3.3.3 Définition 2.05

Pour qu'un processus stochastique $X = X_t$ (à covariance) stationnaire soit à mémoire longue, il doit vérifier les propriétés suivantes:

- $\sum_{k=1}^{\infty} \rho(k) = \infty$;
- la densité spectrale est singulière à l'origine.

Ces deux propriétés sont équivalentes.

2.3.3.4 Propriétés des processus à mémoire longue

Un processus stationnaire est dit à mémoire longue s'il possède une fonction d'autocorrélation qui décroît d'une manière hyperbolique et il doit présenter les propriétés suivantes :

- La fonction d'auto-corrélation est hyperboliquement décroissante :

$$\lim_{k\to\infty} \rho(k) \sim C_1 k^{-a}$$

- La somme des fonctions d'auto-corrélation est divergente:

$$\sum_{k=1}^{\infty} \rho(k) = \infty$$

- $E\left[X^{(m)}\right]$ ne possède pas un bruit de second ordre quand m tend vers ∞
- $Var\left[X^{(m)}\right]$ est asymptotiquement de la forme $m^{-\beta}$ quand m est grand
- $\sum_k Cov(X_n, X_{n+k})$ diverge
- La densité spectrale tend vers $C_2 w^{-(1-a)}$ quand w tend vers 0.

2.4 Modèles auto-similaire du trafic IP

Il existe des modèles appropriés pour des processus à longue mémoire:

- Les processus FARIMA ou Fractional Auto-Regressive Integrated Moving Average, ce sont des processus en temps discret. C'est une extension des processus ARIMA (Auto-Regressive Integrated Moving Average). Les processus ARIMA ou Auto Régressif Intégré à moyenne mobile caractérisés par leur simplicité et flexibilité sont devenus très répandus dans leur application dans l'analyse des séries de temps (séries chronologiques). [8]
- Les processus FBM ou Fractional Brownian Motion. Ce sont des processus dérivés du processus FGN ou Fractional Gaussian Noise. Ce sont des processus en temps continu.

2.4.1 Processus ARIMA

2.4.1.1 Définition 2.06

Un processus X_t est dit AR (Auto Régressif) s'il est obtenu par la somme d'une fonction linéaire de son propre retard $(X_{t-1}, X_{t-2}, \dots, X_{t-q})$ et d'un bruit aléatoire ε_t :

$$X_t = a_1 X_{t-1} + a_2 X_{t-2} + \dots + a_q X_{t-q} + \varepsilon_t \qquad (2.12)$$

2.4.1.2 Définition 2.07

Un modèle MA (Moving Average) ou à moyenne mobile est un modèle où X_t dépend des chocs aléatoires persistants. Un choc aléatoire est peut influencer plusieurs périodes :

$$X_t = b_1 \varepsilon_{t-1} + b_2 \varepsilon_{t-2} + \dots + b_p \varepsilon_{t-p} \qquad (2.13)$$

2.4.1.3 Définitions 2.08

- Soit B l'opérateur retard (Backshift) défini par :

$$BX_t = X_{t-1} \; et \; B^2 X_t = X_{t-2} \dots$$

- La différence peut être exprimée par :

$$X_t - X_{t-1} = (1 - B)X_t \; et \; (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = (1 - B)^2 X_t \dots$$

- Soient les paramètres p et q des entiers définis par :

$$\phi(x) = 1 - \sum_{j=1}^{p} \phi_j X^j \; et \; \psi(x) = 1 + \sum_{j=1}^{q} \psi_j X^j$$

- Soit $\{\varepsilon_t\}$ des variables normales identiquement et indépendamment distribuées (i.i.d) et qui forment un processus bruit blanc centré, de variance finie σ et d'espérance nulle.

2.4.1.4 Définition 2.09

Un modèle ARMA est la combinaison des deux modèles AR et MA. C'est donc un modèle plus complet qui prend en compte à la fois son propre retard et des chocs aléatoires persistants.
Le modèle ARMA de paramètres p et q est définie par l'équation :

$$\phi(B)X_t = \psi(B)\varepsilon_t \qquad (2.14)$$

Ce modèle est surtout utilisé dans le cas des processus à courte mémoire.

2.4.1.5 Définition 2.10

Pour les séries temporelles présentant des variations qui pourront violer l'hypothèse de la stationnarité, la stationnarisation par différenciation à l'ordre d peut être appliquée.
En effet, la différenciation est la méthode la plus utilisée pour rendre les données stationnaires.
Etant donné que la différenciation d'ordre d d'une série X_t est la série $(1-B)^d X_t$, nous pouvons ainsi effectuer plusieurs différenciations d'une façon répétée (d fois) jusqu'à ce que la série devienne stationnaire.
Une série temporelle X_t suit un processus Auto-Régressif à moyenne mobile Intégré ou ARIMA de paramètres p, d, q, si la série $(1-B)^d X_t$ suit un processus ARMA.
En d'autres termes, le modèle ARIMA (p, d, q) est définie par l'équation :

$$\phi(B)(1-B)^d X_t = \psi(B)\varepsilon_t \qquad (2.15)$$

2.4.2 Processus FARIMA

2.4.2.1 Définition 2.11

Le processus ARIMA fractionnaire (FARIMA) est la généralisation du modèle ARIMA (p, d, q) lorsque le degré de différenciation d peut avoir des valeurs réelles et non seulement des entiers naturels. [9]
Ainsi une série X_t suit un processus FARIMA (p, d, q) si celui-ci est défini par l'équation (2.15) avec $d \in \mathbb{R}$. Le paramètre d est l'indicateur de la force de la dépendance à long terme.
Dans le cas où p = 0 et q = 0, le processus FARIMA (0, d, 0) est appelé Fractional Differencing Noise model (FDN). C'est la forme la plus simple du processus FARIMA.

2.4.2.2 Analyse du paramètre d

Des études ont montré que suivant les valeurs du paramètre d on a [8] :

- Si $d \in]-\frac{1}{2}; \frac{1}{2}[$ alors X_t est appelé processus de FARIMA ;
- Si $d \in]0; \frac{1}{2}[$ alors X_t processus est à mémoire longue et a le comportement d'un modèle auto-similaire.
- Si $d > \frac{1}{2}$ alors le processus n'est plus stationnaire.

2.4.2.3 Relation entre les paramètres d et H

Le paramètre d du processus FARIMA (0, d, 0) indique le niveau de dépendance à long terme tout comme le paramètre H de Hurst.

Il a été démonté que pour $d \in [0; \frac{1}{2}[$, le processus FARIMA (0, d, 0) présente une dépendance à long terme avec un paramètre de Hurst tel que $H = d + \frac{1}{2}$ [8]

2.4.2.4 Synthèse

Le processus FARIMA (p, d, q) peut être vu comme la combinaison du processus ARMA avec le FARIMA (0, d, 0). Ainsi, il est capable de modéliser les processus présentant des dépendances à court et à long terme.

L'inconvénient de ce modèle est la complexité des calculs pour estimer les différents paramètres.

2.4.3 Processus de mouvements browniens fractionnaires FBM

2.4.3.1 Mouvement Brownien

Le mouvement brownien est considéré comme la mère de tous les processus [8]. Ce mouvement est aléatoire et le déplacement est en moyenne quadratique nul : il n'y a pas de mouvement d'ensemble, contrairement à un vent ou un courant. Il est caractérisé par la moyenne quadratique $\sqrt{< X^2 >}$ et non par la moyenne arithmétique des positions$< X >$.

Si x(t) est la distance de la particule à sa position de départ à l'instant t, alors :

$$< X^2(t) >= \frac{1}{t}\int_0^t x^2(\tau)\, d\tau \qquad (2.16)$$

Le mouvement brownien $(B_t)_{t\geq 0}$ est un processus stochastique dépendant du temps t et vérifiant les propriétés suivantes [8]:

- le processus $(B_t)_{t\geq 0}$ est un processus gaussien, c'est-à-dire pour tout temps $t_1 \leq t_2 \leq \cdots \leq t_n$, le vecteur $(B_{t_1}, B_{t_2}, \ldots, B_{t_n})$ est un vecteur gaussien ;
- $(B_t)_{t\geq 0}$ est presque surement continu, c'est-à-dire pour toute réalisation, la fonction $t \to B_t(w)$ est continue ;
- $\forall\, s,t$ la moyenne $E(B_t) = 0$ et la covariance $E(B_S, B_t) = \min(s,t)$

2.4.3.2 Mouvement Brownien Fractionnaire

Le processus mouvement Brownien fractionnaire (Fractional Brownian Motion FBM) a été défini par Mandelbrot et Ness en1968. [10]

Un processus $\{X(t); t \geq 0\}$ est un mouvement Brownien Fractionnaire si c'est un processus gaussien d'espérance nulle et vérifie :

$$Cov(X(s), X(t)) = \frac{1}{2}(t^{2H} + s^{2H} - |t-s|^{2H}) \qquad (2.17)$$

Avec $Cov(X(s), X(t))$ est la covariance du processus entre les instants s et t, et H est le paramètre de Hurst tel que $\frac{1}{2} < H < 1$.

La modélisation du trafic Internet par un FBM a été proposée par Norros en 1995 puis par Norros et Pruthi en 1996. Le trafic IP est alors modélisé par un processus $\{Y(t); t \geq 0\}$ défini par :

$$Y(t) = \mu t + \sqrt{\sigma^2 m} X(t) \qquad (2.18)$$

Le processus Y(t) possède trois paramètres : μ, σ^2 et H. Le paramètre μ représente le débit moyen du trafic, le paramètre σ^2 représente la variance du débit du trafic et X (t) est un processus FBM de paramètre H.

2.4.4 Modèle Bruit Fractionnaire Gaussien FGN

Soit $\{X(t); t \geq 0\}$ un mouvement Brownien fractionnaire. Le processus $\{Z(t)\}$ tel que $Z(t) = X(t+1) - X(t)$ est appelé bruit fractionnaire Gaussien de même paramètre de Hurst H que $\{X(t)\}$.

Dans ce cas, le trafic IP est modélisé par un processus $\{Y(t); t \geq 0\}$ défini par :

$$Y(t) = \mu + \sigma Z(t) \qquad (2.19)$$

Le processus Y(t) possède trois paramètres μ, σ^2 et H : le paramètre μ représente le débit moyen du trafic, le paramètre σ^2 représente la variance du débit du trafic et Z(t) est un processus FGN (Fractional Gaussian Noise) de paramètre H.

L'avantage des modèles FBM et FGN est la simplicité de leurs formules (2.18) et (2.19) pour prévoir le débit du trafic, une fois que les différents paramètres sont estimés. Cependant, leur inconvénient est leur incapacité de capturer les dépendances à court terme du trafic puisque $H > \frac{1}{2}$.

2.5 Conclusion

Ce chapitre nous a permis d'étudier en profondeur les aspects du trafic IP qui diffèrent des trafics classiques. Nous avons vu que c'est un processus auto-similaire, à longue dépendance et avec des apparitions sporadiques.

Ces caractéristiques limitent l'exploitation des processus de Poisson et Markovien pour modéliser le trafic IP.

De nouveaux modèles tels que les processus ARIMA, FARIMA, FBM et FGN peuvent alors le représenter au mieux.

Dans le chapitre suivant, nous allons nous intéresser aux performances des réseaux ainsi que de tous les aspects qui peuvent avoir des impacts sur ces performances dans le but de mieux les gérer et les maintenir.

CHAPITRE 3
PERFORMANCES RESEAUX

3.1 Introduction

La notion de performance est toujours liée à la qualité de service. Celle-ci est relative à la perception qu'a l'usager de la réponse du réseau à sa demande. Il ne faut jamais perdre de vue cette finalité, car c'est elle qui guide l'efficacité économique, et non pas la performance pour la performance. [3]
Cependant, l'atteinte de cet objectif ne peut être que complexe compte tenu à la fois de la diversité des requêtes, de la diversité des équipements mis en jeux et de la complexité des réseaux utilisés.
La qualité de service vue de l'usager sera en fait le résultat d'un ensemble cohérent de performances de tous les éléments de réseaux.
Pour pouvoir évaluer les performances du réseau, plusieurs paramètres sont requis. C'est l'essence même de ce chapitre.

3.2 Qualité de service

La QoS (Quality of Service) ou Qualité de service est définie dans la recommandation E-800 de l'UIT par un effet global produit par la qualité de fonctionnement d'un service qui détermine le degré de satisfaction de l'usager d'un service. [11]
La QoS désigne donc la capacité d'un réseau à fournir un service dans une condition satisfaisante. C'est un concept de gestion qui permet de garantir de bonnes performances aux applications critiques et qui a pour but d'optimiser les ressources d'un réseau.
Une autre définition énonce aussi que la qualité de service désigne la capacité d'un réseau à fournir un service préférentiel au trafic réseau sélectionné. L'objectif premier de la QoS est de fournir une priorité, y compris une bande passante dédiée, un contrôle de la gigue et de la latence, ainsi qu'une réduction de la perte de paquets. [12]
Les utilisateurs perçoivent la qualité de service en fonction de deux critères : la vitesse à laquelle le réseau réagit à leurs requêtes et la disponibilité des applications qu'ils souhaitent utiliser.
Dans un réseau sans QoS, tous les paquets reçoivent le même traitement et les applications temps réel s'en ressentent. En effet, certaines applications sont extrêmement sensibles aux besoins en matière de bande passante, aux retards de paquets, à la gigue du réseau et à la perte éventuelle de paquets, notamment la téléphonie sur IP en temps réel et la lecture vidéo en continu.

La QoS ne crée pas réellement plus de bande passante. Elle définit plutôt les priorités d'utilisation de la bande passante pour la prise en charge des applications qui en ont le plus besoin.

3.3 Indicateurs de performances ou KPI

Les KPI, pour Key Performance Indicators, ou indicateurs clés de performance, sont utilisés pour déterminer les facteurs pris en compte pour évaluer l'efficacité d'un réseau. Ces indicateurs permettent donc de mesurer l'efficacité de certains paramètres et servent pour le diagnostic et la supervision de réseau.

Il existe plusieurs indicateurs de performances réseaux dont les cinq majeurs sont la bande passante, la latence, la gigue, le débit et la perte de paquets. [13]

3.3.1 La bande passante

La bande passante est le volume maximal de données pouvant transiter sur un chemin de communication au cours d'une période donnée. Elle est exprimée en bits/s (Kbits/s, Mbits/s).

Cette valeur reste cependant théorique à cause de différents facteurs comme les unités d'interconnexions de réseaux ; les autres utilisateurs utilisant le réseau, leur nombre, leurs ordinateurs, le type de données transmises, etc.

3.3.2 La latence du réseau (Latency)

C'est le temps nécessaire pour véhiculer un paquet au travers d'un réseau, c'est-à-dire entre l'émission et la réception.

La latence peut être mesurée de plusieurs façons : la latence d'aller-retour (ou Round Trip Time RTT), la latence unilatérale (One way). Elle est exprimée en seconde (s, ms, µs, …).

La latence est impactée par tout élément dans la chaîne utilisée pour transporter les données : station cliente, liens WAN, routeurs, réseau local LAN, serveurs ; et pour les très grands réseaux, elle est limitée par la vitesse de la lumière.

$$\text{Latence} = T_T + T_P + T_A \tag{3.01}$$

Avec :

- T_T la durée de transmission : c'est le temps nécessaire pour transmettre les données (les envoyer sur le réseau)

$$T_T = \frac{TailleMessage}{Débit} \tag{3.02}$$

- T_P le temps de propagation: c'est le temps nécessaire pour que les données aillent de l'émetteur au récepteur

$$T_P = \frac{d}{V_P} \tag{3.03}$$

Avec d la distance entre l'émetteur et le récepteur et V_P la vitesse de propagation

- T_A le temps d'attente: c'est le temps "perdu" par le système de communication (notamment à cause de l'occupation des ressources)

La latence de base est un délai incompressible correspondant au temps écoulé avant de recevoir le premier bit d'un message, elle est définie par :

$$\text{Latence de Base} = T_P + T_A \tag{3.04}$$

On peut aussi la mesurer à l'aide de la commande « ping » au niveau d'un nœud tel qu'un routeur.

3.3.3 La gigue (Jitter)

C'est la variation de la latence. Les paquets arrivent de manière irrégulière en fonction du trafic réseau. Elle est donc déterminante dans le cas des services en temps réel tels que la VoIP. Plus la gigue augmente plus la conversation devient hachée.

3.3.4 Le débit réseau (Throughput)

C'est la quantité de données réellement envoyées et reçues par unité de temps. Il définit le taux de transfert des données obtenu en combinant les effets de bande passante et de latence.

Pour simplifier, la bande passante est ce que nous payons et le débit est ce dont nous disposons réellement. Il est exprimé en bits/s (Kbits/s, Mbits/s…).

3.3.5 La perte de paquets

Il s'agit du nombre de paquets perdus par rapport à 100 paquets émis par un hôte sur le réseau. Elle correspond donc au taux de perte qui oblige à la retransmission des données. Les pertes de paquet affectent considérablement la qualité du lien réseau.

Ces pertes dépendent de :

- la fiabilité du support des liaisons : un support peu fiable entraîne une corruption fréquente des paquets. Si un paquet s'avère corrompu à l'issue de sa vérification grâce aux bits de contrôle (checksum), il sera détruit. On peut toutefois estimer qu'actuellement la fiabilité

des liens dans le réseau Internet est relativement élevée, et que par conséquent, les pertes de paquets dues à des défauts de support sont très faibles.

- l'occurrence de surcharges locales (congestions) dans le réseau : les paquets devant être placés dans une file d'attente pleine sont détruits.

Lorsque le protocole de transport garantit l'arrivée de l'information transmise, les paquets perdus doivent être renvoyés. Les pertes ont donc une influence directe sur l'augmentation du délai et sur la gigue.

Par ailleurs, il existe d'autres facteurs qui peuvent affecter les performances du réseau [14] :

- Utilisation : l'utilisation est le pourcentage de durée occupé par le câble et comprend une retransmission en échec de la trame (collisions et erreurs). Le terme câble occupé peut également être utilisé. En règle générale, la durée d'inactivité et d'utilisation donnent 100%
- Instabilité : l'instabilité correspond aux retards variables sur un réseau.
- Jabotage : c'est est un flux continu de données aléatoires transmises sur un réseau en raison d'un dysfonctionnement.
- Goulot d'étranglement : un goulot d'étranglement est un retard survenant lorsqu'une partie d'un réseau est plus lente que les autres et entrave donc le débit général.
- Collisions : les collisions sont des trames dont l'envoi a échoué sur un support partagé car les expéditeurs ont tenté d'envoyer plusieurs trames simultanément.

3.3.6 Valeurs des indicateurs de performances

Le tableau (3.01) donne les valeurs à respecter pour considérer un réseau de qualité :

Tableau 3.01: *Valeurs des indicateurs de performance*

Réseau	Perte de paquets	Latence	Gigue
Sur un réseau local	< 0.5 %	< 10 ms	< 5 ms
Sur un réseau WAN	< 1 %	< 40 ms	< 10 ms
Internet ou VPN sur Internet	< 2 %	< 100 ms	< 30 ms

3.4 Impact du routage IP sur la performance réseau

Le routage IP constitue l'un des aspects fondamentaux d'un réseau. En effet, pour assurer la fiabilité de la communication grâce à ses multiples interconnexions, le réseau nécessite des capacités de

routage dynamique pour contourner les pannes et les encombrements. Ainsi, il influe considérablement au bon fonctionnement du réseau.

Dans cette partie, nous allons étudier les différents algorithmes utilisés par les protocoles de routage permettant de déterminer le plus court chemin lors de la commutation des paquets vers la destination.

