

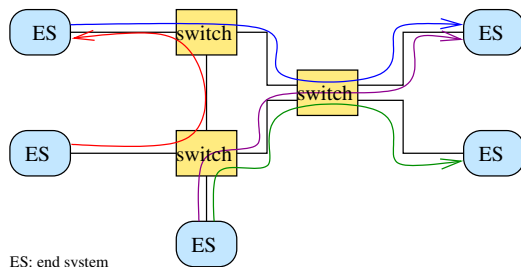
Algorithms for Network calculus

Anne Bouillard

École Normale Supérieure / INRIA

13 juin 2014

AFDX network



AFDX (Avionics Full Duplex):

- 10 switches
 - 100 network elements
 - 10^4 flows
- Flow crossing the system
 - packet size
 - inter-arrival time (with jitter)
 - Network element (server)
 - service rate
 - internal modeling
 - service policy

Objective: deterministic performance guarantees

Compute the maximum time it takes for a packet to cross the system (Worst-case delay)

Network calculus

Real data

Real input traffic
network element



Delay

(min,plus) functions

arrival curve
service curve

↓ (min,plus)-operators

Upper bound on the delay

abstraction
→

abstraction
→

pessimism
→

Two kinds of pessimism

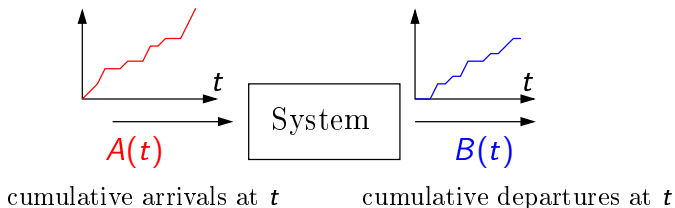
- 1 The abstraction
- 2 The (min,plus) operations

Network calculus

- Theory developed in the 1990's by R.L. Cruz, then developed and popularized by C.S. Chang and J.-Y. Le Boudec.
- Filtering theory in the (\min, plus) algebra.
- Applications:
 - Internet: video transmission (VoD),
 - Load-balancing in switches [Birkoff-von Neumann switches, C.S. Chang]
 - Embedded systems: AFDX (Avionics Full Duplex) [Rockwell-Collins software used to certify A380], Networks-on-chip
- Extensions / variations:
 - Real-Time Calculus [L. Thiele, S. Chakraborty]
 - Stochastic network calculus [C.S. Chang, Y.M. Jiang, F. Ciucu, J. Schmitt]

- 1 Network calculus framework
- 2 Exact worst-case delays in FIFO networks
- 3 Supervision of a flow
- 4 Conclusion and perspectives

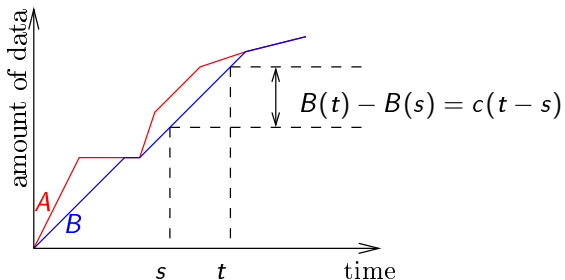
Cumulative processes



- $A : \mathbb{R}_+ \rightarrow \mathbb{R}_{\min+}$: process of the cumulative arrivals, non-decreasing function
- $B : \mathbb{R}_+ \rightarrow \mathbb{R}_{\min+}$: process of the cumulative departures, non-decreasing function
- Causality constraint: $A \geq B$

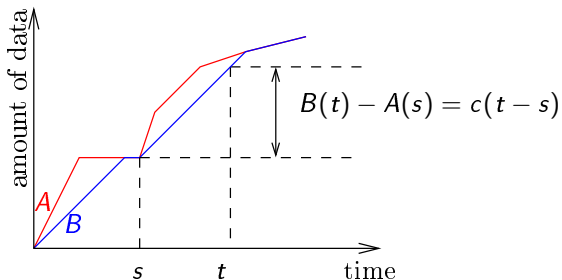
Relation between A and B

Suppose that the system serves c bits of data per unit of time.



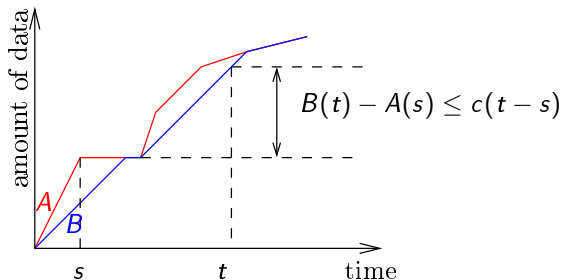
Relation between A and B

Suppose that the system serves c bits of data per unit of time.



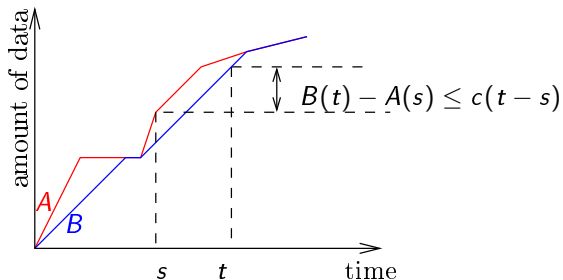
Relation between A and B

Suppose that the system serves c bits of data per unit of time.



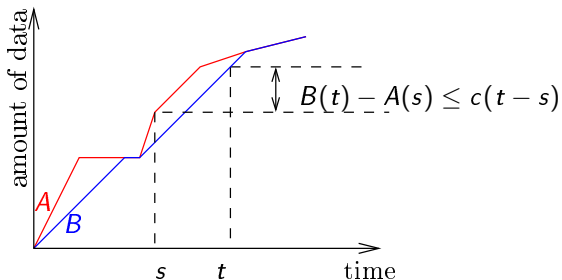
Relation between A and B

Suppose that the system serves c bits of data per unit of time.



