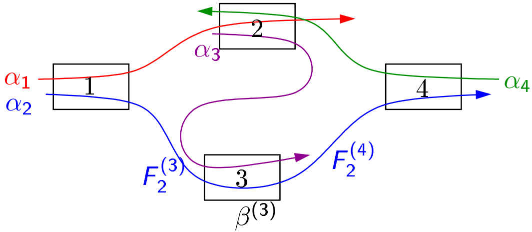
# Worst-case performance bounds in tree-network and application to networks with cyclic dependencies

Anne Bouillard (ENS, Paris)

Woneca 2016

# Model and hypotheses




## Objective

Is the network stable? performance bounds?

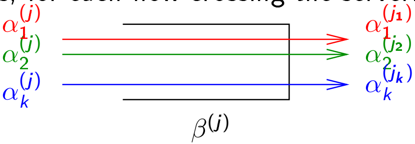
## Hypotheses

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

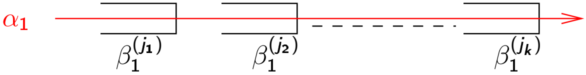
# Separated Flow Analysis (SFA) method

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

$$\beta_1^{(j)} = (\beta^{(j)} - \sum_{i=2}^{k} \alpha_i^{(j)})_+$$
$$\alpha_1^{(j_1)} = \alpha_1^{(j)} \oslash \beta_1^{(j)}$$




2. For the flow of interest (flow 1)




$$\beta = \beta_1^{(j_1)} * \beta_1^{(j_2)} * \cdots * \beta_1^{(j_k)}$$

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

- Efficient algorithms
- Pessimistic performance bounds
- Symbolic computation (for simple classes of functions)

# Greedy algorithm 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:*

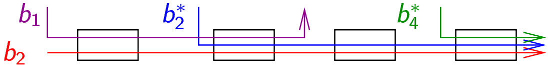
$$D = \sum_{j \in [\![n]\!]} \lambda_j T_j + \sum_{i \in [\![m]\!]} \mu_i b_i$$

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

- Efficient algorithm in tandem network
- Tight delay bound
- Symbolic on some parameters

# Greedy algorithm

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:*

$$B = \sum_{j \in [\![n]\!]} \lambda_j T_j + \sum_{i \in [\![m]\!]} \xi_i b_i + \sum_{i \in [\![p]\!]} b_i^*$$

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

# Computing the worst-case backlog

**begin**

$\xi_n^n \leftarrow (\sum_{i\leq n} r_i^*)(R_n - r_n^n)^{-1}$;

**for** $j$ **from** $n-1$ **to** $1$ **do**

$k \leftarrow n$;

**while** $\xi_{j+1}^k < (\sum_{i\leq j} r_i^* + \sum_{\ell>k} \xi_{j+1}^\ell r_j^\ell)(R_j - \sum_{\ell=j}^k r_j^\ell)^{-1}$ **do**

$\xi_j^k \leftarrow \xi_{j+1}^k$;

$k \leftarrow k-1$;

**for** $\ell$ **from** $j$ **to** $k$ **do**

$\xi_j^\ell \leftarrow (\sum_{i\leq j} r_i^* + \sum_{\ell>k} \xi_{j+1}^\ell r_j^\ell)(R_j - \sum_{\ell=j}^k r_j^\ell)^{-1}$;

**for** $j$ **from** $1$ **to** $n$ **do** $\lambda_j \leftarrow \sum_{i\leq j} r_i^* + \sum_{k\leq j} \xi_j^k r_j^k$;

**end**

# Stability in cyclic networks

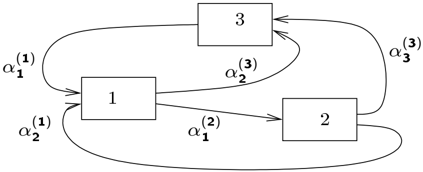
Consider a server offering a strict service curve $\beta : t \mapsto R(t - T)_+$ and a flow crossing it, with arrival curve $\alpha : t \mapsto b + rt$.

- This server is said *unstable* if its worst-case backlog is bounded: $R < r$;
- 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.

