Kalray MPPA®

Massively **P**arallel **P**rocessor **A**rray

MPPA®-256 Bostan Manycore Processor
Guaranteed Services

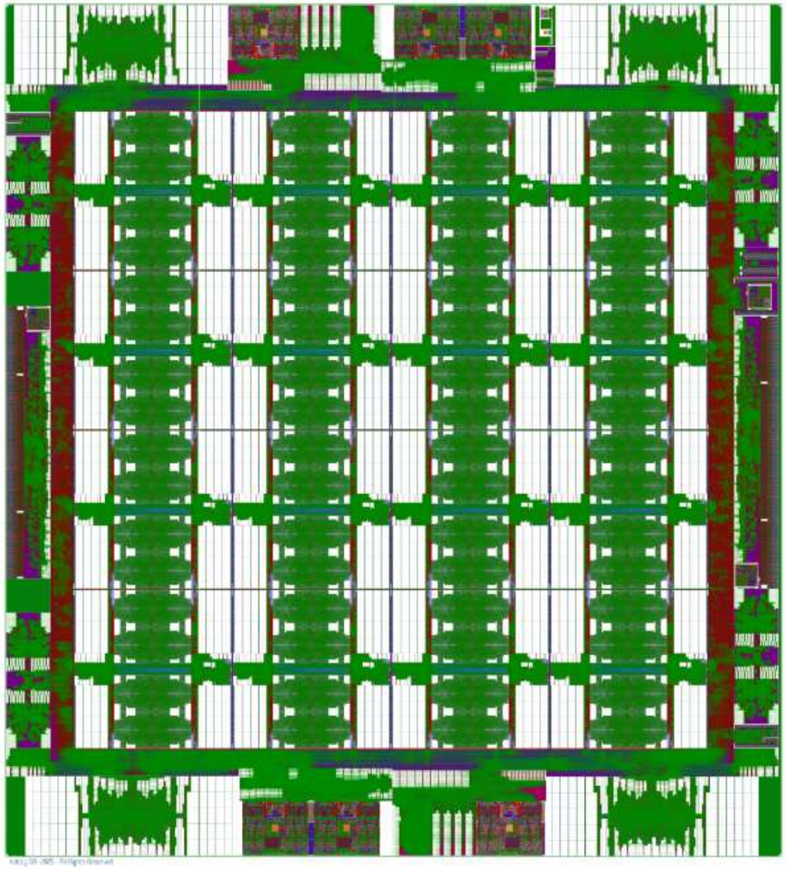
Benoît Dupont de Dinechin, CTO

Outline

- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

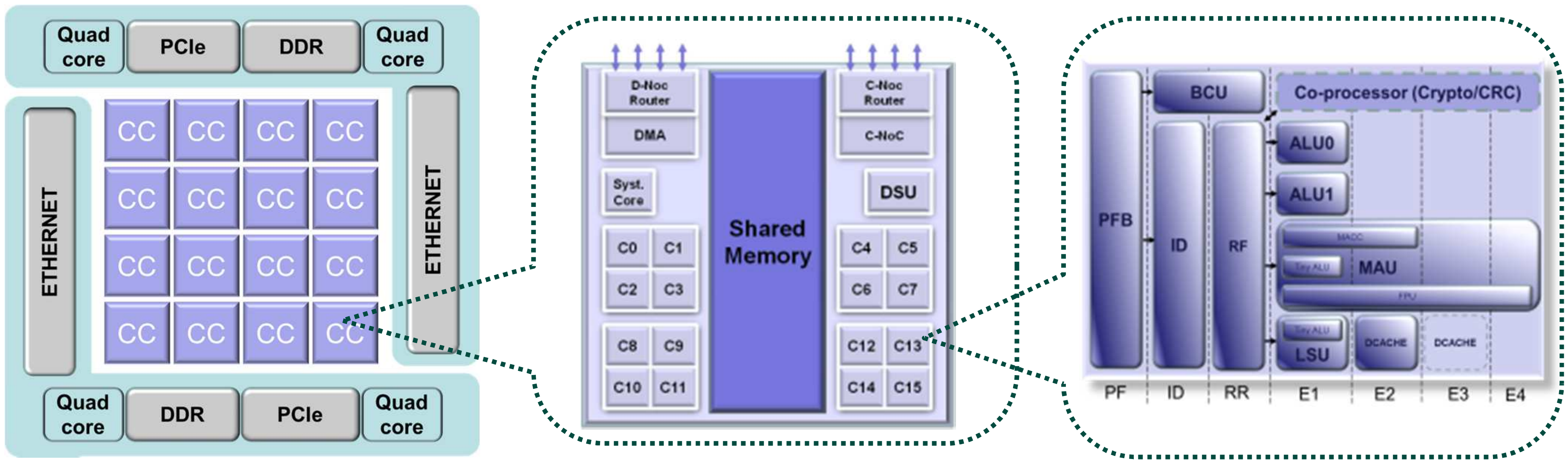
MPPA®-256 Bostan Processor

256 + 32 VLIW cores / 18 address spaces / 2D Torus dual NoC



- Physical characteristics
  - TSMC CMOS 28HP
  - 100µW/MHz per core + L1 caches
  - 2W to 3W leakage
- Processor interfaces
  - 2x DDR3 Memory interfaces
  - 2x PCIe Gen3 8-lane interface
  - 8x 1G/10G or 2x 40G Ethernet interfaces
  - SPI/I2C/UART interfaces
  - Universal Static Memory Controller (NAND/NOR/SRAM)
  - GPIOs with Direct NoC Access
  - NoC extension through Interlaken interface (NoCX)

MPPA®-256 Bostan Processor Architecture



Manycore Processor	Compute Cluster	VLIW Core

Manycore Processor

- 16 compute clusters
- 2 I/O clusters each with quad-core CPUs, DDR3, 4 Ethernet 10G and 8 PCIe Gen3
- Data and control networks-on-chip
- Distributed memory architecture
- 634 GFLOPS SP for 25W @ 600Mhz

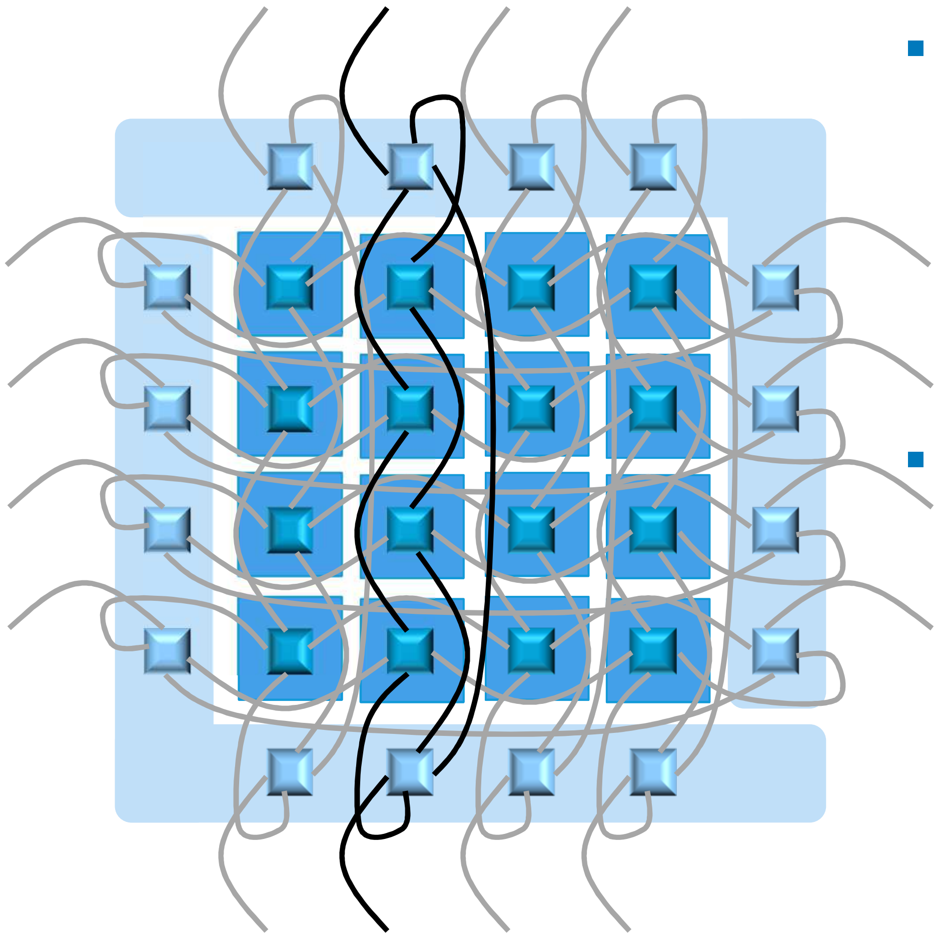
Compute Cluster

- 16 user cores + 1 system core
- NoC Tx and Rx interfaces
- Debug & Support Unit (DSU)
- 2 MB multi-banked shared memory
- 77GB/s Shared Memory BW
- 16 cores SMP System

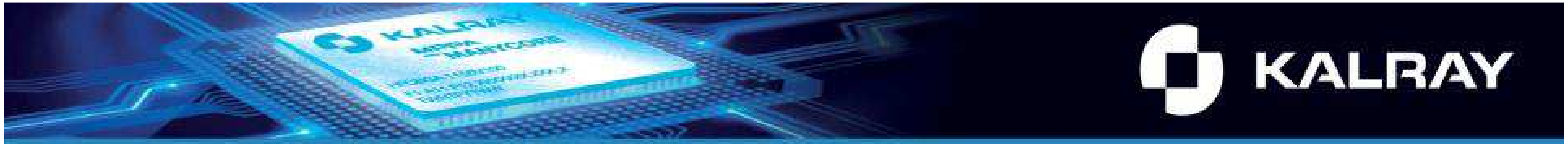
VLIW Core

- 32-bit or 64-bit addresses
- 5-issue VLIW architecture
- MMU + I&D cache (8KB+8KB)
- 32-bit/64-bit IEEE 754-2008 FMA FPU
- Tightly coupled crypto co-processor
- 2.4 GFLOPS SP per core @600Mhz

