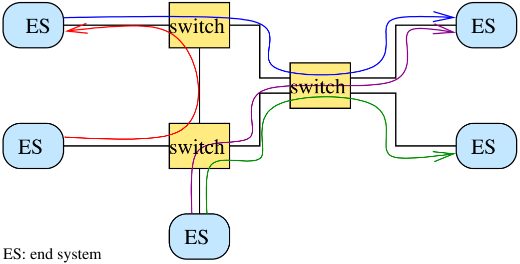
NOKIA Bell Labs

Stability and performance in cyclic network

Anne Bouillard (Nokia Bell Labs France)

FORMATS 2019

Performances in networks



Objective: deterministic performance guarantees

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

Network calculus

Real data		**(min,plus) functions**
Real input traffic	$\xrightarrow{\text{abstraction}}$	arrival curve
network element	$\xrightarrow{\text{abstraction}}$	service curve
$\Downarrow$		$\Downarrow$ (min,plus)-operators
Delay / backlog	$\xrightarrow{\text{pessimism}}$	Upper bound on the delay / backlog

Two kinds of pessimism

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

   NOKIA Bell Labs

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 [Birkhoff-von Neumann switches, C.S. Chang]
  - Embedded systems: AFDX (Avionics Full Duplex) [Rockwell-Collins software used to certify A380], Networks-on-chip

State of the art and contribution: Feed-forward networks

Many recent results for computing tight bounds feed-forward networks:

- PBOO/PMOO phenomena [Schmitt et al 2008]
- Linear programming solutions [B. et al, 2010]
  - tight bounds
  - the problem is NP-hard
  - polynomial for tadem networks
- Exhaustive search / pay segregation only once [Bondorff et al, 2016]
  - good heuristics to approximate the worst-case performance bounds
- Neural networks [Geyer, 2018]
  - learning the good heuristic

State of the art and contribution: Feed-forward networks

Many recent results for computing tight bounds feed-forward networks:

- PBOO/PMOO phenomena [Schmitt et al 2008]
- Linear programming solutions [B. et al, 2010]
  - tight bounds
  - the problem is NP-hard
  - polynomial for tadem networks
- Exhaustive search / pay segregation only once [Bondorff et al, 2016]
  - good heuristics to approximate the worst-case performance bounds
- Neural networks [Geyer, 2018]
  - learning the good heuristic
- Tight bounds for tree-networks for the backlog of a subsets of flows and delay

NOKIA Bell Labs

State of the art and contribution: Cyclic networks

Few results in networks with cyclic dependencies

- Computing good stability conditions and performance guarantees for network with cyclic dependencies is an open issue
- Obtaining such guarantees would enable more flexible design of systems, with fewer switches.

State of the art and contribution: Cyclic networks

Few results in networks with cyclic dependencies

- Computing good stability conditions and performance guarantees for network with cyclic dependencies is an open issue
- Obtaining such guarantees would enable more flexible design of systems, with fewer switches.

- Flow-based bounds: fix-point/stopped time method
  - ”classical” [Cruz 1994]
  - PMOO [Amari et Mifdaoui, 2017]
- Backlog-based bounds: ”stability” of the ring [Tassiulas, Georgiadis, 1996] Additional assumption: the traffic is upper-bounded in each link.
- instability results from adversarial method [Andrews, 2001]

State of the art and contribution: Cyclic networks

Few results in networks with cyclic dependencies

- Computing good stability conditions and performance guarantees for network with cyclic dependencies is an open issue
- Obtaining such guarantees would enable more flexible design of systems, with fewer switches.

- Flow-based bounds: fix-point/stopped time method
  - ”classical” [Cruz 1994]
  - PMOO [Amari et Mifdaoui, 2017]
- Backlog-based bounds: ”stability” of the ring [Tassiulas, Georgiadis, 1996]
  Additional assumption: the traffic is upper-bounded in each link.
- instability results from adversarial method [Andrews, 2001]
- Improve the fix-point method to combine flow and backlog-based bounds