# Stopped-time/fix-point method




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

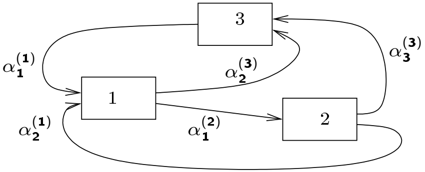
$$\alpha_1^{(2)} = H_1^{(1)}(\alpha_1^{(1)}, \alpha_2^{(1)})$$
$$\alpha_2^{(3)} = H_2^{(1)}(\alpha_1^{(1)}, \alpha_2^{(1)})$$
$$\alpha_1^{(1)} = H_2^{(3)}(\alpha_3^{(3)}, \alpha_2^{(3)})\ldots$$

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

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

- We can assume w.l.o.g. that $\mathbf{H}$ is non-decreasing in any variable
- If $\boldsymbol{\alpha}$ is a solution of $\boldsymbol{\alpha} = \mathbf{H}(\boldsymbol{\alpha})$, is it a family of arrival curves for the intermediate flows?
- If service curves are rate-latency and arrival curves token bucket, this is a linear equation: $\mathbf{b} = M\mathbf{b} + N$.

# Stopped-time/fix-point method




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

$$\alpha_1^{(2)} = H_1^{(1)}(\alpha_1^{(1)}, \alpha_2^{(1)})$$
$$\alpha_2^{(3)} = H_2^{(1)}(\alpha_1^{(1)}, \alpha_2^{(1)})$$
$$\alpha_1^{(1)} = H_2^{(3)}(\alpha_3^{(3)}, \alpha_2^{(3)})\dots$$

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 $\boldsymbol{\alpha} = (\alpha_{i,j})_{i,j}$ of arrival curves for the flows $(F_i^{(j)})$ such that $\boldsymbol{\alpha} \leq \mathbf{H}(\boldsymbol{\alpha})$.*

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

# Stopped times

From [Le Boudec, Thiran, 2001]

Let $\boldsymbol{\alpha}_0$ be the greatest finite solution of $\boldsymbol{\alpha} \leq \mathbf{H}(\boldsymbol{\alpha})$.
Then $\boldsymbol{\alpha}$ is a family of arrival curves for the network.

## Stopped times at $\tau > 0$

Exogenous arrivals in the network are stopped at time $\tau$:
arrival curves of type

$$\alpha^\tau = \alpha(t \wedge \tau).$$

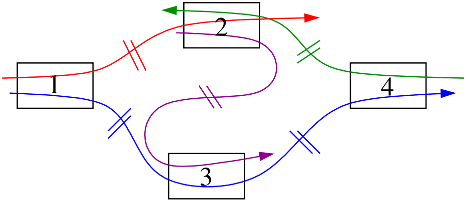
For all $\tau$, the system is stable (finite amount of arrivals in the system), so there exists a solution

$$\boldsymbol{\alpha}^\tau \leq \mathbf{H}(\boldsymbol{\alpha}^\tau) \quad \text{so} \quad \boldsymbol{\alpha}^\tau \leq \boldsymbol{\alpha}^0.$$

As $\mathbf{H}$ is non-decreasing, $\mathbf{H}(\boldsymbol{\alpha}^\tau) \leq \mathbf{H}(\boldsymbol{\alpha}_0) = \boldsymbol{\alpha}_0$.
So $\boldsymbol{\alpha} = \sup_\tau \boldsymbol{\alpha}^\tau$ is a solution, and $\boldsymbol{\alpha} \leq \boldsymbol{\alpha}_0$, which is also a solution.