MPPA®-256 Bostan Network-on-Chip (NoC)

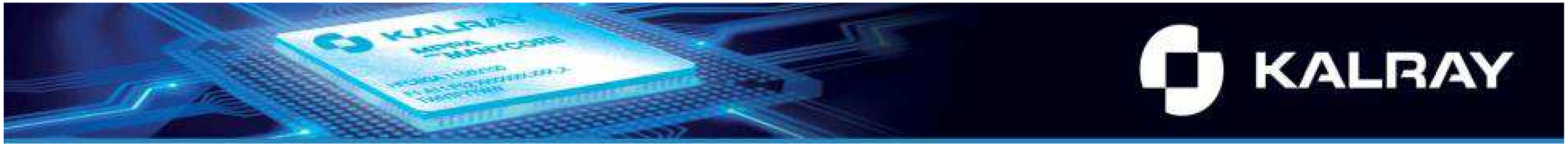


- Dual 2D-torus NoC
  - D-NoC: high-bandwidth RDMA
  - C-NoC: low-latency mailboxes
  - 4B/cycle per link direction per NoC
  - Nx10Gb/s NoC extensions for connection to FPGA or other MPPA®
- Predictability
  - Data NoC is configured by selecting routes and injection parameters
  - Routing ensure deadlock-free traffic
  - Injection parameters are the (σ,ρ) or (burst, rate) of network calculus



Interconnection Network Concepts

- Topology
  - How the nodes are connected together
  - Direct network if routing nodes can be endpoints
- Switching
  - Allocation of network resources (bandwidth, buffer capacity, …) to information flows
- Flow control
  - How a downstream node forwards availability to an upstream node
  - Applies at hop level, entry-to-exit level, and transport level
- Routing
  - Path selection between a source and a destination node in a particular topology

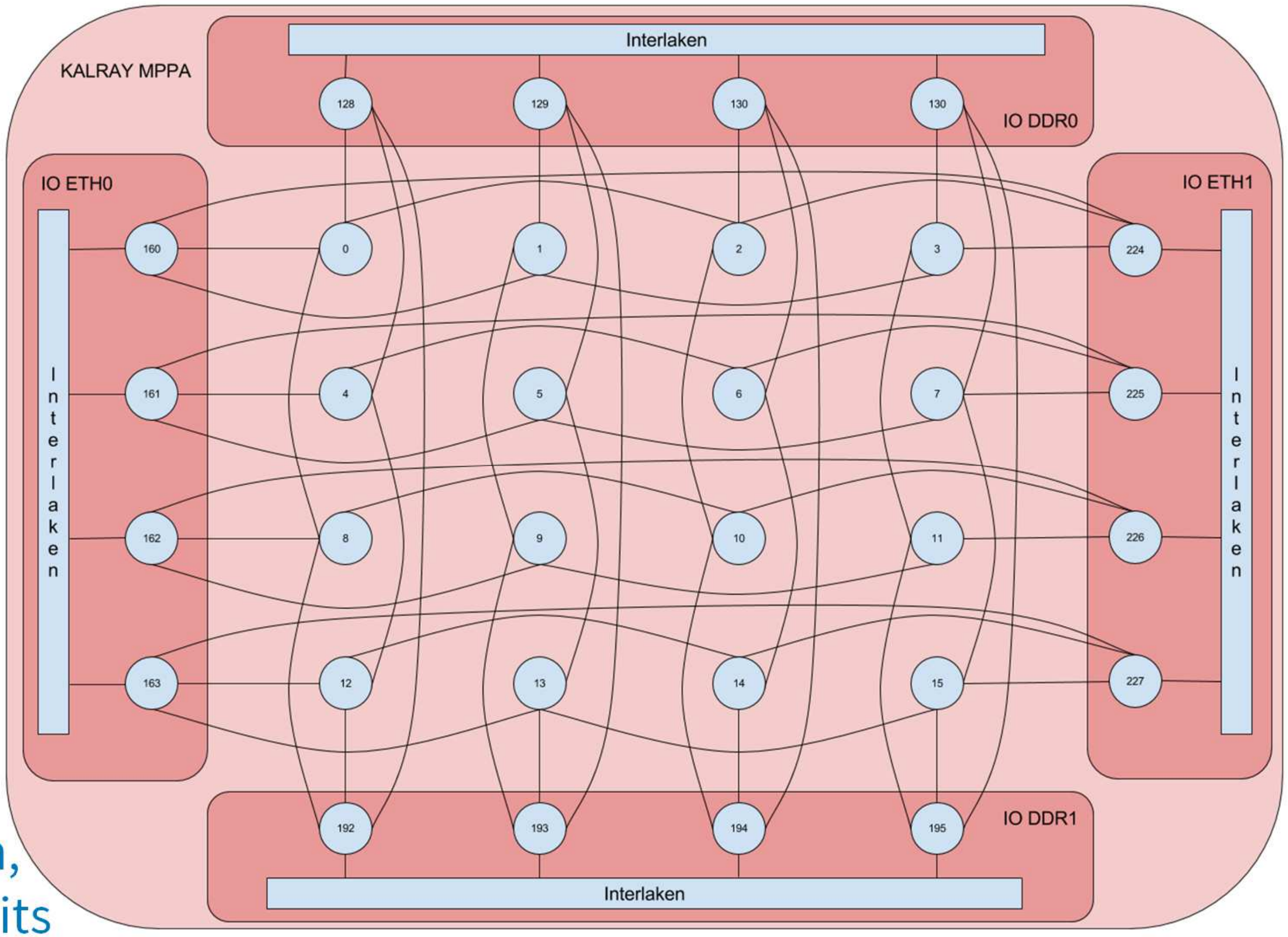


Outline

- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

MPPA®-256 Bostan NoC Topology

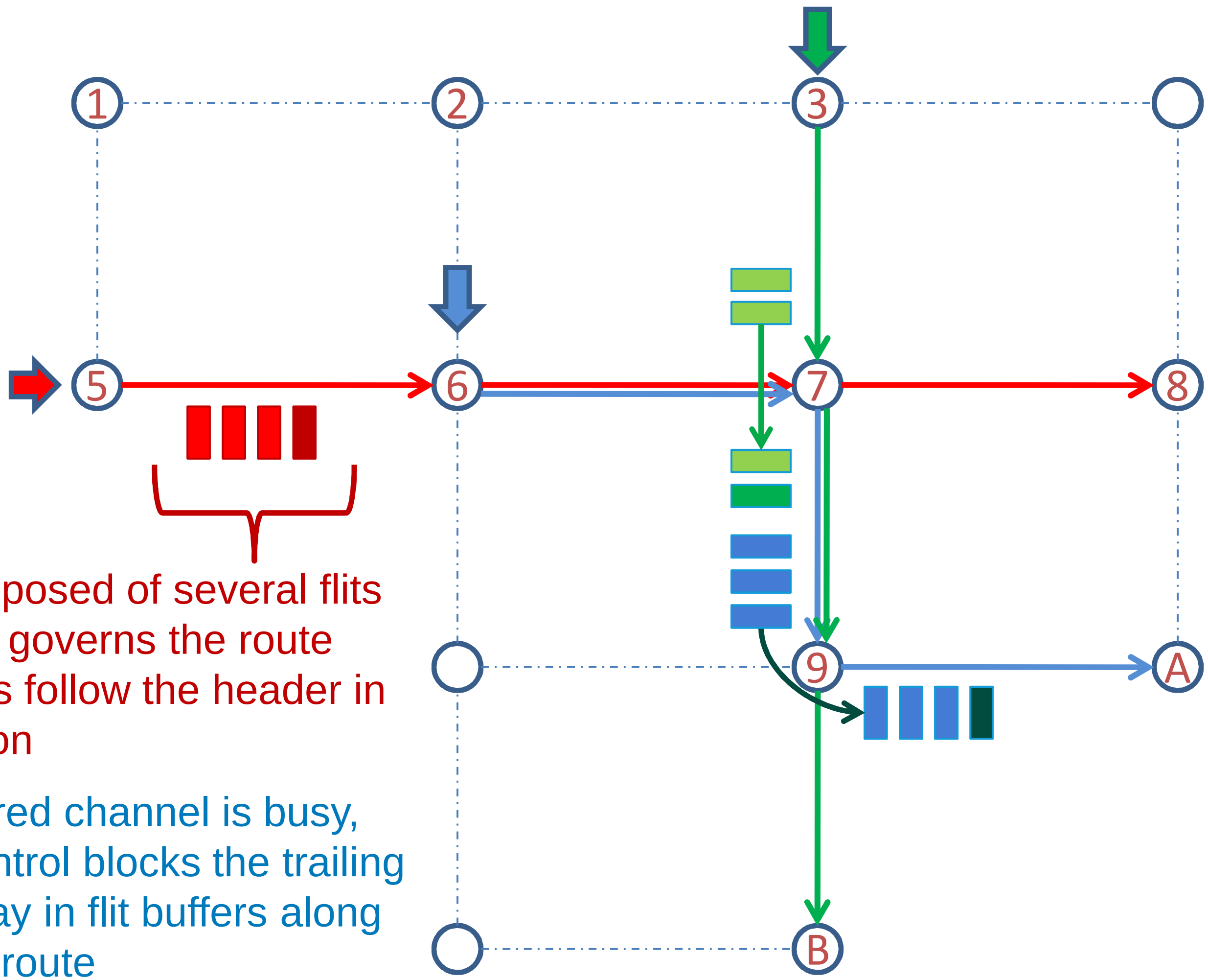
- 2-D Torus
  - Direct
  - Folded
  - I/O nodes
  - No virtual channels
- Dual NoC
  - D-NoC for DMA transfers
  - C-NoC for mailboxes, synchronization, and D-NoC credits