3.4.1 Concepts de graphes valués

Un graphe (orienté ou non) G = (S, A) est valué si il est muni d'une application v qui sera appelée valuation :

$$v : A \rightarrow R \qquad (x, y) \rightarrow v\,(x, y) \tag{3.05}$$

On peut étendre la valuation en une fonction $S \times S \rightarrow R \cup \{+\infty\}$ en posant $v(x, y) = +\infty$ si $(x, y) \notin A$.

La figure (3.01) illustre des exemples de graphes valués.

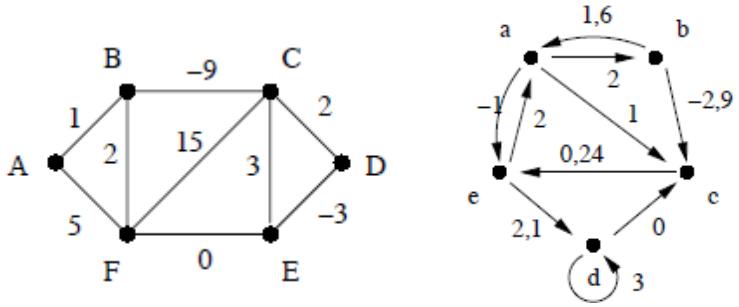
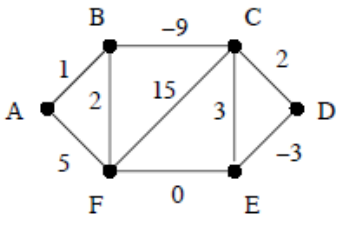


Figure 3.01 : *Exemples de graphes non orienté et orienté valués*

On peut représenter un graphe valué par une matrice carrée, dont les coefficients correspondent à la valuation des arcs (Figure 3.02).

Soit G = (S, A, v) un graphe valué dont on a numéroté les sommets de 1 à n. La matrice de valuation de G est la matrice carrée $M = (m_{ij})$, de taille $n \times n$, définie par :

$$m_{ij} = \begin{cases} v(I, j) & \text{si } (I, j) \in A \\ +\infty & \text{sinon} \end{cases} \tag{3.06}$$



$$\begin{pmatrix} \infty & 1 & \infty & \infty & \infty & 5 \\ 1 & \infty & -9 & \infty & \infty & 2 \\ \infty & -9 & \infty & 2 & 3 & 15 \\ \infty & \infty & 2 & \infty & -3 & \infty \\ \infty & \infty & 3 & -3 & \infty & 0 \\ 5 & 2 & 15 & \infty & 0 & \infty \end{pmatrix}$$

(a) (b)

Figure 3.02 : *(a) graphe valué, (b) représentation matricielle du graphe*

3.4.1.1 Valuation d'un chemin

Soit G = (S, A, v) un graphe valué. La valuation ou longueur d'un chemin (ou d'une chaîne) est la somme des valuations de chacun des arcs qui le composent.
La valuation d'un chemin ne comportant pas d'arcs est égale à 0. [15]
Considérons l'exemple de la figure 3.02 (a)
La valuation de la chaîne (A, F, C, E, D) est 5 + 15 + 3 – 3 = 20

3.4.1.2 Distance et plus court chemin

Soit G = (S, A, v) un graphe valué et soient x, y deux sommets de G.

- On appelle distance de x à y notée d(x, y) le minimum des valuations des chemins ou chaînes allant de x à y (Figure 3.03)
- On appelle plus court chemin ou plus courte chaîne de x à y tout chemin ou chaîne dont la valuation est égale à d(x, y).

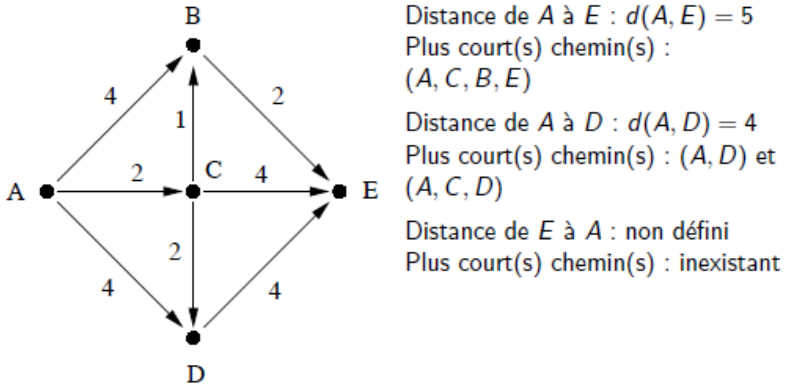


Figure 3.03 : *Exemple illustrant la distance et le plus court chemin*

Ces notions nous amènent alors à modéliser le routage dans les réseaux de télécommunication comme des recherches de plus courts chemins dans des graphes valués.
Nous avons les correspondances représentées sur le tableau 3.02 :

Tableau 3.02: *Correspondance entre graphe et réseau*

Graphe	Réseau de communication
Sommets	Routeurs
Arcs	Lignes de communication
Longueurs	Délais

On étudiera principalement donc des algorithmes qui résolvent le problème suivant : étant donné un sommet x, déterminer pour chaque sommet y la distance et un plus court chemin de x à y.

3.4.1.3 Circuit absorbant

Un circuit absorbant est un circuit de valuation négative. Si un graphe possède un circuit absorbant, alors il n'existe pas de plus courts chemins entre certains de ces sommets.
On définit de la même manière un cycle absorbant dans un graphe non orienté. Le théorème reste vrai en remplaçant chemin par chaine.
Pour la suite, on considère donc que le graphe ne possède pas de circuits absorbants.

3.4.1.4 Principe des algorithmes dans le cas général

Etant données un graphe valué G = (S, A, v) et un sommet x_0, on veut déterminer, pour chaque sommet s, la distance et un plus court chemin de x_0 à s.
Les algorithmes de recherche de distance et de plus court chemin dans un graphe valué fonctionnent de la façon suivante :

- On calcule les distances d (x_0, s) par approximations successives. A un stade donné de l'algorithme on dispose d'estimations d(s) (éventuellement égales à $+\infty$) pour ces distances, et de la donnée d'un prédécesseur P(s) pour les plus courts chemins.
- A chaque étape, on considère un sommet x et un successeur y de x. On compare la valeur d(y) à celle que l'on obtiendrait en passant par x, c'est-à-dire d(x) + v(x, y).

  - Si cette deuxième valeur est plus petite que d(y), on remplace l'estimation d(y) par : d(x) + v(x, y) et le père P(y) par x. [15]

3.4.2 Algorithme de Bellman-Ford

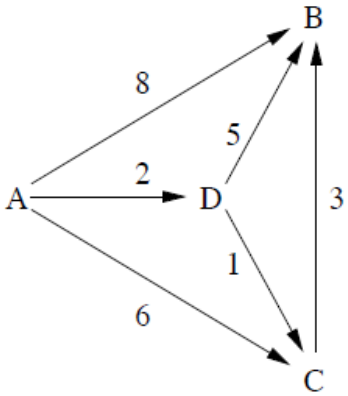
On applique le principe précédent en explorant systématiquement tous les sommets et tous leurs successeurs et on s'arrête quand les valeurs des distances sont stabilisées. C'est algorithme de Bellman-Ford [15].

On a :

Initialisation :
$d(x_0) = 0$, $P(x_0) = nul$
pour tout $s \in S$, $s \neq x_0$ **répéter**
    $d(s) = +\infty$, $P(s) = nul$
$fin = FAUX$

tant que $fin = FAUX$ **répéter**
    $fin \leftarrow VRAI$
    pour tout $x \in S$ **répéter**
        pour tout $y \in G(x)$ **répéter**
            si $d(x) + v(x, y) < d(y)$ **alors**
                $d(y) \leftarrow d(x) + v(x, y)$ (màj distance)
                $P(y) \leftarrow x$ (màj père)
                $fin \leftarrow FAUX$ (pas encore stabilisé)
fin tant que

La figure 3.04 montre l'exemple d'application de l'algorithme de Bellman-Ford.



Itération	$d(A)$	$d(B)$	$d(C)$	$d(D)$	$P(A)$	$P(B)$	$P(C)$	$P(D$
Initialisation	0	$+\infty$	$+\infty$	$+\infty$	nul	nul	nul	nul
1	0	7	3	2	nul	D	D	A
2	0	6	3	2	nul	C	D	A
3	0	6	3	2	nul	C	D	A

Figure 3.04 : *Exemple d'application de l'algorithme de Bellman-Ford*

Les valeurs se stabilisent après, au plus, n passages dans la boucle principale avec n étant le nombre de sommets du graphe.

Les valeurs d(s) obtenues à la fin de l'algorithme sont bien les distances de x_0 aux sommets s.

Pour trouver un plus court chemin entre x_0 et s, on se sert du tableau des pères. On construit le chemin à partir de la fin : on part de s, le tableau des pères donne son prédécesseur p le long d'un plus court chemin, et on recommence avec p.

En considérant l'exemple précédent on a :

$P(A)$	$P(B)$	$P(C)$	$P(D)$
nul	C	D	A

Un plus court chemin de A à B est
A→D→C→B

3.4.3 Algorithme de Dijkstra

L'algorithme de Dijkstra est un autre algorithme de recherche de distance et de plus court chemin. Il est plus efficace que Bellman-Ford, mais ne fonctionne que dans le cas où toutes les valuations des arcs sont positives. [15]

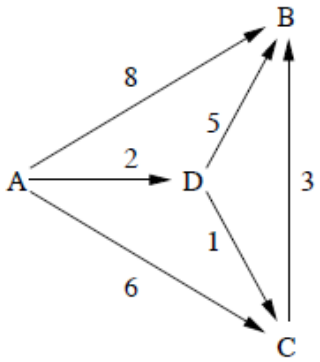
Son principe est le suivant :

- On construit petit à petit, à partir de $\{x_0\}$, un ensemble M de sommets marqués. Pour tout sommet marqué s, l'estimation d(s) est égale à la distance d (x_0, s).
- A chaque étape, on sélectionne le sommet non marqué x dont la distance estimée d(x) est la plus petite parmi tous les sommets non marqué.
- On marque alors x (on rajoute x à M), puis on met à jour à partir de x les distances estimées des successeurs non marqués de x.
- On recommence, jusqu'à épuisement des sommets non marqués.

Initialisation
$d(x_0) = 0$, $P(x_0) = nul$
aucun sommet n'est marqué
$min\_dist\_M = 0$ (minimum des distances estimées des sommets non marqués)
pour tout $s \in S$, $s \neq x_0$ **répéter**
    $d(s) = +\infty$, $P(s) = nul$

répéter
    chercher x non marqué tel que $d(x) = min\_dist\_M$
    marquer x
    pour tout $y \in G(x)$, y non marqué **répéter**
        si $d(x) + v(x,y) < d(y)$ **alors**
            $d(y) \leftarrow d(x) + v(x,y)$ (màj distance)
            $P(y) \leftarrow x$ (màj père)
    $min\_dist\_M = \min\{d(s),\ s \notin M\}$
jusqu'à ce que $min\_dist\_M = +\infty$

Un exemple illustrant l'application de l'algorithme de Dijkstra est représenté à la figure 3.05.



M	$d(A)$	$d(B)$	$d(C)$	$d(D)$	$P(A)$	$P(B)$	$P(C)$	$P(D$
$\emptyset$	0	$+\infty$	$+\infty$	$+\infty$	nul	nul	nul	nul
$\{A\}$		8	6	2		A	A	A
$\{A, D\}$		7	3			D	D	
$\{A, D, C\}$		6				C		
$\{A, D, C, B\}$	0	6	3	2	nul	C	D	A

Figure 3.05 : *Exemple d'application de l'algorithme de Dijkstra*

Quand on connaît les distances de x_0 à s pour tout sommet s, on peut déterminer les plus courts chemins de x_0 à s pour tout sommet s. Si on dispose du tableau des pères, il est facile de trouver un plus court chemin. Pour l'exemple de la figure 3.05, on a :

s	A	B	C	D
$d(A, s)$	0	6	3	2

Un plus court chemin de A à B est :
$A \rightarrow D \rightarrow C \rightarrow$B

3.5 Assurance de la QoS dans les réseaux

Plusieurs recherches ont été faites pour résoudre les problèmes de la qualité de service dans les réseaux. Plusieurs techniques ont été proposées par la suite. Dans ce paragraphe, nous allons nous intéresser à quelques-unes qui sont le contrôle de flux, le contrôle de débit, la réservation de ressources et la différenciation de services.

3.5.1 Concept de contrôle de flux

Il arrive souvent dans les réseaux que le trafic instantané entrant soit supérieur à la capacité de traitement disponible. Ceci est particulièrement visible dans les réseaux de transmission de données à commutation de paquets, où les données arrivent à des instants aléatoires.

De plus, dans un réseau à commutation par paquet offrant un service orienté connexion, bien que le réseau réserve des ressources au moment de l'admission de l'appel, il y a toujours un risque de dépassement de capacité (congestion) à cause de la nature variable du trafic (sporadicité) et du multiplexage statistique de ces flux. C'est là qu'intervient alors le contrôle de flux.

3.5.1.1 Définition de la congestion

Les performances réseaux se dégradent lorsque le trafic entrant augmente : diminution du débit utile et augmentation du temps de transit. En effet, quand un paquet arrive dans un routeur, il est placé dans une file d'attente. Si le débit des paquets en entrée du routeur est plus grand que le débit de sortie, le nombre de paquets dans la file d'attente augmente. Cependant la taille des files est limitée, une congestion survient lorsque la file est pleine. Les paquets devant être placés dans la file pleine sont alors détruits.

En d'autres termes, on est face à une congestion lorsque le trafic entrant entraîne la diminution du débit utile du réseau. Ce phénomène est en rapport avec la saturation des ressources qui ne peut être évitée qu'en diminuant le débit des sources.

Le contrôle de flux est ainsi nécessaire afin de limiter l'apparition de congestion et sa durée. La fonction du contrôle de flux est de donc de réguler le trafic entrant, à savoir le ralentir dès l'apparition de la surcharge et ensuite permettre l'écoulement total lorsque le réseau retourne à son fonctionnement normal.

Les travaux de Van Jacobson se sont intéressés sur ce sujet et ont permis à de nombreuses techniques de naitre afin de contrôler les débits des sources et ainsi résoudre les problèmes de congestion. [16]

3.5.1.2 Démarrage lent (Slow Start)

Dans la couche de transport, le protocole TCP permet d'envoyer dès le début de la transmission, un nombre de paquets correspondant à la taille de sa fenêtre d'émission. La taille de cette fenêtre est déterminée à l'issue d'une négociation entre la source et la destination des données. Cela peut être la cause de congestions dans le réseau. Van Jacobson propose donc d'utiliser une fenêtre supplémentaire, la fenêtre de congestion. Lorsqu'une nouvelle connexion est établie, la taille de cette fenêtre (cwnd) est initialisée à un segment. Chaque fois qu'un acquittement est reçu par la source, la taille de la fenêtre de congestion est incrémentée d'un segment. Le nombre de segments transmis (taille de la fenêtre d'émission), est borné par le minimum de la taille de la fenêtre de réception de la destination et de la fenêtre de congestion. Ainsi, la source commence par envoyer

un segment, et attend son acquittement. Lorsque l'acquittement est reçu, la fenêtre de congestion est incrémentée d'un segment. La source émet donc deux segments. Pour chacun des deux acquittements reçus pour ces segments, la fenêtre de congestion est à nouveau incrémentée d'un segment, c'est-à-dire passe à 4 segments, et ainsi de suite.
La taille de la fenêtre de congestion augmente ainsi exponentiellement. Lorsqu'un paquet est perdu, cela signifie que la fenêtre de congestion est devenue trop grande, on peut alors faire appel à la technique de l'évitement de congestion.

3.5.1.3 Evitement de congestion (Congestion Avoidance)

La source détecte qu'un paquet a été perdu lorsqu'elle reçoit des acquittements dupliqués ou bien si elle ne reçoit pas l'acquittement d'un paquet après une période donnée (time-out). Le nombre de paquets corrompus à cause d'un défaut de liaison est très faible. On peut considérer que la perte d'un paquet est le signe d'une congestion dans le réseau. Dans ce cas, l'évitement de congestion permet de réduire le débit de la source, en réduisant la taille de sa fenêtre. Pour cela, on applique un seuil au démarrage lent (Slow Start Threshold, ssthresh).
Lorsqu'une perte de paquet est détectée, on affecte à ssthresh une valeur égale à la moitié de la taille de la fenêtre d'émission, et on réduit la taille de la fenêtre de congestion (cwnd) comme suit:

- Si la perte de paquet a été détectée grâce à des acquittements dupliqués, cwnd=ssthresh;
- Si la perte de paquet a été détectée après un time-out, cwnd=1.

Ensuite, l'augmentation de la fenêtre de congestion peut être de deux natures :

- Si cwnd < ssthresh, on applique le démarrage lent. Par conséquent, la taille de la fenêtre de congestion augmente exponentiellement ;
- Si cwnd>ssthresh, on applique l'évitement de congestion. A la réception de chaque acquittement, on augmente cwnd d'une taille fixée. La taille de la fenêtre de congestion augmente donc linéairement.

Cependant, ces deux techniques ne permettent la réduction du débit des sources que lorsqu'une congestion a débuté. L'apparition de congestions n'est donc pas écartée. C'est pourquoi des algorithmes permettant la prévention des congestions, tels que RED et WRED, ont été proposés.

3.5.1.4 RED (Random Early Detection) - WRED (Weighted Random Early Detection)

RED est une technique proposée par Sally Floyd et Van Jacobson en 1993. C'est une technique préventive permettant d'éviter les congestions. Son principe consiste à détruire arbitrairement des

paquets dans les files d'attente, lorsque celles-ci sont remplies au-delà d'un certain seuil. Plus la file se remplit, plus le nombre de flux concernés par la destruction arbitraire de paquets est grand. Grâce au mécanisme d'évitement de congestion, les flux transportés par les paquets détruits voient leur débit réduit.

RED est équitable, puisque les flux qui subissent des destructions de paquets sont choisis arbitrairement. Au contraire, pour privilégier certains flux par rapport à d'autres, on peut appliquer différentes probabilités de destruction. C'est le but des techniques Weighted RED et Enhanced RED. Pour ces derniers, les paquets des flux privilégiés (plus prioritaires) sont moins exposés à la destruction arbitraire. WRED distingue la priorité des paquets grâce au champ de priorité présent dans chaque en-tête.

Ces techniques de régulation des sources présentées ci-dessus s'appliquent aux flux individuellement. Une autre démarche consiste à imposer plus généralement des limitations de débit à chaque routeur.

3.5.2 Contrôle de débit

Le contrôle de débit (traffic shaping), est une approche globale. Son principe est de limiter le débit des routeurs d'accès, et donc le nombre de paquets injectés dans le réseau, afin de ne pas dépasser un certain seuil en termes d'utilisation des ressources disponibles. Tous les flux ne sont pas sujets à l'évitement de congestion.