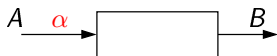
Relation between A and B

Suppose that the system serves c bits of data per unit of time.



$$B(t) = \inf_{0 \leq s \leq t} A(s) + c(t - s) = A * c(t).$$

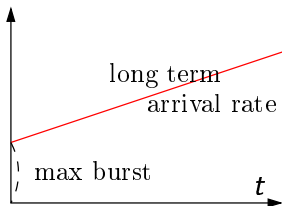
Network calculus in a slide



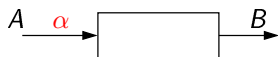
Arrival curve

A is constrained by the function α
if $\forall 0 \leq s \leq t$,

$$A(t) - A(s) \leq \alpha(t - s).$$



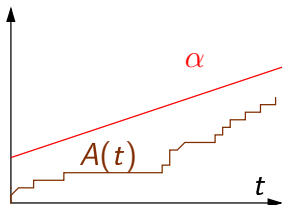
Network calculus in a slide



Arrival curve

A is constrained by the function α
if $\forall 0 \leq s \leq t$,

$$A(t) - A(s) \leq \alpha(t - s).$$



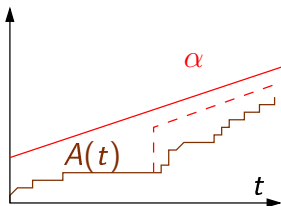
Network calculus in a slide



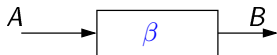
Arrival curve

A is constrained by the function α
if $\forall 0 \leq s \leq t$,

$$A(t) - A(s) \leq \alpha(t - s).$$



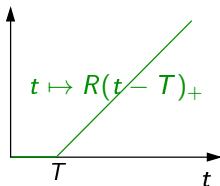
Network calculus in a slide



Arrival curve

A is constrained by the function α
if $\forall 0 \leq s \leq t$,

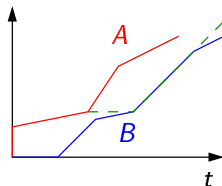
$$A(t) - A(s) \leq \alpha(t - s).$$



Service curve

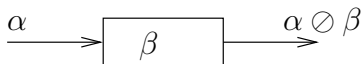
A network element guarantees β
for A if B satisfies

$$B \geq A * \beta.$$



$$A * \beta(t) = \inf_{0 \leq s \leq t} A(s) + \beta(t - s)$$

Network calculus in a slide



Arrival curve $A(t) - A(s) \leq \alpha(t - s)$.

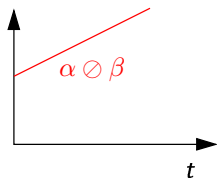
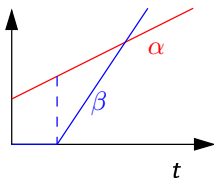
Propagation of constraints

If A is α -constrained and β is a service curve for A , then B is $\alpha \circledast \beta$ -constrained.

Service curve $B \geq A * \beta$.

A network element guarantees β for A if B satisfies

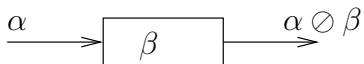
$$B \geq A * \beta.$$



$$\alpha \circledast \beta(t) = \sup_{s \geq 0} \alpha(t + s) - \beta(s)$$

$$A * \beta(t) = \inf_{0 \leq s \leq t} A(s) + \beta(t - s)$$

Network calculus in a slide



Arrival curve $A(t) - A(s) \leq \alpha(t - s)$.

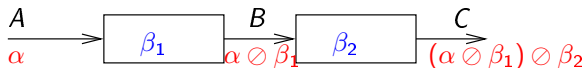
Propagation of constraints

If A is α -constrained and β is a service curve for A , then B is $\alpha \circ \beta$ -constrained.

Service curve $B \geq A * \beta$.

Composition of service curves

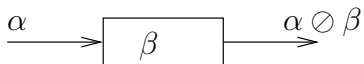
The concatenation of two servers offering respective service curves β_1 and β_2 guarantees a service curve $\beta_1 * \beta_2$ to the flow.



$$\alpha \circ \beta(t) = \sup_{s \geq 0} \alpha(t + s) - \beta(s)$$

$$A * \beta(t) = \inf_{0 \leq s \leq t} A(s) + \beta(t - s)$$

Network calculus in a slide



Arrival curve $A(t) - A(s) \leq \alpha(t - s)$.

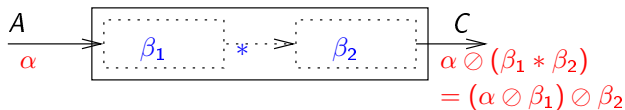
Propagation of constraints

If A is α -constrained and β is a service curve for A , then B is $\alpha \circ \beta$ -constrained.

Service curve $B \geq A * \beta$.

Composition of service curves

The concatenation of two servers offering respective service curves β_1 and β_2 guarantees a service curve $\beta_1 * \beta_2$ to the flow.



$$\alpha \circ \beta(t) = \sup_{s \geq 0} \alpha(t + s) - \beta(s)$$

$$A * \beta(t) = \inf_{0 \leq s \leq t} A(s) + \beta(t - s)$$

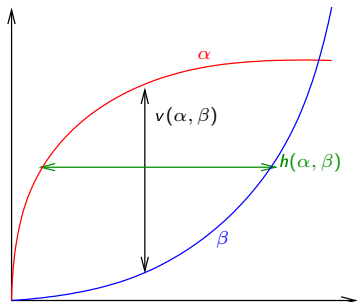
From constraints to performance bounds

Maximum backlog:

$$B_{\max} = \sup_{t \geq 0} A(t) - B(t)$$

Maximum delay:

$$D_{\max} = \inf \{d \mid \forall t \in \mathbb{R}_+, B(t+d) \geq A(t)\}$$

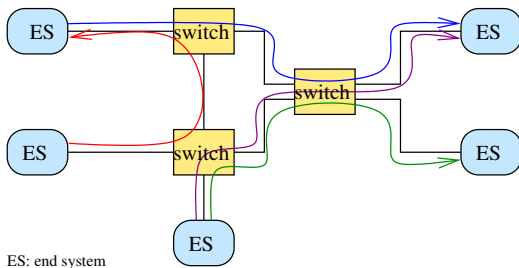


Performance bounds

- $B_{\max} \leq \alpha \oslash \beta(0) = v(\alpha, \beta) = \sup\{\alpha(t) - \beta(t) \mid t \geq 0\}$
- $D_{\max} \leq h(\alpha, \beta) = \inf\{\forall t \geq 0, d \geq 0 \mid \alpha(t) \leq \beta(t+d)\}$ (for FIFO per flow).

