Une brève introduction aux mécanismes pour la Qualité de Service

Chaput Emmanuel

2015-2016

Notes :

Notes :

Introduction

Notes :

Le problème

- Réseaux à commutation de paquets
  - Multiplexage temporel asynchrone
- Pas de circuit virtuel
  - Pas d'identification des flux

Altération du profil temporel des flots de paquets

Inéquité lors de contention entre les taux de perte des flots

Pas de traitement spécifique en fonction de besoins différents

Comportement *best effort*

- Le réseau fait ce qu'il peut
- Sans garantir quoi que ce soit

Notes :

Objectifs

Fournir des mécanismes permettant d'assurer de la "Qualité de service" dans les réseaux de paquets.

- Distribuer "équitablement" le débit des liens
  - Quelle granularité ?
  - Quelle équité ?
  - Quelle métrique ?
- Isoler les trafics entre eux
  - Éviter que le comportement d'un dégrade le service des autres
- Accroître les performances du service fourni

  Délai de bout en bout
  Gigue sur un flux
  Débit sur le chemin
  Perte de paquets

Notes :

Mécanismes =/= architecture

Mise en œuvre de la qualité de service

- Une architecture (*IntServ*, *DiffServ*, ...)
- Des protocoles (RSVP, DSCP, ...)
- Des mécanismes (ordonnancement, lissage, ...)

Les mécanismes sont la "cheville ouvrière" de la QoS

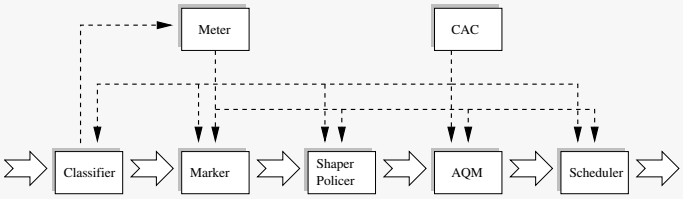
- Mécanismes non liés à une architecture bien que
- Souvent implantés pour assurer des services définis dans le cadre d'une architecture

Notes :

Schéma général

Notes :

Les fonctions



Présence et localisation dépendantes

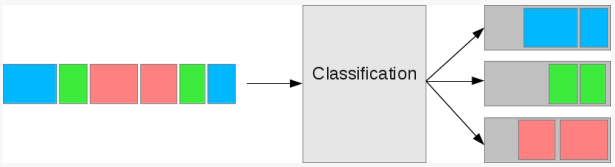
- De l'architecture
  - *IntServ [3], DiffServ [1], . . .*
- De l'entité dans l'architecture
  - Routeur de frontière, de cœur, . . .

Notes :

Classifier

Objectif : réaliser une ségrégation des paquets

- Ségrégation par flot (à la *IntServ*)
- Ségrégation par classe (à la *DiffServ*)
- Fondée sur
  - Des informations contenues dans les paquets
  - Une fonction de hachage
- Nécessaire du fait de l'absence de connexion IP



Notes :

Meter

Objectif : mesurer les caractéristiques (temporelles) d'un ensemble de paquets

- Caractéristiques selon critères de ségrégation
- Vérification de conformité
  - À un profil de trafic négocié
- Comparaison à des seuils
  - Éventuellement négociés
  - Éventuellement dynamiques

Notes :

Marker

Objectif : ajouter des informations pertinentes au paquet

- Emplacement prévu dans les entêtes
  - Champ TOS d'IPv4, ...
- Encapsulation supplémentaire
  - Label MPLS, ...
- Structure de données associée
  - Traitement local
- Pemet un traitement plus rapide en aval

Notes :

Shaper

Objectif : Lisser le trafic

- Le rendre conforme à un profil temporel
- Évite les bursts en aval
- Introduction de "délai" entre paquets
- Implanté dans le scheduler

Notes :

Policer

Objectif : contraindre le trafic

- Le rendre conforme à un contrat
- Destruction/remarquage de paquets
- Pas de lissage
- Isolation des trafics

Notes :

Active Queue Management

Objectif : diminuer les risques de congestion

- Gestion active des files d'attente [2]
- Éviter les files d'attente pleines
  - Capacité à accueillir les bursts
  - Minimiser le temps de traversée
- Prévenir la congestion plutôt que la subir
  - Définition de taux de remplissage seuils
  - Actions curatives (anticipées) ou préventives
  - Actions plus équitables

Notes :

Scheduler

Objectif : réaliser le multiplexage temporel

- Utiliser "équitablement" le débit disponible
- Appliquer les choix faits lors des étapes précédentes
- Implantation intègre les fonctions de shaper/policer
- Éventuellement non work conserving (*cf* page suivante)

Notes :

