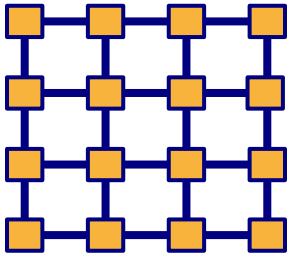


ECE 6115 / CS 8803 - ICN
Interconnection Networks for High Performance Systems
Spring 2020

SMART NETWORK-ON-CHIP

Tushar Krishna

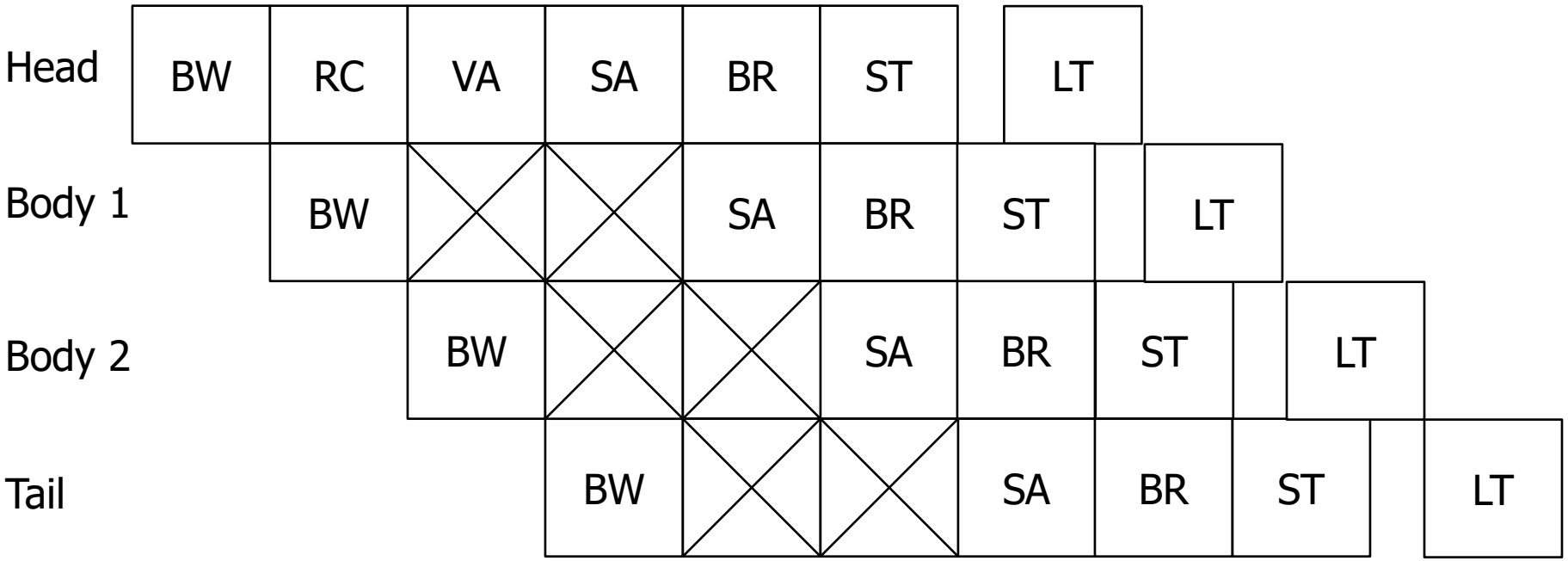
Assistant Professor

School of Electrical and Computer Engineering

Georgia Institute of Technology

tushar@ece.gatech.edu

BASELINE ROUTER PIPELINE



- **Per Packet**
  - RC, VA → done by Head flit
- **Per Flit**
  - BW, SA, BR, ST, LT

WHY DOES ROUTER DELAY MATTER?

$$T_N = (t_r + t_w) \times H + T_c + T_S$$

- T_N: Network delay
- t_r: router pipeline delay
- t_w: wire delay per hop
- H: number of hops
- T_c: contention delay
- T_S: serialization delay (for multi-flit packets)