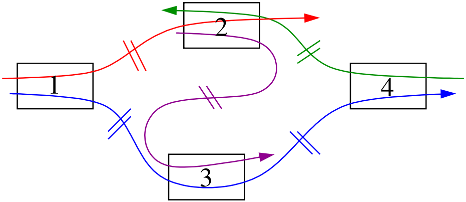
# Decomposition of the network

- SFA decomposition: at each server for each flow

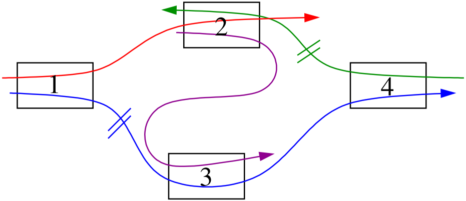


# Decomposition of the network

- SFA decomposition: at each server for each flow

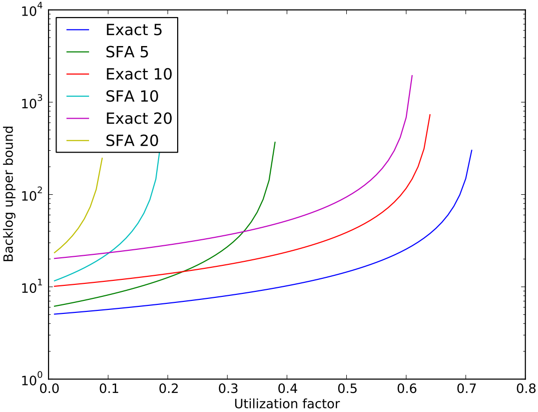



- "Exact" decomposition: decompose into trees



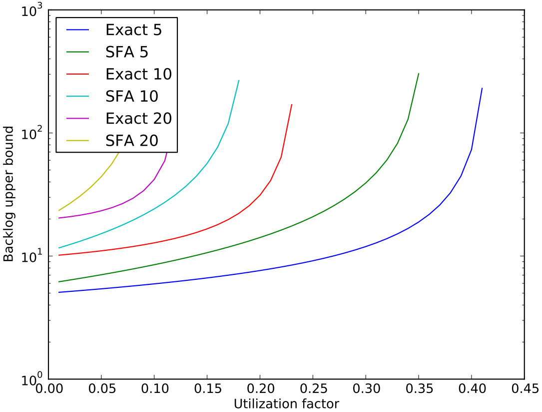
# Numerical comparisons

## Unidirectional ring



# Numerical comparisons

## Bidirectional ring



# Ring stability revisited

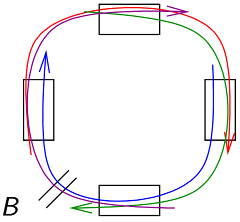
**Theorem (Tassiulas, Georgiadis, 96)**

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

We have

$$B = \sum_{j \in [\![n]\!]} \lambda_j T_j + \sum_{i \in [\![m]\!]} \xi_i b_i + \sum_{i \in [\![p]\!]} b_i^*$$

where the coefficients $\lambda_j$ and $\xi_i < 1$ depend only on the arrival and service rates.

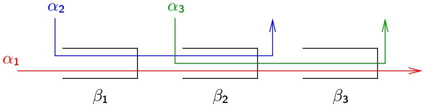



$$B \leq C + \xi B$$

where $\xi = \sup \xi_j^n < 1$ and

$$B \leq \frac{C}{1 - \xi}.$$

# Properties of a worst-case scenario




**(H$_1$)** Service policy is SDF (shortest-to-destination first): for two flows $i$ and $j$, if $d_i < d_j$, then flow $i$ is served with higher priority than flow $j$.

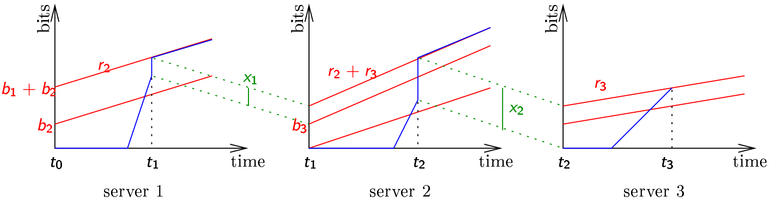
**(H$_2$)** Server $j$ has the unique backlogged period $(t_{j-1}, t_j)$ and provides infinite service outside its backlogged period.

**(H$_3$)** Each server provides exact service in its backlogged period and $t_j - t_{j-1} \geq T_j$.

**(H$_4$)** The new arrivals at server $j$ are maximal from time $t_{j-1}$ on and zero before that.

**(H$_5$)** The flows of interest are always transmitted instantaneously.

# Properties of a worst-case scenario



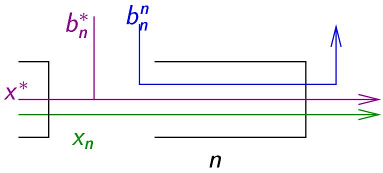

## Theorem

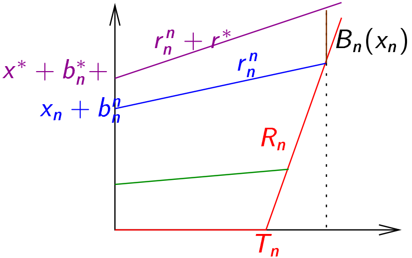
*There exists a worst-case scenario that satisfy* $(\mathbf{H_1}, \ldots, \mathbf{H_5})$.