L'aspect global du contrôle de débit permet, en limitant le débit total des routeurs, d'imposer des limites aux flux indésirables. Il existe deux techniques permettant ce contrôle de débit : le Leacky Bucket et le Token Bucket. [16]

3.5.2.1 Leacky Bucket

Le Leacky Bucket ou « Seau troué » en français, permet de réguler le trafic émis par les routeurs dans le réseau. Ainsi les paquets devant être émis, sont d'abord placés dans une file d'attente puis envoyés à intervalles réguliers. Cette méthode permet donc d'éviter les émissions en rafale. (Figure 3.06)

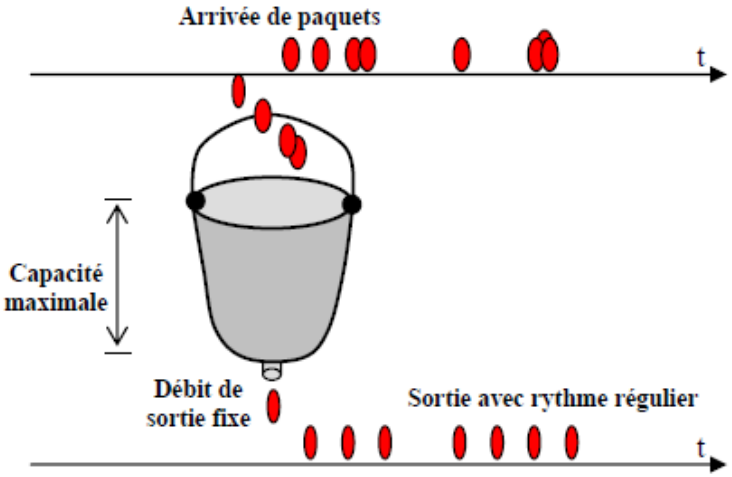


Figure 3.06 : *Fonctionnemet de Leacky Bucket*

La taille de la file d'attente et le débit d'émission sont configurables. Toutefois, cette méthode ne permet pas de s'adapter à la charge du réseau : le débit d'émission reste constant quelle que soit la charge. La deuxième technique dérivée de celle-ci permet au contraire de tenir compte des ressources disponibles dans le réseau.

3.5.2.2 Token Bucket

Pour une meilleure gestion des ressources, il est préférable de moduler le débit des routeurs en fonction de la charge du réseau. Le Token Bucket ou « seau à jetons », utilise la notion de jeton pour valuer la charge du réseau. Le nombre de jetons disponibles est inversement proportionnel à la charge du réseau. Le débit du routeur est conditionné par le nombre de jetons disponibles. Ainsi, le Token Bucket permet l'émission en rafale lorsque le réseau est peu chargé.

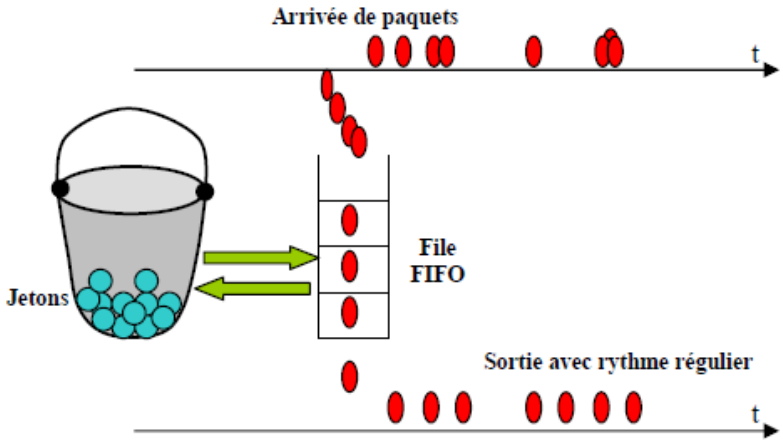


Figure 3.07 : *Fonctionnement du Token Bucket*

3.5.3 Réservation de ressources : IntServ (Integrated Services)

IntServ est un modèle d'architecture qui utilise le Resource Reservation Protocol (RSVP) pour signaler explicitement les besoins en QoS du trafic d'une application sur les périphériques du chemin d'accès de bout en bout via le réseau. Si chaque périphérique réseau du chemin d'accès peut réserver la bande passante nécessaire, l'application d'origine peut commencer à transmettre.

En d'autres termes, IntServ propose de réserver des ressources dans les nœuds intermédiaires avant de commencer à les utiliser. Contrairement au modèle DiffServ, chaque application est libre de demander une qualité de service spécifique à ses besoins. Les routeurs le long du chemin de transmission vérifient s'ils peuvent accorder cette qualité de service ou non, et acceptent ou refusent en conséquence la réservation de l'application. Si la réservation est acceptée, l'application est alors assurée d'obtenir des garanties pour le transfert de données, selon ce qui a été négocié lors de la réservation. [17]

Le principal défaut du modèle IntServ est que tous les routeurs, même les routeurs au cœur du réseau travaillant à haut débit, doivent inspecter plusieurs champs de chaque paquet afin de déterminer la réservation associée à ce dernier. Après la classification, chaque paquet est placé dans la file d'attente qui correspond à la réservation. La classification et la gestion de files d'attentes pour chaque flux individuel empêchent l'utilisation du modèle IntServ dans le réseau à haut débit. De plus, il suffit qu'un seul routeur sur le chemin de transmission n'offre pas le service d'IntServ pour que celui-ci soit mis à mal.

3.5.4 Différenciation de services : DiffServ (Differentiates Services)

Le modèle DiffServ met l'accent sur la QoS agrégée et dimensionnée. Au lieu de signaler les besoins d'une application en termes de QoS, DiffServ utilise un point de code de services différenciés (DSCP) dans l'en-tête IP pour indiquer les niveaux de QoS requis. [17]

Il consiste donc à introduire plusieurs classes de service offrant chacune une qualité de service différente. Chaque flux de trafic se voit attribuer une classe de service appropriée. Cette classification de trafic est effectuée en périphérie du réseau, directement à la source ou sur un routeur d'accès, selon des critères préconfigurés (adresses IP ou ports TCP/UDP). Chaque paquet est marqué par un code qui indique la classe de trafic assignée. Les routeurs dans le cœur du réseau utilisent ce code, transporté dans un champ du datagramme IP, pour déterminer la qualité de service requise pour le paquet. Tous les paquets ayant le même code reçoivent ainsi le même traitement.

Les critères de classification des paquets doivent refléter les besoins réels de l'information qu'ils transportent en termes de largeur de bande, sensibilité aux délais et aux variations de délai (gigue).

3.6 Relation entre routage et contrôle de flux dans la performance réseau

Le routage et le contrôle de flux sont deux mécanismes qui visent les mêmes objectifs généraux tels que :

- maximiser le débit utile du réseau ;
- satisfaire les seuils des indicateurs de performances pour assurer la qualité de service attendue par les utilisateurs : un faible délai de transit dans le réseau ainsi qu'une gigue acceptable, une faible probabilité de perte de paquets, une disponibilité assurée d'une route ou du réseau, une équité entre utilisateurs ;
- minimiser le coût économique à utiliser ;
- éviter les congestions

La figure (3.08) est une illustration simplifiée de la relation entre routage et contrôle de flux.

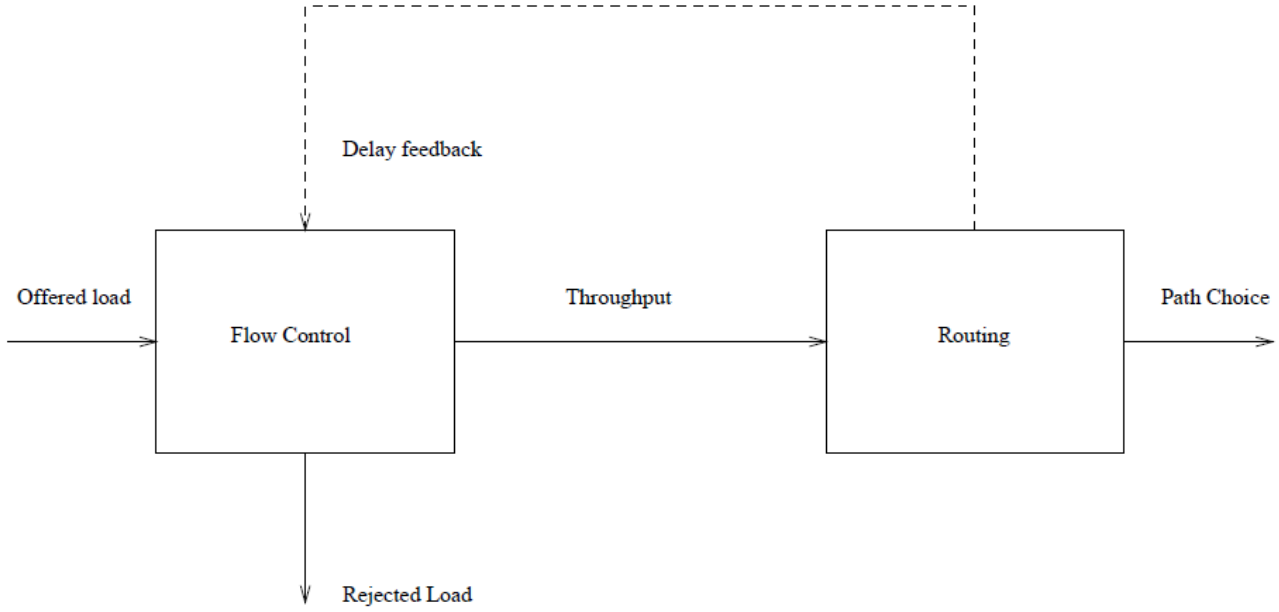


Figure 3.08 : *Relation entre routage et contrôle de flux*

3.7 Conclusion

Ce chapitre nous a permis d'étudier en profondeur les aspects d'un réseau pouvant nous amener à évaluer sa performance. Parmi ces aspects sont la notion de qualité de service QoS pour atteindre les exigences requises pour les différents services ; les indicateurs de performance ainsi que les facteurs pouvant affecter l'état du réseau. Nous avons ensuite vu les différentes techniques qui permettent de prévenir et de réagir face aux congestions.

Dans le chapitre suivant, nous allons nous intéresser aux approches d'analyse des mécanismes de gestion de la qualité de service afin de mettre en évidence leur performance.

CHAPITRE 4
ANALYSE DES MECANISMES DE GESTION DE QOS

4.1 Introduction

Ce chapitre est consacré à l'étude des techniques de gestions de la qualité de service pour des réseaux à commutation de paquets tel qu'Internet.

Avant d'entrer dans le vif du sujet, nous allons exposer les problématiques qui peuvent compliquer la maintenance des performances de ces réseaux. Nous allons ensuite simuler un réseau de base afin d'analyser les différents mécanismes de gestion de QoS pour différents scénarios.

4.2 Problématiques

4.2.1 Niveau trafic IP

Les caractéristiques du trafic IP rendent leur gestion extrêmement difficile. En effet :

- à cause de la sporadicité, le trafic se comporte en rafales et crée beaucoup de problèmes au niveau de la gestion du trafic lors de la réservation des ressources ainsi qu'au niveau des mémoires tampons dans les routeurs.
- le trafic auto-similaire et à longue mémoire est responsable de beaucoup d'effets indésirables dans le réseau, notamment l'importante variance du débit au sein du réseau. Ceci engendre des tailles de files d'attente très variables. De ce fait, les temps d'attente au niveau des routeurs deviennent très variables.

4.2.2 Niveau file d'attente

Les routeurs sont les équipements de base dans les réseaux IP. La file d'attente est une composante centrale dans l'architecture des routeurs. En effet, un routeur est composé de processus asynchrones destinés 4 à : (Figure.01)

- assembler les paquets reçus (file d'attente en entrée) ;
- contrôler l'intégrité du paquet;
- déterminer l'interface de destination ;
- transmettre les paquets (files d'attente en sortie).

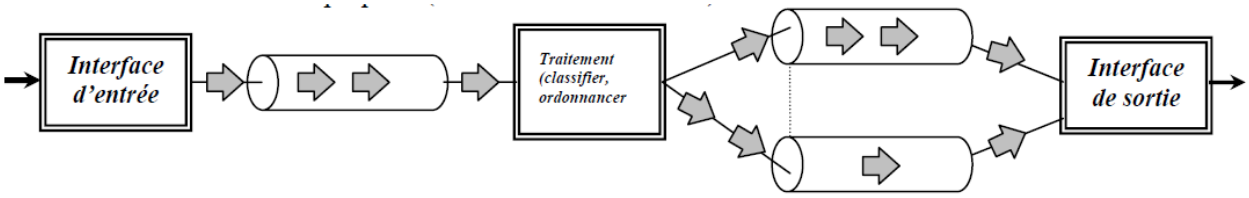


Figure 4.01 : *Architecture d'un routeur*

Les files d'attente dans les routeurs présentent des caractéristiques différentes, en fonction de la charge sur les autres liens directement connectés.

La stratégie de gestion des diverses files d'attente dans un routeur joue un rôle essentiel pour l'assurance de la qualité de service. [18] [19]

Nous allons analyser le comportement des files d'attentes en fonction des charges dans le système. Pour ce faire, nous allons considérer une file M/M/1.

Cette file est caractérisée par une arrivée poissonienne de taux λ et une durée de service exponentielle de taux μ.

On pose :

- la charge ou le taux de trafic donnée par $\rho = \frac{\lambda}{\mu}$
- la probabilité d'état donné par $p_k = p_0\rho^k = (1-\rho)\rho^k$ qui définit la probabilité d'avoir un nombre de clients k dans le système.
- la condition de stabilité $\rho < 1$ qui traduit le fait d'avoir un débit d'entrée inférieur au débit du serveur.

Nous allons alors déterminer le nombre moyen de clients dans le système, le nombre moyen de client dans la file et le temps moyen de séjour d'un client dans le système en fonction de la charge de celui-ci.

4.2.2.1 Nombre moyen de clients dans le système

Le nombre moyen de clients est donné par la formule suivante :

$$N_S = \frac{\rho}{1-\rho} \qquad (4.01)$$

Démonstration :

♣

Le nombre moyen de clients se calcule à partir des probabilités stationnaires de la façon suivante :

$$N_S = \sum_{k=1}^{\infty} k p_k = \sum_{k=1}^{\infty} k\,(1-\rho)\rho^k = (1-\rho)\rho \sum_{k=1}^{\infty} k\rho^{k-1}$$

$$N_S = (1-\rho)\rho \sum_{k=0}^{\infty} (\rho^k)' = (1-\rho)\rho \left(\frac{1}{1-\rho}\right)'$$

$$N_S = (1-\rho)\rho \left(\frac{1}{(1-\rho)^2}\right)$$

D'où :

$$N_S = \frac{\rho}{1-\rho} \tag{4.02}$$

♦

4.2.2.2 Temps moyen de séjour d'un client dans le système

Le temps moyen de séjour d'un client dans le système est donné par :

$$T_S = \frac{1}{\mu}\,\frac{1}{1-\rho} \tag{4.03}$$

Démonstration

♣

Le nombre moyen de client N_S dans le système s'écrit, d'après la loi de Little, par :

$$N_S = \lambda\, T_S \tag{4.04}$$

Avec λ le nombre d'arrivées de clients par seconde et T_S le temps moyen passé par le client dans le système. Ainsi on en déduit d'après (4.02) et (4.04):

$$T_S = \frac{N_S}{\lambda} = \left(\frac{\rho}{1-\rho}\right)\left(\frac{1}{\lambda}\right)$$

D'où :

$$T_S = \frac{1}{\mu}\,\frac{1}{1-\rho} \tag{4.05}$$

♦

4.2.2.3 Nombre moyen de clients dans la file

Le nombre moyen de clients dans la file N_a s'écrit, d'après la loi de Little, par :

$$N_a = \lambda\, T_a \tag{4.06}$$

Avec T_a le temps passé par le client dans la file. Ce temps peut être déduit du temps T_S.
En effet, on peut décomposer T_S par :

$$T_S = \frac{1}{\mu} + \frac{\rho}{\mu(1-\rho)} \tag{4.07}$$

On en déduit de l'équation (4.07) le temps T_a par : [18]

$$T_a = \frac{\rho}{\mu(1-\rho)} \tag{4.08}$$

En utilisant les équations (4.06) et (4.08), on obtient le nombre moyen de clients dans la file :

$$N_a = \lambda \frac{\rho}{\mu(1-\rho)} = \mu\rho \frac{\rho}{\mu(1-\rho)}$$

D'où:

$$N_a = \frac{\rho^2}{(1-\rho)} \tag{4.09}$$

En utilisant MATLAB, on peut présenter sur la figure 4.02 l'état du système et de la file d'attente en fonction du taux de trafic.

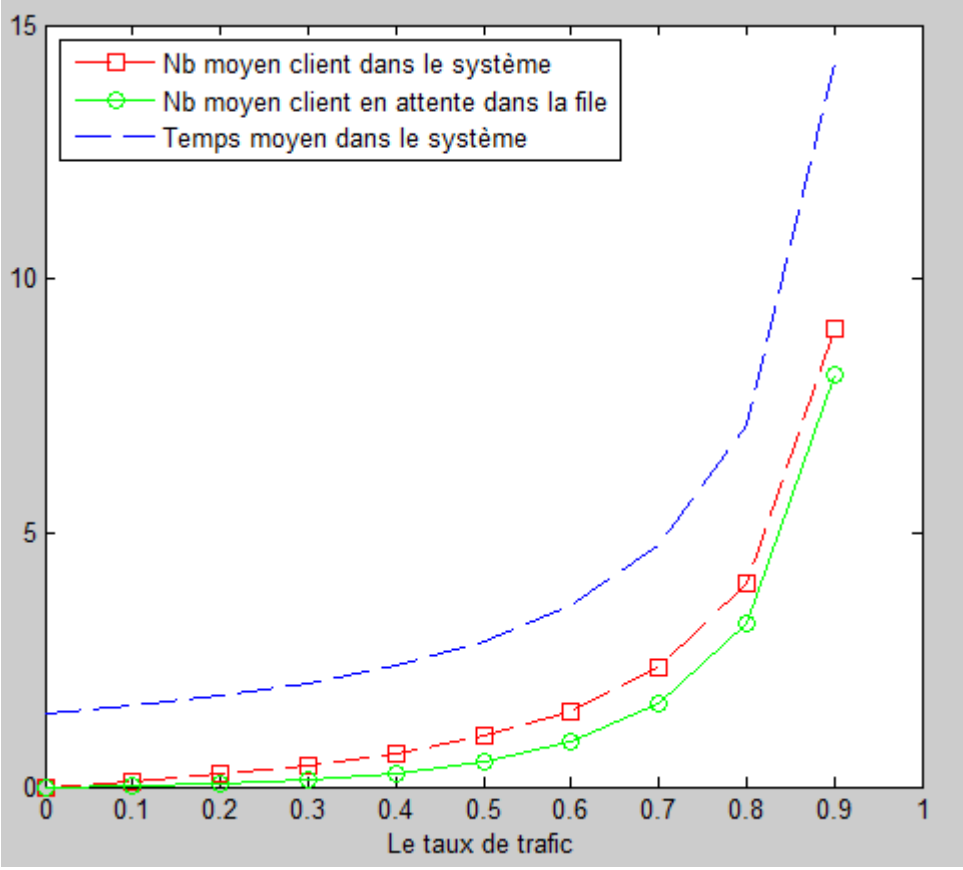


Figure 4.02 : *Etat du système et de la file d'attente en fonction du taux de trafic*

D'après ces courbes, nous pouvons conclure que plus la charge de trafic dans la file d'attente augmente plus le temps passé dans le système augmente. Cette augmentation de temps d'attente

désigne la présence de congestion et affecte ainsi les performances du réseau, surtout pour des applications à temps réels tels que la voix et la vidéo conférence.
La gestion des files d'attente au niveau des routeurs est alors indispensable pour contrôler le bon fonctionnement d'un réseau.