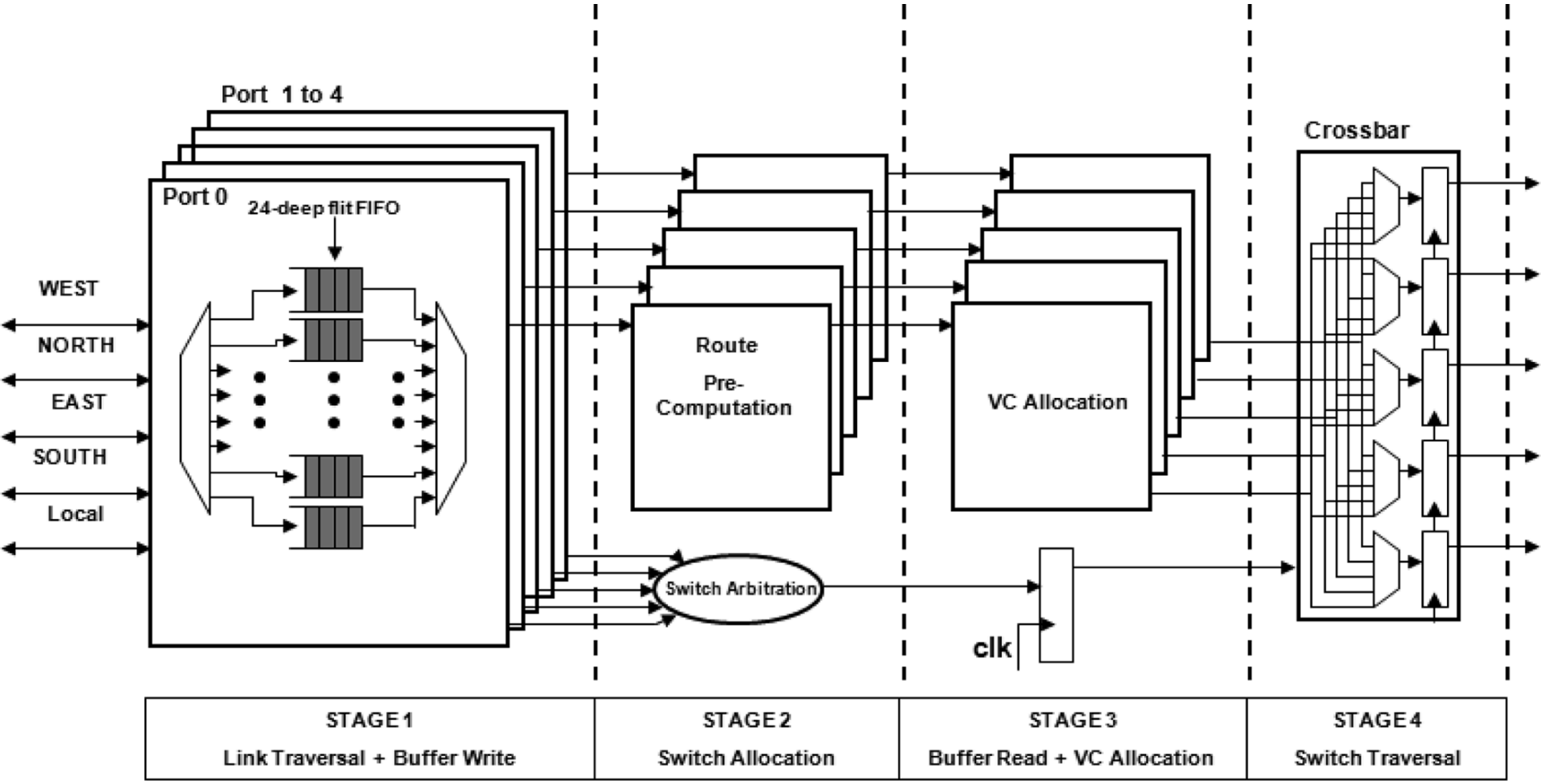
Which of these is static?

t_r t_w T_s

Which of these is dynamic (traffic-dependent)?