MPPA®-256 Bostan NoC Switching

- Network switching techniques
  - Circuit switching: network resources are dedicated over an end-to-end path before transmission starts
  - Packet switching:
    - Store and forward: node buffers entire packet before forwarding
    - Virtual cut-through: node starts forwarding as soon as buffer space for a whole packet is available on the next node
    - Wormhole switching: the packet is decomposed into flits that travel in a pipelined fashion, buffering is applied at flit level
- The MPPA® NoC is wormhole switching with source routing
  - A packet is composed of header flits and payload flits (32-bit flits)
  - The packet follows a route determined by a bit string in the header

Wormhole Switching Illustrated

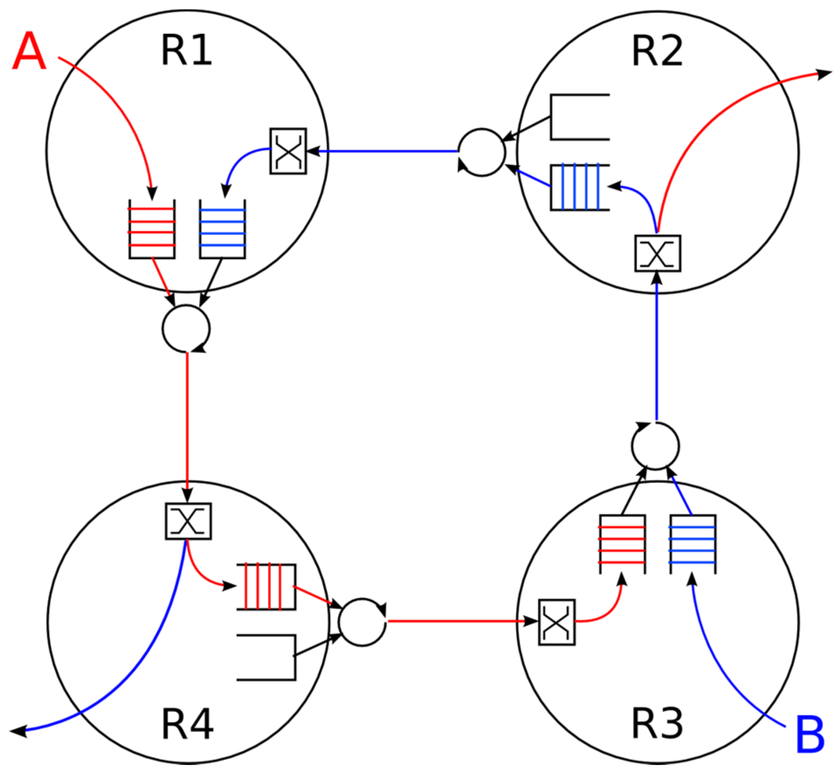


A packet is composed of several flits
The header flits governs the route
The payload flits follow the header in a pipeline fashion

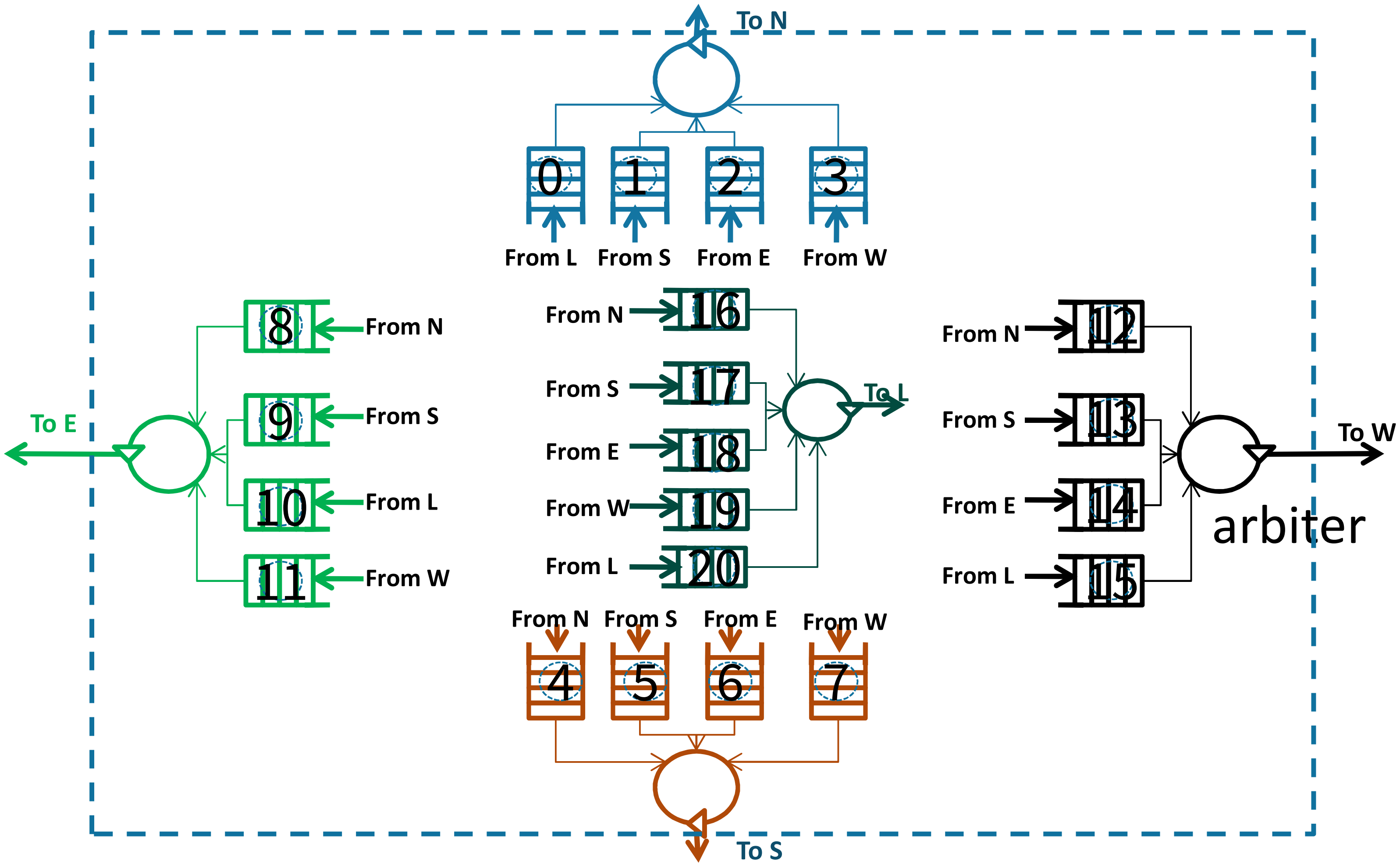
When the required channel is busy, the hop flow control blocks the trailing flits and they stay in flit buffers along the established route

Wormhole Switching NoC Issues

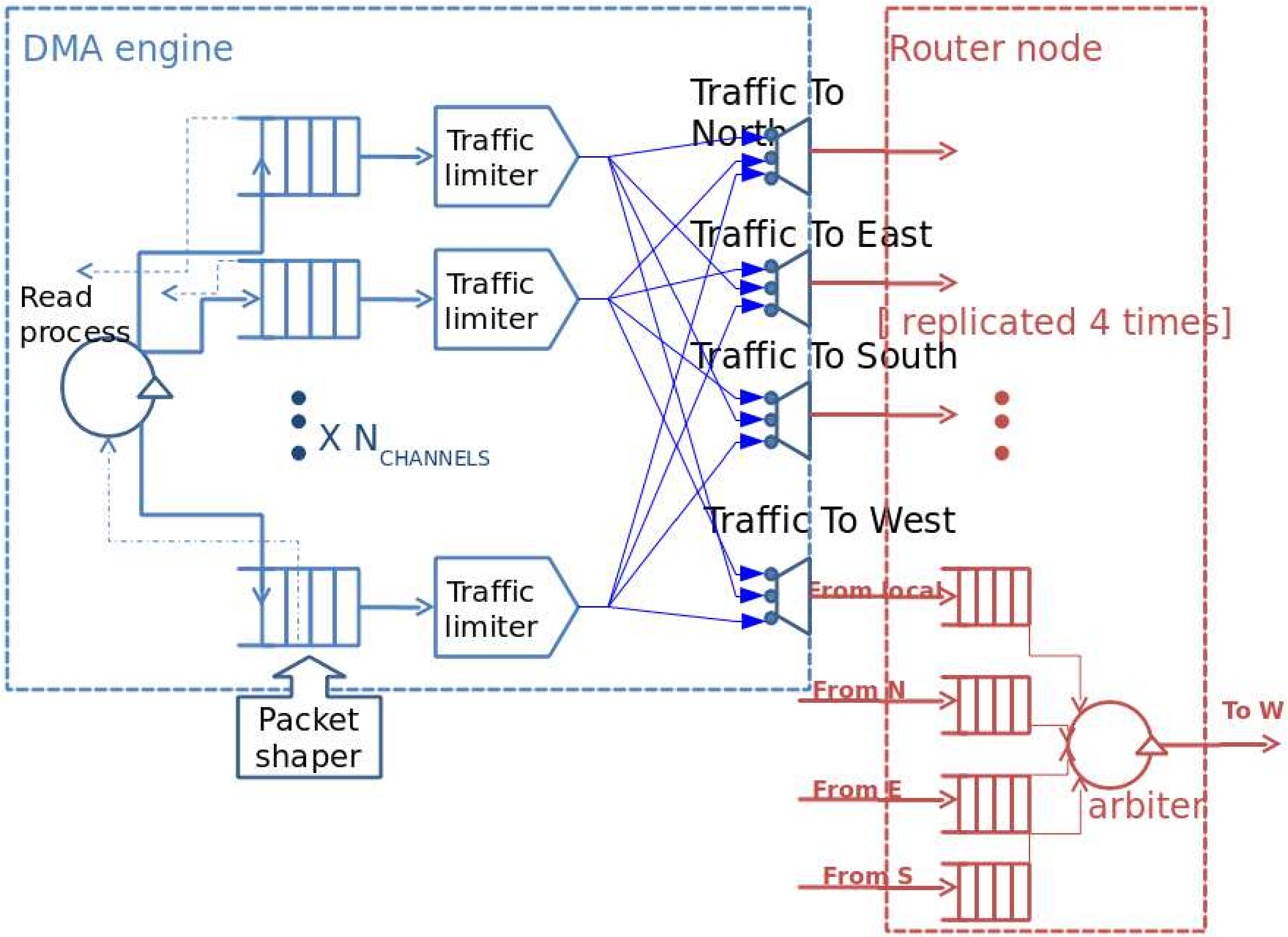
- Complex to implement
  - May be true for input queueing & output matching (e.g. iSLIP)
  - The MPPA® NoC routers only include demultiplexers, output queues and RR arbiters
- Prone to deadlocking
  - In this example, the red flow cannot use R3→R2 because the blue flow is using it
  - Likewise, the blue flow needs R1→R4 held by the red flow
  - Deadlock requires full queues