- 1 Network calculus framework
- 2 Exact worst-case delays in FIFO networks**
- 3 Supervision of a flow
- 4 Conclusion and perspectives

Computing performance bounds in a network

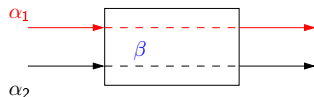


- Each flow i is described by its arrival curve α_i
- Each server h is described by its service curve β_h and a service policy

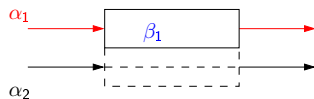
Goal

Compute the maximum time that it takes for a bit of data of a flow to cross the system, when the cumulative processes that respect the constraints given by the curves.

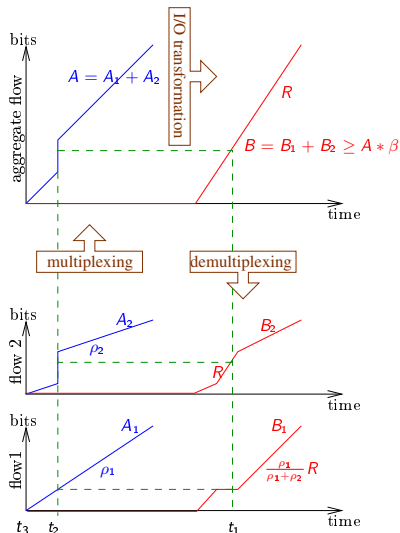
FIFO policy: residual service curves

Multiplexing: $B_1 + B_2 \geq (A_1 + A_2) * \beta$ 

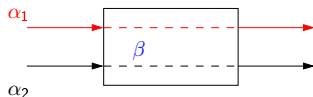
FIFO policy: residual service curves

Multiplexing: $B_1 + B_2 \geq (A_1 + A_2) * \beta$ 

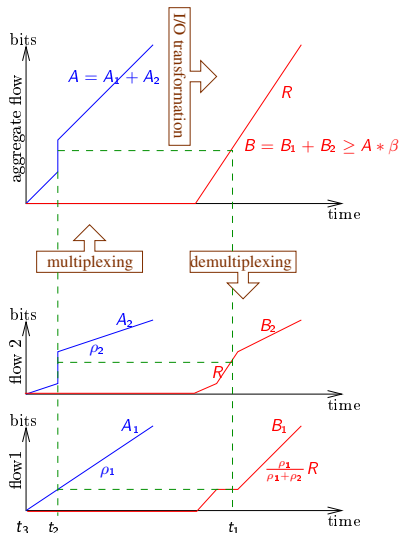
FIFO policy: residual service curves



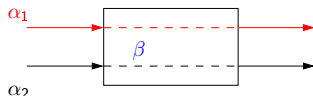
$$\text{Multiplexing: } B_1 + B_2 \geq (A_1 + A_2) * \beta$$



FIFO policy: residual service curves



Multiplexing: $B_1 + B_2 \geq (A_1 + A_2) * \beta$

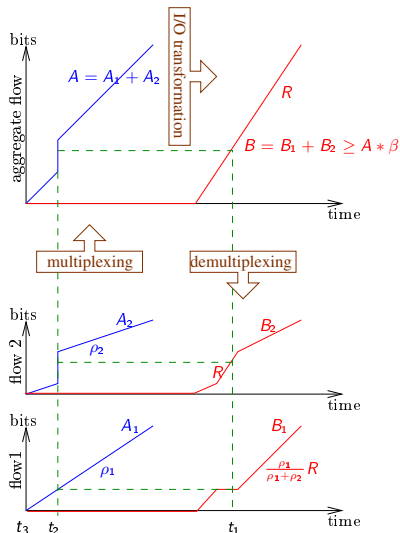


Theorem (Le Boudec, Thiran, 2001)

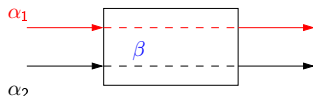
$\forall \theta \in \mathbb{R}_+$, flow 1 is guaranteed service curves

$$\beta_{\theta}^1(t) = [\beta(t) - \alpha_2(t - \theta)]_+ 1_{t > \theta}.$$

FIFO policy: residual service curves



Multiplexing: $B_1 + B_2 \geq (A_1 + A_2) * \beta$



Theorem (Le Boudec, Thiran, 2001)

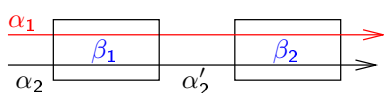
$\forall \theta \in \mathbb{R}_+$, flow 1 is guaranteed service curves

$$\beta_\theta^1(t) = [\beta(t) - \alpha_2(t - \theta)]_+ 1_{t > \theta}.$$

Infinite family of service curves that cannot be compared

Composition of FIFO servers

$$\beta_{\theta}^1(t) = [\beta(t) - \alpha_2(t - \theta)]_+ 1_{t > \theta}.$$

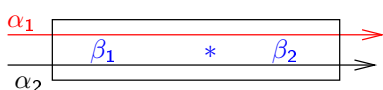


$$\beta_{1,\theta_1}^1 * \beta_{2,\theta_2}^1$$

- Concatenation of servers $1, \dots, N$: how to find the best $(\theta_1, \dots, \theta_n)$, that minimizes the end-to-end delay of a flow? [Lenzini, Stea *et al.*]
- Is the delay the exact worst-case delay bound? NO.
- Software Deborah based on this approach.

Composition of FIFO servers

$$\beta_{\theta}^1(t) = [\beta(t) - \alpha_2(t - \theta)]_+ 1_{t > \theta}.$$



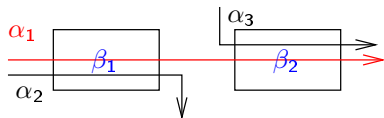
$$\beta_{1, \theta_1}^1 * \beta_{2, \theta_2}^1$$

$$(\beta_1 * \beta_2)_{\theta}^1$$

- Concatenation of servers $1, \dots, N$: how to find the best $(\theta_1, \dots, \theta_n)$, that minimizes the end-to-end delay of a flow? [Lenzini, Stea *et al.*]
- Is the delay the exact worst-case delay bound? NO.
- Software Deborah based on this approach.

Composition of FIFO servers

$$\beta_{\theta}^1(t) = [\beta(t) - \alpha_2(t - \theta)]_+ 1_{t > \theta}.$$



$$\beta_{1,\theta_1}^1 * \beta_{2,\theta_2}^1$$

- Concatenation of servers $1, \dots, N$: how to find the best $(\theta_1, \dots, \theta_n)$, that minimizes the end-to-end delay of a flow? [Lenzini, Stea *et al.*]
- Is the delay the exact worst-case delay bound? NO.
- Software Deborah based on this approach.

