Techniques d'ordonnancement de paquets

Christophe Deleuze

Laboratoire d'Informatique de Paris 6
`Christophe.Deleuze@lip6.fr`

DEA Réseaux

2 mars 2000

Plan

- rôle et place dans l'architecture
- objectifs
- choix de conception
- trafic "au mieux"
- trafic à garanties
- problèmes d'échelle

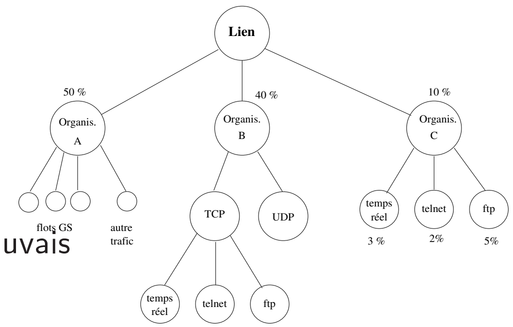
Contexte

- **commutation de circuits**
  - ✓ partage statique
  - ✓ isolation complète
  - ✓ ex : voix
  - ✓ gaspillage de la bande passante
- **commutation de paquets**
  - ✓ partage dynamique (au niveau du paquet)
  - ✓ gain statistique
  - ✓ pas d'isolation

unification des réseaux ➠ intégration de services

Intégration de services

- réseaux de paquets
- offrir tous les services
  - ✓ temps réel
  - ✓ partage équitable
  - ✓ protection contre les mauvais joueurs
  - ✓ partage de lien
  - ✓ modèle hiérarchique

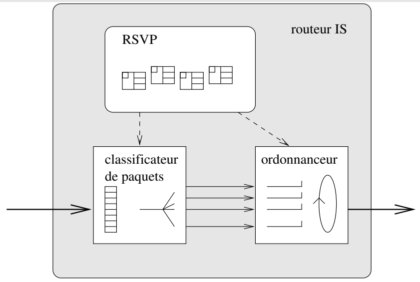


Multiplexage

partage d'une ressource limitée

- ordre de service (temps)
- file d'attente (espace)
- cas de l'accès au lien
  - ✓ bande passante
  - ✓ mémoire tampon
- fonctionnalités
  - ✓ partage équitable (contrôle de congestion)
  - ✓ garanties de performance

Architecture IntServ/RSVP



3 composants :

- RSVP (signalisation)
- classification
- ordonnancement

➠ traitement *par flot*

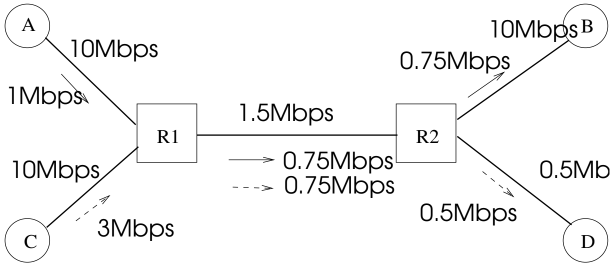
Objectifs

Facilité d'implémentation

- traitement par paquet (hauts débits)
- implémentation matérielle
- quantité d'état par flot (facteur d'échelle)
  - ✓ LAN 50 à 100
  - ✓ WAN env. 100 000 !

Équité et protection

- équité max-min
  - ✓ on commence par les petites demandes
  - ✓ un flot ne reçoit pas plus que sa demande
  - ✓ les flots non satisfaits reçoivent une part égale
- protection ➠ BP mini garantie à chaque flot
- équité globale ≠ equité locale



Garanties

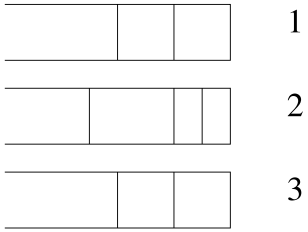
garanties par flot

- réserver des ressources
- bornes déterministes et statistiques
- paramètres de performances : délai, débit, gigue, taux de perte
- couplage délai-débit
- contrôle d'admission
  - ✓ simple
  - ✓ efficace

Choix de conception

Priorités

"multilevel priority with exhaustive service"



- protection unidirectionnelle
- famine ➠ CAC/poliçage aux priorités supérieures

Conservation du travail

Work-conserving

- jamais au repos si des paquets attendent
- peut créer des déformations
- loi de conservation (délais moyens)

Non work-conserving

- peut être au repos même si des paquets attendent
- maintient l'aspect du trafic (gigue)
- augmente le délai moyen global
- limite le besoin de tampons