H T_c

CASE STUDY: INTEL SCC (ISSCC 2010)



CASE STUDY: INTEL SCC (ISSCC 2010)

The 5-port virtual cut-through router (Fig. 5.7.3) used to create the 2D-mesh network employs a credit-based flow-control protocol. Router ports are packet-switched, have 16-byte data links, and can operate at 2GHz at 1.1V. Each input port has five 24-entry queues, a route pre-computation unit, and a virtual-channel (VC) allocator. Route pre-computation for the outport of the next router is done on queued packets. An XY dimension ordered routing algorithm is strictly followed. Deadlock free routing is maintained by allocating 8 virtual channels (VCs) between 2 message classes on all outgoing packets. VC0 through VC5 are kept in a free pool, while VC6 and VC7 are reserved for request classes and response classes, respectively. Input port and output port arbitrations are done concurrently using a wrapped wave front arbiter. Crossbar switch allocation is done in a single clock cycle on a packet granularity. No-load router latency is 4 clock cycles, including link traversal. Individual routers offer 64GB/s interconnect bandwidth, enabling the total network to support 256GB/s of bisection bandwidth.

COMMON PIPELINE OPTIMIZATIONS

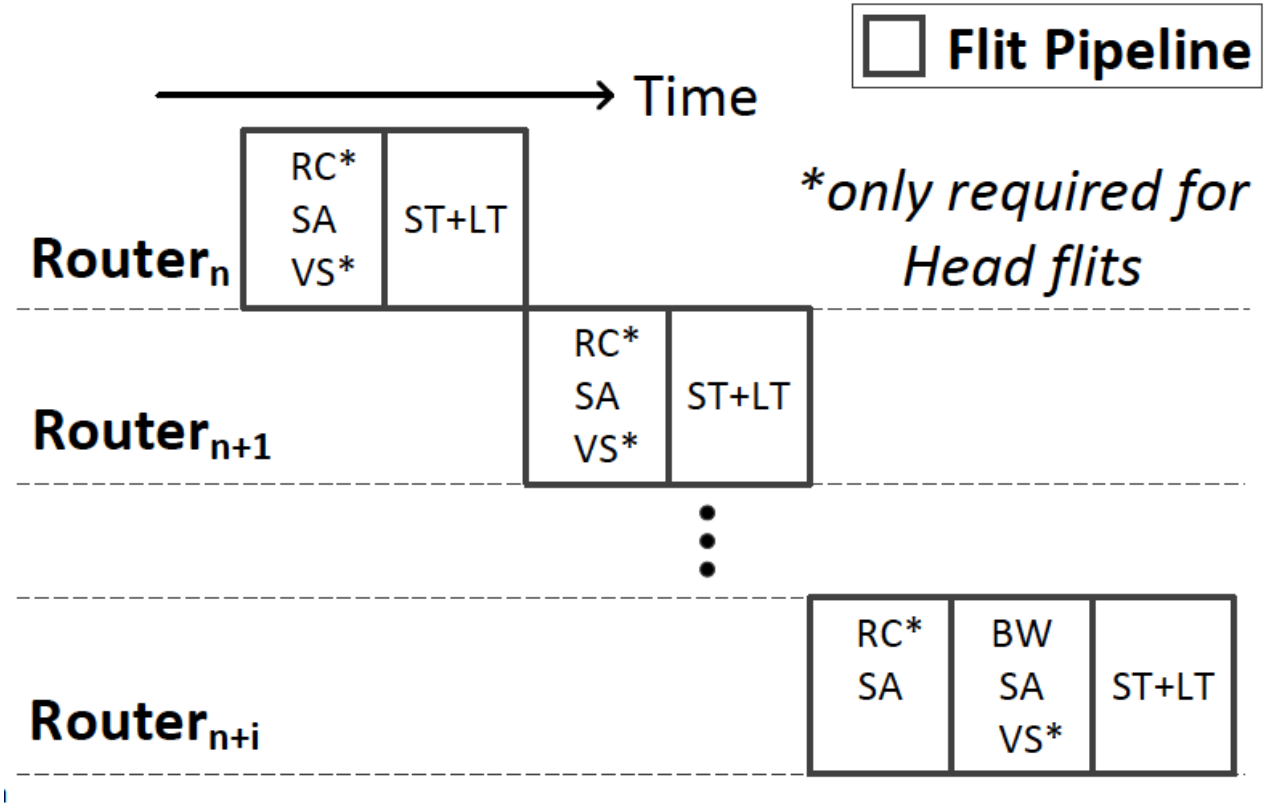
- BW + RC in parallel
  - Lookahead Routing