Linear programming and network calculus

Joint work with G. Stea

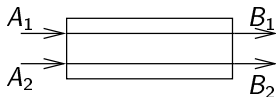
Idea

- 1 Encoding the Network calculus constraints by linear constraints.
- 2 No computation of residual service curves.

Assumptions

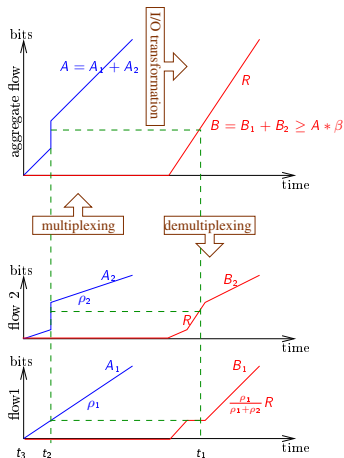
- 1 Feed-forward network.
- 2 Concave piecewise affine arrival curves.
- 3 Convex piecewise affine service curves.
- 4 Left-continuous cumulative processes.

One server and two flows

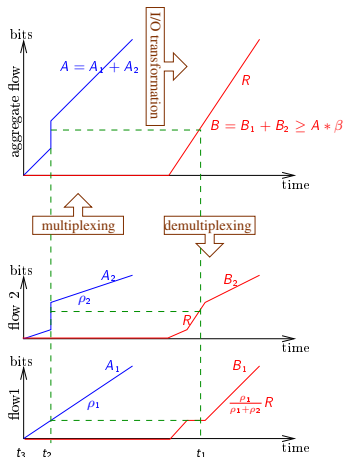
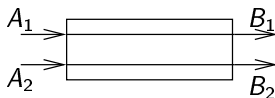


Maximize $t_1 - t_2$ with

- $t_3 \leq t_2 \leq t_1$
- $A_1(t_2) = B_1(t_1)$ **FIFO**
- $A_2(t_2) = B_2(t_1)$ **FIFO**



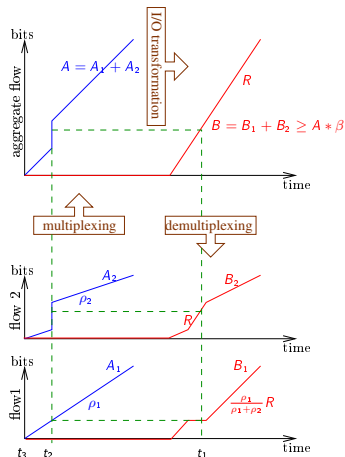
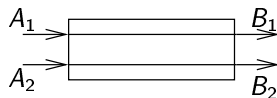
One server and two flows



Maximize $t_1 - t_2$ with

- $t_3 \leq t_2 \leq t_1$
- $A_1(t_2) = B_1(t_1)$
- $A_2(t_2) = B_2(t_1)$
- $(B_1 + B_2)(t_1) \geq (A_1 + A_2)(t_3) + \beta(t_1 - t_3)$
Service $(B_1 + B_2) \geq (A_1 + A_2) * \beta$
 $= \min_t (B_1 + B_2)(t_1) \geq (A_1 + A_2)(t) + \beta(t_1 - t)$

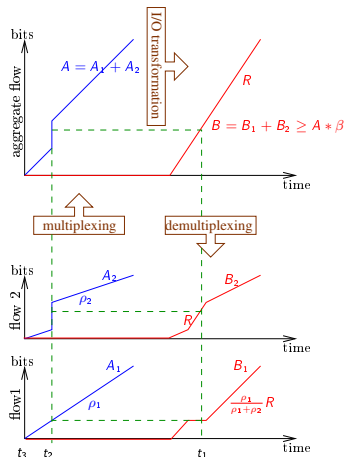
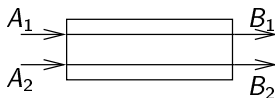
One server and two flows



Maximize $t_1 - t_2$ with

- $t_3 \leq t_2 \leq t_1$
- $A_1(t_2) = B_1(t_1)$
- $A_2(t_2) = B_2(t_1)$
- $(B_1 + B_2)(t_1) \geq (A_1 + A_2)(t_3) + \beta(t_1 - t_3)$
- $A_i(t_2) - A_i(t_3) \leq \alpha_i(t_2 - t_3)$ **Arrival**

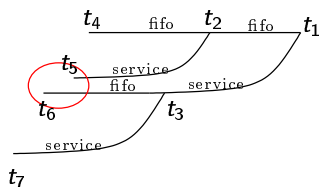
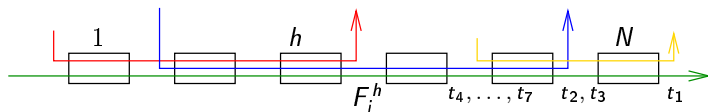
One server and two flows



Maximize $t_1 - t_2$ with

- $t_3 \leq t_2 \leq t_1$
- $A_1(t_2) = B_1(t_1)$
- $A_2(t_2) = B_2(t_1)$
- $(B_1 + B_2)(t_1) \geq (A_1 + A_2)(t_3) + \beta(t_1 - t_3)$
- $A_i(t_2) - A_i(t_3) \leq \alpha_i(t_2 - t_3)$
- $A_i(t_3) \leq A_i(t_2)$ **Monotony**
- $A_i(x) \geq B_i(x)$ **Causality**

Tandem networks



Variables

dates: t_1, \dots, t_{2N+1-1} , where

- t_{2k} FIFO hypothesis with regards to t_k
- t_{2k+1} service curve constraints with regards to t_k

function values: $F_i^h(t_k)$ cumulative process of flow i after server h at time t_k

One Mixed Integer Linear Program (MILP)

We can add Boolean variables ($q \in \{0, 1\}$) so that one linear program totally orders the dates and maintains the monotony properties:

- Take M large enough
- for each couple of dates t_i, t_j , add the variable $q_{i,j} \in \{0, 1\}$ and the constraints:

$$t_i + q_{i,j}M \geq t_j$$

$$t_j + (1 - q_{i,j})M \geq t_i$$

$$f(t_i) + q_{i,j}M \geq f(t_j)$$

$$f(t_j) + (1 - q_{i,j})M \geq f(t_i)$$

for each function f concerning t_i and t_j .