4.3 Principe de gestion la QoS

Le principe de la QoS est basé sur la combinaison de fonctions appliquées sur les paquets transitant dans les routeurs. Ces fonctions opèrent des changements de priorité, bloquent ou laissent passer certains paquets ou encore ordonnancent les paquets en sortie dans un ordre différent que celui d'entrée. [20]
Les fonctions que l'on retrouve sont synthétisées sur la figure 4.03.

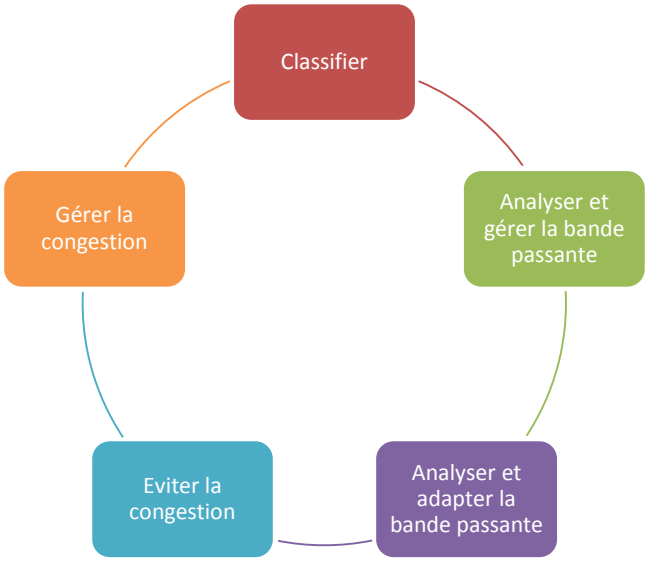


Figure 4.03 : *Fonctions de la QoS*

On a :

- Classifier : il s'agit de mettre en place des listes de contrôles d'accès (ACL) et des filtres plus ou moins évolués.
- Analyser et gérer la bande passante : il s'agit des mécanismes de Leacky Bucket et de Token Bucket qu'on a vu dans le chapitre précédent ainsi que le principe d'utilisation des champs TOS (Type Of Service) et DSCP (Differentiated Services Code Point).
- Analyser et adapter la bande passante : il s'agit du fonctionnement d'un traffic shaper et sa différence avec un policer.

- Eviter la congestion : il concerne la mise en place des algorithmes RED et WRED dont les principes sont explicités dans le chapitre précédent.
- Gérer la congestion : cette fonction se fait typiquement au travers de files d'attente dont l'utilisation suit un algorithme particulier appelé algorithme d'ordonnancement, par exemple le FIFO, le PQ, le WFQ.

Dans ce travail nous allons nous focaliser sur les deux dernières fonctions qui s'opèrent directement sur les files d'attentes. Nous allons mettre en place la simulation d'un réseau afin d'analyser les performances de ces différentes techniques.

4.4 Présentation du réseau à simuler

4.4.1 Environnement de simulation : OPNET Modeler

OPNET (Optimum Network Performance) est un outil de simulation de réseaux très puissant et très complet. Basé sur une interface graphique intuitive, son utilisation et sa prise en main est relativement aisée. De plus, il permet de visualiser la topologie physique d'un réseau local, métropolitain, distant ou embarqué.

OPNET dispose de trois niveaux hiérarchiques imbriqués : le network domain, le node domain et le process domain. [16]

4.4.1.1 Network domain

C'est le niveau le plus élevé de la hiérarchie d'OPNET. Il permet de définir la topologie du réseau en y déployant des hôtes, des liens ainsi que des équipements actifs tels que des switchs ou des routeurs. (Figure 4.04)

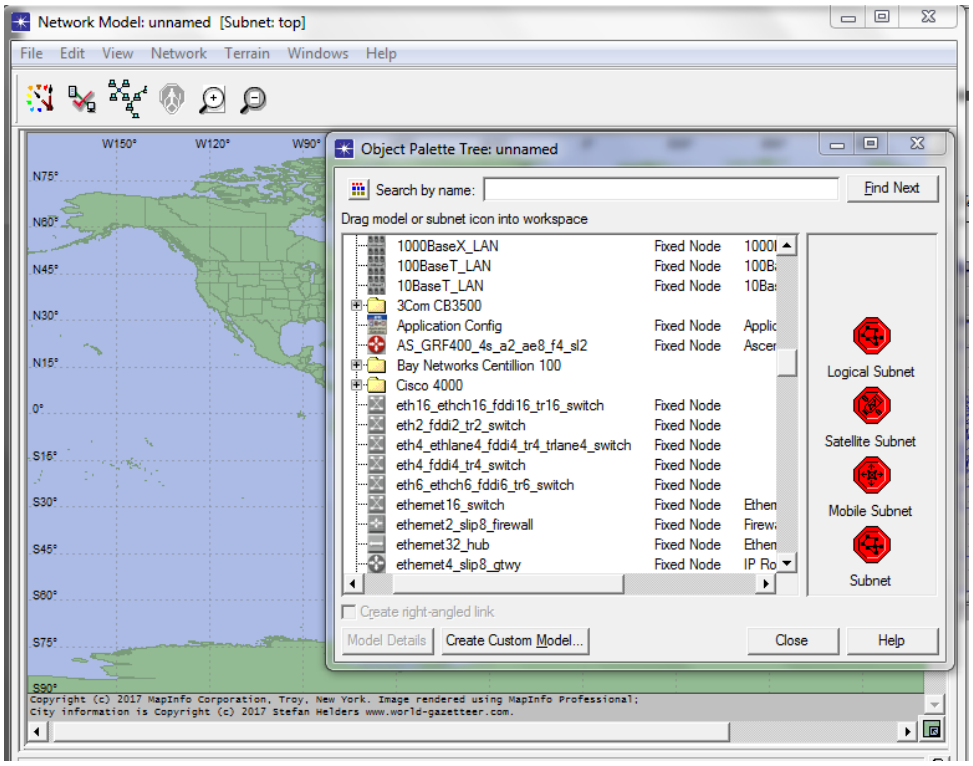


Figure 4.04 : *Network domain*

4.4.1.2 Node domain

Le Node domain permet de définir la constitution des nœuds (routeurs, stations de travail, hub, …). Le modèle est défini à l'aide de blocs appelés modules. (Figure 4.05)

Certains modules sont non programmables : il s'agit principalement des « transmitters » et des « receivers », dont la seule fonction est de s'interfacer entre le nœud et les liens auxquels il est connecté.

Les autres modules sont entièrement programmables : il s'agit des processors et des queues.

Les processors sont des modules qui remplissent une tâche bien précise du nœud : ils symbolisent les différentes parties du système d'exploitation d'une machine, et plus principalement les différentes couches réseau implémentées dans le nœud (Ethernet, IP…).

Les processors peuvent communiquer entre eux via des packets streams (flux de paquets), qui permettent de faire transiter un paquet d'une couche à une autre à l'intérieur d'une même machine.

Cette organisation permet d'avoir une vision claire de la pile de protocoles implémentée dans un nœud, et de connaître rapidement leurs interactions.

Les statistic wires constituent le second type de lien permettant une communication entre modules. Ils permettent de faire remonter des informations de statistiques d'un module à l'autre, comme par exemple la taille et le délai des queues des transmitters.

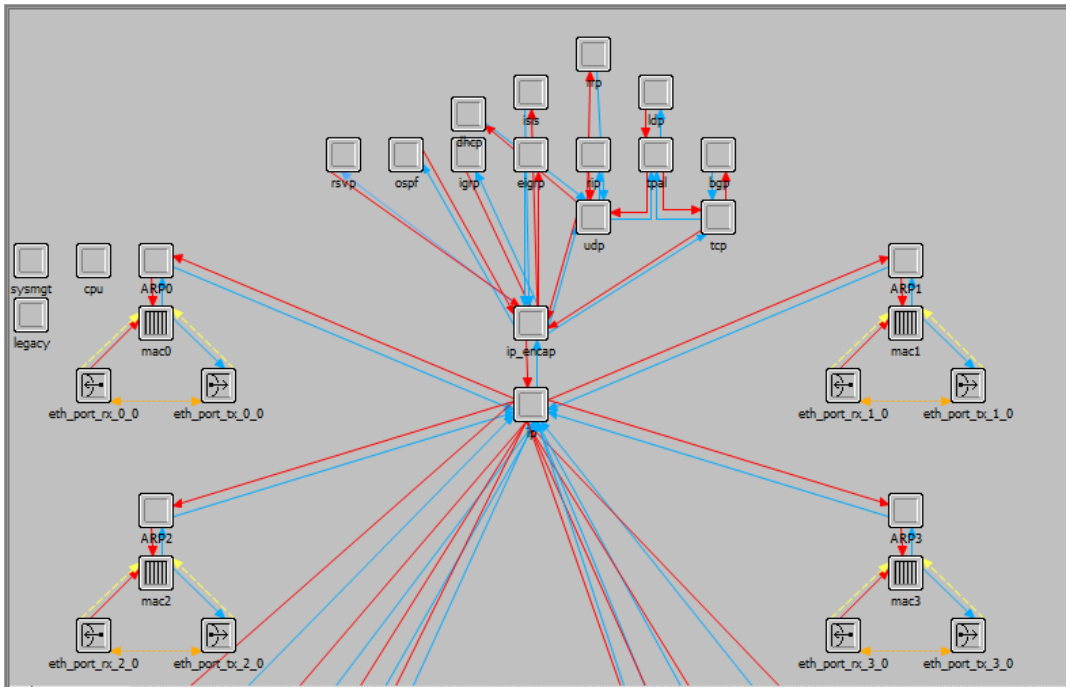


Figure 4.05 : *Node domain*

4.4.1.3 Process domain

C'est à ce niveau que l'on définit le rôle de chaque module programmable. Le rôle d'un module est déterminé par son process model, que l'on décrit sous forme d'une machine à états finis (finite state machine).

Chaque bloc représente un état différent, dans lequel la machine exécute un code déterminé.

Les transitions sont symbolisées par des liens entre blocs et déterminées par des conditions (interruptions, variable ayant une certaine valeur, etc.). (Figure 4.06)

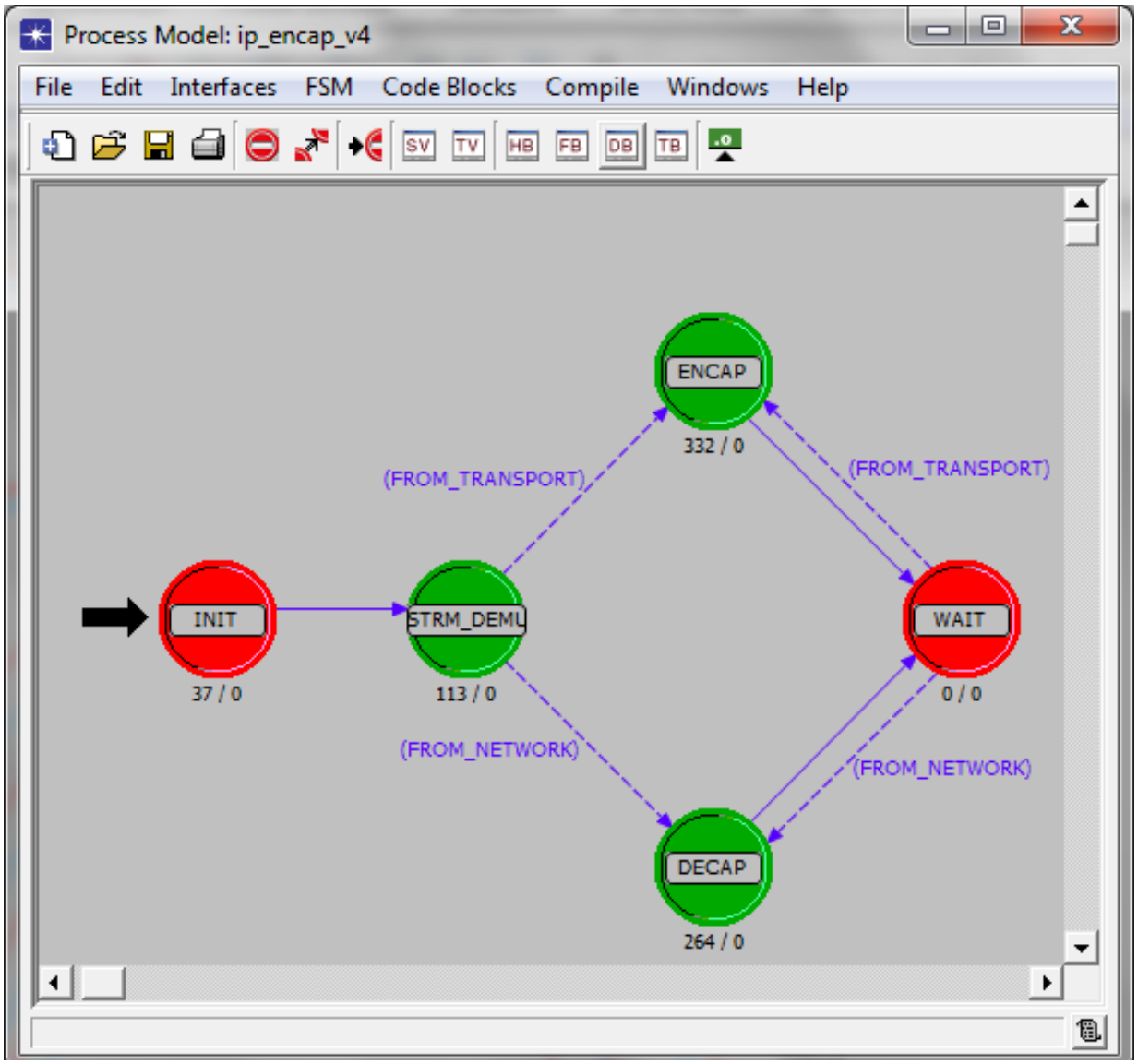


Figure 4.06 : *Process model*

4.4.2 Topologie du réseau à étudier

La topologie du réseau qu'on va étudier sous le simulateur OPNET est donnée par la figure 4.07 suivante.

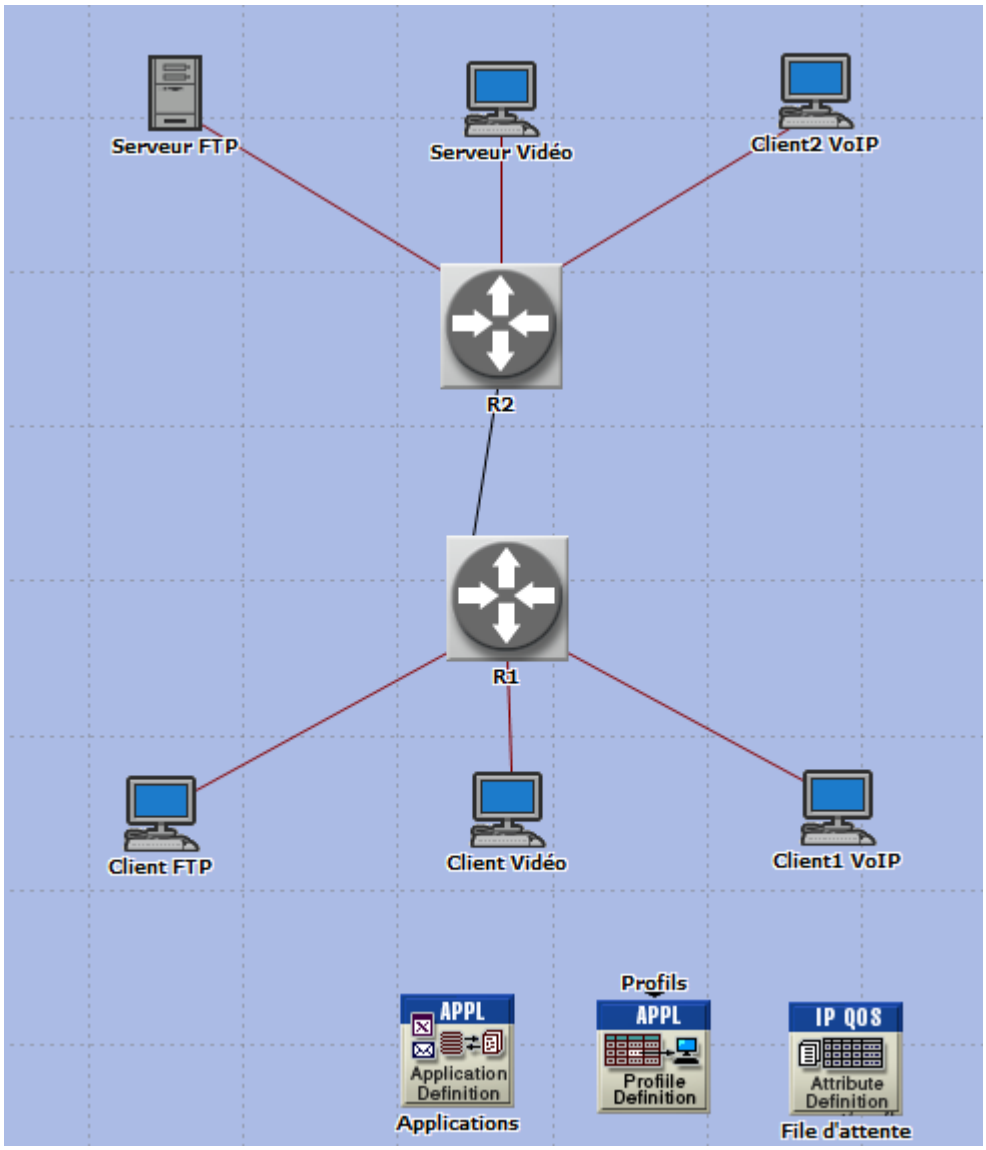


Figure 4.07 : *Topologie du réseau à étudier*

Dans ce réseau, nous allons générer différents trafics (trafic FTP, trafic vidéo et trafic voix) pour illustrer les applications fréquemment utilisées sur Internet. Ainsi, nous avons quatre clients : un Client FTP, un Client Vidéo et deux clients VoIP (Client1 et Client 2) ainsi que deux serveurs : un Serveur FTP et un serveur Vidéo. L'interconnexion entre ces nœuds se fait par deux routeurs R1 et R2.

Sous OPNET, il faut aussi définir les applications et les profils utilisateurs présents dans le réseau. Ces configurations se font sur les deux nœuds nommés « Application » et « Profils ».

Le dernier nœud nommé « File d'attente » permet de définir les configurations de QoS dans notre système.

4.4.3 Caractéristiques des trafics à étudier

Nous avons deux types de trafics :

- le trafic FTP de type élastique qui n'exige pas des contraintes extérieures telles qu'une bande passante insuffisante sans pour autant remettre en cause la viabilité du service. Les paquets FTP sont de grande taille.
- les trafics vidéo et voix qui sont de type « streaming » ou « à temps réel » qui ne tolèrent pas des délais et des pertes de paquets élevés. La voix est encore plus sensible à la latence par rapport à la vidéo.

Ainsi, nous avons trois applications avec trois types de services (ToS) différents.

Le ToS est assigné dans l'en-tête de chaque paquet IP. C'est un attribut qui permet de fournir pour chaque paquet le service qui lui est approprié dans les files d'attente.

Ainsi, le trafic FTP est de type « Best Effort » correspondant à un ToS = 0, le trafic vidéo est de type « Streaming Multimédia » correspondant à un ToS = 4, le trafic voix est de type « Interactive Voice) correspondant à un ToS = 6.

La mise en place de ces types de trafics consiste à étudier le comportement de notre système face aux attentes en termes de performances des différentes applications.

