Computing Routes and Delay Bounds for the Network-on-Chip of the Kalray MPPA2 Processor

4th Workshop on Network Calculus (WoNeCa-4)

Marc Boyer (ONERA) Benoît Dupont de Dinechin (Kalray)
Amaury Graillat (Verimag, Karlay)
Lionel Havet (RealTime-at-Work)

February 28, 2018, Erlangen

ONERA

THE FRENCH AEROSPACE LAB

retour sur innovation

Outline

Introduction

- application of Deterministic Network Calculus to a real NoC
- results presented in [2]
- slides have been presented at ERTSS 2018
- new slides done for WoNeCa
- augmented with on-going works (no peer review, done at airport during connection...)

Outline

The MMPA processor and its NoC

From NoC to network calculus

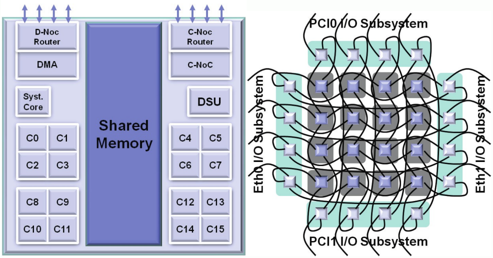
Network calculus modelling

Network calculus computation

Experiments

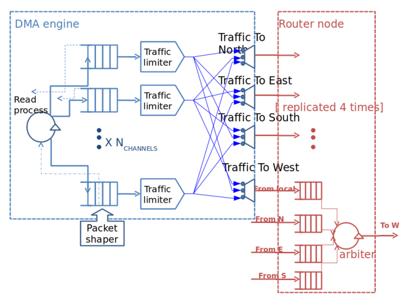
Conclusion

Kalray architecture



- A 256-cores chip [4]
- torus topology
- 16 tiles
- 16 “simple” cores per tile

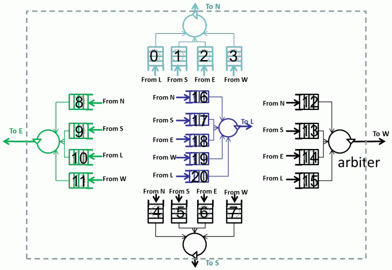
Kalray Network Adapter



- 8 channels [4]
- explicit communications
- per channel traffic limiter (token-bucket)

⇒ HW support for latency computation

Kalray Network router



- virtual cut-through forwarding
- round-robin arbitration
- buffers large enough to store several messages
- wormhole switching $\Rightarrow$ back pressure in case of buffer overflow
- link throughput: one flit per cycle

ONERA
THE FRENCH AEROSPACE LAB

Outline

Why the NoC fits NC

- assume static routing
- static number of flows
- traffic limiter $\Rightarrow$ token-bucket arrival curve
- round-robin arbiter $\Rightarrow$ RR residual service
- avoid buffer overflow $\Rightarrow$ no wormhole back-pressure
    - how to avoid buffer overflow ?

Why the NoC fits NC

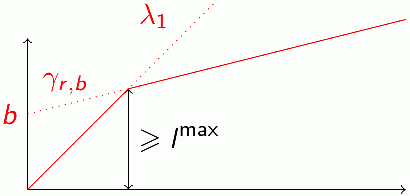
- assume static routing
- static number of flows
- traffic limiter $\Rightarrow$ token-bucket arrival curve
- round-robin arbiter $\Rightarrow$ RR residual service
- avoid buffer overflow $\Rightarrow$ no wormhole back-pressure
  - how to avoid buffer overflow ?
  - reduces load (burst and rate of token-bucket)

Outline

Flow modelling

Obvious:

- traffic limiter is token-bucket shaper
  $\Rightarrow$ arrival curve $\gamma_{r,b}$
- output link has maximal capacity of one flit per cycle
  $\Rightarrow$ arrival curve λ_1
- take minimum of both
- Rq: burst must be sufficient to send one packet