Granularité

- un seule classe
- traitement par flot
- traitement par classe/ par agrégat

Agrégation

- réduit le nombre d'états
- perd la protection
- partage la gigue

➠ protection/partage

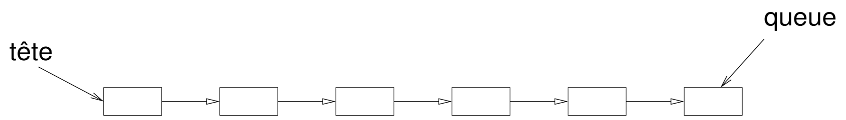
Ordre de service

À l'intérieur d'un niveau de priorité

- calcul d'estampille
- maintien d'une structure de données
  - ✓ liste chainée
  - ✓ tas
  - ✓ ...

Utilisation d'une liste chainée

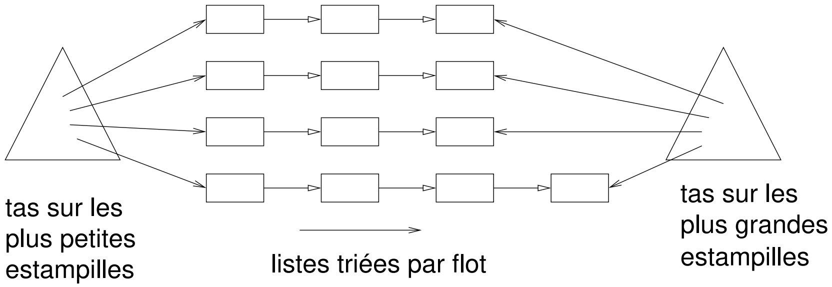
- accès au premier $\theta(1)$
- accès au dernier $\theta(1)$
- insertion θ(longueur) Aie !



Utilisation d'un "tas"

tas = arbre binaire "à plat"

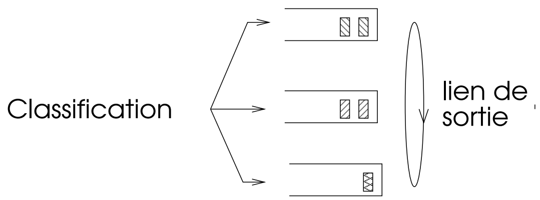
- insertion : $\theta(1)$!
- accès au paquet de plus petite estampille : $\theta(logN)$
- accès au paquet de plus grande estampille : $\theta(logN)$



Ordonnancement pour le trafic "au mieux"

Tourniquet (*Round robin*)

- proposition de Nagle [Nag87] ➠ contrôle de congestion
- files séparées ➠ service *round robin*
- pb : équité comptée en paquets
- sémantique d'un flot

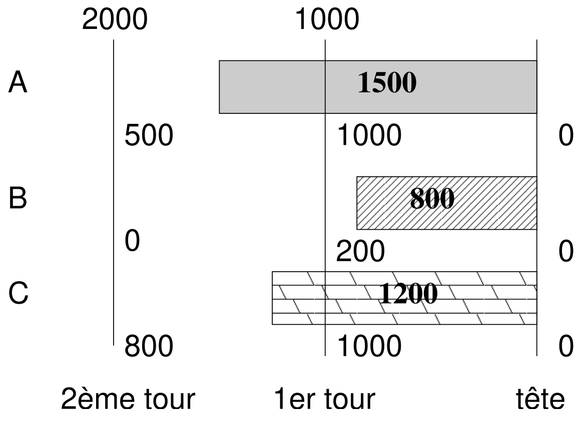


Deficit Round Robin

- compteur de déficit cd pour chaque flot
- initialisé à 0
- quantum de service q
- paquet de taille L
  - ✓ $L > q + cd$ ➠ $cd+ = q$
  - ✓ $L \leq q + cd$
    - ☞ sert le paquet
    - ☞ $cd- = L$
- choix du quantum $> L_{max}$
- équité à échelle de temps > cycle

Deficit Round Robin

- quantum = 1000



Partage équitable *(Fair queueing)*

- émulation d'un round robin *bit-à-bit* [DKS89]
- $R(t)$ nombre de cycles
- $N_{ac}(t)$ flots actifs
- paquet de taille L, commence à t_0 ➠ finit à $R(t) = R(t_0) + L$

$$F_i^k = S_i^k + L \quad \text{et} \quad S_i^k = max(F_i^{k-1}, R(t_i^k))$$

- paquets triés sur le numéro de cycle de leur dernier bit F_i^k
- gestion de file
- retard max
  - ✓ délai $\frac{L_{max}}{C}$
  - ✓ service L_{max}