Network calculus framework

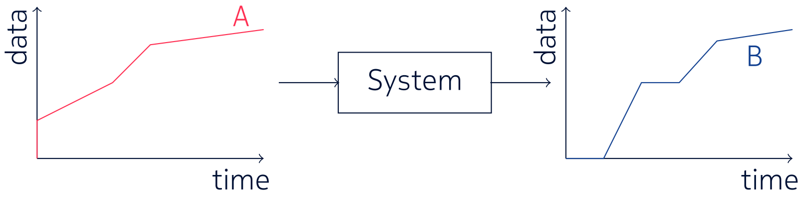
Networks wih cyclic dependencies

Computing performance bounds in feed-forward networks

Performances in cyclic network

Conclusion

Cumulative processes



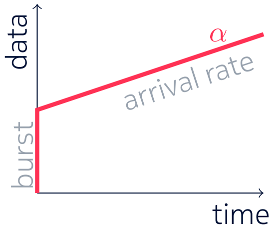
- $A : \mathbb{R}_+ \to \mathbb{R}_{\min,+}$: process of the cumulative arrivals, non-decreasing function
- $B : \mathbb{R}_+ \to \mathbb{R}_{\min,+}$: process of the cumulative departures, non-decreasing function
- Causality constraint: $A \geq B$

Arrival and service curves

Arrival curve

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

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



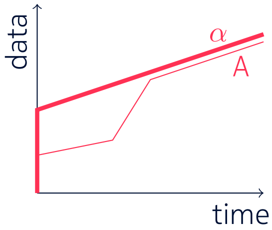
  Public NOKIA Bell Labs

Arrival and service curves

Arrival curve

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

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

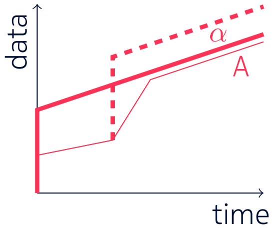


Arrival and service curves

Arrival curve

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

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



  Public NOKIA Bell Labs

Arrival and service curves

Arrival curve

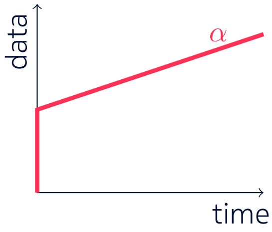
A is constrained by the function α if $\forall 0 \leq s \leq t$,

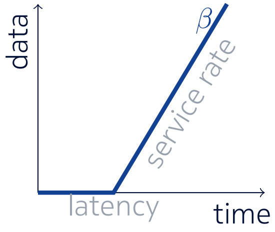
$$A(t) - A(s) \leq \alpha(t - s).$$

Strict service curve

A network element guarantees β for A if, while system not empty, B satisfies

$$B(t) \geq B(s) + \beta(t - s).$$





NOKIA Bell Labs

Arrival and service curves

Arrival curve

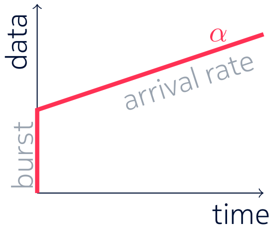
A is constrained by the function α if $\forall 0 \leq s \leq t$,

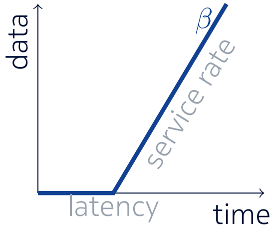
$$A(t) - A(s) \leq \alpha(t - s).$$

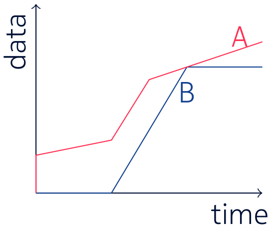
Strict service curve

A network element guarantees β for A if, while system not empty, B satisfies

$$B(t) \geq B(s) + \beta(t - s).$$





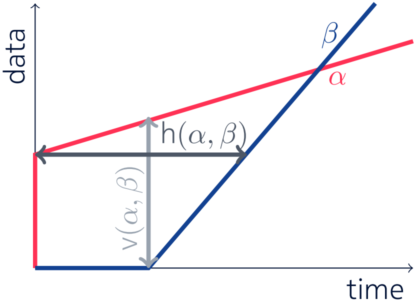


  Public NOKIA Bell Labs

From constraints to performance bounds

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

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

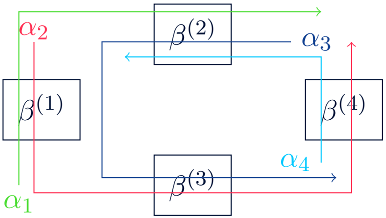


Performance bounds

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

  Public NOKIA Bell Labs

Model and hypotheses



Hypotheses

- m token-bucket arrival curves: $\alpha_i(t) = b_i + r_i t$;
- n rate-latency strict service curves: $\beta^{(j)}(t) = R_j(t - T_j)_+$.