ONERA THE FRENCH AEROSPACE LAB

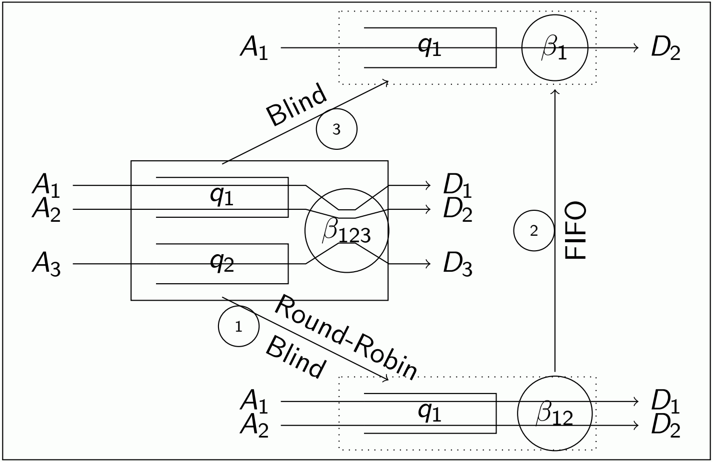
Router modelling: principles

- Round-Robin betwen queues
- FIFO between flows in the same queue
- but FIFO/RR can be approximated per blind arbiter also

Modeling principles

1. Residual service per queue (RR or blind)
2. FIFO per flow in queue
3. or Blind per flow

Router modelling: full picture

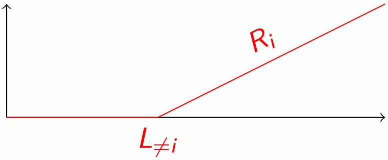


ONERA THE FRENCH AEROSPACE LAB

Round Robin arbiter

- Round-Robin residual service

$$\beta_i^{RR} = \beta_{R_i, L_{\neq i}} \qquad L_{\neq i} = \sum_{j \neq i} L_j^{max} \qquad R_i = \frac{l_i^{min}}{l_i^{min} + L_{\neq i}} \tag{1}$$



- Blind multiplexing

$$\beta_i^{Blind} = \left[\beta - \sum_{j \neq i} \alpha_j \right]_{\uparrow}^{+} \tag{2}$$

- Depending on load, packet sizes, β_i^{RR} and β_i^{Blind} incomparable

FIFO results

- local global delay (TFA-like)

$$\beta_i^{g-\text{FIFO}} = \delta_d \qquad d = d\left(\sum_{i=1}^{n} \alpha_i, \beta\right)$$

- the θ result: for all $\theta \geqslant 0$

$$\beta_i^{\theta-\text{FIFO}} = \left[\beta - \sum_{j \neq i} \alpha_j * \delta_\theta\right]_{\uparrow}^{+} \wedge \delta_\theta$$

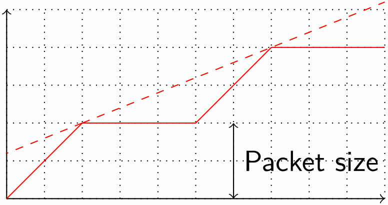
- the specific case of linear curves (token-bucket γ_{r_i,b_i} and rate-latency $\beta_{R,T}$)

$$\beta_i^{l-\text{FIFO}} = \beta_{R_i,T_i} \quad R_i = R - \sum_{j \neq i} r_j \quad T_i = T + \frac{\sum_{j \neq i} b_j}{R} \tag{3}$$

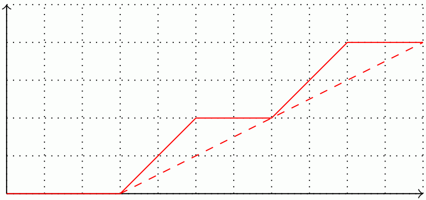
- LP: transformation into linear-programming problem, exact result

The specific case of constant size packets

Smaller arrival curve



Bigger service curve



Considering packets of constant size

- better arrival curves
- better service curves

On-going work, not presented in [2].