GPS – *Generalized processor sharing*

discipline idéale (fluide), assure l'équité max-min [PG93]

- sert chaque flot en tourniquet
- unité infinitésimale (tous les flots sont servis "en même temps")
- version *pondérée*

plus formellement : intervalle de temps [τ,t], flots actifs i et j de poids ϕ_i et ϕ_j

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} = \frac{\phi_i}{\phi_j}$$

- délai de paquetisation

PGPS – *Packet by packet GPS*

idem *fair queueing* !

- notion de "temps virtuel"

$$F_i^k = max(F_i^{k-1}, V(a_i^k)) + \frac{L_i^k}{\phi_i}$$

- en temps virtuel, le service reçu par chaque flot est constant
- complexité du calcul
- effacements en cascade *(iterated deletions)*

Implémentation WFQ/PGPS

un paquet arrive

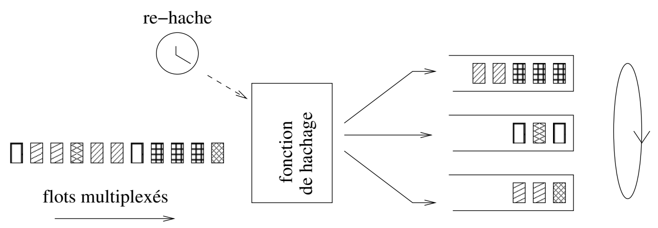
- mise à jour du temps virtuel
- calcul de l'estampille
- insertion du paquet à sa place

WF^{2}Q

équité WFQ peut être améliorée [BZ96b]

- retard de service borné par L_{max}
- *avance* de service ...
- Worst-case Fair Weighted Fair Queueing
- prendre le plus petit F_i^k **parmis** ceux qui ont $S_i^k < tv$
- est *work-conserving*

Stochastic Fair Queueing



- contrôle de congestion [McK90]
- nombre limité de files
- répartition aléatoire
- un peu d'isolation
- re-hachage ➠ équité à long terme

Fair Queueing hiérarchique

préserver l'équité quand la capacité du lien est variable

- WFQ/PGPS non !
- Start-time Fair Queueing ➠ SFQ
- Worst-case Fair Weighted Fair Queueing Plus ➠ WF^{2}Q+

aussi pour les liens sans fil

Conclusion partage équitable

- WFQ
  - ✓ **positif**
    - ☞ protection
    - ☞ équité locale
    - ☞ incitation à utiliser un contrôle de flux
    - ☞ ➠ équité globale ?
  - ✓ **négatif**
    - ☞ état par flot
    - ☞ effacements en cascade
    - ☞ maintien structure triée
- autres...
  - ✓ RED

Ordonnancement pour les garanties

GPS — Garanties de débit

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} = \frac{\phi_i}{\phi_j}, j = 1, 2, ...N$$

pour les sessions i actives sur tout l'intervalle $[\tau, t]$

$$S_i(\tau,t) \cdot \phi_j = \phi_i \cdot S_j(\tau,t)$$

somme sur j $\quad S_i(\tau,t) \cdot \sum \phi_j \geq \phi_i \cdot \sum S_j(\tau,t)$

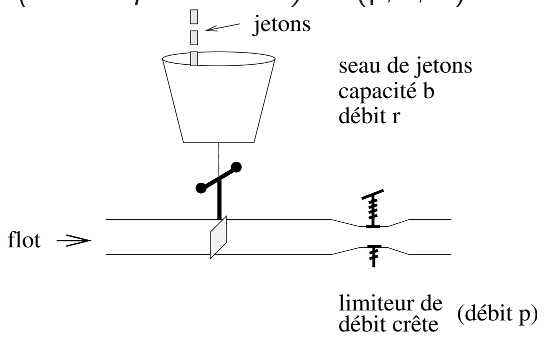
$$S_i(\tau,t) = \frac{\phi_i}{\sum \phi_j} \cdot C \cdot (t-\tau)$$

d'où :