Ordonnancement *Work Conserving*

- Ordonnancement *Work Conserving*
  - Un paquet en attente est émis dès que le support est disponible
  - Utiliser au mieux la bande passante
- Ordonnancement *Non Work Conserving*
  - Un paquet en attente doit de plus être éligible pour être émis
  - Attendre un paquet plus prioritaire
  - Mettre en place du lissage

Notes :

Connexion Admission Control

Objectif : Vérifier qu'un contrat peut être assuré

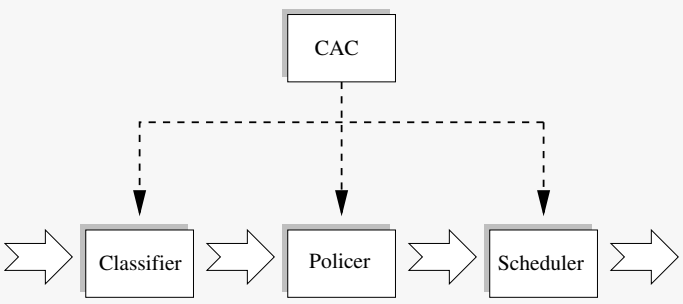
- Négociation de paramètres de QoS
- Associé à de la réservation de ressource
- Nécessite un protocole orienté connexion ou un équivalent
- Peut permettre l'isolation des trafics

Notion de SLA

- *Service Level Agreement*
- Accord entre le client et l'opérateur
- Description des paramètres de QoS

Notes :

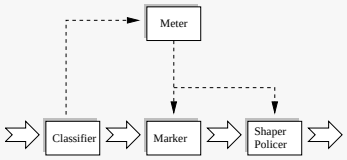
Un exemple : *IntServ*



- Ressources réservées par RSVP

Notes :

Un exemple : *DiffServ* (edge router)



- Ségrégation évoluée (MF-classifier)
- Marquage simple (DSCP) pour le domaine
- Policing (ingress router)
- Shaping (egress router)

Notes :

Un exemple : *DiffServ* (core router)

- Ségrégation simple
- Comportement conforme à un PHB

Notes :

Exemples de classifieurs

Notes :

Deep packet inspection

Notes :

Deep packet inspection

- Analyser le contenu des paquets
  - Aux niveaux réseau ou transport
    - Par exemple Netfilter sous Linux
  - Niveau applicatif
    - Par exemple L7 filter ou Hippie sous Linux
- Effectuer un traitement en fonction de cette analyse
- Très gourmand en ressources
- Confidentialité ?

Notes :

Le champ DSCP

Notes :

Le champ DSCP

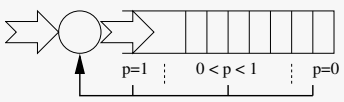
- DiffServ Code Point
- Dans IPv4
  - Utilisation de l'ex champ "Type of Service"
- Dans IPv6
  - Utilisation du cham "Traffic Class"
- 6 bits permettant de définir
  - Quelques classes
    - EF, AF, BE
  - Èventuellement des priorités
    - Classe AF)

Notes :

Exemples d'AQM

Notes :

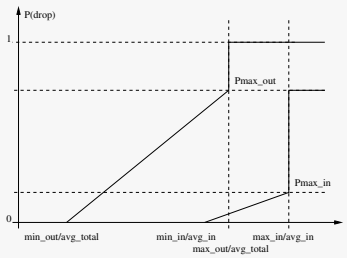
Random Early Detection



- Observation de l'état de la file
- Définition d'une probabilité en fonction de l'état
- Marquage/destruction de paquets choisis aléatoirement
- Désynchronisation des TCPs
- Première proposition en 1993 [6]

Notes :

RED with In and Out



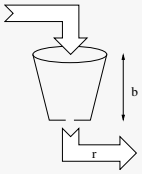
- Double RED [4]
  - Paquets marqués *in* ou *out*
- Perdre prioritairement des paquets hors-profil
- Indépendant du marquage des paquets
- Quel paramétrage ?

Notes :

Exemples de shaper

Notes :

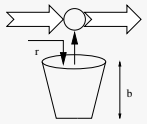
Leaky bucket



- Les paquets entrent dans le “seau”
- Sortie du seau à débit borné (`r`)
- Ici, le leaky bucket est utilisé pour faire du *shaping*
- Si capacité (`b`) finie : policing

Notes :

Token bucket



- Chaque paquet (octet) consomme un jeton
- Bursts possibles en sortie (limités par b)
- Paquets hors profil (pas de jeton)
  - Retardés (shaper)
  - Détruits/déclassés (policer)
- Le *leaky bucket* peut également être utilisé de la sorte

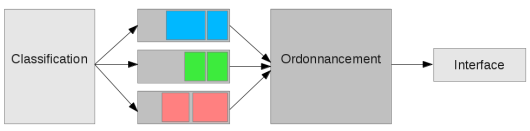
Notes :