Outline

A linear model

Principle:

- consider token-bucket arrival curves
- per queue residual service: either RR or blind residual service
- per queue residual service: FIFO result for linear curves
- per flow shaping
- delay by SFA algorithm

Resolution:

- a linear programming problem
- goals
  - chose routing
  - maximise per flow rate (while ensuring fairness)
  - avoid buffer overflow

Local delays

Assume that:

- routing is done
- flow parameters are set

Resolution: adaptation of an AFDX tool

- per queue residual service: either RR or blind residual service
- per flow residual service: the aggregate queue delay (TFA-like)
- per group link shaping
- with arrival and service curves
  - linear model
  - constant packet size enhancement, not presented in [2]

LP approach

Assume that:

- routing is done
- flow parameters are set

Resolution:

- per queue residual service: either RR or blind residual service computed from local delay tool
- per flow residual service: resolution by LP problem, exact solution
- interfering flow arrival curves: linear model, from local delay tool
- no shaping

On-going work, not presented in [2].

SFA approach

Assume that:

- routing is done
- flow parameters are set

Resolution: adaptation of an AFDX tool

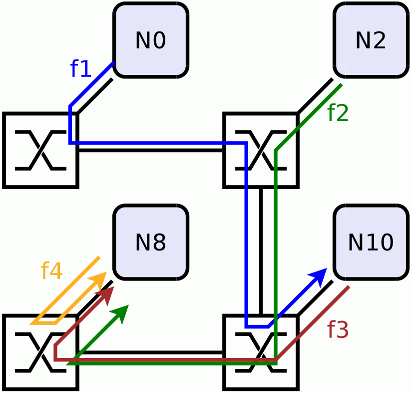
- per queue q residual service: either RR or blind residual service β_q
- per flow residual service: $\beta^{\theta-\text{FIFO}}$ with $\theta = \sup\{ t \mid \beta_q(t) = 0 \}$
- constant packet size enhancement
- end-to-end convolution $\beta^{\text{NoC}} = \beta^{\theta-\text{FIFO}}_{q_1} * \beta^{\theta-\text{FIFO}}_{q_2} * \cdots * \beta^{\theta-\text{FIFO}}_{q_n}$ with $q_1, q_2 \ldots q_n$ the sequence of crossed queues.

On-going work, not presented in [2].

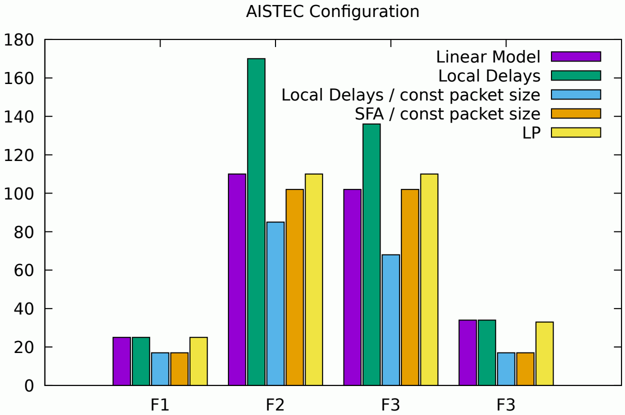
Outline

First experiment: test case

- Example from [3]
- Four flows $f_1, \ldots, f_4$
- Maximum packet size: 17flits