$$B_i = \frac{\phi_i}{\sum_j \phi_j} \cdot C$$

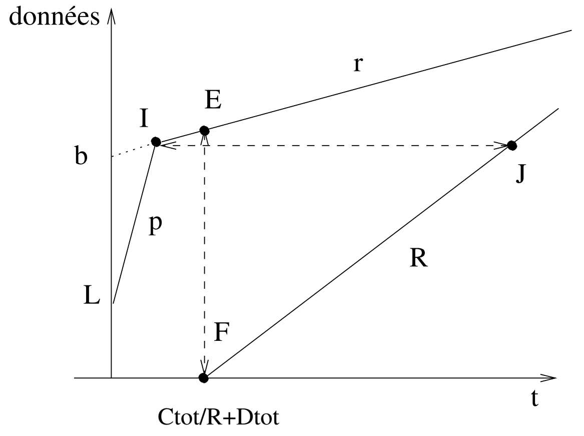
WFQ — Garanties de délai

- délai de propagation
- délai de paquetisation (GPS)
- délai de serialisation (WFQ)
- délai d'attente dans les files
  - ✓ contrat de trafic
  - ✓ TSPEC *(traffic specification)* ➠ (p, r, b)



Garanties de délai

- $[IJ]$ délai max
- $[EF]$ tampons max
- très conservateur



Virtual Clock

- Lixia Zhang [Zha90]
- émulation de TDM
- estampille de fin de service en TDM
- pour chaque flot i
  - ✓ débit moyen AR_i
  - ✓ horloge virtuelle VC_i
  - ✓ paquet de taille L ➠ $VC_i = max(date, VC_i) + \frac{L}{AR_i}$
- comme PGPS mais temps réel ➠ pas équitable
- mêmes garanties de délai
- gestion de file

Famille nombreuse...

- WFQ/PGPS
- SCFQ Self Clocked Fair Queueing [Gol94]
- WF^{2}Q Worst-case Fair Weighted Fair Queueing [BZ96b]
- WF^{2}Q+ [BZ96a]
- SFQ Start time Fair Queueing [GVC96]
- Stochastic FQ
- VC Virtual Clock [Zha90] (pas équitable !)
- ...

Critères d'évaluation

- qualité de l'équité
- borne sur les délais ajoutés
- complexité d'implémentation
- adaptation à un modèle hiérarchique
- ...

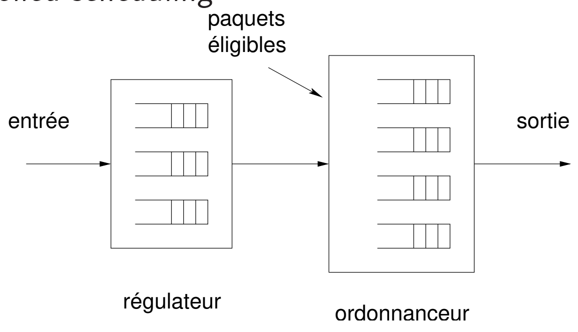
choix en fonction des objectifs

Delay EDD et Jitter EDD

- *Earliest Due Date*
  ✓ estampillage et service dans l'ordre
- Delay EDD
  ✓ contrat sur le débit crête
  ✓ estampille correspond au débit crête
  ✓ ➠ lissage si nécessaire
  ✓ découplage délai/débit
  ✓ MAIS réservation au débit crête
- Jitter EDD
  ✓ contrôle en plus la gigue
  ✓ retarde les paquets

Ordonnancement à contrôle de débit

rate-controlled scheduling



- régulateur retient les paquets jusqu'à ce qu'ils soient éligibles
- ordonnanceur : FCFS, WFQ, priorités, ...
- peut émuler beaucoup d'ordonnanceurs

Class Based Queueing – CBQ (1)

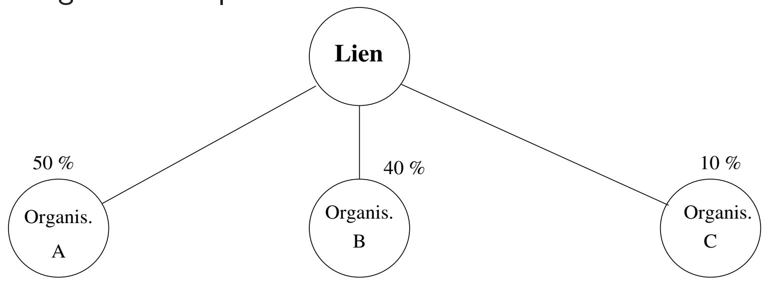
ensemble de composants pour un routeur IS [FJ95]
partage de lien hiérarchique

- classificateur
- estimateur ➠ BP de la classe
- sélecteur ➠ prochaine classe à émettre
- régulateur ➠ si une classe consomme trop

garanties à une échelle de temps plus grande que WFQ

CBQ (2)

- partage de lien "plat"



- régulation si *(link sharing guidelines)*
  ✓ classe overlimit
  ✓ une autre classe est insatisfaite
- si partage hiérarchique, ça se complique !