4.5 Simulation 1 : analyse des algorithmes de gestions de file d'attente

4.5.1 Principes des gestions de file d'attente pour l'évitement de congestion

Un inconvénient induit par le mécanisme de comportement adaptatif de TCP (algorithme de « congestion avoidance ») apparaît lorsque des points de congestion se développent dans un réseau chargé et entraîne la baisse de performance globale. En effet, chaque flux TCP actif peut détecter des pertes et agir en conséquence. Ainsi, les centaines ou milliers de flux TCP simultanés vont repasser en mode « slow-start » et tenter de réémettre les paquets perdus, contribuant ainsi l'accroissement de la congestion. Ce phénomène est connu sous le nom de « global synchronization » et met en évidence l'importance de mécanismes de prévention de la congestion tels que RED et WRED. [11]

4.5.1.1 RED

RED ou Random Early Detection est un algorithme de gestion des files d'attente.

Son but est de prévenir les congestions avant qu'elles n'interviennent, en détectant à l'avance le remplissage des buffers, sans attendre leur saturation. Le mécanisme prend l'initiative de rejeter certains paquets TCP par anticipation, ce qui entraîne la source à réduire son débit.

Le type de comportement adopté par cet algorithme lors de la réception d'un paquet dépend de la valeur «*avg* » de la taille moyenne de la file d'attente à cet instant par rapport aux deux seuils min_{th} et max_{th} (Figure 4.08). [21]

Soit p_b la probabilité pour que le paquet soit pénalisé. Trois cas se présentent alors à chaque arrivée d'un paquet :

- Si $avg < min_{th}, p_b = 0$, le paquet n'est pas rejeté;
- Si $min_{th} < avg < max_{th}$, la probabilité p_b se calcule par extrapolation linéaire d'après la valeur de « *avg* » à l'aide de la formule :

$$p_b = p_{max} \frac{avg - min_{th}}{max_{th} - min_{th}} \tag{4.10}$$

  Avec p_{max} la probabilité maximale atteinte lorsque avg vaut max_{th}.
- Si $avg > max_{th}, p_b = 1$, le paquet est rejeté.

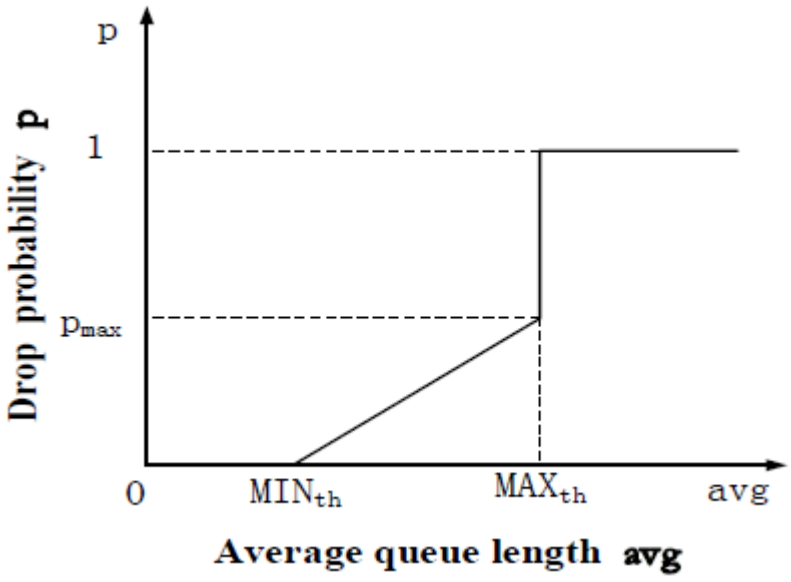


Figure 4.08 : *Principe de RED*

La taille moyenne de la file est calculée à l'aide de la formule :

$$avg = (1 - w_q)\, avg + w_q\, q \tag{4.11}$$

Où q représente la taille courante de la file et w_q le poids qui détermine l'influence d'une modification de q sur avg. Plus w_q est grand, plus les changements de valeur de q influeront sur la valeur de avg. Toutefois, si w_q est trop grand, les congestions momentanées et les courtes rafales de paquets auront une trop grande importance sur la taille moyenne de la file. Dans le cas contraire, avg ne convergera pas assez rapidement vers la taille moyenne réelle de la file d'attente.

4.5.1.2 WRED

Le principal inconvénient de RED est le manque de discernement dans son rejet des paquets. Certaines applications critiques sont très sensibles à la perte ; d'autres, comme les transmissions multimédia, fonctionnant sur UDP, ne disposent pas de mécanisme de contrôle de flux tel que la fenêtre de congestion. [11]

Ainsi, le WRED (Weighted Random Early Detection) est une extension de l'algorithme RED afin de promouvoir une politique avec plusieurs niveaux de priorité. En effet, WRED est la combinaison de RED et de PQ ou Priority Queuing (qu'on va aborder plus tard). Il permet de définir des différents seuils min_{th} et max_{th} ainsi que des probabilités maximales p_{max} différentes selon le type de service (ToS) du paquet. (Figure 4.09)

Il ne rejettera donc que les paquets dont la classe de service indique une faible sensibilité aux pertes.

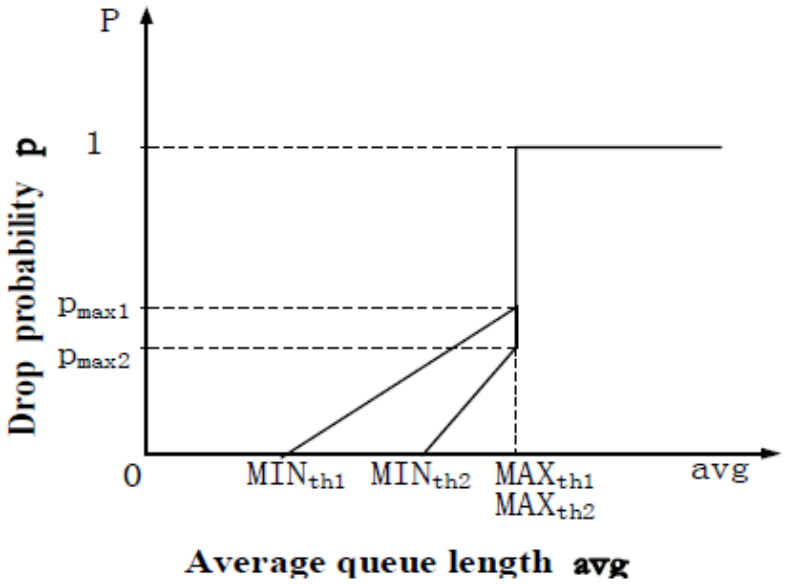


Figure 4.09 : *Principe de WRED*

4.5.2 Présentation des scénarii de simulation

Nous avons mis en place trois scénarii de simulation décrits par le tableau 4.01 suivant.

Tableau 4.01: *Scénarii de la simulation 1*

Scénario	Ordonnanceur	Durée de simulation
Scénario 1	FIFO	10 min
Scénario 2	FIFO + RED	
Scénario 3	FIFO + WRED	

4.5.3 Paramètres de configurations utilisés

Les paramètres utilisés pour les configurations de RED et WRED sont donnés par le tableau 4.02 suivant.

Tableau 4.02: *Paramètres de configuration*

Paramètre	RED	WRED
Seuil minimal min_{th}	100	100
Seuil maximal max_{th}	200	200
Probabilité de rejet maximal p_{max}	0.1	1, 0.2, 0.5 (respectivement pour FTP, vidéo et voix)

4.5.4 Paramètres de sortie à collecter

4.5.4.1 Temps d'attente dans la file

Il s'agit du temps d'attente dans chaque file, exprimé en secondes. C'est un paramètre défini dans chaque routeur du réseau au niveau des interfaces connectés.

4.5.4.2 Utilisation du « buffer »

Il correspond à la variation de la taille de la file d'attente utilisée pour chaque interface connecté des routeurs. Il est exprimé en octets.

4.5.4.3 Nombre de paquets détruits

Il s'agit du nombre de datagrammes IP détruits par tous les nœuds du réseau à travers toutes les interfaces. Il est exprimé en paquets/secondes.

4.5.5 Interprétations des résultats

Les courbes suivantes montrent les résultats des algorithmes de gestions de files RED et WRED.

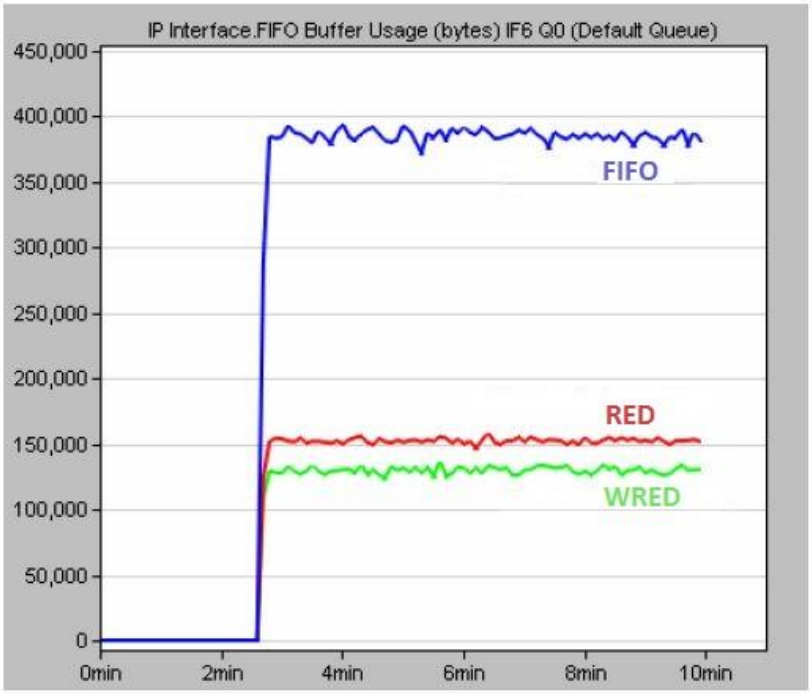


Figure 4.10 : *Utilisation du « Buffer »*

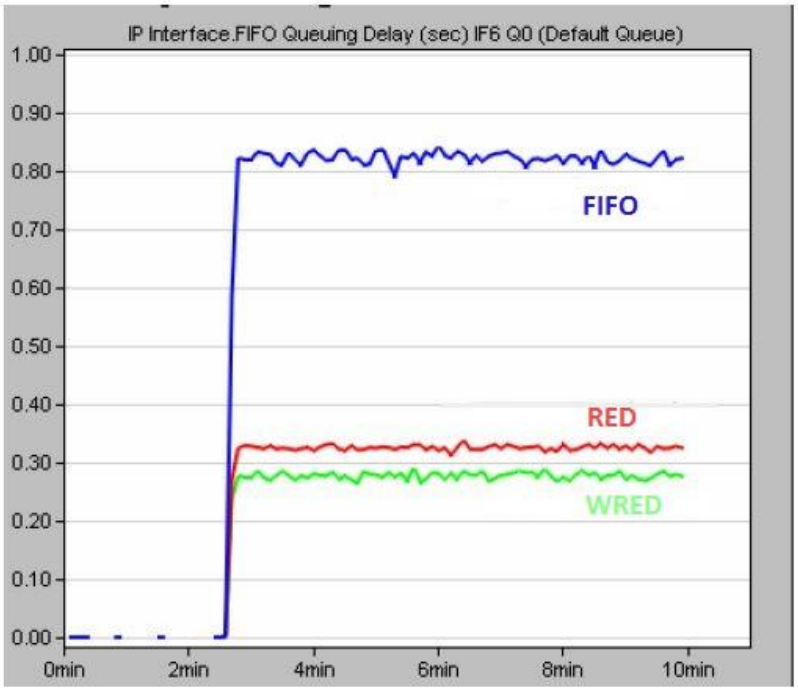


Figure 4.11 : *Temps d'attente*

La figure 4.10 nous montre que la mémoire tampon de la file est largement utilisée avec FIFO contrairement avec RED et WRED. Ceci peut s'expliquer par le fait que FIFO utilise la taille de la file jusqu'à ce que celle-ci soit pleine avant de rejeter les autres trafics entrants. Or pour le cas de

RED et WRED, le rejet des paquets n'attend pas que la file soit pleine mais il se déclenche en fonction de la taille moyenne de la file calculée par rapport aux seuils minimal et maximal posé. Cette technique permet ainsi de réduire le temps d'attente de paquets et de diminuer ainsi les risques de congestion. La figure 4.11 illustre cette diminution de temps d'attente grâce à RED et WRED par rapport à l'algorithme FIFO.

La perte de paquet se produit principalement lorsque l'intensité du trafic sur les liens de sortie devient supérieure à leur capacité d'écoulement. La perte de paquet est une indication de congestion. D'après la figure 4.12, on constate qu'avec RED et WRED on n'a pas de perte de paquets ce qui vérifie alors que ces algorithmes permettent d'éviter l'apparition de congestion.

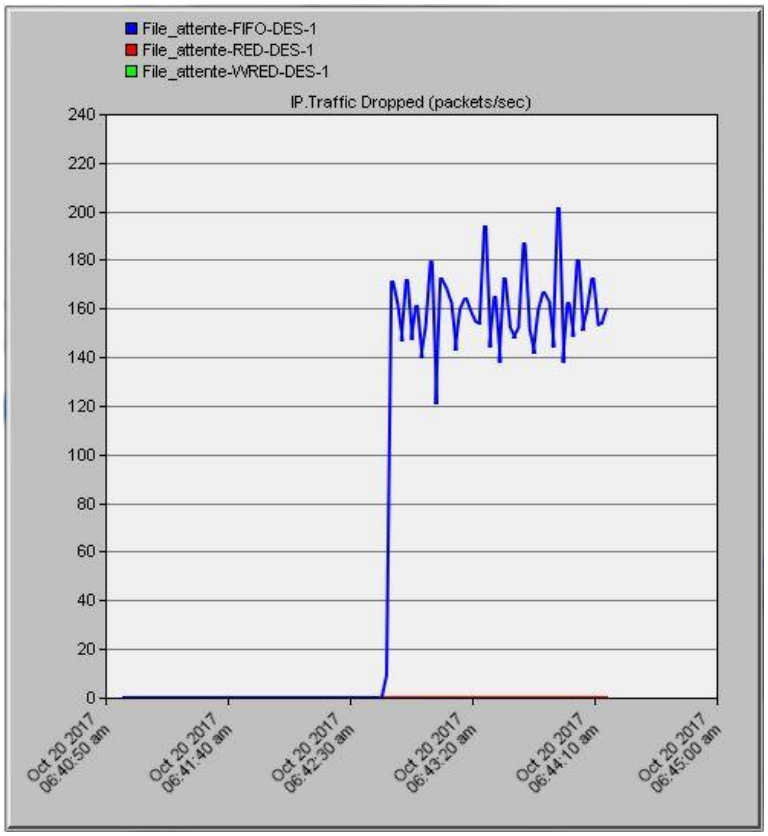


Figure 4.12 : *Perte de paquets IP*

4.5.6 Synthèse des résultats

Les résultats observés nous permettent de faire la synthèse suivant :

- Un système sans gestion de la file d'attente utilise la totalité de sa taille avant de rejeter les paquets excédants ce qui incite la retransmission des paquets perdus augmentant le risque d'apparition de congestion ; contrairement à RED et WRED qui rejettent les paquets en fonction de la taille de la file, donc de l'état de congestion du réseau.

- En comparant les deux algorithmes RED et WRED, WRED présente de meilleurs avantages du fait que lors des rejets des paquets, il prend en compte les priorités de chaque paquet. Ainsi, les paquets de faible priorité (ici les trafics FTP) sont plus sujets à être détruits quand la taille moyenne de la file dépasse le seuil maximal, assurant ainsi la transmission de l'intégrité des trafics à temps réel.

4.6 Simulation 2 : analyse des mécanismes d'ordonnancement

4.6.1 Principe des algorithmes d'ordonnancement

Chaque routeur du réseau utilise une certaine politique d'ordonnancement pour déterminer l'ordre dans lequel les paquets de différents flux partageant une même ligne de sortie seront transmis. Dans le plus simple des cas, on utilise une discipline de type FIFO qui permet de transmettre les paquets dans l'ordre dans lequel ils sont arrivés. Mais cette technique ne permet pas de donner plus d'importance (de poids) à un flux donné par rapport aux autres, alors que c'est nécessaire pour implanter des qualités de service. [21] [22]

Il existe une panoplie d'algorithmes qui visent à résoudre ce problème. Dans cette partie, en commençant par la discipline FIFO, nous allons décrire les algorithmes de PQ (Priority Queuing) et WFQ (Weighted Fair Queuing).

4.6.1.1 FIFO

La discipline FIFO (First In, First Out), traduite par premier arrivé, premier servi, est la méthode standard de gestion de trafic entre une interface d'entrée et une interface de sortie. Les paquets sont placés dans la file de sortie selon l'ordre dans lequel ils sont reçus, comme le montre la figure (4.13). C'est le fonctionnement par défaut de tout type d'interface. [11]

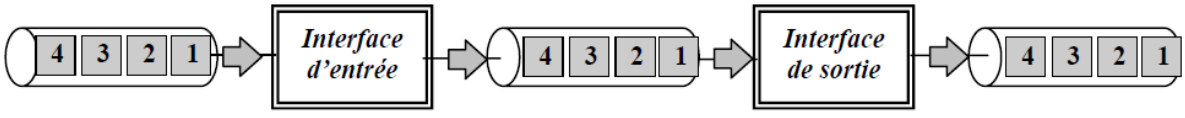


Figure 4.13 : *First In First Out (FIFO)*

La technique FIFO est simple et rapide. Cependant il n'existe qu'une seule file avec une seule priorité statique pour les premiers arrivés. Son principal inconvénient est donc le fait que FIFO ne gère pas la QoS.

4.6.1.2 PQ

L'algorithme de PQ (Priority Queuing) est la forme primitive de différenciation des services. Dans ce mode de gestion chaque interface de sortie a quatre files pour ordonnancer les paquets. Les priorités de ces files sont absolues, c'est-à-dire que la file 1 est moins prioritaire que la file 2, la file 2 est moins prioritaire que la file 3 qui elle-même est moins prioritaire que la file 4. (Figure 4.14)
L'algorithme pour servir les différentes files est le suivant [20]:

- tant qu'il y a des paquets dans la file "high"(file 4) alors router le paquet, sinon passer à la file inférieure,
- tant qu'il y a des paquets dans la file "medium" (file 3) alors router le paquet, sinon passer à la file inférieure,
- tant qu'il y a des paquets dans la file "normal" (file 2) alors router le paquet, sinon passer à la file inférieure,
- router le paquet dans la file "low" (file 1), puis réitérer.

Rappelons la définition du routage qui consiste à aiguiller les paquets d'une source vers une sortie pour une destination précise.

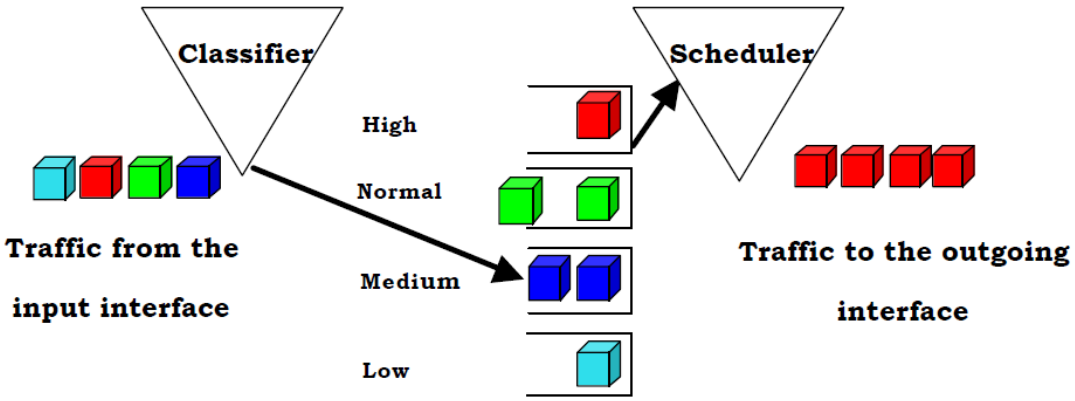


Figure 4.14 : *Priority Queuing PQ*

Les avantages de PQ sont sa simplicité et sa rapidité ainsi que la mise en place des priorités absolues permettant de privilégier de manière absolue un trafic par rapport à un autre.