Algorithmes d'ordonnancement

Notes :

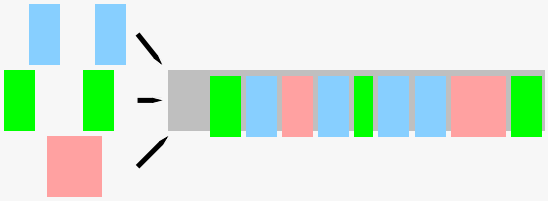
Positionnement

- Sur un routeur
- Après la phase de routage
- Après une éventuelle classification
- Juste avant l'émission sur l'interface de sortie
  - Indépendance entre les interfaces



Notes :

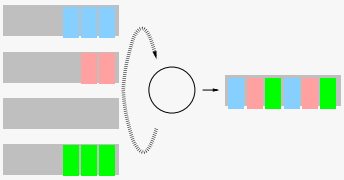
First Come First Served



- Fonctionnement le plus simple à implanter
  - Mécanisme le plus “naturel”
- Éventuelles pertes gérées selon la politique “*Tail drop*”
- Pas de distinction entre trafics
  - Pas de classification en amont

Notes :

Round Robin



- Classification du trafic
  - Besoin d'un mécanisme qui place les paquets dans des files virtuelles
- Simple à implanter
- Certaine isolation entre les trafics
  - Un paquet de chaque file est émis à chaque tour
- Équitable en paquets : en débit ?
- Quelle pondération entre les flux ?
  - *Weighted Round Robin*

Notes :

Deficit Round Robin

- Constat : packet round robin non équitable
  - Lorsque la taille des paquets est variable
- Service pondéré (comme WFQ) :
  - Q_i bits *max* par tour pour la file i
- Prise en compte du déficit (Q_i - service) au tour suivant
  - Profiter du retard pris sur les tours précédents
- Initialement défini par [12]

Notes :

Deficit Round Robin : algorithme

Q_i nombre max de bits du flux i transmis par cycle
DC_i déficit du flux i
L_i^k nombre de bits du paquet k du flux i

Algorithme

- $DC_i = 0$
- À chaque cycle, pour chaque flux i actif
  - $DC_i \leftarrow DC_i + Q_i$
  - tant qu'il existe un paquet k en attente tel que $L_i^k \leq DC_i$
    - émettre le paquet
    - $DC_i \leftarrow DC_i - L_i^k$

Notes :

Fair Queuing

- Simulation d'un round-robin fluide
  - Algorithme théorique bit à bit
- À la réception d'un paquet
  - Calcul de sa date de départ théorique
- Émission ordonnée selon les dates théoriques
  - Mais à des dates différentes ...
- [9] [10] [5]

Notes :

Fair Queuing : principe

Notation

$R(t)$ nombre de cycles du round robin à t
$N_a(t)$ nombre de flux actifs à t
a_i^k date d'arrivée du paquet i du flux k
L_i^k nombre de bits du paquet i du flux k
S_i^k et F_i^k dates de début et fin de service du paquet i du flux k *en nombre de cycles*

Propriétés

$N_a(t) = card(\{k, F^k_{max(i, a_i^k \leq t)} \geq t\})$

$F_i^k = S_i^k + L_i^k$

$S_i^k = max(F_{i-1}^k, R(a_i^k))$ (respect de l'ordre !)

$\frac{\partial R(t)}{\partial t} = \frac{1}{N_a(t)}$ si 1 bit par unité de temps

Notes :

Fair Queuing : algorithme

- Réception d'un paquet : calcul de sa date de fin de service par un round-robin bit à bit
- Support libre : émission du paquet de plus faible F_i^k
- On ne s'intéresse pas aux dates réelles puisque $r(t)$ est strictement croissante
- Durée de cycle variable (arrivée d'un paquet sur un flux inactif)
  - Dates (réelles) de départ théorique à recalculer
  - Dates en $R(t)$ inchangées !

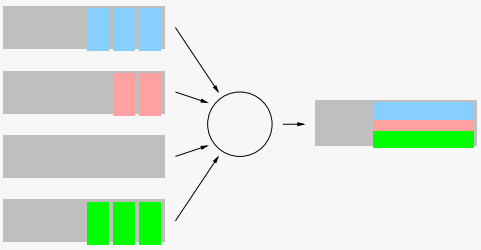
Notes :

Stochastic Fair Queuing

- Constat du Fair Queuing : lourd à implanter à haut débit (gestions des files)
- Flots répartis selon une méthode de hachage
- Hachage modifiée régulièrement
- [8] [7]

Notes :

Generalized Processor Sharing (GPS)