Worst-case delay for a tandem network

$\Delta = \max t_1 - t_{2N}$ with

- Date ordering, monotony
- FIFO hypothesis ($A_1(t_{2k}) = B_1(t_k)$);
- service curve constraint
 $((B_1 + B_2)(t_k) \geq (A_1 + A_2)(t_{2k+1}) + \beta(t_k - t_{2k+1}))$;
- arrival curve ($A_i(t_h) - A_i(t_k) \leq \alpha_i(t_h - t_k)$)

Theorem

The worst-case delay for the flow crossing the system is Δ .

Worst-case delay for a tandem network

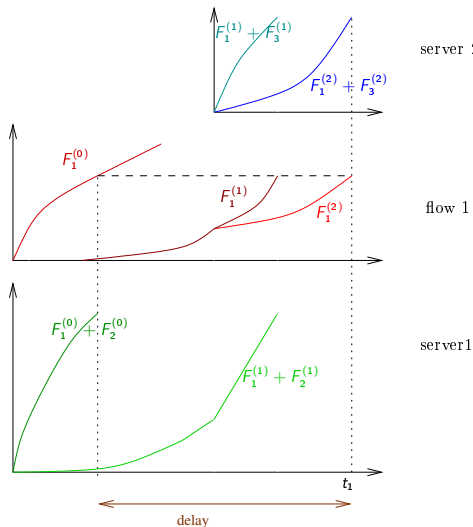
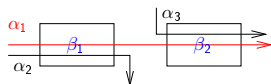
$\Delta = \max t_1 - t_{2N}$ with

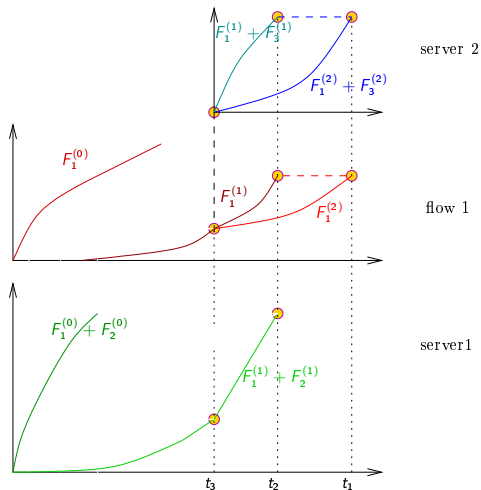
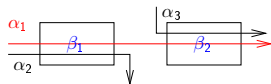
- Date ordering, monotony
- FIFO hypothesis ($A_1(t_{2k}) = B_1(t_k)$);
- service curve constraint
 $((B_1 + B_2)(t_k) \geq (A_1 + A_2)(t_{2k+1}) + \beta(t_k - t_{2k+1}))$;
- arrival curve ($A_i(t_h) - A_i(t_k) \leq \alpha_i(t_h - t_k)$)

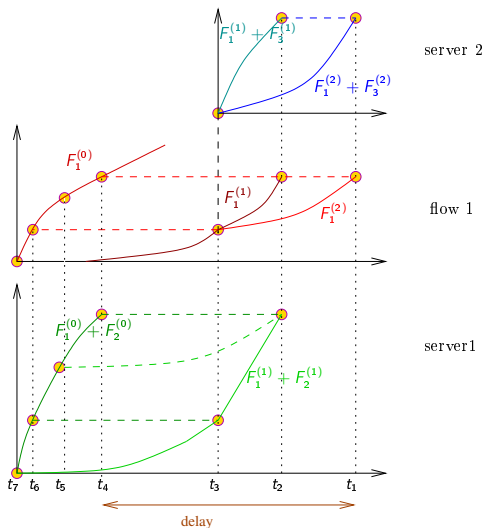
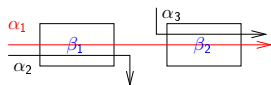
Theorem

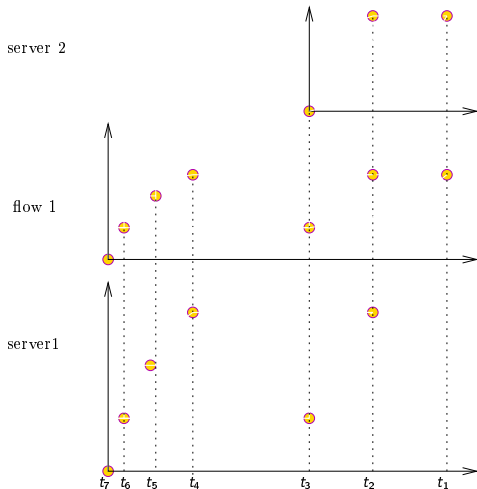
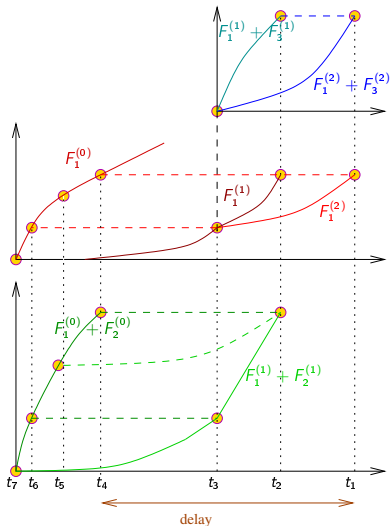
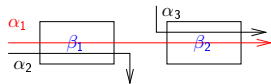
The worst-case delay for the flow crossing the system is Δ .

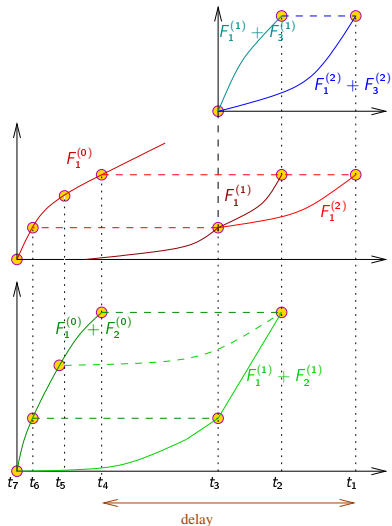
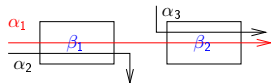
Exponential number of variables and constraints, Boolean variables: in practice, this delay is computable for networks with up to 6 servers.