Cependant, il est limité par ses quatre niveaux de priorité. En effet, le problème se pose lorsqu'on est face à plus de 4 classes de trafics à prendre en compte.
De plus les priorités absolues décrites par l'algorithme PQ peuvent causer les problèmes de congestions artificielles dues à des problèmes de « famine », c'est-à-dire dans le cas où la file « High » présentent trop de paquets à ordonnancer pénalisant ainsi les autres files inférieures.

4.6.1.3 WFQ

Le WFQ (Weighted Fair Queuing) est un algorithme qui propose n+8 files d'attente sur chaque interface de sortie. Le paramètre n est fonction de la bande passante de l'interface et les 8 autres files sont réservées par Cisco pour ne pas mettre en concurrence certains flux avec ceux des utilisateurs. L'ordonnancement des paquets dans les files est dit "fair" et "weighted" [20] :

- « fair » car il y a un ordonnancement "juste" des paquets en entrée en fonction des flux détectés. Le principe est assez simple : le routeur va créer dynamiquement une file par flux détecté (limité à n files). On entend ici par flux le n-uplet [adresse IP source, adresse IP destination, protocole, port source, port destination, champ TOS]. Ainsi tous les paquets d'une même session seront en attente dans une même file. L'ordonnanceur en sortie consulte les différentes files les unes après les autres avançant octet par octet et dès qu'un paquet entier peut être transféré il le fait. L'algorithme exécuté recalcule en quelque sorte les priorités de chaque file à chaque octet.
- « weighted » car l'ordonnanceur en sortie est sensible au champ TOS ou DSCP. En effet, pour un même flux entre deux machines, des paquets marqués avec une « IP precedence » à 5 seront servis plus rapidement que des paquets marqués à 0. Ceci se fait en ajoutant une pondération à chaque file en fonction de la priorité du marquage détecté.

Chaque file d'attente est servie avec un poids qui représente une fraction garantie de la bande passante. Ce poids peut se traduire par un nombre de paquets émis par tour de service ou, par une fréquence et un schéma de service particulier. La figure (4.15) illustre le principe général du WFQ. [22]

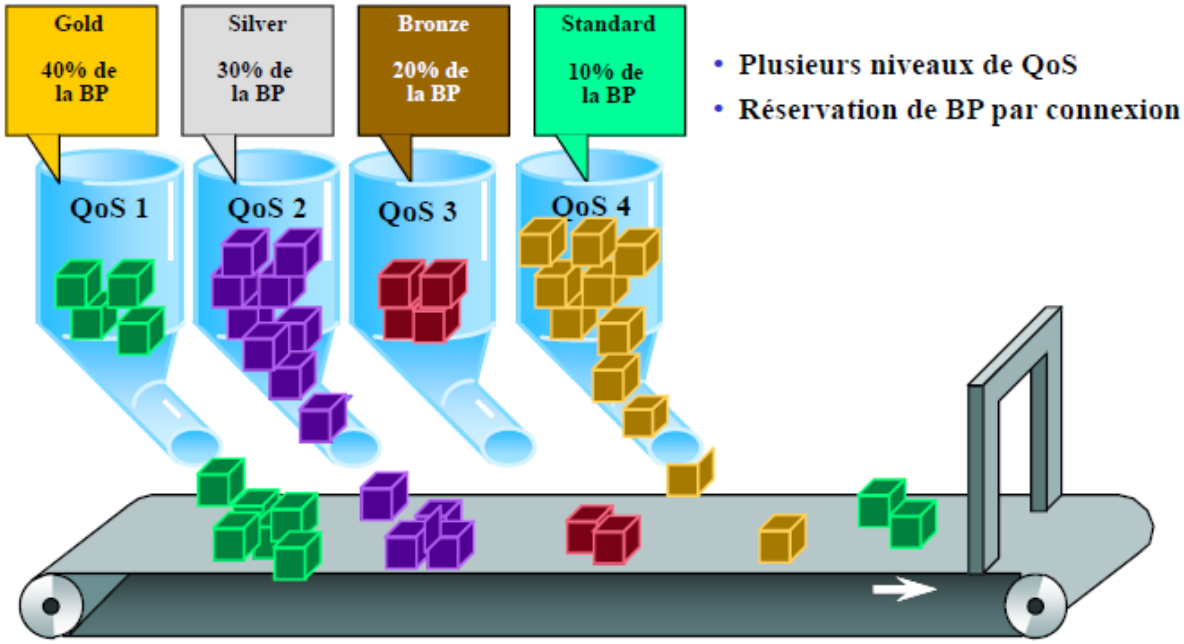


Figure 4.15 : *Weighted Fair Queuing WFQ*

WFQ est donc un apport significatif en termes de mécanisme de QoS par rapport aux deux approches précédentes du fait de son aspect dynamique tant au niveau de l'ordonnancement des flux en entrée que du service des files en sortie. L'ordonnancement est juste et pondéré pour chaque paquet en entrée.

4.6.2 Présentation des scénarii de simulation

Le tableau 4.03 décrit les trois scénarii mis en place.

Tableau 4.03: *Scénarii de la simulation 2*

Scénario	Ordonnanceur	Durée de simulation
Scénario 1	FIFO	200s
Scénario 2	PQ	
Scénario 3	WFQ	

4.6.3 Paramètres de sortie à collecter

Pour cette simulation, nous avons collecté quatre paramètres de sortie.

4.6.3.1 Nombre de paquets IP détruits

Il s'agit du nombre de datagrammes IP détruits par tous les nœuds du réseau à travers toutes les interfaces. Il est exprimé en paquets/secondes.

4.6.3.2 Temps d'attente de bout-en-bout

Il représente le temps mesuré depuis l'entrée d'un paquet dans chaque file d'attente jusqu'à ce que le dernier bit de ce paquet soit transmis. Il est exprimé en seconde.

4.6.3.3 Nombre de paquets reçus

C'est le nombre moyen de paquets transmis, en bits/secondes, à travers les couches de transport dans le réseau. Nous avons récolté ce paramètre pour les trois applications FTP, vidéo et voix.

4.6.3.4 Latence pour le trafic voix

Il s'agit de la latence de tous les paquets voix englobant la latence du réseau, la latence de codage et de décodage, la latence de la compression et de la décompression. Il est exprimé en seconde.

4.6.4 Interprétations des résultats

Nous allons analyser les résultats obtenus par application afin de mettre en évidence l'effet de chaque mécanisme d'ordonnancement sur celles-ci.

4.6.4.1 Nombre de paquets IP détruits

La figure 4.16 représente l'évolution du nombre de paquets perdus par seconde pour les trois mécanismes proposés.

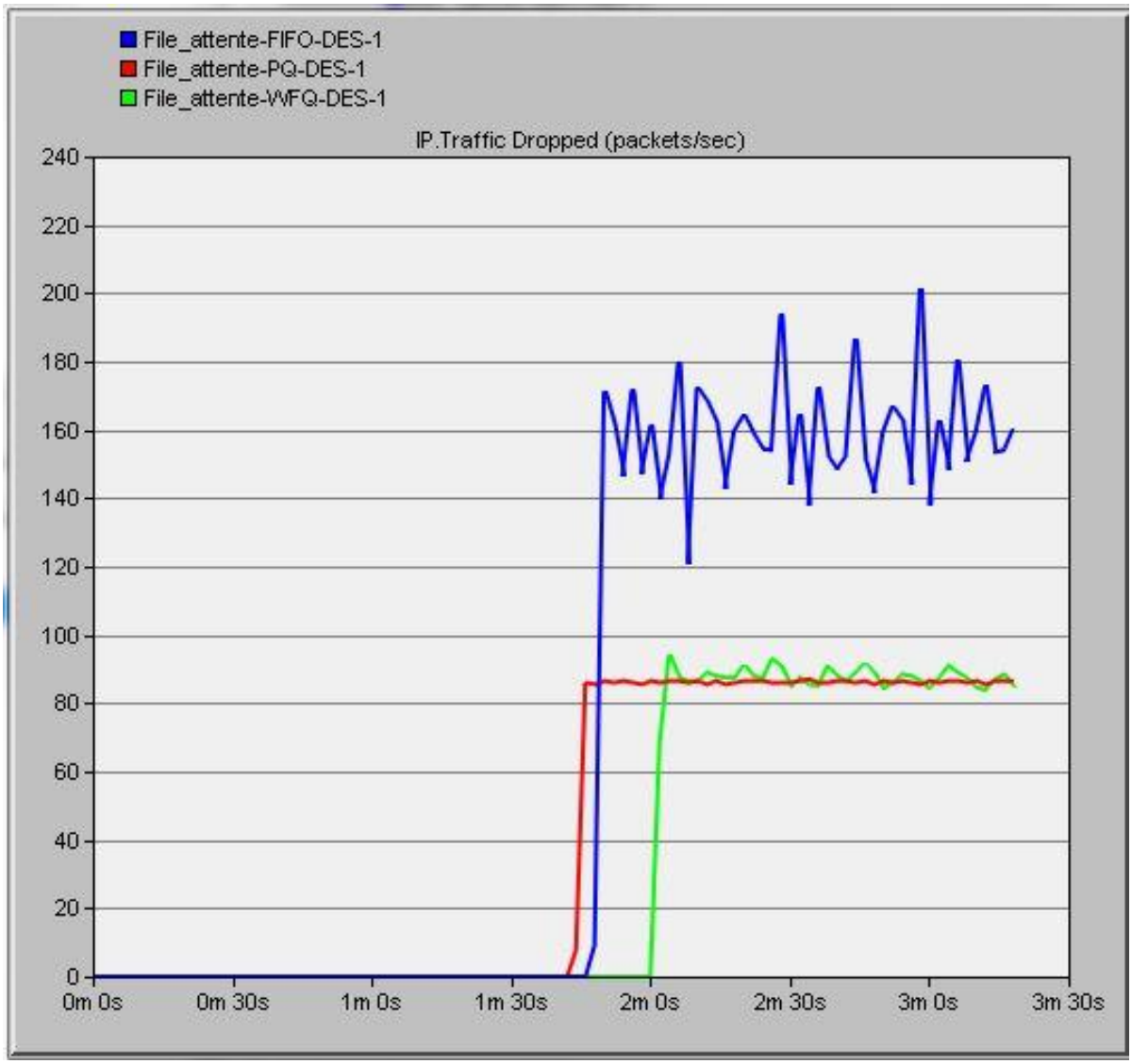


Figure 4.16 : *Trafics IP détruits*

Ces courbes permettent de dire que PQ et WFQ présentent de meilleures performances que FIFO. Ceci peut s'expliquer du fait que PQ et WFQ réalisent des différenciations de services permettant d'éviter la monopolisation des ressources par les trafics de grandes tailles et la perte des autres qui arrivent après.

Elles indiquent alors que PQ et WFQ réduisent les pertes de paquets et donc arrivent à remédier contre la congestion.

4.6.4.2 Temps d'attente moyen de bout-en-bout

L'évolution du temps d'attente moyen est représentée par la figure 4.17.

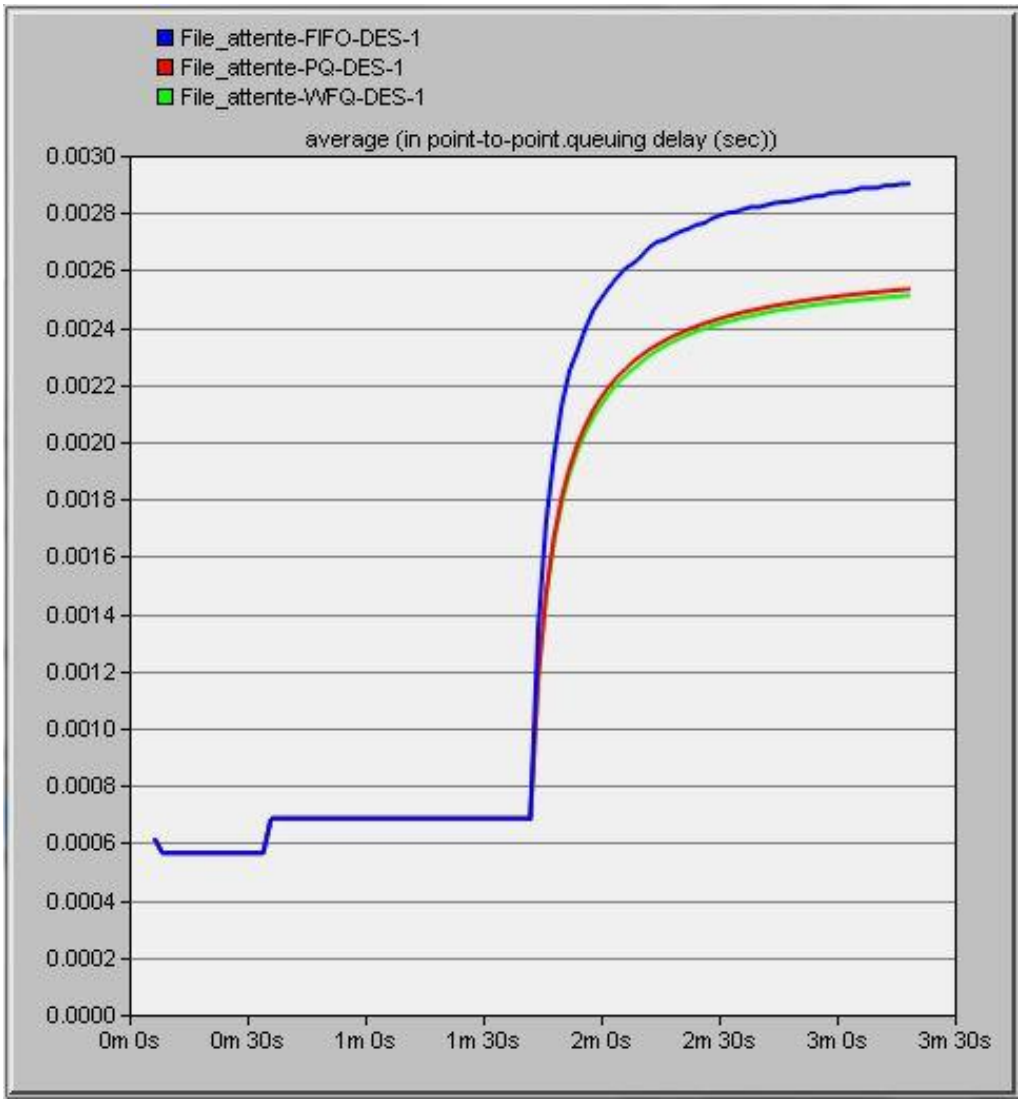


Figure 4.17 : *Temps d'attente moyen*

D'après la figure 4.17, on peut dire que PQ et WFQ améliorent le temps d'attente des paquets observé dans un réseau avec un simple FIFO.

En effet avec FIFO, tous les paquets arrivant attendent tous dans une même file jusqu'à ce que le traitement des paquets déjà dans la file soit terminé. Contrairement aux processus de PQ et WFQ qui présentent plusieurs files pour chaque application permettant ainsi d'améliorer les traitements d'ordonnancement au niveau du routeur.

4.6.4.3 Trafic FTP

Les paquets reçus FTP sont illustrées sur la figure 4.18

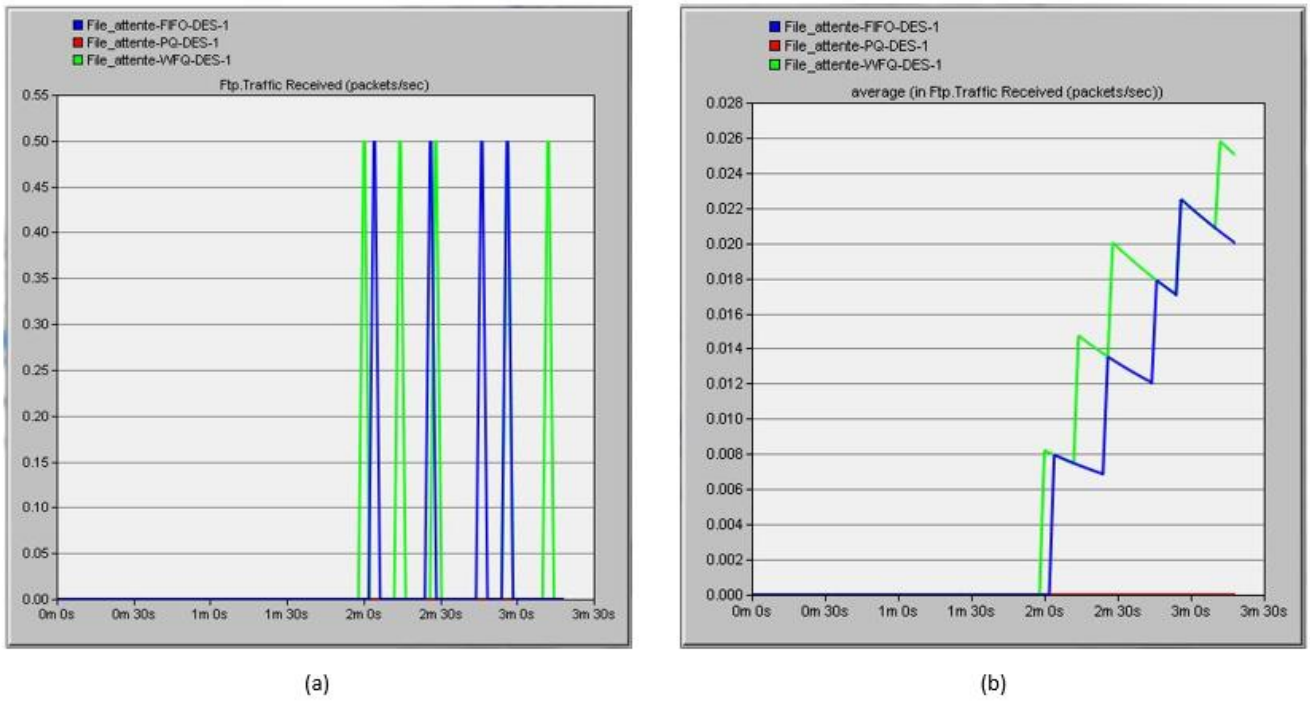


Figure 4.18 : *Paquets FTP : (a) reçus, (b) moyens reçus*

Sur la figure, on observe que les paquets FTP sont transmis par les algorithmes FIFO et WFQ tandis que PQ (courbe en rouge) ne laisse passer aucun.

En effet, dès qu'un paquet FTP arrive dans la file FIFO, elle le laisse passer. WFQ, lui, dirige les paquets FTP dans sa propre file et les laissent passer.

Pour PQ, dans notre cas, il existe trois files pour les trois applications :

- la file 1 de haute priorité pour la voix de ToS = 6
- la file 2 avec une priorité moyenne pour la vidéo de ToS = 4
- la file 3 avec une priorité normale pour FTP de ToS = 0

Ainsi, tant que les files de haute priorité ne sont pas vide, PQ ne traite pas les paquets FTP. Ce qui explique donc les absences des paquets FTP.