MPPA® NoC Router

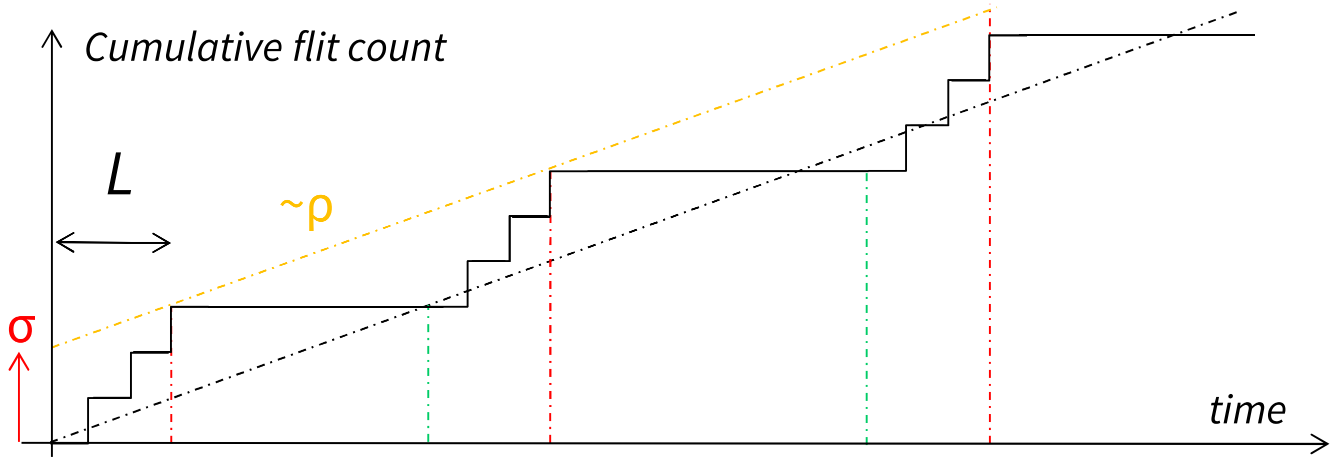


MPPA®-256 Data NoC Tx



Design of the MPPA® NoC Guaranted Services

- Data NoC packet injection implements a (σ,ρ)regulation
  - No more than σ+ρ(*t*-*s*) flits are injected for any interval [*s*,*t*]



- Application of Network Calculus prevents NoC congestion and provides bounds on end-to-end delays
- Determining routes and solving the Network Calculus equations by (integer) linear programming is effective

Outline

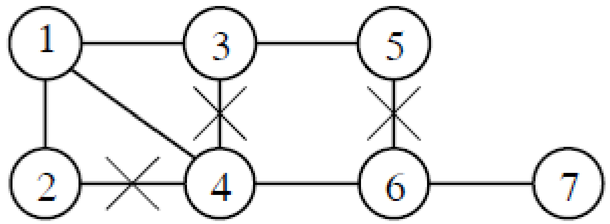
- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

Views of the MPPA® NoC Guaranted Services

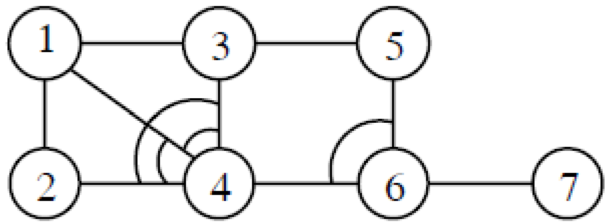
- Initial view
  - Selecting the (σ, ρ) packet injection parameters through Network Calculus prevents router queue filling so deadlocking is avoided
  - The end nodes have the capacity to accept full NoC Rx bandwidth
- Corrected view
  - Some nodes (e.g. DDR & I/O interfaces) may not accept full Rx traffic
    - Need 'entry-to-exit flow control' => use the C-NoC to carry credits
  - Network Calculus key results only apply to feed-forward networks
    - A network is feed-forward if it is possible to find a numbering of its links such that for any flow through the network, the numbering of its traversed links is an increasing sequence
    - The directed graph G= (link→node, turn→arc) must be cycle-free

Ensuring the Feed-Forward Property

- Spanning tree routing
  - Construct a spanning tree of the network graph and prohibit use of links outside the spanning tree



- Up-Down routing
  - Construct a spanning tree of the network graph, order nodes according to their tree level, and prohibit turns (a,b,c) such that a < b and c < b



- Turn prohibition [Starobinsky et al. 2003]
  - Recursively break all the link cycles and preserve global connectivity
- Work on the network graph and assume bi-directional links

Deadlock-free Message Routing

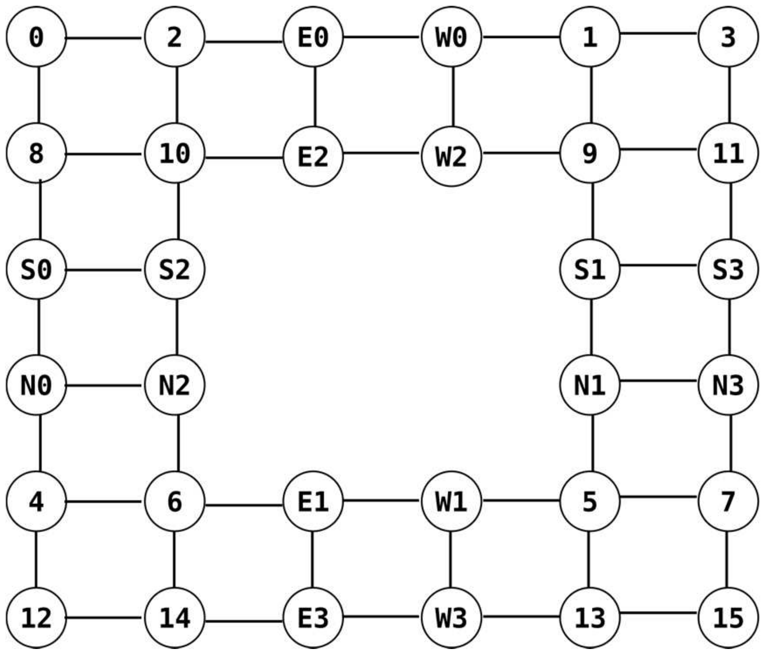
- Deadlock results from circuits of **agents** and **resources** connected by a **wait-for** relation [Dally & Seitz 1998]
  - Circuit switching: agents are connections; resources are channels
  - Wormhole switching: agents are packets; resources are link buffers
- Resource dependence graph
  - Whenever an agent is holding resource R_i while waiting for resource R_j, a dependence between R_i and R_j exists
  - Deadlock can be avoided by eliminating circuits in dependence graph
- Deadlock-free packet switching
  - Restrict routing to remove enough dependences from the graph
  - There must be a numbering of the links such that each allowed route traverses increasingly numbered links

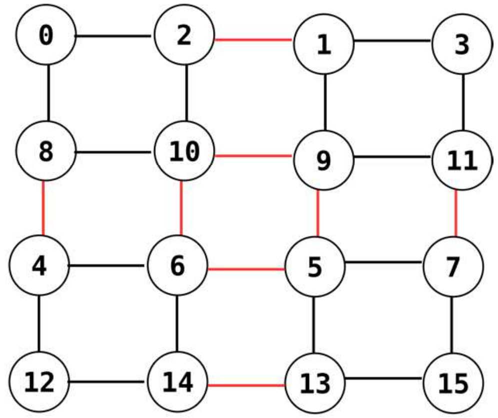
Deadlock-Free Routing Wormhole Switching