CSZ

Clark-Shenker-Zhang [CSZ92], [BCS94] (RFC 1633)

- trafic guaranti
- trafic prédictif
- WFQ
  - ✓ une file par flot garanti
  - ✓ tout le reste dans la même file
    - ☞ avec des priorités
    - ☞ classes prédictives
    - ☞ au mieux
    - ☞ FIFO+

Scalabilité

Scalabilité

pb : maintient d'état **par flot**

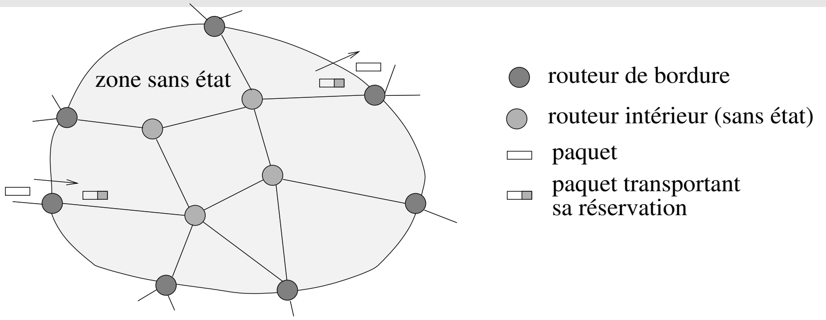
$F_i^k = max(F_i^{k-1}, A_i^k) + \frac{L_i^k}{\phi_i}$ ➠ R_i, F_i^{k-1}

R_i dans un champ d'en tête, et F_i^{k-1} ?

Fonction = retarder les paquets "en avance"

- source émet à débit plus élevé
- paquet plus petit
- gigue créée par les nœuds précédents

Architecture proposée

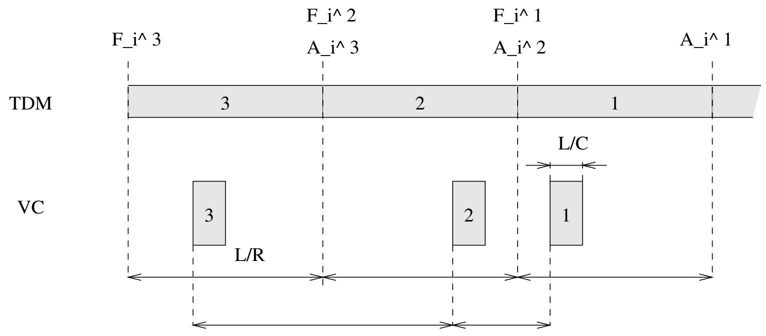


- lissage des flots en entrée du cœur
- taille maximum de paquet

Stateless Virtual Clock ➠ approximation $F_i^k = A_i^k + \frac{L_i^k}{R_i}$

Création de la gigue par les nœuds

- incertitude sur le moment d'émission de chaque paquet

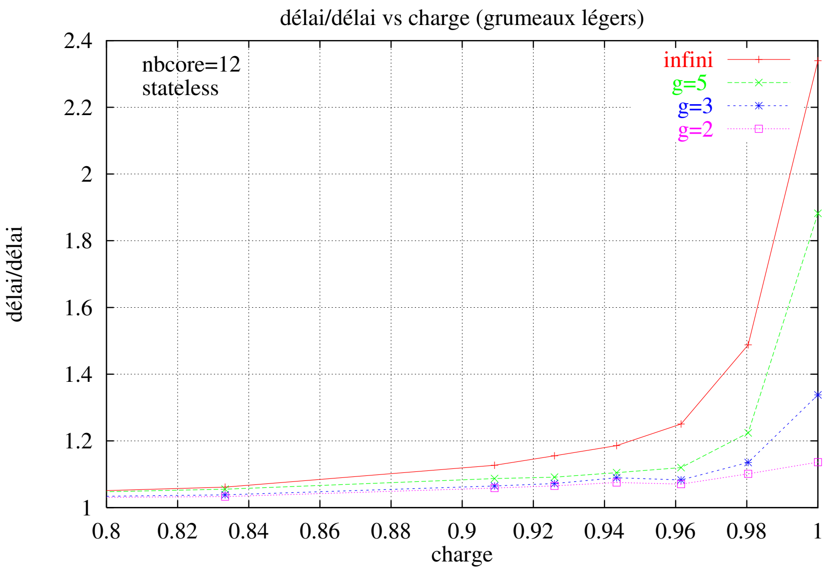


CJVC

Core Jitter Virtual Clock [SZ99]