4.6.4.4 Trafic vidéo

La figure 4.19 illustre les trafics vidéo.

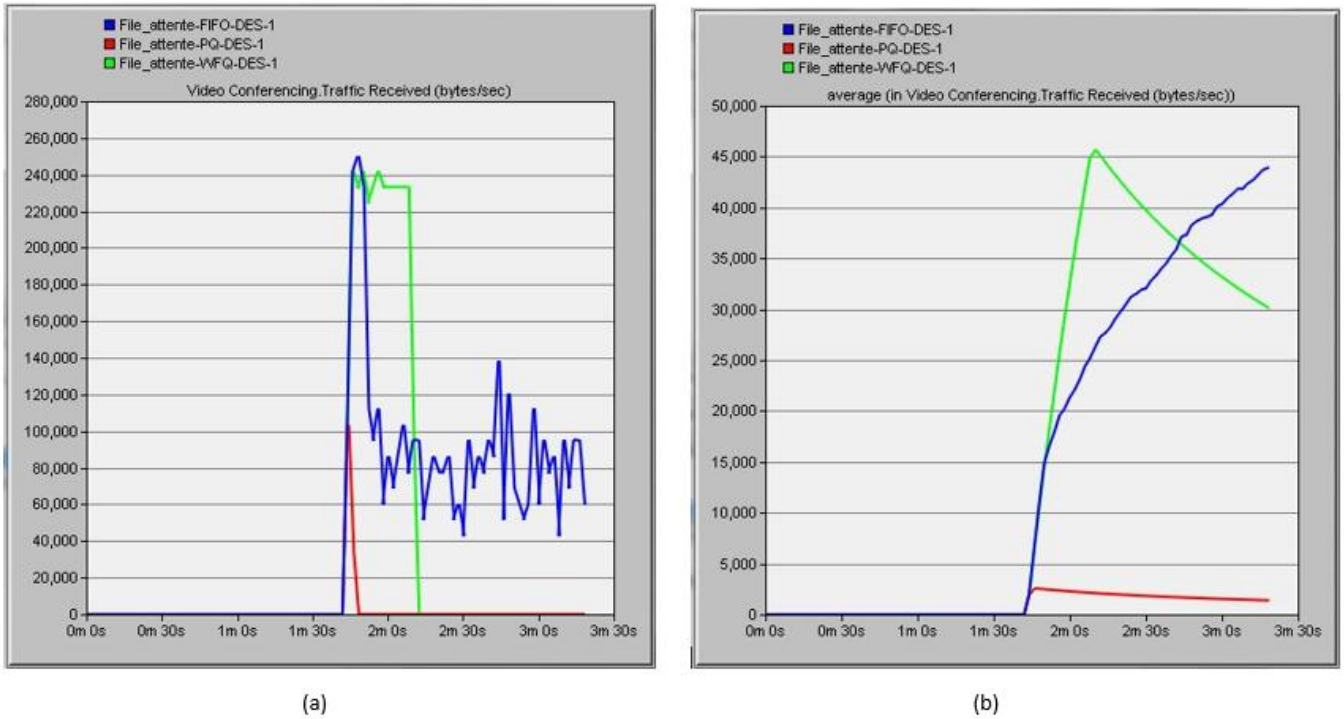


Figure 4.19 : *Paquets vidéos : (a) reçus, (b) moyens reçus*

Ces courbes nous montrent que les trafics vidéo sont transmis par les algorithmes FIFO et WFQ tandis que les paquets vidéo transmis par PQ sont presque nuls.

Ceci peut encore s'expliquer par les ordres de traitement utilisés par l'algorithme PQ : les trafics vidéo avec un ToS = 4 sont envoyés dans la file de priorité moyenne et attendent que la file supérieure (celle traitant les trafics voix) soit traitée avant de passer à son tour.

Cependant, on peut encore observer que WFQ permettent d'envoyer le plus de paquets vidéo par rapport à FIFO. Sur la figure 4.19 (a), les paquets envoyés par FIFO sont largement faibles par rapport à WFQ car il se peut que d'autres paquets intercalent entre les paquets vidéo dans la file FIFO.

4.6.4.5 Trafic Voix

La figure 4.20 illustre les trafics voix reçus.

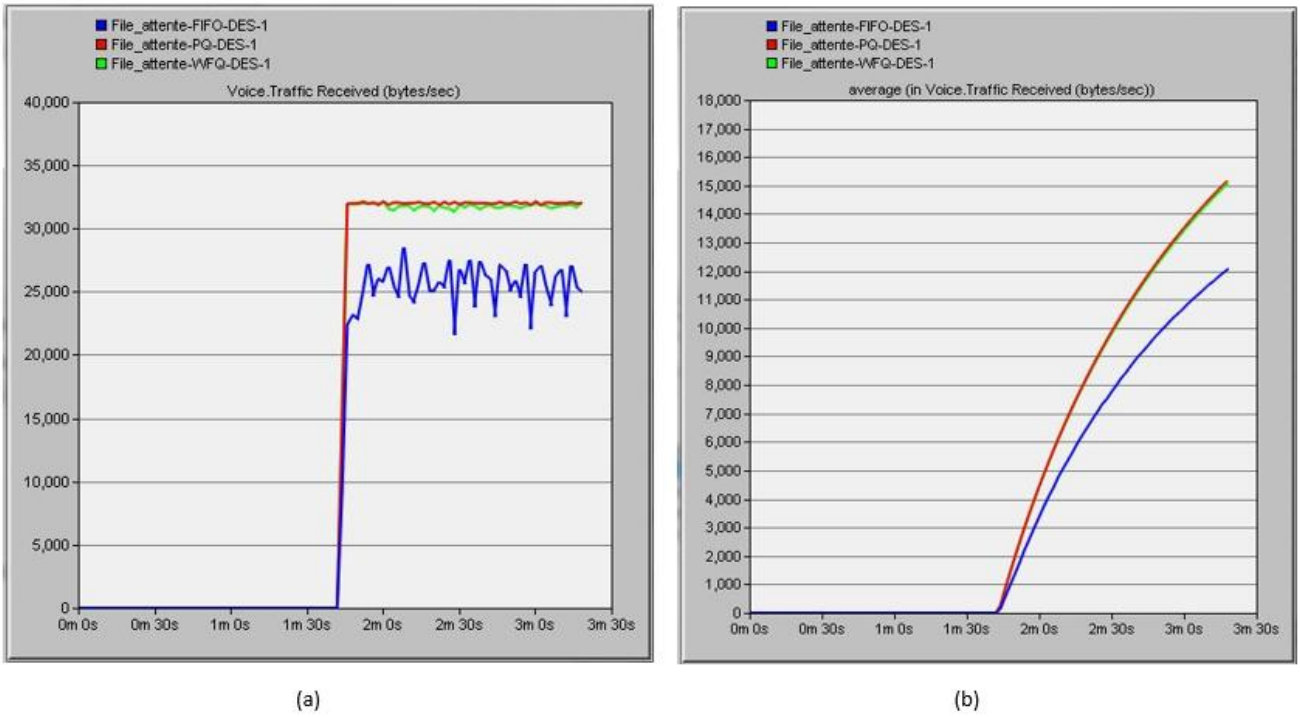


Figure 4.20 : *Paquets voix :(a) reçus, (b) moyens reçus*

Dans les figures 4.20 (a) et 4.20 (b), on observe que PQ et WFQ sont meilleurs par rapport à FIFO pour la transmission des trafics voix.

En effet, la voix présente un ToS=6 (le plus élevé) donc elle reçoit les meilleurs des services avec l'ordre prioritaire au niveau des files d'attente dans le cas des algorithmes PQ et WFQ.

Pour FIFO, les paquets voix ne sont transmis que lorsque les paquets qui les précèdent soient traités. Ce qui explique le nombre de paquets envoyés moins élevés.

D'autre part, les trafics voix sont de type « Streaming » ou à temps réel. Ils présentent des exigences en termes de latence (temps de transit) et de gigue.

Le tableau 4.04 présente les valeurs de ces paramètres avec leur niveau de qualité de service respectif.

Tableau 4.04: *Valeurs seuils des métriques de performance de la VoIP*

Niveau de service	Bon	Moyen	Mauvais
Délai de transit	D < 150 ms	150 ms < D < 400 ms	400 ms < D
Gigue	G < 20 ms	20 ms < G < 50 ms	50 ms < D

Comme la gigue est la variation du temps de transit, alors l'observation de ce dernier peut refléter la valeur de la gigue (Figure 4.21)

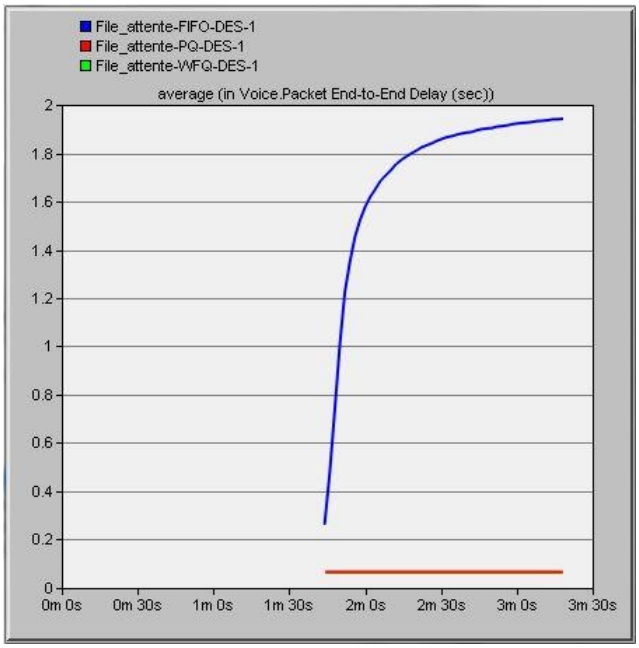


Figure 4.21 : *Latence moyenne des trafics voix*

On voit que les latences produites par PQ et WFQ sont presque la même et très faibles par rapport à celle de FIFO qui peut atteindre jusqu'à 2 secondes. D'après le tableau 4.04, on peut dire que l'ordonnanceur FIFO n'est pas performant pour l'application VoIP car le délai est supérieur à 400ms ce qui offre un très mauvais service en termes de qualité de la voix.

Si on fait un gros plan sur les courbes de latences pour le PQ et le WFQ, on a la figure 4.22 :

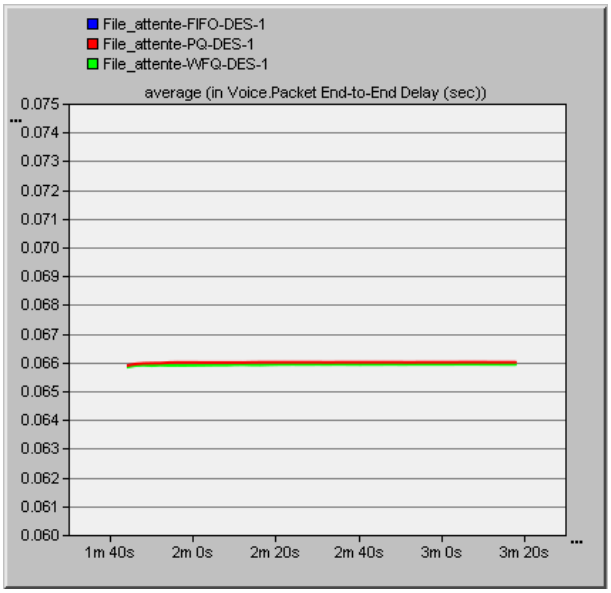


Figure 4.22 : *Zoom sur latences PQ et WFQ*

On observe que la valeur moyenne de la latence des paquets voix utilisant PQ et WFQ est de 0.066 secondes soit aux environs de 66ms. Cette valeur correspond à un très bon service et à une très bonne qualité de la voix d'après le tableau 4.04.
Ainsi, on peut dire que PQ et WFQ sont adaptés pour les applications critiques telles que la VoIP.

4.6.5 Synthèse des résultats

Les résultats obtenus nous permettent de faire les constatations suivantes :

- FIFO ne permet pas de donner plus d'importance (de poids) à un flux donné par rapport aux autres, alors que c'est nécessaire pour implanter des qualités de service.
- PQ présente cause les problèmes de « famines », c'est-à-dire dans le cas où la file « High » présentent trop de paquets à ordonnancer pénalisant ainsi les autres files inférieures. Dans notre cas, nous avons constaté que seuls les trafics voix arrivent à passer au détriment des trafics FTP et vidéo de priorités inférieures.
- WFQ est un apport significatif en termes de mécanisme de QoS par rapport aux deux approches précédentes du fait de son aspect dynamique tant au niveau de l'ordonnancement des flux en entrée que du service des files en sortie: aucun flux (FTP, vidéo ou voix) n'est négligée par rapport à d'autres. Ce qui le rend très performant.

4.7 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux mécanismes de gestion de la QoS dans les réseaux IP. Nous avons simulé un réseau incluant trois types d'applications (FTP, vidéo, VoIP) afin d'analyser les performances des algorithmes de gestion de file d'attente (RED, WRED) et les algorithmes d'ordonnancement à la sortie des files d'attente (FIFO, PQ et WFQ).
Nous avons constaté la performance de RED et WRED pour diminuer et éviter l'apparition de congestion dans le réseau. WRED présente un surplus au RED du fait qu'il prend en compte la classification des paquets afin de ne rejeter que ceux de faible priorité en premier en cas de congestion.
Pour les ordonnanceurs, nous pouvons dire que parmi les trois (FIFO, PQ et WFQ), WFQ est le plus performant et le plus adapté pour la gestion de la QoS pour les différents types d'application.

CONCLUSION GENERALE

La mise en œuvre de la QoS par classes de services nécessite l'implémentation dans les routeurs des mécanismes respectant la priorité et assurant le contrôle d'admission. De tels mécanismes font partie de l'ingénierie de trafic qui consiste essentiellement à établir un certain nombre de règles sur le dimensionnement des réseaux, le contrôle de leur performance, la gestion de leur ressource et leur planification.

Dans notre travail, les bases fondamentales du télétrafic dont l'étude des files d'attente et la modélisation des trafics ont été abordées.
Dans un second temps, nous avons aussi démontré que le trafic IP diffère du trafic classique par son caractère auto-similaire et à mémoire longue. Plusieurs modèles tels qu'ARIMA, FARIMA, FBM et FGN ont été exposés pour représenter ce trafic.
Les caractères du trafic IP ainsi que l'évolution de l'état de la file d'attente en fonction de la charge du réseau ont montré la nécessité d'implémenter des solutions de QoS au niveau des routeurs afin de contrôler et remédier au phénomène de congestion.

Les algorithmes de gestion de la file d'attente tels que RED et WRED sont des solutions préventives de la congestion tandis que les algorithmes d'ordonnancement (FIFO, PQ et WFQ) sont des solutions curatives en cas d'apparition de congestion.
D'après les résultats des différentes simulations qu'on a effectuées, nous pouvons dire qu'avec RED et WRED, la conservation d'une taille de la file d'attente relativement basse offre plusieurs avantages : augmenter la capacité d'absorber les rafales sans supprimer de paquets, diminuer le gaspillage de bande passante dû aux retransmissions de paquets, diminuer le délai moyen de passage des paquets dans le routeur, ce qui diminue les délais de transmission des paquets entre l'expéditeur et le destinataire. Cependant, WRED devance RED du fait qu'il utilise la classification de service lors du rejet de paquets permettant ainsi de garantir les exigences de certaines applications qui sont à temps réel tels que la voix.
Pour les différents ordonnanceurs, WFQ défie largement le mode classique FIFO et l'algorithme PQ car il offre plusieurs files pour chaque flux de paquets et traite cycliquement chaque file avec un poids correspondant à sa classe. Il permet ainsi de satisfaire tout type d'application circulant dans le réseau.

Nous pouvons déduire donc que ces deux types de solutions sont complémentaires pour assurer la bonne gestion active de la QoS dans les réseaux IP : le WRED pour éviter l'apparition des congestions et le WFQ pour y remédier quand le cas critique persiste.

L'étude sur les files d'attente qu'on a effectuée reste un vaste sujet de recherche pour trouver de meilleure solution de gestion.

Ainsi, en termes de perspective, notre analyse sur ces différents mécanismes de gestion de la QoS nous a permis de mettre en évidence la possibilité d'une nouvelle approche qui consistera à combiner en un seul et même algorithme les concepts d'évitement et de gestion des congestions en se basant sur les principes des algorithmes existants.

ANNEXE 1

RAPPELS MATHEMATIQUES

A1.1 Rappels sur les probabilités

A1.1.1 Probabilité

On se donne un phénomène que l'on observe au travers de la réalisation des événements. Ceux-ci sont notés A, B, etc. Sur cet ensemble d'événements, on définit la notion de probabilité.

On appelle probabilité d'un événement A, que l'on note P(A), un nombre réel positif attaché à cet événement, qui vérifie les conditions :

- $0 \leq P(A) \leq 1$;
- $P(\Omega) = 1$, c'est l'évènement certain et il a une probabilité égale à 1 ;
- Si A et B sont deux événements exclusifs, $P(A \cup B) = P(A) + P(B)$;
- $P(\emptyset) = 0$, c'est l'évènement impossible et il a une probabilité nulle.

A1.1.2 Variable aléatoire

Une variable aléatoire est une fonction qui associe un nombre réel x à la réalisation d'un événement aléatoire. Plus formellement, une variable aléatoire X est une application mesurable X de Ω dans R (Ensemble des nombres réels).

La fonction de répartition d'une variable aléatoire X à valeur réelle est définie par :

$$F(x) = P\{X < x\} \qquad \text{(A.01)}$$

A1.1.3 Produit de convolution

Soit deux fonctions f(t) et g(t), on appellera fonction de convolution ou encore produit de convolution de ces deux fonctions, la fonction suivante :

$$f(\tau) = \int_0^{\tau} f(t)\, g(\tau - t) dt \qquad \text{(A.02)}$$

Suivant cette définition, on voit immédiatement son intérêt pratique. Ainsi, si f(t) et g(t) représentent respectivement la densité de probabilité du temps de traversée de deux éléments en série dans un réseau, la probabilité que le temps de traversée global soit τ est bien la probabilité que le temps de traversée du premier élément soit t et celui du deuxième τ-t, pour toutes les valeurs de t possibles.

Le produit de convolution exprime la densité de probabilité de la somme de variables aléatoires

A1.1.4 Transformée de Laplace

La transformée de Laplace d'une fonction f(t) est définie par:

$$F^*(s) = \int_{-\infty}^{\infty} f(t)\, e^{-st} dt \tag{A.03}$$

Dans la plupart de nos cas d'application, nous aurons des fonctions telles que f(t) = 0 si t < 0. Alors, en faisant attention que la limite inférieure correspond en réalité à 0-, on écrira :

$$F^*(s) = \int_{0}^{\infty} f(t)\, e^{-st} dt \tag{A.04}$$

A1.1.5 Fonction caractéristique

On désigne par fonction caractéristique d'une variable aléatoire X, de fonction de répartition F(x), la fonction suivante :

$$\phi(u) = \int_{-\infty}^{\infty} e^{iux} dF(x) \tag{A.05}$$

Nous avons aussi :

$$\phi(is) = \int_{-\infty}^{\infty} e^{-sx} dF(x) = F^*(s) \tag{A.06}$$

On passera ainsi aisément de la fonction caractéristique à la transformée de Laplace.