- Deadlock-free routing implies feed-forward networks
  - Wormhole switching resources are the link flit buffers
    - Links between routers **and** links internal to routers (‘turns’)
  - The (link, turn) graph considered for feed-forward networks is the vertex contraction of this resource dependence graph
- Special cases when the network topology is a 2D mesh
  - Dimension order (X-Y on 2D meshes)
  - Turn model [Glass & Ni 1994] (not the same as ‘turn prohibition’)
  - Odd-Even [Chiu 2000], H. Odd-Even [Bahrebar & Stroobandt 2015]
- Strategy for the MPPA® NoC
  - Isolate a 2D mesh in topology and applies deadlock-free routing
  - Resulting flows are feed-forward so Network Calculus applies

2D Mesh Topology on the MPPA® NoC

- The NoC nodes in the I/O clusters can be abstracted away





- The NoC can be partitioned in two or four along the I/O links
  - The NoC has lockout bits that disable links until next reset

Outline

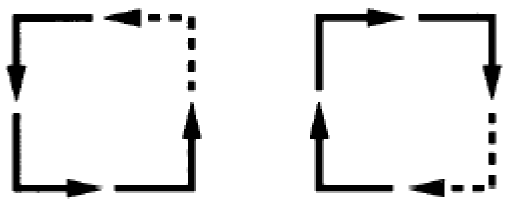
- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

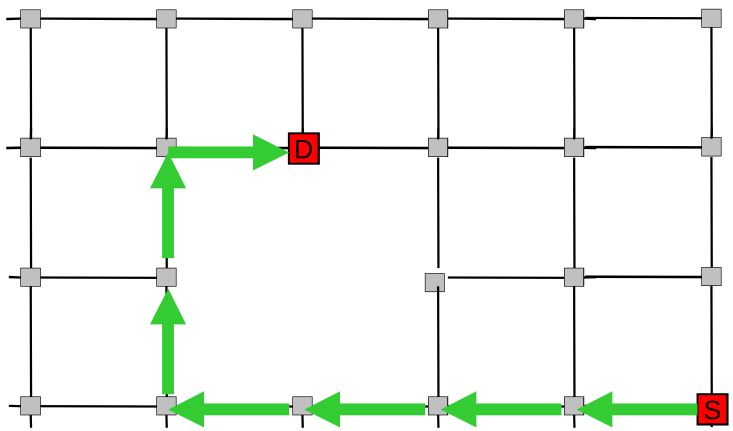
Turn Model for Adaptive Routing

- Principle [Glass & Ni 1994]
  - Analyze directions in which packets can turn in the network
  - Determine the cycles that such turns can form
  - Prohibit just enough turns to break all cycle
- n-dimensional meshes
  - Prohibits n(n-1) 90 degree turns to prevent deadlock
  - One half of all possible 180 degree turns must be prohibited
    - All-but-one-negative-first (West-First)
    - All-but-one-positive-last (North-Last)
    - Negative-First
- k-ary n-cubes
  - Allows to use the wraparound channels

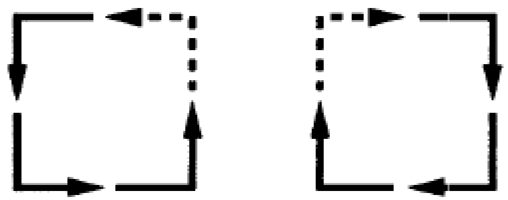
2-D Mesh Turn Models

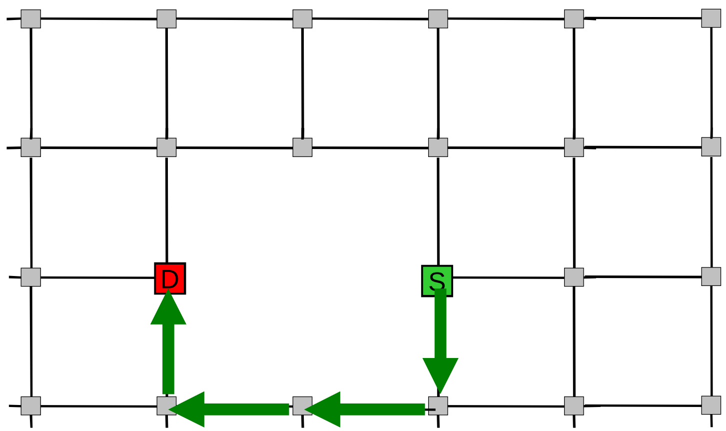
- West First
  - No North-West turn
  - No South-West turn





- North Last
  - No North-West turn
  - No North-East turn



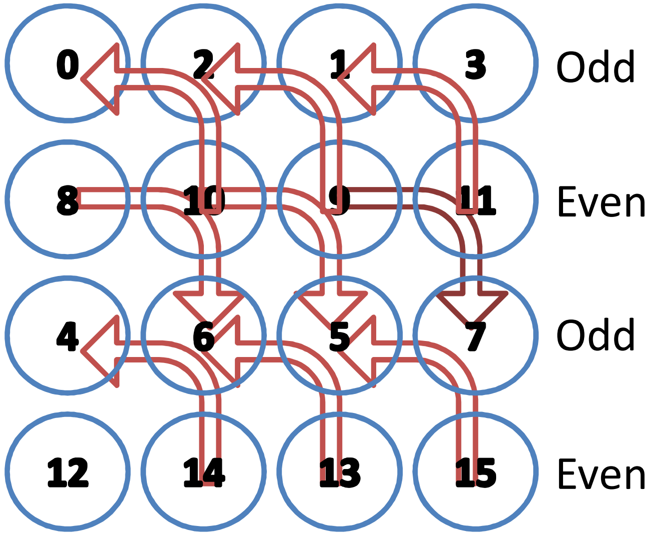


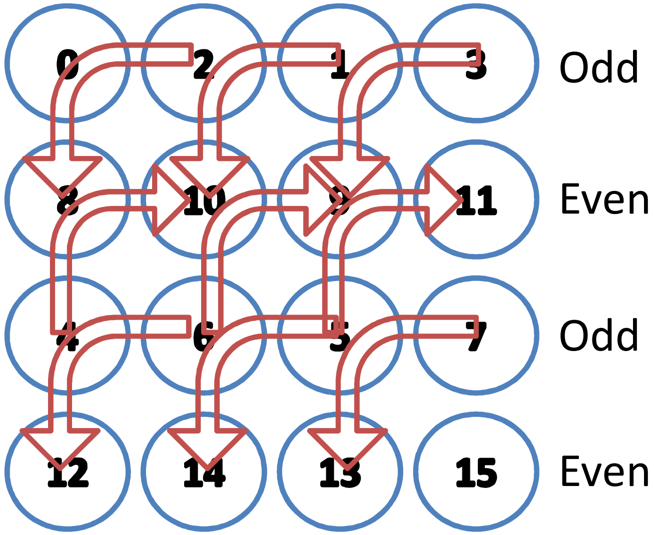
Odd–Even Routing

- The adaptiveness of the Glass & Ni turn model is uneven
  - At least half of the source-destination pairs are restricted to having only one minimal path [Chiu 2000]
- The Odd-Even turn model [Chiu 2000] is fully adaptive
  - Even columns: East-North and East-South turns are prohibited
  - Odd columns: North-West and South-West turns are prohibited
  - 180-degree turns are prohibited
- Hamiltonian-based Odd–Even [Bahrebar & Stroobandt 2015]
  - Designed to be compatible with the Multi-Path (MP) and the Column-Path (CP) routing algorithms for path-based multicast
  - Considers Odd/Even rows instead of Odd/Even columns
  - 180-degree turns are prohibited

Hamiltonian Odd-Even Prohibited Turns

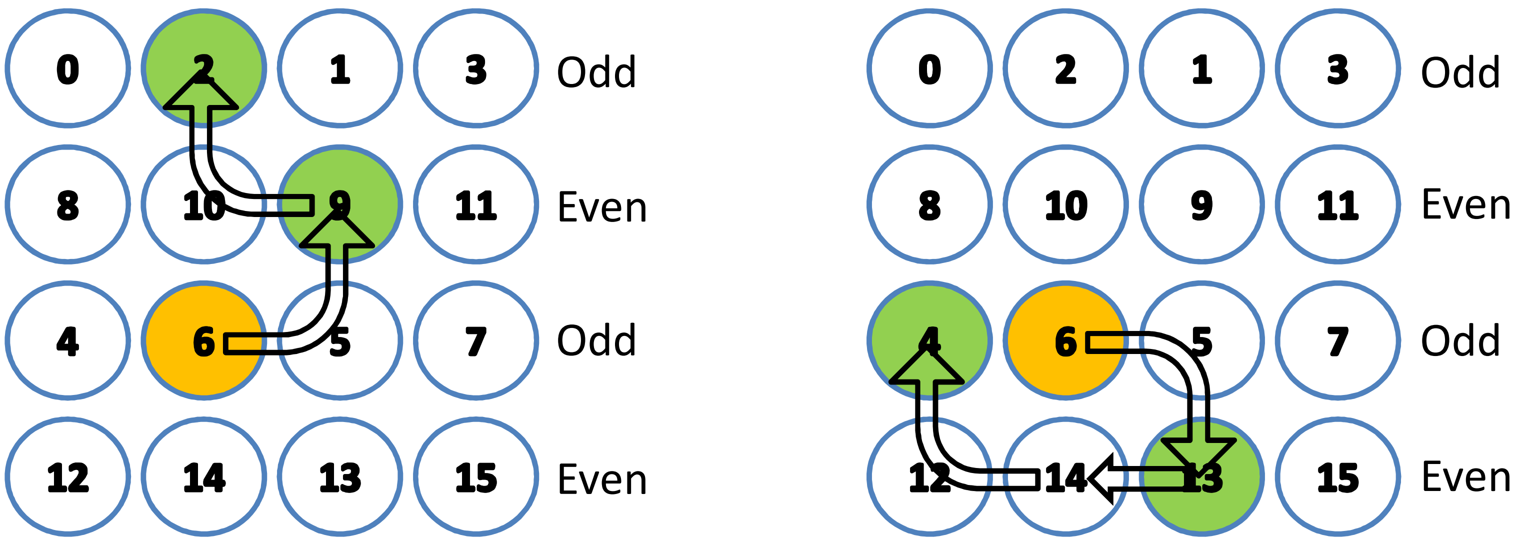
- Even rows
  - East-South turn prohibited
  - North-West turn prohibited
- Odd rows
  - North-East turn prohibited
  - West-South turn prohibited