- au nœud n,
  - ✓ F_i^k estampille
  - ✓ f_i^k date effective d'émission
  - ✓ $a = F_i^k - f_i^k$ "avance" placée dans un champ d'en- tête
- au nœud $n+1$
  - ✓ $F_i^k = A_i^k + a + \frac{L_i^k}{R_i}$
- non *work-conserving*
- traitement par paquet élevé

Stateless Virtual Clock



Un mot sur les pertes de paquets

- granularité
- équité max-min, protection
- priorité de perte ➠ "push-out"
  - ✓ ATM CLP
  - ✓ DiffServ "three color marker"
- perte précoce (ex RED – *Random Early Detection*)
- position
  - ✓ queue, tête, au hasard

Et ça marche ?

- qualité de service ➠ "buzzword"
- implémentations disponibles
- noyau Linux
- CBQ pour Solaris
- AltQ dans FreeBSD
- routeurs CISCO, ...

- MAIS jamais vraiment testé...
- ex RSVP implémentation de l'ISI utilisée par CISCO plante au $6^{ième}$ flot !
- ➠ "quick and dirty" ?

Conclusions

- beaucoup de mécanismes
- de quoi a-t-on besoin ?
- solutions théoriques maitrisées
- manque d'expérience pratique

Références bibliographiques

[BCS94] R. Braden, D. Clark et D. Shenker. Integrated services in the Internet architecture: an overview. RFC 1633, IETF, juillet 1994.

[BZ96a] Jon C.R. Bennett et Hui Zhang. Hierarchical packet fair queueing algorithms. In *Proceedings of SIGCOMM'96* [SIG96], pages 143–156. Aussi dans *IEEE/ACM Transactions on Networking* oct 97. v5n5.

[BZ96b] Jon C.R. Bennett et Hui Zhang. WF^2Q: Worst-case fair weighted fair queueing. In *Proceedings of INFOCOM'96*, pages 1d.3.1–1d.3.9, San Francisco, California, mars 1996.

[CSZ92] David Clark, Scott Shenker et Lixia Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanisms. In *Proceedings of SIGCOMM'92*, volume 22, Baltimore, Maryland, USA, août 1992.

[DKS89] Alan Demers, Srinivasan Keshav et Scott Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of SIGCOMM'89*, volume 19, pages 1–12, Austin, Texas, septembre 1989.

[FJ95] Sally Floyd et Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, août 1995.

[Gol94] S. Jamaloddin Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of INFOCOM'94*, pages 636–646, Toronto, Canada, juin 1994.

[GVC96] Pawan Goyal, Harrick M. Vin et Haichen Cheng. Start-time Fair Queueing: A scheduling algorithm for integrated services packet switching networks. In *Proceedings of SIGCOMM'96* [SIG96], pages 157–168. Aussi dans *IEEE/ACM Transactions on Networking* oct 97 v5n5.

[McK90] P. McKenney. Stochastic fair queueing. In *Proceedings of INFOCOM'90*, San Francisco, California, juin 1990.

[Nag87] John B. Nagle. On packet switches with infinite storage. *IEEE Transactions on Communications*, COM-35(4):435–438, avril 1987.

[PG93] A. K. Parekh et R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks – the single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, juin 1993.

[SIG96] *SIGCOMM '96 SYMPOSIUM Communications Architectures & Protocols*, 1996.

[SZ99] Ion Stoica et Hui Zhang. Providing guaranteed services without per flow management. In *Proceedings of SIGCOMM'99*, pages 81–94, Boston, MA, septembre 1999.

[Zha90] Lixia Zhang. Virtual Clock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM'90*, volume 20, pages 19–29, Philadelphia, Pennsylvania, septembre 1990.

Fin

Document réalisé avec LaTeX.
Style de document `foils`.
Dessins réalisés avec `xfig`.

Christophe Deleuze, `Christophe.Deleuze@lip6.fr`
`http://www-rp.lip6.fr/~deleuze`