A1.1.6 Processus aléatoire

On désigne par $(\Omega, \mathcal{F}, P)$ un espace de probabilité, T un ensemble arbitraire souvent appelé ensemble des indices et S espace métrique (espace d'état) muni de la tribu borélienne noté $\mathcal{B}(S)$. T peut faire référence au temps, à l'espace ou aux deux à la fois.

Un processus aléatoire est une famille de variables aléatoires X_i (c'est-à-dire, des applications mesurables) définies sur le même espace de probabilité $(\Omega, \mathcal{F}, P)$ indexé par T et à valeurs dans S. Un processus aléatoire, dont l'index est le temps, est un processus stochastique noté $\{X_t\}_{t\in T}$.

A1.2 Processus de Poisson

A1.2.1 Processus de comptage

Ce processus qui sera noté $N = \{N_t; t \geq 0\}$ représente un compteur qui s'incrémente à chaque arrivée d'un client (paquet, unité de donnée, ...) dans le système, avec N_t une variable aléatoire discrète positive qui représente le nombre de clients arrivés dans l'intervalle [0, t] et t est une

variable d´déterministe réelle positive qui représente le temps. Le processus de comptage est dit simple si les arrivées sont simples. Dans ce cas le compteur s'incrémente d'une unité juste après chaque arrivée. Les instants d'arrivée T = {Ti; i = 1, 2, …} sont supposées être aléatoires.

A1.2.2 Processus de comptage à accroissement indépendant

Le processus de comptage N est dit à accroissements indépendants (ou encore sans m´mémoire) si pour tout instant t et tout temps s, le nombre d'arrivées dans l'intervalle [t, t+s] donné par $N_{t+s} - N_t$ est indépendant du passé avant l'instant

A1.2.3 Processus de comptage à accroissements stationnaires

Le processus de comptage N est dit à accroissements stationnaires si le nombre d'arrivées dans un intervalle de durée s ne d´dépend que de la durée de cet intervalle et non de la position de cet intervalle sur l'axe du temps. En d'autres termes, la distribution statistique de N_s est la même que celle de $N_{t+s} - N_t$ pour tout $t \geq 0$, en posant $N_0 = 0$.

A1.2.4 Définition du processus de Poisson

On appelle processus de Poisson $N = \{N_t; t \geq 0\}$ de paramètre λ un processus de comptage simple à accroissements indépendants et stationnaires tel que :

$$P\{N_t = k\} = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \qquad \text{(A.07)}$$

A1.3 Processus de Markov

On dit qu'un processus $X = \{X_n; n \geq 0\}$ est une chaîne de Markov sur un espace E de probabilité de transition P si l'on a, pour tout $x_0, \dots, x_n, x_{n+1}$, on a :

$$P(X_{n+1} = x_{n+1} | X_n = x_n, \dots, X_0 = x_0) = P(X_{n+1} = x_{n+1} | X_n = x_n) = P(x_n, x_{n+1}) \qquad \text{(A.08)}$$

Pour de tels processus, la meilleure prévision qu'on puisse faire du futur, connaissant le passé et le présent, est identique à la meilleure prévision qu'on puisse faire du futur, connaissant uniquement le présent : si on connait le présent, la connaissance du passé n'apporte pas d'information supplémentaire utile pour la prédiction du futur.

En d’autres termes, la notion de propriété markovienne définit donc une classe de processus discrets ou continus, à valeurs discrètes ou continues, qui repose sur l'hypothèse selon laquelle l'avenir ne dépend que de l'instant présent.

Le processus de Markov fournit un outil simple de modélisation d’une classe particulière de systèmes à espace d’états discret. L’analyse des processus de Markov est un préliminaire nécessaire à l’étude des systèmes de files d’attente.

ANNEXE 2
PROTOCOLES DE ROUTAGES DYNAMIQUES

Bien que l'idée d'une route statique est séduisante par sa facilité de mise en œuvre, elle est devient fastidieuse pour l'administrateur réseau pour l'organisation de grands réseaux où le nombre de routeurs est considérable.

La présence de plusieurs routes possibles pour rejoindre une destination implique de facto l'usage d'un protocole de routage dynamique et l'existence de ces plusieurs routes est une nécessité pour assurer la redondance du service, voire même l'équilibrage du trafic sur plusieurs liens.

Les protocoles de routage permettent donc aux routeurs de gérer et de mettre à jour automatiquement leurs tables de routage.

Si plusieurs chemins existent pour atteindre une destination précise, ils choisissent le plus optimal tout en mettant en réserve un autre meilleur chemin en cas d'indisponibilité du premier.

Il existe deux familles de protocoles de routage dynamique :

- L'IGP ou Interior Gateway Protocol
- L'EGP ou Exterior Gateway Protocol

L'IGP permet d'acheminer les données à l'intérieur d'un système autonome tandis que l'EGP permet de router les données entre différents systèmes autonomes contrôlés par des administrateurs différents. Un système autonome est un ensemble de réseaux gérés par un administrateur commun et qui suit les même règles de routage, c'est-à-dire qu'il y a partage d'une stratégie de routage commune.

Les protocoles de routage intérieurs IGP sont classés, selon l'algorithme utilisé, en protocole à vecteur de distance et en protocole à état de lien (ou à état de liaison).

A2.1 Algorithme à vecteur de distances

Les algorithmes de routage à vecteur de distances se basent sur l'algorithme de Bellman-Ford. Ils conduisent les routeurs à transmettre à leurs voisins réseaux immédiats une copie de leur table de routage. Ces tables se modifient au fur et à mesure de leur propagation car chaque route est associée à une métrique qui croît par défaut d'une unité au passage de chaque routeur. La mise à jour se fait donc de manière périodique.

Le terme vecteur de distances est employé parce que la propagation des routes s'effectue sous la forme de vecteurs : " Pour atteindre telle destination, il faut passer par ce routeur et la métrique associée vaut cette valeur " ; donc une direction et une métrique, d'où l'analogie avec un vecteur.

Le choix de la meilleure route est établi par chaque routeur en considérant la valeur minimale de cette métrique pour toutes les routes qui aboutissent à la même destination. Seule la meilleure route est propagée, les autres sont oubliées. Pour ces considérations on dit que le calcul de la route est distribué et par conséquence chaque routeur n'a pas la connaissance de la topologie globale du réseau : il n'en connait qu'une version interprétée par ses voisins.

Les protocoles à vecteur de distance sont les protocoles RIP, IGRP, EIGRP

A2.1.1 RIP

La métrique utilisée pour déterminer le meilleur chemin est le nombre de sauts, c'est-à-dire le nombre de routeurs que le paquet doit traverser avant d'arriver à la destination finale. Le nombre de sauts maximal avant que le paquet soit éliminer est 15. Les mises à jour de routage se font tous les 30 secondes. Il existe deux versions de RIP : le RIPv1 et RIPv2.

Tableau A.01 : *RIPv1 et RIPv2*

RIPv1	**RIPv2**
Facile à configurer	
Utilisation de protocole de routage par classe (classful)	Utilisation de protocole de routage CIDR (classless)
Les mises à jour se font par broadcast	Les mises à jour se font par multicast sur 224.0.0.9
Aucune information sur les sous-réseaux dans la mise à jour	Masques de sous-réseaux envoyés avec la mise à jour
Aucune authentification	Authentification des voisins dans les mises à jour
Utilisation d'un même masque de sous-réseau	Support du VLSM

Malgré ses avantages, RIP présente un temps de convergence lent et l'utilisation du nombre de sauts comme métrique n'est pas très efficace pour choisir le meilleur chemin ; de plus ce nombre de sauts est limité à 15.

A2.1.2 IGRP et EIGRP

L'IGRP (Interior Gateway Routing Protocol) et EIGRP (Enhanced IGRP) sont des protocoles de routage à vecteur de distance développés par Cisco. EIGRP est une version améliorée d'IGRP mais les deux protocoles sont compatibles.

Tableau A.02 : *IGRP et EIGRP*

	IGRP	**EIGRP**
Classe de Protocole	A vecteur de distance	A vecteur de distance mais à état de lien lors de la mise à jour
Nombre maximum de sauts	255	224
Métriques utilisées	Bande passante, fiabilité, délai, charge ; 24bits	Métriques composites ; 32 bits
Mise à jour	Toutes les 90s de façon multicast	Lors modification du réseau de façon multicast
Distance administrative	100	90
Bande passante	Consommation de la bande passante	Moins de bande passante utilisée
Spécificités		Support du CIDR et VLSM
		Découverte des voisins

A2.2 Algorithme à états de liens

Les algorithmes à états de liens bâtissent les tables de routages différemment. Chaque routeur est responsable de la reconnaissance de tous ses voisins, plus ou moins lointains, à qui il envoie une liste complète des noms et des coûts (en termes de bande passante, par défaut) contenu dans une base de données à sa charge et qui représente l'intégralité de tous les routeurs du nuage avec lesquels il doit travailler.

Chaque routeur a donc une connaissance exhaustive de la topologie du " nuage " dans lequel il se situe et c'est à partir de cette représentation qu'il calcule ses routes à l'aide d'un algorithme connu de recherche du plus court chemin dans un graphe : celui de Dijkstra. Sa réponse est plus rapide car la mise à jour se fait seulement en cas de modification du réseau.

A2.2.1 OSPF (Open Shortest Path First)

Le protocole de routage à états liens le plus répandu est l'OSPF ou Open Shortest Path First. Il permet la connaissance exacte de la topologie du réseau avec découverte des voisins. La métrique utilisée pour la sélection de la meilleure route est la bande passante des liaisons. Plus la bande

passante est grande plus le chemin est optimal. La mise à jour de routage ne se fait qu'à chaque modification topologique du réseau à partir d'une adresse multicast diminuant ainsi l'utilisation de la bande passante. C'est aussi un protocole de routage classless supportant le VLSM dont le domaine est dépourvu de boucle de routage.

A2.2.2 Valeur des états de liens

Le coût des liens, nommé également la métrique, agit directement sur le choix d'une route plutôt qu'une autre. Le constructeur Cisco préconise une formule qui est reprise partout :

$$cost = \frac{100000000}{bande\ passante\ en\ bits/s} \tag{4.12}$$

Tableau A.03 : Valeur du coût pour des débits connus

Média	**Coût**
Liaison série 56 Kbits/s	1785
T1 (série 1544 Kbits/s)	64
E1 (série 2048 Kbits/s)	48
Token Ring 4 Mbits/s	25
Ethernet 10 Mbits/s	10
Token Ring 16 Mbits/s	6
Ethernet 100 Mbits/s	1

ANNEXE 3

EXTRAITS DE CODE ET CONFIGURATIONS

A3.1 Extrait codes sous MATLAB

```
% Nombre moyen de client dans le système en fonction du taux de trafic
Ro=[0:0.1:1];
Ns=Ro./(1-Ro);
subplot(3,1,1), plot(Ro,Ns,'--rs','LineWidth',2,'Color','blue')
xlabel('Le taux de trafic');
ylabel('Nombre de client du système');
Title('Nombre moyen de client dans le système en fonction du taux de trafic');
%Nombre moyen de client dans la file d'attente en fonction du taux de
%trafic
Ro=[0:0.1:1];
Na=(Ro.^2)./(1-Ro);
subplot(3,1,2), plot(Ro,Na,'--rs','LineWidth',2,'Color','green');
xlabel('Le taux de trafic');
ylabel('Nombre de client dans la file');
Title('Nombre moyen de client en attente en fonction du taux de trafic');
```

A3.2 Configurations des applications (FTP, Vidéo, Voix) sous OPNET

A3.2.1 Application FTP

Les paramètres utilisés pour l'application FTP sont donnés par la figure A.01.

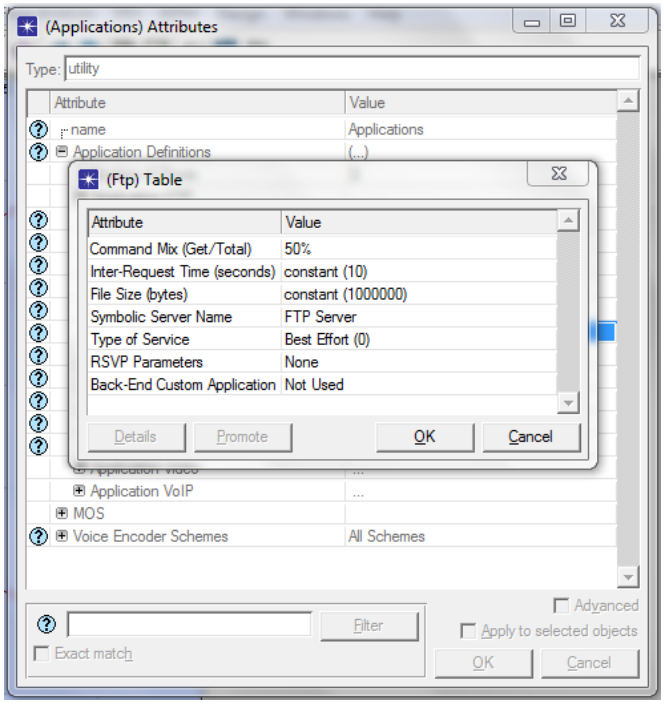


Figure A.01 : *Configuration FTP*

A3.2.2 Application Vidéo

Les paramètres utilisés pour l'application Vidéo sont donnés par la figure A.02.

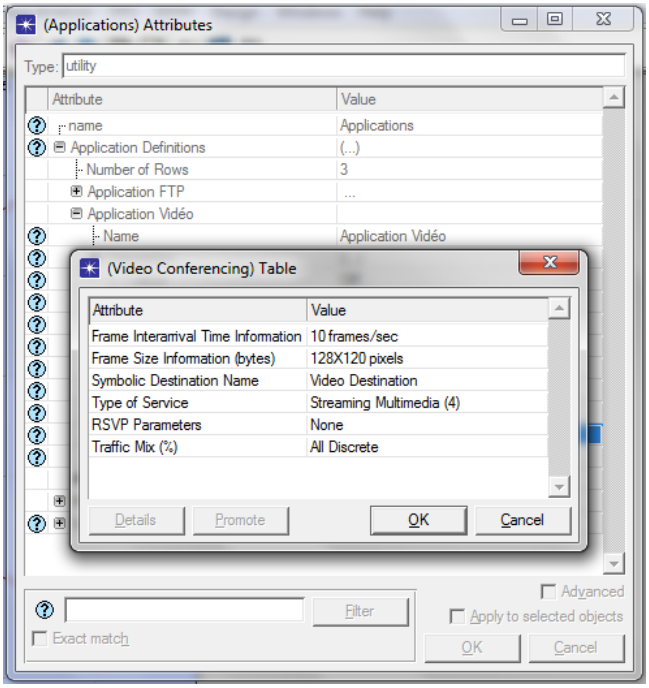


Figure A.02 : *Configurations Vidéo*

A3.2.3 Application Voix

Les paramètres utilisés pour l'application Vidéo sont donnés par la figure A.03.

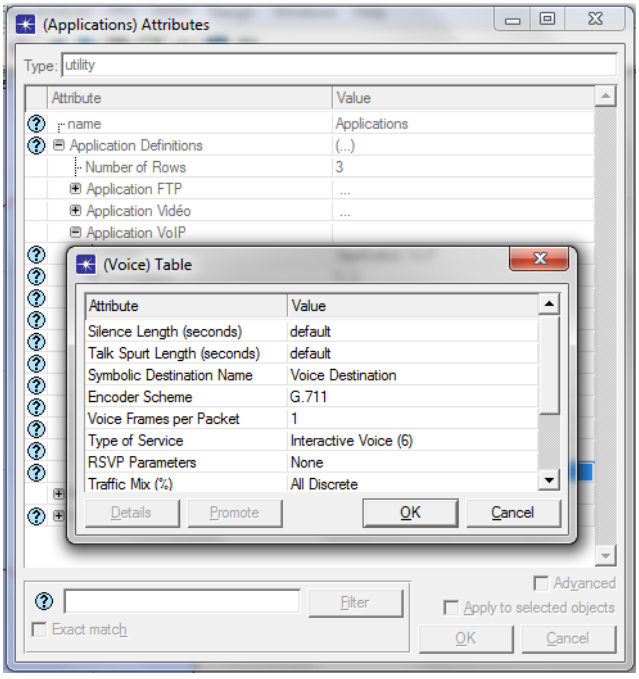


Figure A.03 : *Configurations voix*

BIBLIOGRAPHIE

[1] J. Roberts, *«Engineering for Quality of Service", In Self-similar network traffic and performance evaluation, »*,2000

[2] A. Proutiere, *« Insensibilité et bornes stochastiques dans les réseaux de files d'attente »*, France Telecom R&D - Ecole Polytechnique, Thèse de Doctorat, 2003

[3] G. Fiche, G. Hébuterne, *« Trafic et performances des réseaux de télécoms »*, Hermes Science Publications, 2005

[4] S.Tohmé et R.Naja, *« Dimensionnement de réseaux »*, Master IRS, Mars 2005

[5] D. T. T. Ha, *« Les files et les réseaux zéro-automatique »*, Université Paris Didérot, Thèse de Doctorat, Décembre 2007

[6] D. Tilloy, *« Introduction aux réseaux TCP/IP »*, Support de cours, Institut Universitaire de Technologie d'Amiens, A.U : 1998-1999

[7] Cisco Networking Academy, *« CCNA 2.1.2 »*, Cisco Systems, 2000

[8] T. J. Rakotonirina, *« Modélisation et performance des trafics ip dans les réseaux de télécommunication »*, ESPA, DEA TASI, Février 2015

[9] M. F. Zhani, « *Prévision du trafic internet -modèles et applications »*, Université du Québec à Montréal, Thèse de Doctorat, Juin 2011

[10] M. Chtioui, « *L'exploitation de l'auto-similarité pour la prédiction du trafic internet »*, Université du Québec à Montréal, Thèse Doctorat, Mars 2006

[11] J. Kashmar, « *Gestion adaptative de la Qualité De Service dans un réseau actif IP »*, INT-Evry France, DEA, Mars 2004

[12] Cisco Networking Academy, *« CCNA Discovery 4.0 »*, Cisco Systems, 2007-2008

[13] L.Peterson, B. S. Davie, *« Computer Networking, a system approach »*, 3è Edition, 2003

[14] Microsoft Training and Certification, « *Module 2 : Planification et optimisation d'un réseau TCP/IP physique et logique* », Microsoft Corporation, 2003

[15] G. Montcouquiol, *« Théorie des graphes »*, Support de cours, IUT Orsay, AU : 2006-2007

[16] S. HOCEINO, *« Techniques d'Apprentissage par Renforcement pour le Routage Adaptatif dans les Réseaux de Télécommunication à Trafic Irrégulier »*, UNIVERSITE PARIS XII – VAL DE MARNE, Thèse de Doctorat, 2004

[17] Natachia K. J-S., *« QoS IP : Modèles IntServ/DiffServ »*, ENIC Telecom Lille 1, France, 2003

[18] A. Lagnoux, *« Processus stochastiques et modélisation»*, Support de cours, Le Mirail Université de Toulouse, AU : 2012-2013

[19] Stephan R., *« Modélisation stochastique - Files d'attente »*, HEIG-Vd, Master of Science and Engineering, 2009

[20] A. Simon, *« Principes de mécanismes de Qualité de Service, mesures de performances et mises en oeuvre sur Lothaire »*, Centre Interuniversitaire de Ressources Informatiques de Lorraine, France, 2011

[21] G. Leduc, L. Kuty, « *Spécification formelle des mécanismes de support des qualités de service dans l'Internet »*, Support de cours, Université de Liège, Institut Montefiore, 2010

[22] Z. Mammery, *« Ordonnancement de paquets »*, Support de cours Master de recherche, option IT, 2008