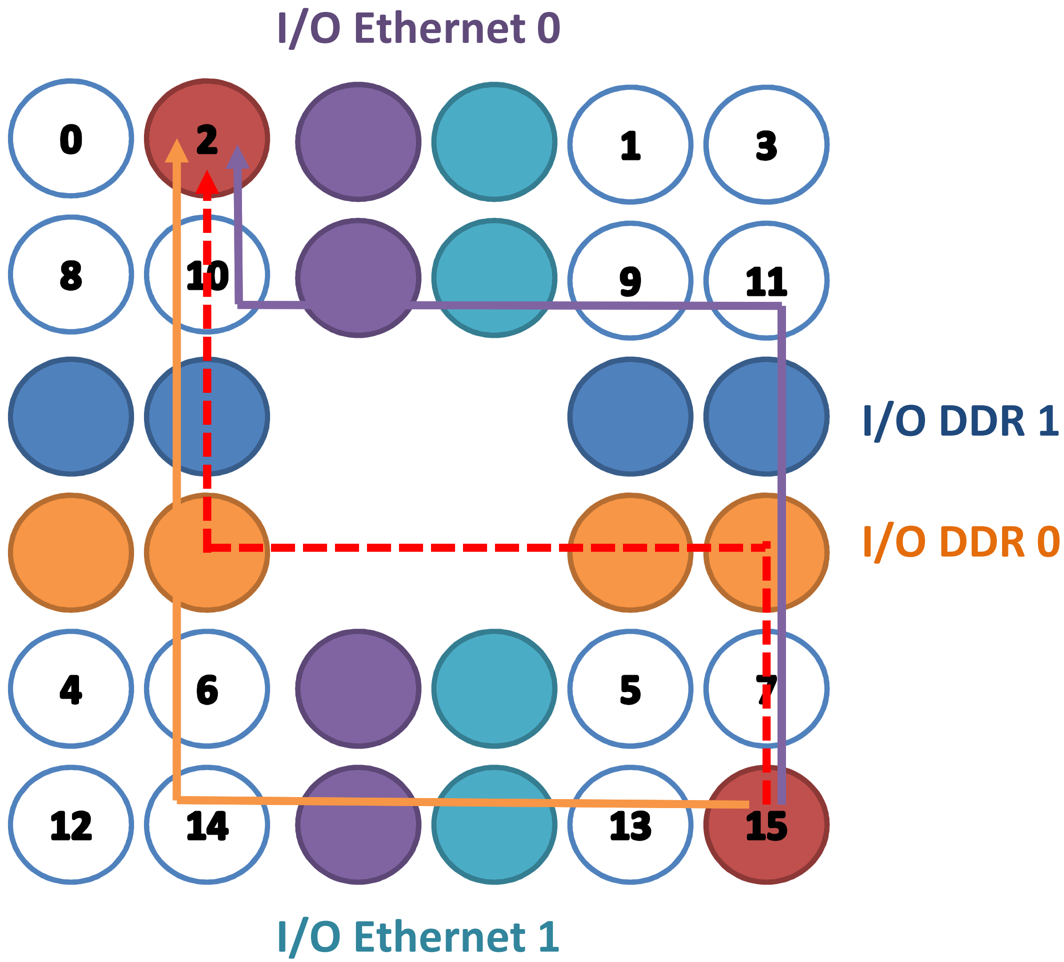
Hamiltonian Odd-Even Path-Based Multicast

- Path-based multicast: use a series of paths
  - Example for node 6 to nodes 2, 4, 9, 13
    - First path: node 6 to nodes 9 and 2
    - Second path: node 6 to nodes 13 and 4



Hamiltonian Odd-Even on the MPPA® NoC

- Routing between compute clusters
  - Routes generated assuming a 6x6 mesh
  - Impossible routes are discarded
- Routing between I/O and compute cluster
  - Always possible, thanks to the 4 NoC nodes per I/O cluster

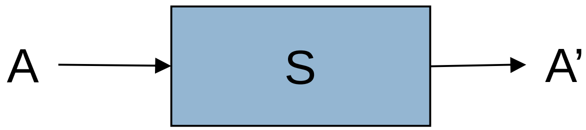


Outline

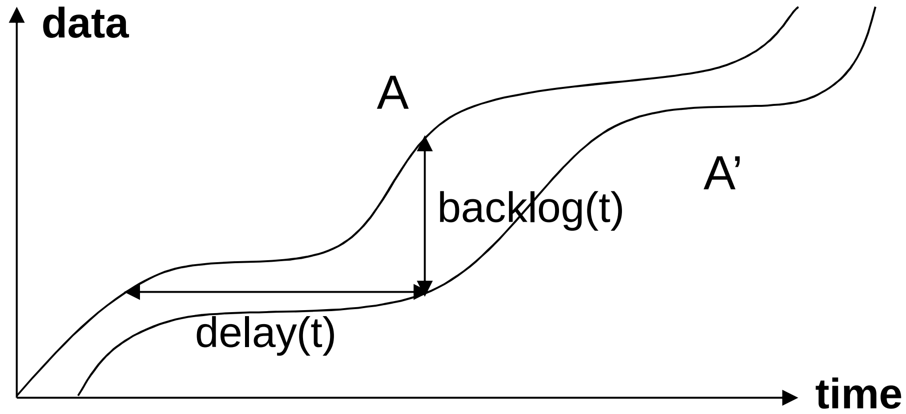
- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

Network Calculus

- Compute deterministic upper/lower bounds in communication networks
- Flows are represented by cumulative data transferred up to time t
- Servers are abstracted as relations between input and output flows



$A \geq A'$



- Framework based on (min,+) dioid instead of (+,*) ring or field

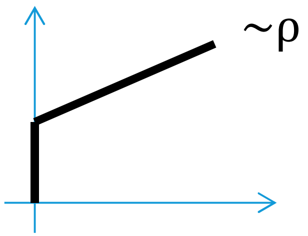
$(f \otimes g)(t) = inf_{0 \leq s \leq t}(f(t-s) + g(s))$ convolution

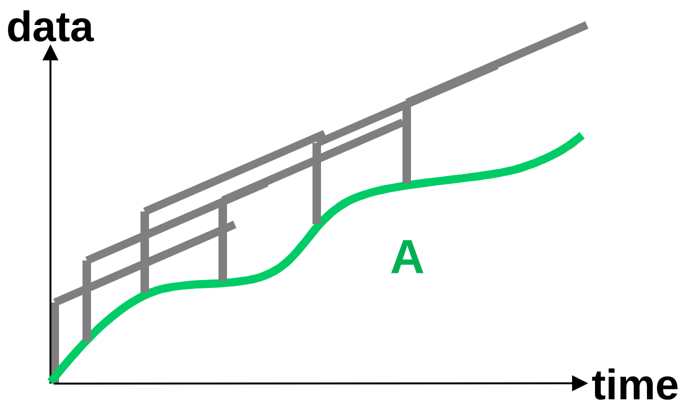
$(f \oslash g)(t) = sup_{s>0}(f(t+s) - g(s))$ deconvolution

$f \oslash g \leq h \Leftrightarrow f \leq h \otimes g$

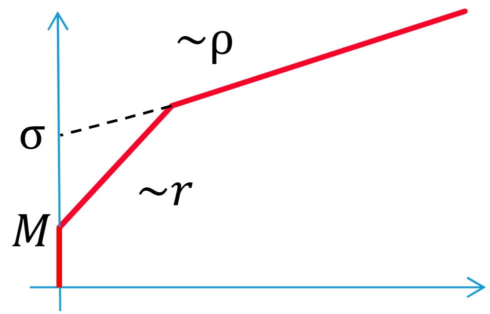
Arrival Curves

- An arrival curve α(t) is a traffic contract on a flow A(t):
  - $\forall t, d \geq 0, A(t+d) - A(t) \leq \alpha(d)$ equivalent to $A \leq A \otimes \alpha$
  - Leaky-bucket arrival curve:
    $\alpha(t) = (\sigma + \rho t)_{1_{t>0}}$