Network calculus framework

Networks wih cyclic dependencies

Computing performance bounds in feed-forward networks

Performances in cyclic network

Conclusion

Stability in cyclic networks

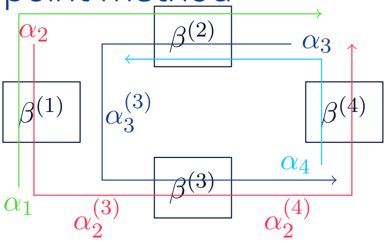
Consider a server offering a strict service curve $\beta : t \mapsto R(t - T)_+$ and a flow crossing it, with arrival curve $\alpha : t \mapsto b + rt$.

- This server is said unstable if its worst-case backlog is unbounded: $R < r$;
- This server is said critical if its worst-case backlog is bounded, but the lengths of its backlogged periods are not bounded bounded: $R = r$;
- This server is said stable if the length of its backlogged periods is bounded: $R > r$.

Definition (Global stability)

A network is globally stable if for all its servers, the length of the maximal backlogged period is bounded.

Fix-point method



(service curves and arrival curves of exogenous arrivals are constants of the problem)

$$\alpha_2^{(3)} = \mathsf{H}_2^{(1)}(\alpha_1, \alpha_2)$$
$$\alpha_2^{(4)} = \mathsf{H}_2^{(3)}(\alpha_1^{(3)}, \alpha_3^{(3)})\ldots$$

We write this equation for each output flow at each server and obtain a system

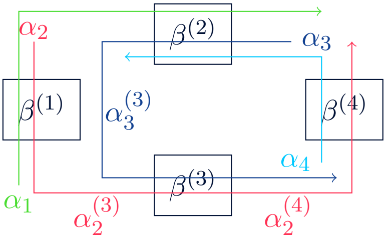
$$\boldsymbol{\alpha} = \mathbf{H}(\boldsymbol{\alpha})$$

Lemma

If the system is stable, then there exists a family $\boldsymbol{\alpha} = (\alpha_{\mathsf{i,j}})_{\mathsf{i,j}}$ of arrival curves for the flows $(\mathsf{F}_{\mathsf{i}}^{(\mathsf{j})})$ such that $\boldsymbol{\alpha} \leq \mathbf{H}(\boldsymbol{\alpha})$.

Take the best arrival curves, they will satisfy every inequality.

Fix-point method



(service curves and arrival curves of exogenous arrivals are constants of the problem)

$$\alpha_2^{(3)} = \mathsf{H}_2^{(1)}(\alpha_1, \alpha_2)$$
$$\alpha_2^{(4)} = \mathsf{H}_2^{(3)}(\alpha_1^{(3)}, \alpha_3^{(3)})\ldots$$

We write this equation for each output flow at each server and obtain a system

$$\boldsymbol{\alpha} = \mathbf{H}(\boldsymbol{\alpha})$$

- If service curves are rate-latency and arrival curves token bucket, this is a linear equation: $\mathbf{b} = \mathsf{M}\mathbf{b} + \mathsf{N}$.

   Nokia Bell Labs

Network calculus framework

Networks wih cyclic dependencies

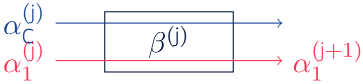
Computing performance bounds in feed-forward networks

Performances in cyclic network

Conclusion

Generic (min,plus) method

1. In the topological order of the servers, for each flow crossing the server:



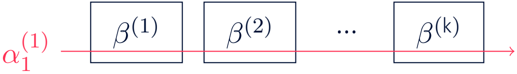
Residual service curve:

$$\beta_1^{(j)} = (\beta^{(j)} - \alpha_C^{(j)})_+$$

Output arrival curve:

$$\alpha_1^{(j+1)} = \alpha_1^{(j)} \oslash \beta_1^{(j)}$$

2. For the flow of interest



End-to-end service curve:

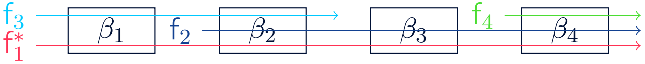
$$\beta = \beta_1^{(1)} * \beta_1^{(2)} * \cdots * \beta_1^{(k)}$$

3. Delay bound: $h(\alpha_1, \beta)$,
   Backlog bound: $v(\alpha_1, \beta)$

  Public NOKIA Bell Labs

Tight worst-case delays for tandem networks

Joint work with Thomas Nowak [Performance 2015]



Theorem

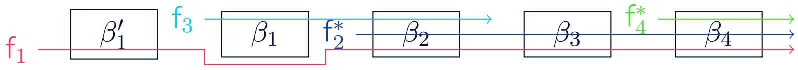
Consider a tandem network of n servers. The worst-case delay is linear in the bursts and latencies:

$$D = \sum_{j \in \mathbb{N}_n} \lambda_j T_j + \sum_{i \in \mathbb{N}_m} \mu_i b_i$$

where the coefficients λ_j and μ_i depend only on the arrival and service rates and can be effectively computed in time $O(n^2 + m)$.

Tight worst-case delays and backlog for tree networks

This theorem can be adapted to backlog at server n and for tree-topologies:



Theorem

Consider a **tree** network of n servers, and p **flows of interest** at server n. The worst-case backlog at server n for the flows of interests is linear in the bursts and latencies:

$$B = \sum_{j \in \mathbb{N}_n} \rho_j T_j + \sum_{i \in \mathbb{N}_m} \varphi_i b_i + \sum_{i \in \mathbb{N}_p} b_i^*$$

where the coefficients ρ_j and $\varphi_i < 1$ depend only on the arrival and service rates and can be effectively computed in time $O(n^2 + m + p)$.

  Public NOKIA Bell Labs

Sketch of the proof

1. Find properties of a worst-case scenario for the network
   - SDF (Shortest-to-destination first) service policy
   - one backlogged period for each server
   - minimum service
   - maximum arrivals
2. Backward induction on the servers
   - Only the dates of the start of backlogged periods need to be computed
   - they depend only on the amount of data transmitted at those time.

Comparison of the approaches

1. (min,plus):
   - Efficient algorithms (linear in the total length of the flows)
   - Pessimistic performance bounds (as soon as two servers and two flows)
   - Linear in T_j and b_i
2. Our approach:
   - Quadratic algorithm in tree network
   - Tight delay bound
   - Linear in T_j and b_i

Network calculus framework

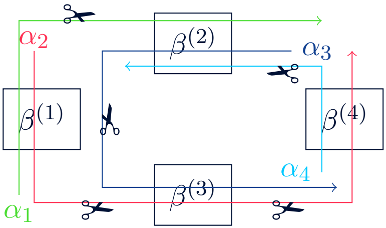
Networks wih cyclic dependencies

Computing performance bounds in feed-forward networks

Performances in cyclic network

Conclusion

Flow-based decomposition of the network

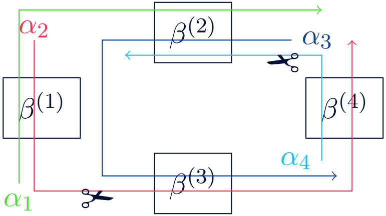


- (min,plus) approach: at each server for each flow

$$b_{i,k+1} \leq b_{i,k} + \frac{r_i}{R_j - \sum_{p \in Fl(j) \setminus \{i\}} r_p} \Big(\sum_{s \in S_j \setminus \{(i,k)\}} b_s + R_j T_j \Big).$$

  Public NOKIA Bell Labs

Flow-based decomposition of the network



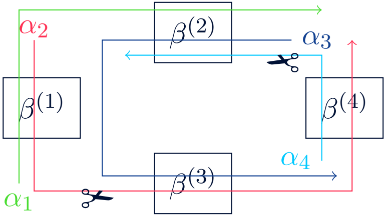
- (min,plus) approach: at each server for each flow

$$b_{i,k+1} \leq b_{i,k} + \frac{r_i}{R_j - \sum_{p \in Fl(j)\backslash\{i\}} r_p} \Big(\sum_{s \in S_j \backslash \{(i,k)\}} b_s + R_j T_j \Big).$$