**Consequence:** we only have to optimize on the dates of start of backlog period $t_0, t_1, \ldots, t_n$.

# A backward computation

**Server** $n$**:**



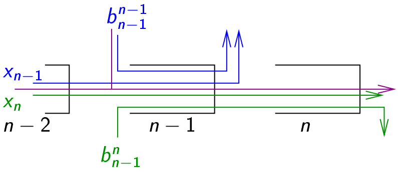

$$B_n(x_n, x^*) = b_n^* + x^* + r^*\left(T_n + \frac{x_n + b_n^n + r_n^n T_n}{R_n - r_n^n}\right)$$

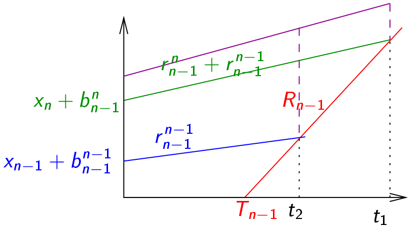
$$= b_n^* + x^* + \lambda_n T_n + \frac{r^*}{R_n - r_n^n}(x_n + b_n^n).$$

$\xi_n^n = \frac{r^*}{R_n - r_n^n} < 1.$

# A backward computation

**Server** $n-1$**:** $(q_i = x_i + b^i_{n-1} + r^i_{n-1} T_{n-1};\ T = T_{n-1})$



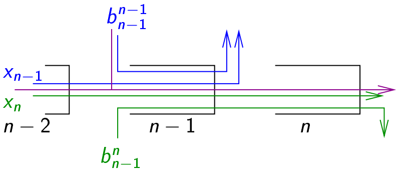



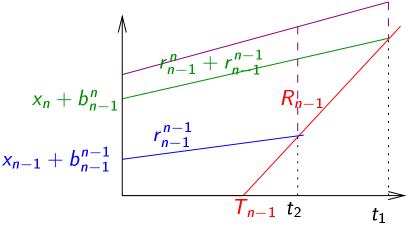

$$B^1_{n-1}(x_{n-1}, x_n, x^*) = B_n(0, b^* + r^* t_1)$$

$$B^2_{n-1}(x_{n-1}, x_n) = B_n(q_n + r^n_{n-1} t_2, b^* + r^* t_2)$$

# A backward computation

**Server** $n-1$**:** $(q_i = x_i + b^i_{n-1} + r^i_{n-1}\,T_{n-1};\ T = T_{n-1})$




$$B^1_{n-1} > B^2_{n-1} \Leftrightarrow \frac{1}{R_n - r^n_n} < \frac{1}{R_{n-1} - r^n_{n-1} - r^{n-1}_{n-1}}.$$

Delay from server $j$ when flows ending after server $k$ are served instantaneously.

$$B_j^k(x_j^j, \dots, x_j^n) = B_{j+1}(0, \dots, 0, x_{j+1}^{k+1}, \dots, x_{j+1}^n, x^* + r^* t_j)$$

with $t_j = \frac{\sum_{\ell=j}^k Q_j^\ell}{R_j - \sum_{\ell=j}^k r_j^\ell}$, $x_{j+1}^\ell = Q_j^\ell + r_j^\ell t_j$, and $Q_j^\ell = x_j^\ell + r_j^\ell T_j$.

**Lemma**

*There exists $k$ such that $\forall x_j, \dots, x_n$,*

$$B_j(x_j, \dots, x_n) = B_j^k(x_j, \dots, x_n).$$

Finally,

$$B = B_1(0, \dots, 0).$$

# Computing the worst-case backlog

**begin**
- $\xi_n^n \leftarrow (\sum_{i\leq n} r_i^*)(R_n - r_n^n)^{-1}$;
- **for** $j$ **from** $n-1$ **to** $1$ **do**
  - $k \leftarrow n$;
  - **while** $\xi_{j+1}^k < (\sum_{i\leq j} r_i^* + \sum_{\ell > k} \xi_{j+1}^\ell r_j^\ell)(R_j - \sum_{\ell=j}^k r_j^\ell)^{-1}$ **do**
    - $\xi_j^k \leftarrow \xi_{j+1}^k$;
    - $k \leftarrow k-1$;
  - **for** $\ell$ **from** $j$ **to** $k$ **do**
  - $\xi_j^\ell \leftarrow (\sum_{i\leq j} r_i^* + \sum_{\ell > k} \xi_{j+1}^\ell r_j^\ell)(R_j - \sum_{\ell=j}^k r_j^\ell)^{-1}$;
- **for** $j$ **from** $1$ **to** $n$ **do** $\lambda_j \leftarrow \sum_{i\leq j} r_i^* + \sum_{k\leq j} \xi_j^k r_j^k$;

**end**

# Conclusion and future work

## 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

## Future work

- Application to stochastic network calculus
- Extension to feed-forward networks (we conjecture that a simple generalization can lead to the same approximation with one linear program)
- Improvement of the conditions with the “ring trick”
- Extension to some service policies (FIFO for example)