- SA + VA in parallel
  - VC Select (switch output port winner selects VC from pool of free VCs)
  - Speculative VA (if VA takes long, speculatively allocate a VC while flit performs SA) (Peh and Dally, HPCA 2001)
    - If SA and VA both successful, go for ST
    - If SA or VA fails, retry next cycle
- BR + SA in parallel
  - The winner of Input Arbitration is read out and sent to the input of the crossbar speculatively
- Low-load Bypassing
  - When no flits in input buffer
    - Speculatively enter ST
    - On port conflict, speculation aborted

EXPRESS VIRTUAL CHANNELS (ISCA 2007)

- Analogy – Express Trains and Local Trains
- Flits on Express VCs do not get buffered at intermediate routers
  - Send a "lookahead" to ask local flits to wait (i.e., kill switch allocation)

MODERN PIPELINES



1-cycle for arbitration (tr), 1-cycle for traversal (tw)

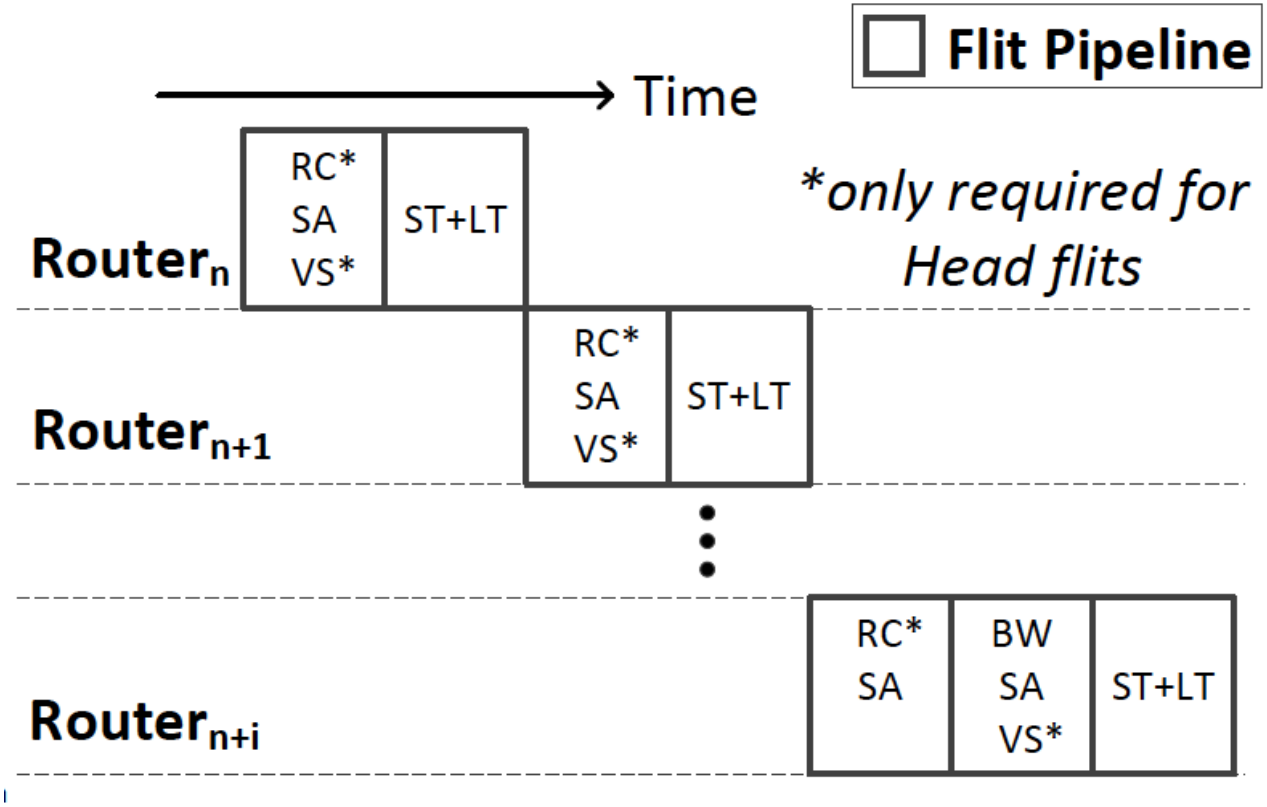
Used by Tilera's iMesh, Intel's Ring, NoC prototypes (Park et al., DAC 2012)