Proof (delay at t_1)

Proof (delay at t_1)

Proof (delay at t_1)

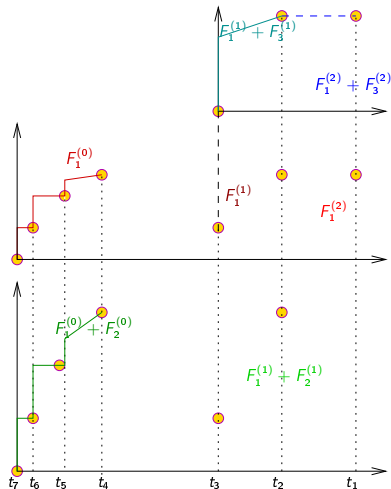
Proof (delay at t_1)

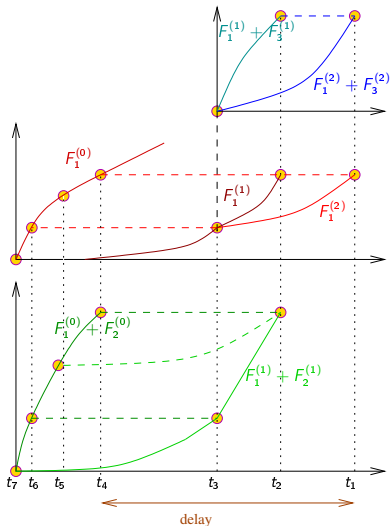
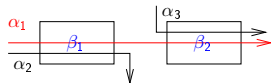
Proof (delay at t_1)

server 2

flow 1

server 1

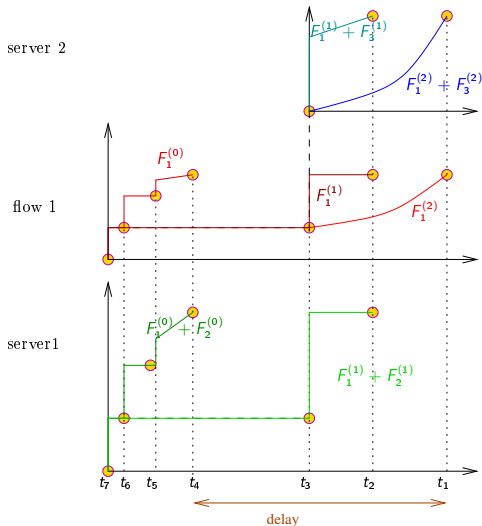
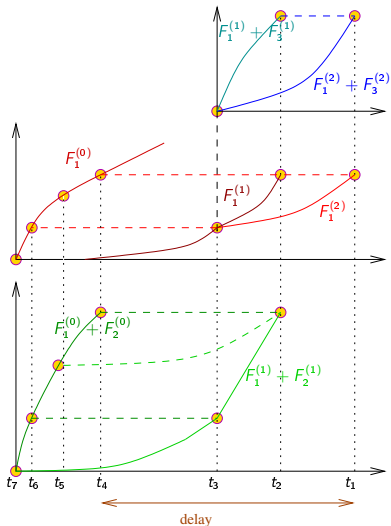
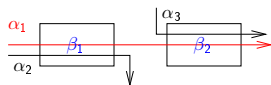


Proof (delay at t_1)

server 2

flow 1

server 1

Proof (delay at t_1)

An upper and lower bounds

Upper bound

Relax the monotony and arrival constraints: keep a partial order on the dates.

- Exponential number of dates
- No Boolean variable

An upper and lower bounds

Upper bound

Relax the monotony and arrival constraints: keep a partial order on the dates.

- Exponential number of dates
- No Boolean variable

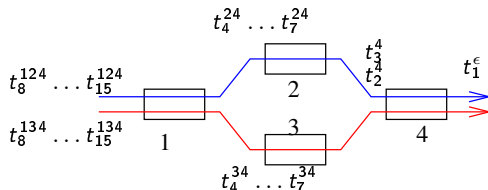
Lower bound

Impose some additional constraints on the dates:

$$t_{2k+1} = t_{2k'+1} \quad \text{for} \quad 2^h \leq k, k' < 2^{h+1} - 1$$

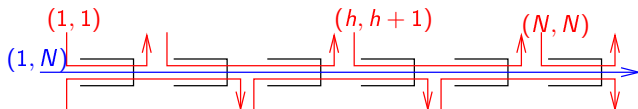
- Quadratic number of dates
- complete order on the dates

Generalization to feed-forward networks

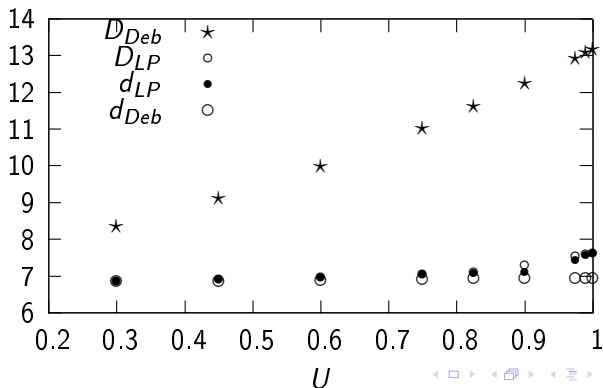


The number of dates increases exponentially again, but the MILP will be similar.

Numerical experiments



$$\beta(t) = 10(t - 1)_+, \sigma(t) = 1 + \rho t \text{ and } U = 3\rho/10.$$



Other results obtained using linear programming

Arbitrary multiplexing [B., Thierry, Jouhet]

- Exact worst-case delay and backlog bounds;
- polynomial algorithm for tandem networks;
- no simplification using (MILP);
- NP-hard problem for general feed-forward networks.

Fixed priorities [B., Junier]

- Improved known bounds;
- exact worst-case only in some limited cases;
- general networks (the priorities break the cyclic dependencies).

- 1 Network calculus framework
- 2 Exact worst-case delays in FIFO networks
- 3 Supervision of a flow**
- 4 Conclusion and perspectives

Supervision of a flow

Joint work with A. Junier and B. Ronot

Learn the arrival rate of the packets of a flow and detect the changes of the arrival rates over time.

Let (x_n) be the non-decreasing sequence of arrival times of the packets.