- Service fluide pondéré (débit de sortie ϕ_i pour flux i)
- Chacun obtient $\frac{\phi_i}{\sum \phi_i}$
- Extrèmement équitable
- Impossible à implanter

Notes :

Packet GPS

- Weighted Fair Queuing (pondéré par ϕ_i, où $\sum \phi_i = 1$)
- Pour chaque paquet : estimation de sa date de sortie sur un GPS
- Paquets émis dans l'ordre de ces dates
- [5] [11]

Notes :

Packet GPS : principes

Notations du Fair Queuing plus

$V(t)$ temps virtuel évoluant proportionnellement à $\frac{1}{\sum_{i \in \{actifs\}} \phi_i}$ si 1 bit par unité de temps

Propriétés

$F_i^k = S_i^k + \frac{L_i^k}{\phi_i}$

$S_i^k = max(F_{i-1}^k, V(a_i^k))$

$\frac{\partial V(t)}{\partial t} = \frac{1}{\sum_{i \in \{actifs\}} \phi_i}$ si 1 bit par unité de temps

Notes :

Virtual Clock

- Inspiré du TDM
- Calcul d'une date de départ théorique
- Date calculée en supposant un débit constant
- Émission ordonnée selon les dates théoriques
- Isolation des flux
- [13]

Notes :

Virtual Clock : principes

Notations

r_i débit moyen associé au flot i ($i \in [1 \dots n]$)

S_i^k et VC_i^k dates de début et fin de service du paquet i du flux k

a_i^k date d'arrivée du paquet i du flux k

L_i^k nombre de bits du paquet i du flux k

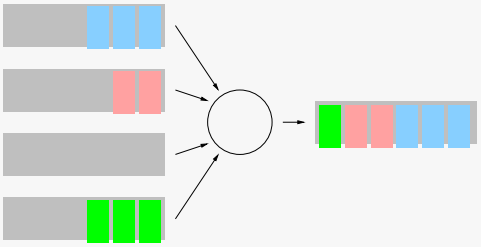
Propriétés

$VC_i^k = S_i^k + \frac{L_i^k}{r_i}$

$S_i^k = max(F_{i-1}^k, a_i^k)$ (respect de l'ordre !)

Notes :

Priority Queuing



- Chaque file est dotée d'une priorité
- File de plus faible priorité servie si prioritaire vide
- Généralement pas de préemption
- Risque de famine

Notes :

[1] S. Blake, D. Black, M. Carlson, E. Davies, and Z. Wang January.
RFC 2475 - an architecture for differentiated service.
Informational, IETF, December 1998.

[2] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang.
RFC 2309 : Recommendations on queue management and congestion avoidance in the internet.
Technical report, IETF, April 1998.
Category : Informational.

[3] R. Braden, D. Clark, and S. Shenker.
Integrated services in the internet architecture : an overview.
Technical report, Internet Engineering Task Force, United States, 1994.

[4] David D. Clark and Wenjia Fang.
Explicit allocation of best-effort packet delivery service.

Notes :

IEEE/ACM Trans. Netw., 6(4) :362–373, 1998.

[5] A. Demers, S. Keshav, and S. Shenker.
Analysis and simulation of a fair queueing algorithm.
In *SIGCOMM '89 : Symposium proceedings on Communications architectures & protocols*, pages 1–12, New York, NY, USA, 1989. ACM Press.

[6] Sally Floyd and Van Jacobson.
Random early detection gateways for congestion avoidance.
IEEE/ACM Transactions on Networking, 1(4) :397–413, 1993.

[7] Paul E. McKenney.
High-speed event counting and classification using a dictionary hash technique.
Proceedings of the International Conference on Parallel Processing, 3 :71–75, 1989.

[8] P.E. McKenney.
Stochastic fairness queuing.

Notes :

In IEEE, editor, INFOCOM *'90 proceedings*, volume 2, pages 733–740, June 1990.

[9] J. Nagle.
On packet switches with infinite storage.
Technical report, IETF, United States, 1985.

[10] J. B. Nagle.
On packet switches with infinite storage.
Innovations in Internetworking, pages 136–139, 1988.

[11] Abhay K. Parekh and Robert G. Gallager.
A generalized processor sharing approach to flow control in integrated services networks : the single-node case.
IEEE/ACM Trans. Netw., 1(3) :344–357, 1993.

[12] M. Shreedhar and George Varghese.
Efficient fair queueing using deficit round-robin.
IEEE/ACM Trans. Netw., 4(3) :375–385, 1996.

[13] L. Zhang.

Notes :

Virtual clock : a new traffic control algorithm for packet switching networks.
In *SIGCOMM '90 : Proceedings of the ACM symposium on Communications architectures & protocols*, pages 19–29, New York, NY, USA, 1990. ACM Press.

Notes :