ALTERNATE ROUTER MICROARCHITECTURES

- Output Queues
  - "Virtual" Output Queues (== Virtual Channels)
- Centralized Buffers
- Rotary Router (in paper presentations)

SMART NOCS

MODERN PIPELINES



1-cycle for arbitration (t_r), 1-cycle for traversal (t_w)

Used by Tilera's iMesh, Intel's Ring, NoC prototypes (Park et al., DAC 2012)

IS THAT THE BEST WE CAN DO?

$$T_N = (t_r + t_w) \times H + T_c + T_S$$

1 $t_w = 1$

Latency α Hops

fundamental limitation?

Can we remove the dependence of latency on hops?

Stay tuned!

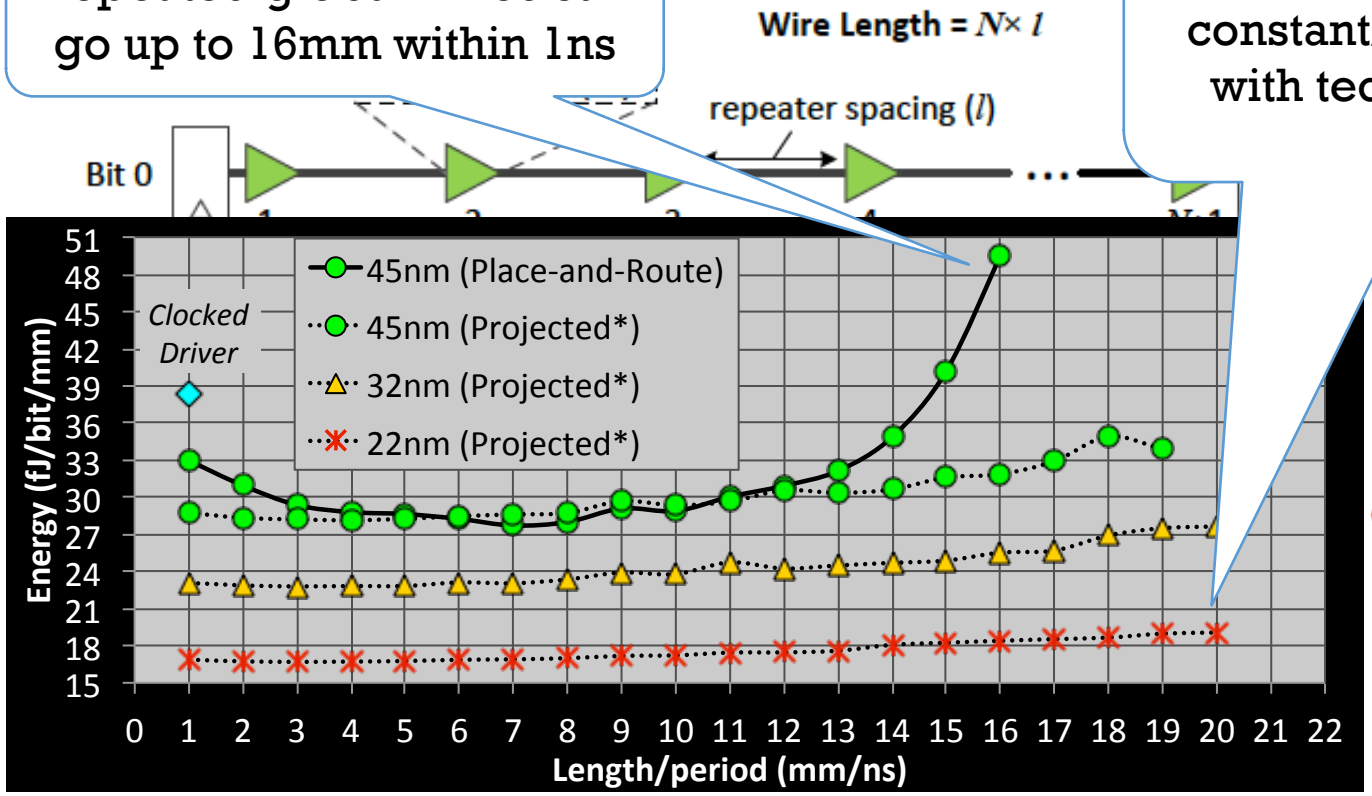
DESIGNING A 1-CYCLE NETWORK

What limits us from designing a 1-cycle network?

Is it the wire delay?

Repeated global wires can go up to 16mm within 1ns

Repeated global wire delay expected to remain constant/decrease slightly with technology scaling.



Metal Layer = M6

Repeater Spacing = 1mm

Wire Width = DRC_{min}

Wire Spacing = $3 \times DRC_{min}$ ***(coupling cap → 0)***

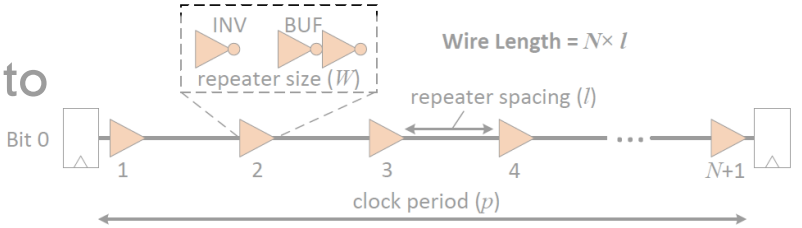
**DSENT (NOCS 2012): Timing-driven NoC Power Estimation Tool*

DESIGNING A 1-CYCLE NETWORK

What limits us from designing a 1-cycle network?