What Network calculus usually does:

- Compute the arrival cumulative process: $A(t) = \max\{n \mid x_n \leq t\}$.
- Compute the best arrival curve $\alpha = A \otimes A$

Limitation of this approach:

- Only one curve that describes the flow at all times
- Very loose if there are drastic changes in the arrival rate
- No efficient on-line computation as

$$\alpha_n(t) = \max(\alpha_{n-1}(t), \max\{j \mid x_n - x_{n-j} \leq t\}).$$

Idea: use simple curves on finite intervals

Curves on a finite interval

The flow (x_n) is $(\underline{\alpha}, \bar{\alpha})$ -constrained on the interval $[m, n]$ if
 $\forall m \leq m' \leq n' \leq n,$

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}).$$

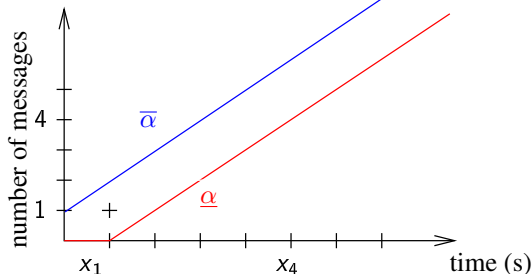
Simple curves

$$\bar{\alpha}_{\rho, \sigma} : t \mapsto \sigma + \rho t$$

$$\underline{\alpha}_{\rho, T} : t \mapsto \rho(t - T)_+.$$

constant time complexity

- check the constraints are satisfied
- update the strongest constraint



Idea: use simple curves on finite intervals

Curves on a finite interval

The flow (x_n) is $(\underline{\alpha}, \bar{\alpha})$ -constrained on the interval $[m, n]$ if
 $\forall m \leq m' \leq n' \leq n,$

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}).$$

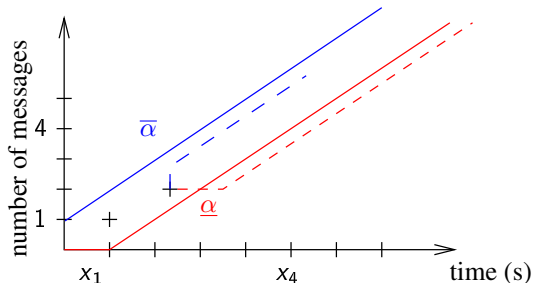
Simple curves

$$\bar{\alpha}_{\rho, \sigma} : t \mapsto \sigma + \rho t$$

$$\underline{\alpha}_{\rho, T} : t \mapsto \rho(t - T)_+.$$

constant time complexity

- check the constraints are satisfied
- update the strongest constraint



Idea: use simple curves on finite intervals

Curves on a finite interval

The flow (x_n) is $(\underline{\alpha}, \bar{\alpha})$ -constrained on the interval $[m, n]$ if

$$\forall m \leq m' \leq n' \leq n,$$

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}).$$

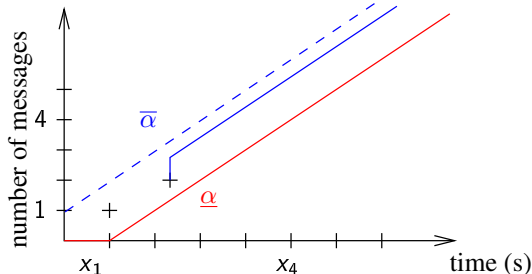
Simple curves

$$\bar{\alpha}_{\rho, \sigma} : t \mapsto \sigma + \rho t$$

$$\underline{\alpha}_{\rho, T} : t \mapsto \rho(t - T)_+.$$

constant time complexity

- check the constraints are satisfied
- update the strongest constraint



Idea: use simple curves on finite intervals

Curves on a finite interval

The flow (x_n) is $(\underline{\alpha}, \bar{\alpha})$ -constrained on the interval $[m, n]$ if
 $\forall m \leq m' \leq n' \leq n,$

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}).$$

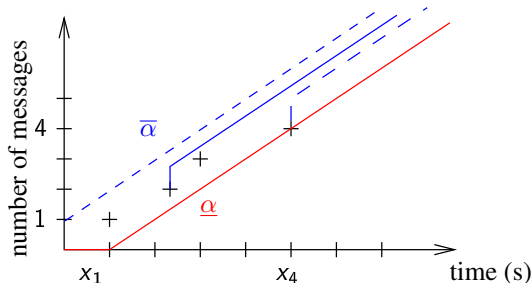
Simple curves

$$\bar{\alpha}_{\rho, \sigma} : t \mapsto \sigma + \rho t$$

$$\underline{\alpha}_{\rho, T} : t \mapsto \rho(t - T)_+.$$

constant time complexity

- check the constraints are satisfied
- update the strongest constraint



Idea: use simple curves on finite intervals

Curves on a finite interval

The flow (x_n) is $(\underline{\alpha}, \bar{\alpha})$ -constrained on the interval $[m, n]$ if
 $\forall m \leq m' \leq n' \leq n,$

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}).$$

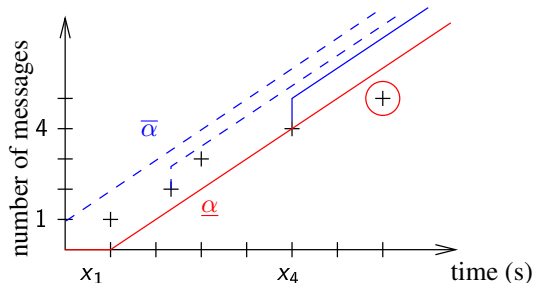
Simple curves

$$\bar{\alpha}_{\rho, \sigma} : t \mapsto \sigma + \rho t$$

$$\underline{\alpha}_{\rho, T} : t \mapsto \rho(t - T)_+.$$

constant time complexity

- check the constraints are satisfied
- update the strongest constraint



Idea: use simple curves on finite intervals

Curves on a finite interval

The flow (x_n) is $(\underline{\alpha}, \bar{\alpha})$ -constrained on the interval $[m, n]$ if

$$\forall m \leq m' \leq n' \leq n,$$

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}).$$

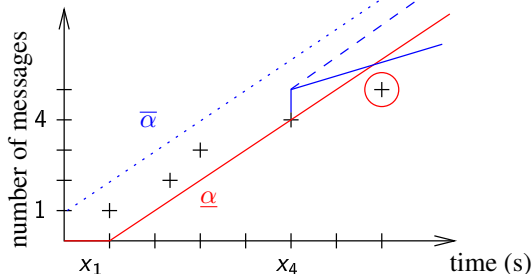
Simple curves

$$\bar{\alpha}_{\rho, \sigma} : t \mapsto \sigma + \rho t$$

$$\underline{\alpha}_{\rho, T} : t \mapsto \rho(t - T)_+.$$

constant time complexity

- check the constraints are satisfied
- update the strongest constraint

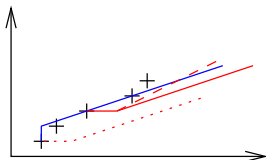
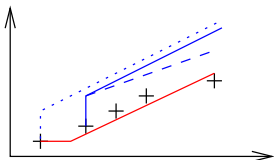


Core algorithm

σ and T are fixed.