- Our approach, with a tree decomposition:

$$b_{i,k+1} \leq \sum_{s \in S} \varphi_s^{i,k+1} b_s + \sum_{\{j | j \rightsquigarrow j_1\}} \rho_j^{i,k+1} T_j,$$

Flow-based decomposition of the network



Linear problem:

$$\boxed{\begin{array}{l}\text{Maximize } Q\mathbf{b} + C \\ \text{ such that } \mathbf{b} \leq M\mathbf{b} + N.\end{array}}$$

- (min,plus) approach: at each server for each flow

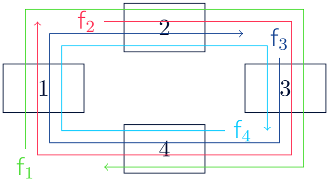
$$b_{i,k+1} \leq b_{i,k} + \frac{r_i}{R_j - \sum_{p \in Fl(j)\setminus\{i\}} r_p}\Big(\sum_{s \in S_j \setminus \{(i,k)\}} b_s + R_j T_j\Big).$$

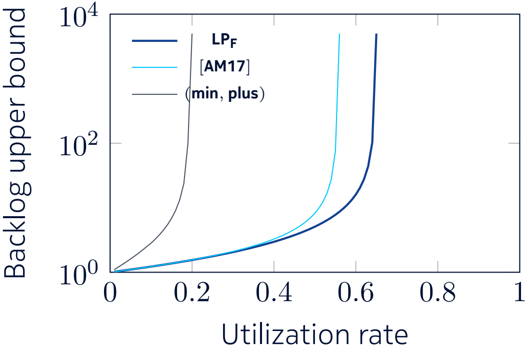
- Our approach, with a tree decomposition:

$$b_{i,k+1} \leq \sum_{s \in S} \varphi_s^{i,k+1} b_s + \sum_{\{j \mid j \rightsquigarrow j_1\}} \rho_j^{i,k+1} T_j,$$

  Public NOKIA Bell Labs

Comparison on the ring network





$\mathsf{n} = 10$ servers, $\mathsf{R} = 100\mathsf{Mb.s}^{-1}$, $\mathsf{L} = 1\mathsf{ms}$,
$\mathsf{b} = 1\mathsf{Mb}$, $\mathsf{r} = \frac{\mathsf{uR}}{\mathsf{n}}$

Backlog-based decomposition

Idea:

- worst-case bounds for flows ending at the same server do not happen at the same time;
- computing worst-case backlog of all flows following an arc might takes this phenomenon into account.

$$
\begin{aligned}
B_a &\le \sum_{s\in S} \varphi_s^a x_s + \sum_{\{j \mid j \rightsquigarrow j_1\}} \rho_j^a T_j \\
&\le \sum_{a'\in \mathbb{A}'} \left[\left(\max_{s\in S'_{a'}} \varphi_s^a \right) \left(\sum_{s\in S'_a} x_s \right) \right] + \sum_{i=1}^{m} \varphi_{(i,1)}^a b_i + \sum_{\{j \mid j \rightsquigarrow j_1\}} \rho_j^a T_j \\
&\le \sum_{a'\in \mathbb{A}'} \left(\max_{s\in S'_{a'}} \varphi_s^a \right) B_{a'} + \sum_{i=1}^{m} \varphi_{(i,1)}^a b_i + \sum_{\{j \mid j \rightsquigarrow j_1\}} \rho_j^a T_j .
\end{aligned}
$$

Ring stability revisited

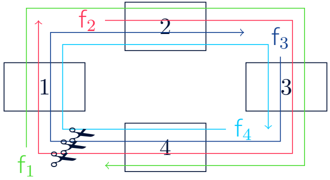
Theorem (TG96, LT04)

"The ring is stable" under assumption for stability of each server
Additional assumption: the traffic is upper-bounded in each link.

Our approach

$$B = \sum_{j \in \mathbb{N}_n} \rho_j T_j + \sum_{i \in \mathbb{N}_m} \varphi_i b_i + \sum_{i \in \mathbb{N}_p} b_i^*$$

where the coefficients ρ_j and $\varphi_i < 1$ depend only on the arrival and service rates.