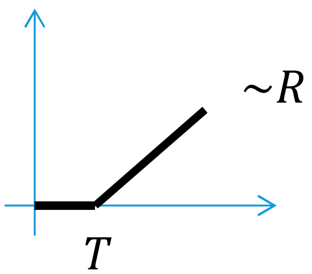


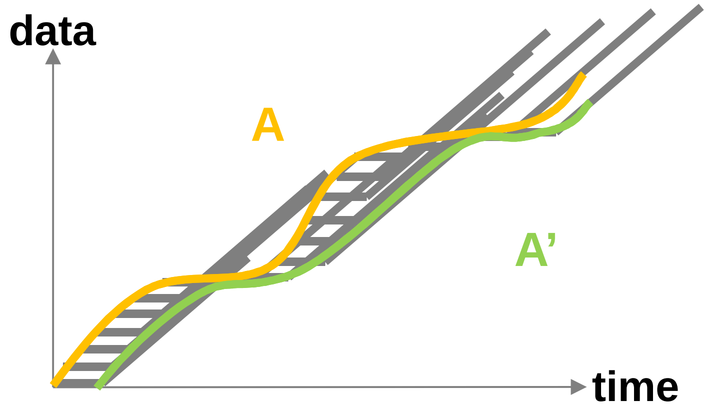
  - TSPEC arrival curve:
    $\alpha(t) = \min(M + rt, \sigma + \rho t)_{1_{t>0}}$



Service Curves

- A server has a lower service curve β(t) iff for any input A(t):
  - Output flow A'(t) satisfies $A' \geq A \otimes \beta$ and $\beta(0) = 0$
  - Rate-latency service curve:
    $\beta(t) = R\,[t - T]_+$

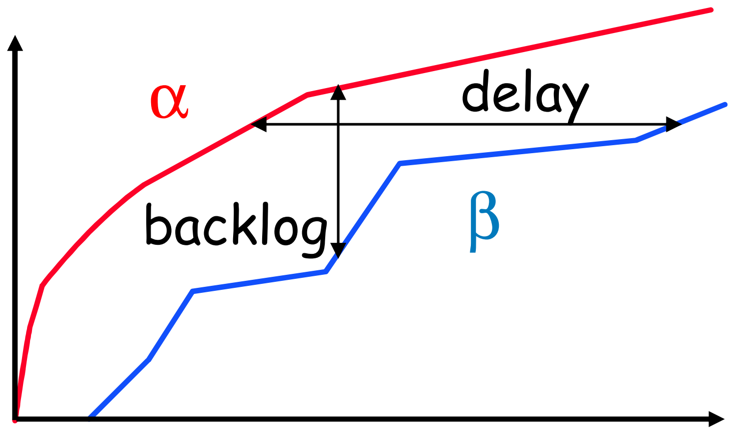




- A server has a strict service curve β(t) iff for any input A(t):
  - For any period (s, t] during which the flow is backlogged
    $$A'(t) - A'(s) \geq \beta(t - s)$$

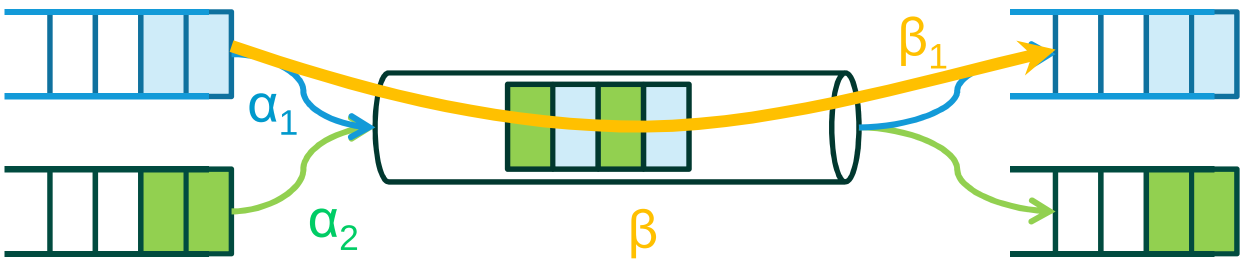
Main Rules

- Constraint propagation rule
  - A flow A(t) with arrival curve $\alpha(t)$ that traverses a server with service curve $\beta(t)$ results in a flow A'(t) constrained by arrival curve $\alpha \oslash \beta(t)$
- Tandem composition rule
  - The service curve of a tandem of two of servers with respective service curves $\beta_1(t)$ and $\beta_2(t)$ is the convolution $\beta_1 \otimes \beta_2(t)$
- Tight delay and backlog bounds
  - If flow has arrival curve $\alpha(t)$ and node offers service curve $\beta(t)$:
    - backlog = $\max_{t\geq 0} (\alpha(t) - \beta(t))$
    - delay = $h(\alpha, \beta) = \max_{t\geq 0} \{ \inf\ s\geq 0 : \alpha(t) \leq \beta(t+s) \}$

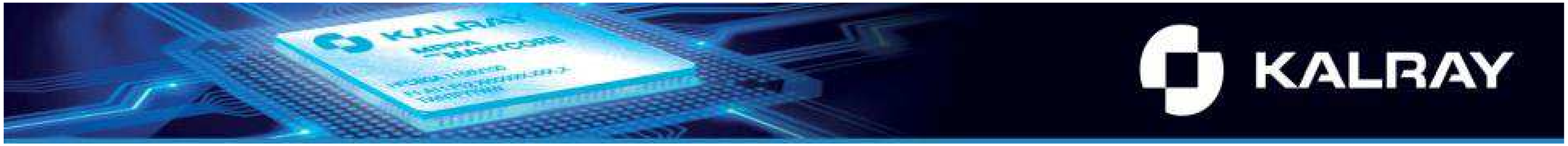


Flow Aggregation

- Blind multiplexing (flows served in arbitrary order)
  - Assume a node serving the aggregate of two flows with the **strict** service curve β(t); assume flow 2 is α_2-smooth
  - Then a service curve for flow 1 is $\beta_1(t) = [\beta(t) - \alpha_2(t)]^+$

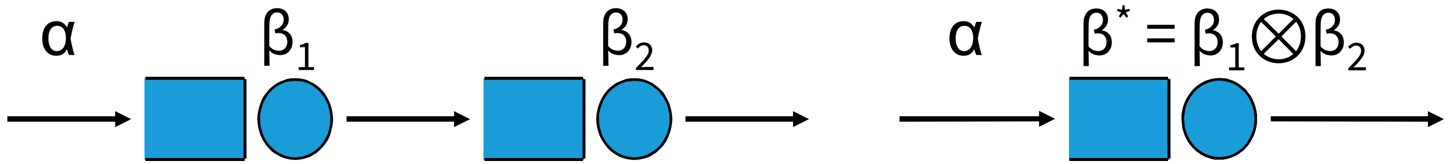


- FIFO multiplexing (flows buffered in the same queue)
  - Assume a node serving the aggregate of two flows in FIFO order with the **lower** service curve β(t); assume flow 2 is α_2-smooth; define the β_θ^1 family as $\beta_\theta^1(t) = [\beta(t) - \alpha_2(t - \theta)]^+ 1_{\{t>\theta\}}$
  - For θ≥ 0, if β_θ^1 is wide-sense increasing, it is a service curve for flow 1



Computation of the End-to-End Delay

- Without aggregation: use tandem composition (PBOO)
  - Delay = h(α, β*) α the arrival and β* the convolution of service curves



- With aggregation [Bouillard & Stea 2015]:
  - Separated-Flow Analysis (SFA)
    - First compute the equivalent service curves for tagged flow
    - Then compute the convolution of the curves thus obtained
  - Pay Multiplexing Only Once (PMOO)
    - First compute the convolution of the service curves
    - Then compute the equivalent service for tagged flow
  - Neither method is tight or best, however the SFA is more generic

Outline

- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

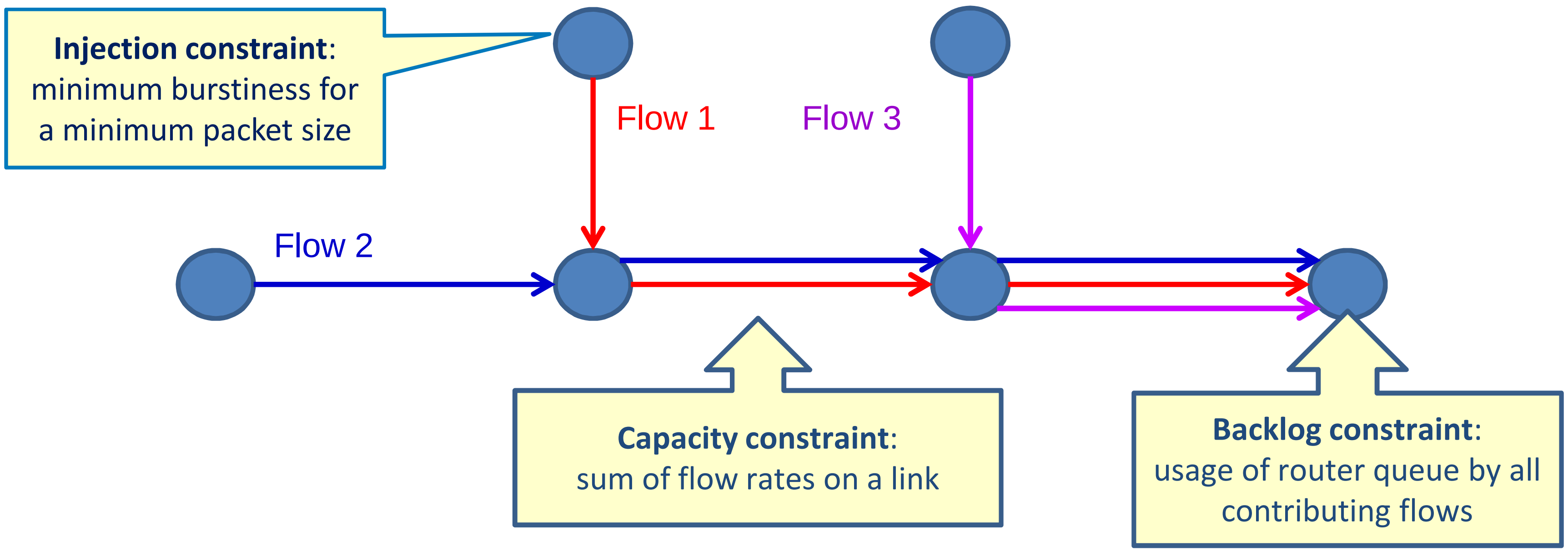
MPPA® NoC Services Objectives (I)