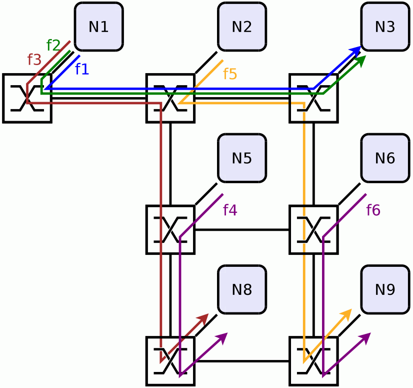


First experiment: Results

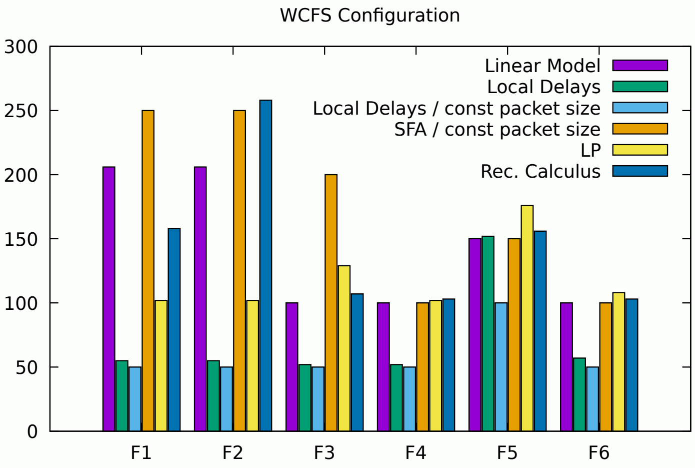


Second experiment: test case

- Example from [1]
- Six flows $f_1, \ldots, f_4$
- Maximum packet size: 50flits (unrealistic)
- Very small rate
- Comparaison with Recursive Calculus

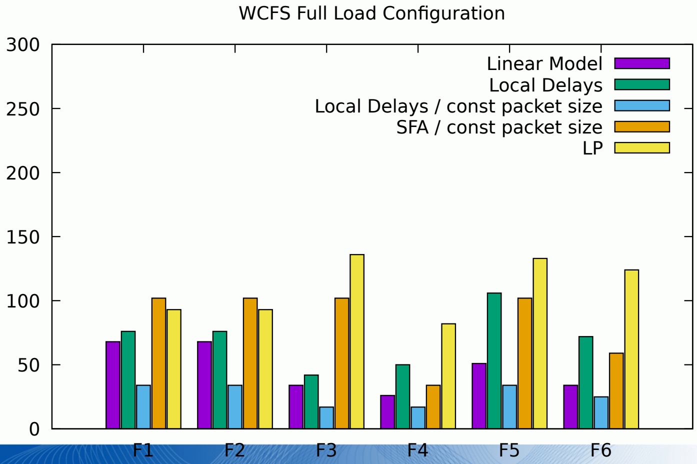


Second experiment: Results



Third experiment: Results

- Same as previous
- Except higher rate (full link use) and smaller bursts



Outline

Conclusion

No clear conclusion can be done from only 3 examples with 4-6 flows, but...

- NoC can provide guaranteed service
- Modelling shaping is important
- Modelling packet size may improve
- More work still to be done

Bibliography I

Hamdi Ayed, Jérôme Ermont, Jean-luc Scharbarg, and Christian Fraboul.
Towards a unified approach for worst-case analysis of tilera-like and kalray-like noc architectures.
In *Proc. of the 12th IEEE World Conf. on Factory Communication Systems (WFCS 2016), WiP Session*, Aveiro, Portugal, 2016. IEEE.

Marc Boyer, Benoît Dupont de Dinechin, Amaury Graillat, and Lionel Havet.
Computing routes and delay bounds for the network-on-chip of the kalray mppa2 processor.
In *Proc. of the 9th European Congress on Embedded Real Time Software and Systems ($ERTS^2$ 2018)*, Toulouse, France, January 2018.

B. Dupont de Dinechin and A. Graillat.
Network-on-chip service guarantees on the Kalray MPPA-256 Bostan processor.
In *Proc. of the 2nd Inter. Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems – AISTECS '17*, 2017.

Benoît Dupont de Dinechin, Yves Durand, Duco van Amstel, and Alexandre Ghiti.
Guaranteed services of the noc of a manycore processor.
In *Proc. of the 7th Int. Workshop on Network on Chip Architectures (NoCArc'14)*, Cambridge, United Kingdom, December 2014.

Bibliography II