At each arrival of a packet

- 1 Check that its arrival time respects the constraints;
- 2 If yes, update the strongest constraints
- 3 Otherwise, compute a new rate.



Convergence of the algorithm

Proposition

If (x_n) is a periodic and $(\underline{\alpha}_{\rho, T}, \bar{\alpha}_{\rho, \sigma})$ -constrained flow, then either the algorithm finds the rate ρ in finite time (and the rate will not be updated anymore) or it ultimately has a periodical behavior.

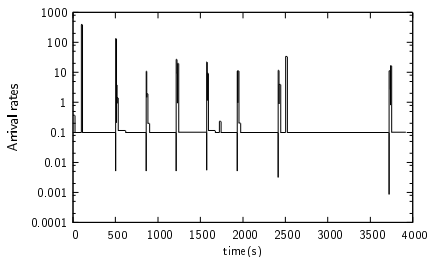
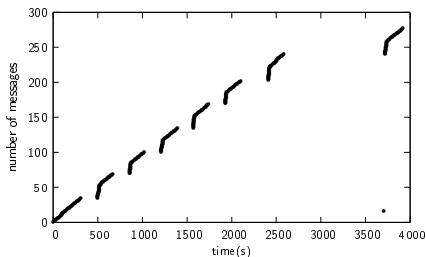
Theorem

If (x_n) is balanced, the algorithm always converges.

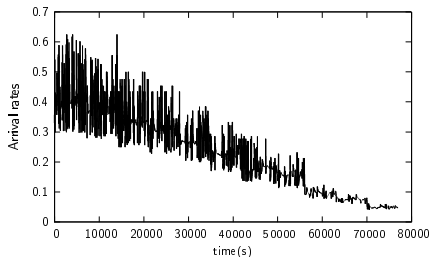
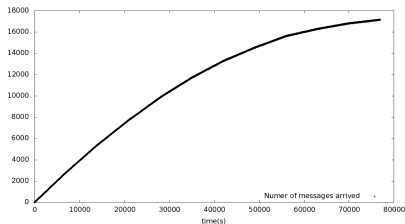
Example of a flow from an OSPF router

(OSPF: Open Shortest Path First)

Flow of messages received by a router for the OSPF protocol when the router periodically reboots.

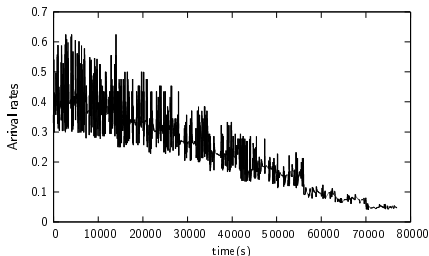
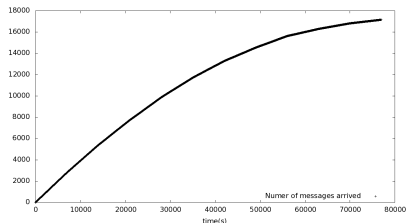


Example of a random flow with decreasing arrival rate



Output: sequence of rates and packets breaking the constraints.

Example of a random flow with decreasing arrival rate

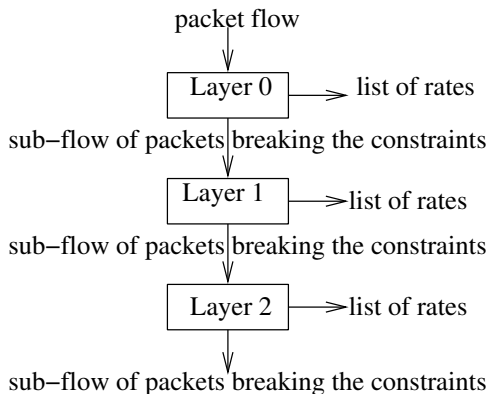


Output: sequence of rates and packets breaking the constraints.

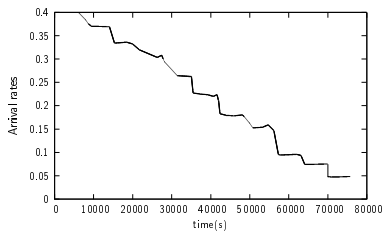
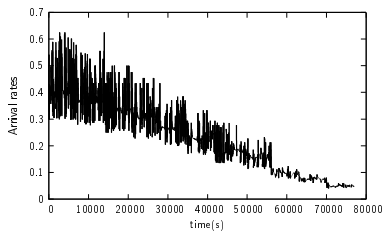
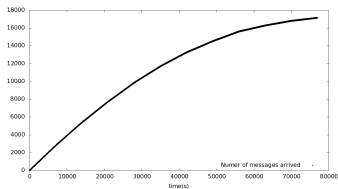
Limitations:

- ① σ and T must be fixed in advance
→ develop learning/adaptive algorithms
- ② Detailed vs. rough view of the flow
→ use the algorithm several times.

Multilayer algorithm



Numerical results



- 1 Network calculus framework
- 2 Exact worst-case delays in FIFO networks
- 3 Supervision of a flow
- 4 Conclusion and perspectives

Conclusion and perspectives

- 1 Exact worst-case performance in feed-forward networks.
 - Arbitrary multiplexing and FIFO
 - NP-hard problem.
- 2 Supervision of a flow: a simple algorithm to supervise a flow using the concept of curves from NC.

Conclusion and perspectives

- 1 Exact worst-case performance in feed-forward networks.
 - Arbitrary multiplexing and FIFO
 - NP-hard problem.
- 2 Supervision of a flow: a simple algorithm to supervise a flow using the concept of curves from NC.

Perspectives and challenges

- Algorithmic cost too high to be used in large networks
- Closed formula for a residual service curve?
- Fluid model vs. packet model
- Stochastic network calculus
- Network with cyclic dependences?