- For each flow, select a single path among those proposed by adaptive routing so as to maximize a network utility function
  - This is a multi-commodity flow problem that can be solved using linear programming
  - Adding the single path constraints makes the problem NP-hard but practical instances are solved with mixed integer programming
  - Utility function is the proportional fairness of flow rates [Kelly 1998]
    - Optimal rates $\Gamma^* = \{\rho^*_1, .. \rho^*_n\}$ such that for any solution Γ, $\sum_i \frac{\rho_i - \rho^*_i}{\rho^*_i} \leq 0$
  - Numerical results for all pairs of flows between 8 clusters (55 flows)

Flow	Route	Bandwidth
F48	R1	0.441901304029376
F**49**	R**1**	**0**
	R**2**	**0.375611423014079**
	R**3**	**0**
	R**4**	**0**
F50	R1	0.13862940041161
F51	R1	0.23889251894024
	R2	0

MPPA® NoC Services Objectives (II)

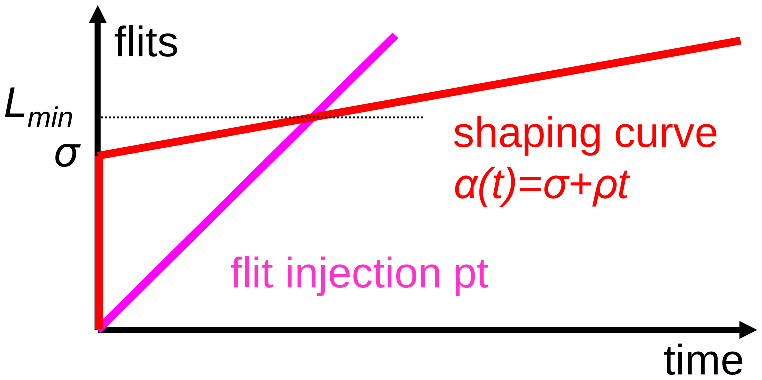
- Compute the D-NoC (σ,ρ) injection parameters along the path obtained from the single-path routing problem
  - Assume a single maximum packet size L_{max} for all interfering flows

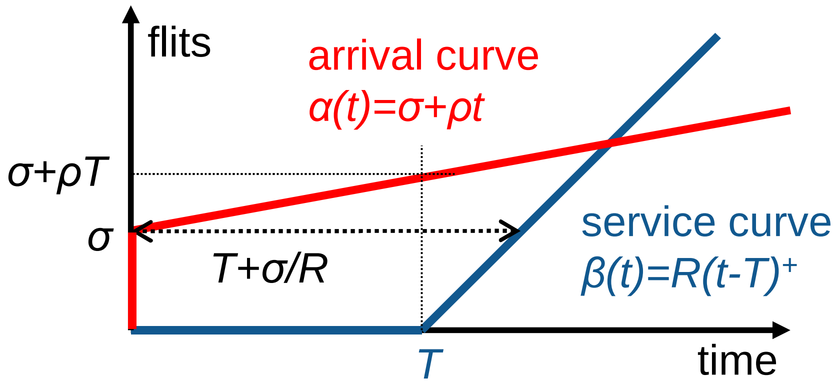


Linear Programming Formulation (a)

- Link capacity constraints
  - For each link traversed by a set of flows { (σ_i, ρ_i) } : $\sum_i \rho_i \leq R = 1$
- Queue backlog constraints
  - For each queue buffering flows { (σ_i, ρ_i) } : $\sum_i (\sigma_i + \rho_i d_L) \leq Q_{size}$

  $T = d_L = (n_L - 1)L_{max}$ with n_L the count of active queues for link L
- Packet injection constraints
  - $\sigma_i \geq L_{min}(p - \rho_i)/p$ with $p = 1$ injection rate, L_{min} min packet size



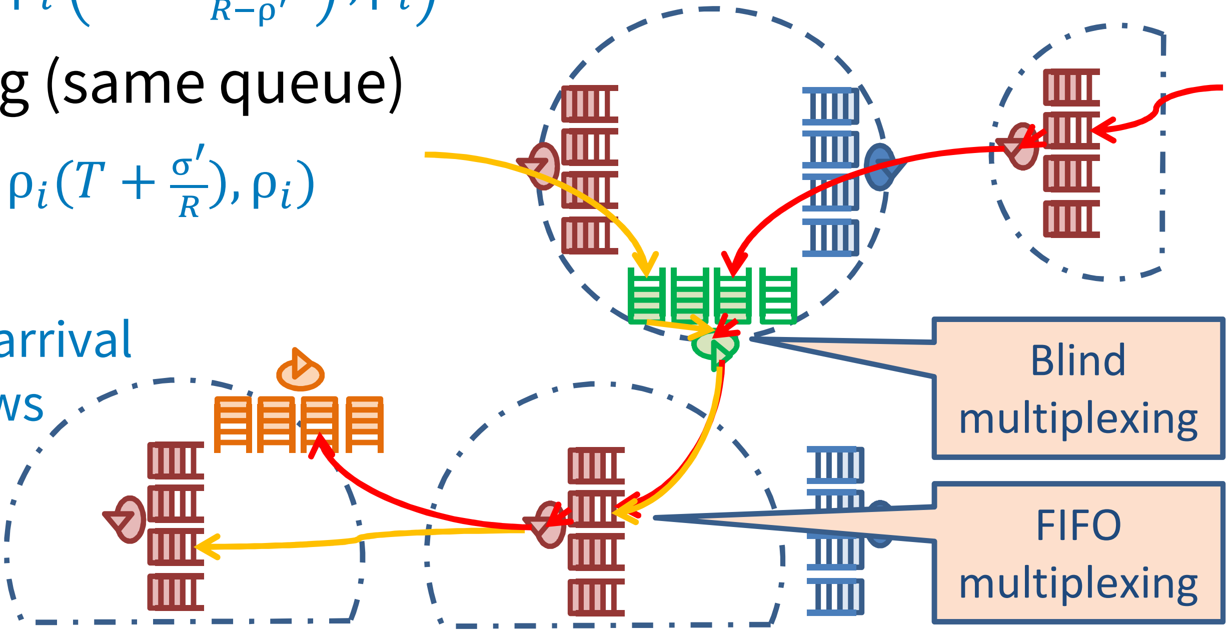


KALRAY

Linear Programming Formulation (b)

- Link arbiter service curves
  - Approximated by latency-rate $\beta(t) = R[t - T]_{+}$ with $R = 1$ and $T = d_L$
- Blind multiplexing (different queues)
  - $(\sigma_i, \rho_i) \rightarrow \left(\sigma_i + \rho_i\left(T + \frac{\sigma' + \rho' T}{R - \rho'}\right), \rho_i\right)$
- FIFO multiplexing (same queue)
  - $(\sigma_i, \rho_i) \rightarrow (\sigma_i + \rho_i(T + \frac{\sigma'}{R}), \rho_i)$

(σ', ρ') the sum of arrival curves of other flows in the link arbiter



MPPA® NoC Services Objectives (III)

- Compute upper bounds on flow end-to-end delays
  - Upper bound = $h(\alpha_i, \beta^*{}_i)$ with $\alpha_i(t) = (\sigma_i + \rho_i t)_{1_{t>0}}$ the flow i shaping curve at injection and $\beta^*{}_i$ the convolution of the left-over service curves $\beta'{}_i$ for this flow in the link arbiters along path
- Link arbiter left-over service curves
  - Let (σ’, ρ’) be the sum of arrival curves of interfering flows in arbiter
  - Case of FIFO multiplexing: $\beta'(t) = (R - \rho')[t - T - \sigma'/R]_+$
  - Case of blind multiplexing: $\beta'(t) = [R\,[t - T]_+ - (\sigma' + \rho' t)_{1_{t>0}}]_+$
- The arrival curves of interfering flows in front of each link arbiter are obtained from the linear program
  - See slide « Linear Programming Formulation (b) »

Outline

- Introduction
- MPPA®-256 NoC
- Feed-Foward Flows
- Routing Techniques
- Network Calculus
- MPPA® NoC Services
- Conclusions

Conclusions

- The MPPA® NoC implement wormhole switching
  - Packet switching enables dynamic resource sharing
  - Wormhole switching is implemented with minimal complexity
- We address both deadlock-freedom and QoS in the D-NoC
  - Deadlock-free routing ensures feed-forward network flows
  - Hamiltonian Odd-Even routing over a 2-D mesh subset of the D-NoC
  - Solve a mixed integer program to select paths between endpoints
  - Solve a linear program to compute the D-NoC injection parameters
- Work on-going for the QoS of traffic from/to DDR
  - Assume each compute cluster works in its private DDR bank
  - Configure the DDR controller to prevent request reordering
  - Try to apply Network Calculus or Sensor Calculus model to DDR

Thank you

KALRAY S.A.
Paris - France

86 rue de Paris,
91 400 Orsay
France

Tel: +33 (0) 184 00 00 45
email: info@kalray.eu

KALRAY S.A.
Grenoble - France

445 rue Lavoisier,
38 330 Montbonnot
France

Tel: +33 (0)4 76 18 09 18
email: info@kalray.eu

KALRAY INC.
Los Altos - USA

4962 El Camino Real
Los Altos, CA
USA

Tel: +1 (650) 469 3729
email: info@kalrayinc.com