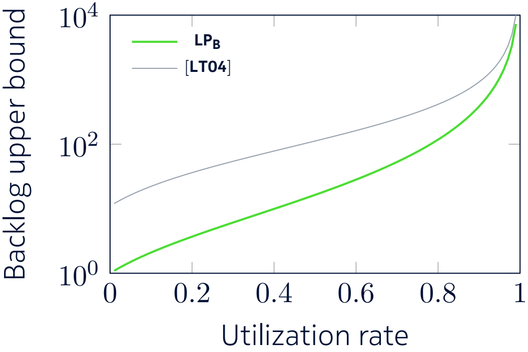
$$B \leq C + \varphi B$$

where $\varphi = \sup \varphi_j^n < 1$ and $B \leq \frac{C}{1-\varphi}$.

  Public

NOKIA Bell Labs

Comparison on the Ring Network



$$n = 10 \text{ servers}, R = 100\text{Mb.s}^{-1}, L = 1\text{ms}, b = 1\text{Mb}, r = \frac{uR}{n}$$

  Public NOKIA Bell Labs

Combining Flow-based and Backlog-based bounds

New set of linear constraints:

$$\mathcal{L} = \left\{ \begin{array}{ll} b_s \leq \sum_{s' \in S} \varphi^s_{s'} x^s_{s'} + C_s, & \forall s \in S \\ B_a \leq \sum_{s' \in S} \varphi^a_{s'} x^a_{s'} + C_a, & \forall a \in \mathbb{A}^r \\ 0 \leq x^s_{s'} \leq b_{s'}, & \forall s' \in S, s \in S \cup \mathbb{A}^r \\ \sum_{s' \in a} x^s_{s'} \leq B_a, & \forall a \in \mathbb{A}^r, s \in S \cup \mathbb{A}^r \end{array} \right\},$$

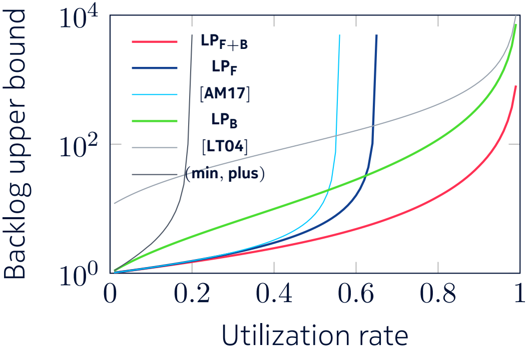
Number of constraints:

- Flow-based: $O(k)$
- Backlog-based: $O(a)$
- Combination: $O(k+a)^2$

where k is the number of flows is the tree-decomposition and a the number of arcs removed for this decompostion.

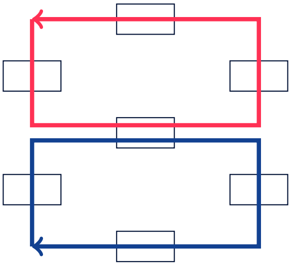
  Public NOKIA Bell Labs

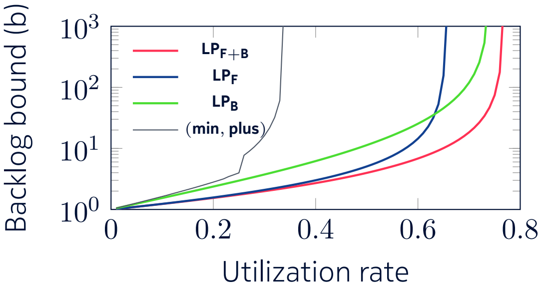
Comparison on the Ring Network



$n = 10$ servers, $R = 100\text{Mb.s}^{-1}$, $L = 1\text{ms}$, $b = 1\text{Mb}$, $r = \frac{uR}{n}$

Two-Ring network





Network calculus framework

Networks wih cyclic dependencies

Computing performance bounds in feed-forward networks

Performances in cyclic network

Conclusion

Conclusion and futurework

Conclusion

- A new efficient algorithm to compute tight worst-case delays and backlog.
- Application to networks with cyclic dependencies:
  - best stability conditions
  - stability of the ring without additional assumptions
- Implementation in a Python package: `https://github.com/nokia/NCBounds`

Future work

- Extension to feed-forward networks (we conjecture that a simple generalization can lead to the same approximation with one linear program)
- what is the best decomposition?
- Extension to some service policies (FIFO, GPS for example), maximum service rate...

  Public NOKIA Bell Labs

NOKIA