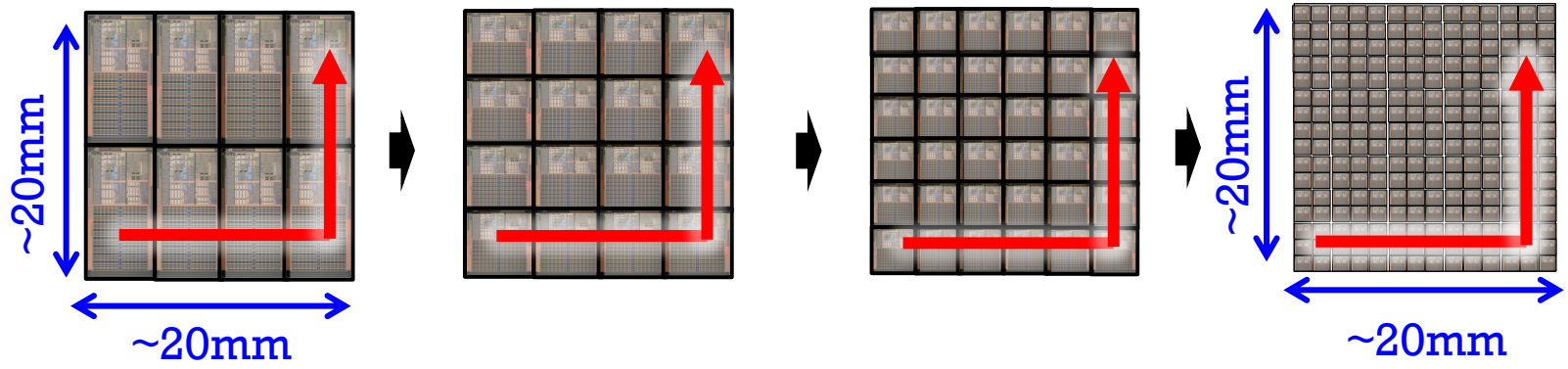
Is it the wire delay?

- Global repeated wires can transmit up to 10-16mm within 1ns (1GHz)



- Glo
  - Ch
  - Clock frequency expected to remain similar (power wall)

On-chip wires fast enough to transmit across the chip within 1-2 cycles at 1GHz even as technology scales



DESIGNING A 1-CYCLE NETWORK

What limits us from designing a 1-cycle network?

Is it the wire delay? No!

Classic scaling challenge with wires
Wire-delay increases relative to logic delay

But …

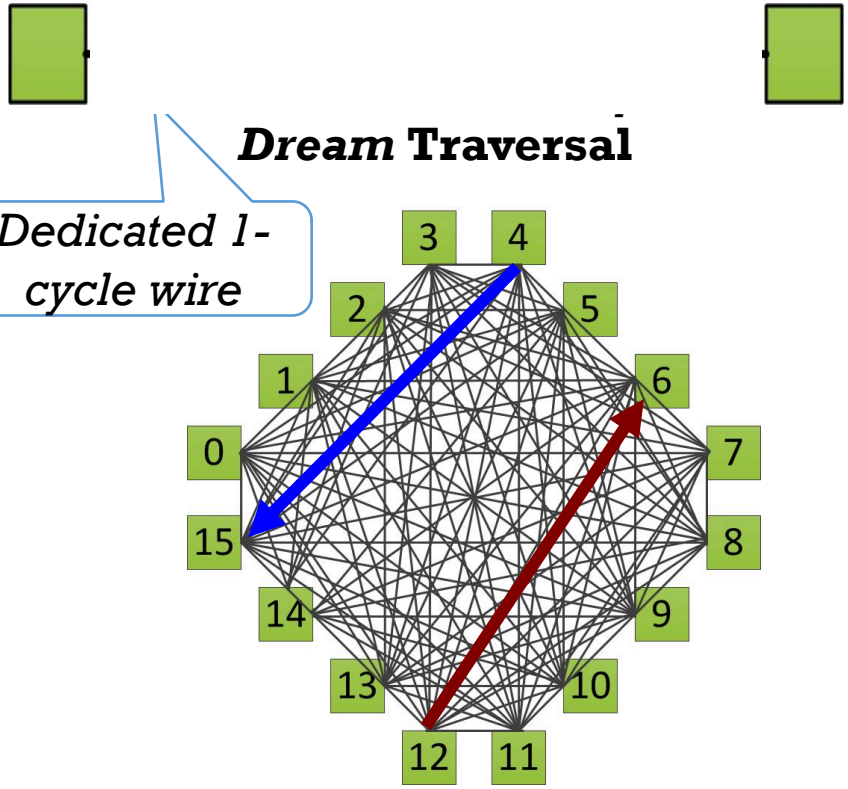
Wire-delay in *cycles* expected to remain constant.

Wires fast enough to transmit across chip in 1-2 cycles today and in future.

DESIGNING A 1-CYCLE NETWORK

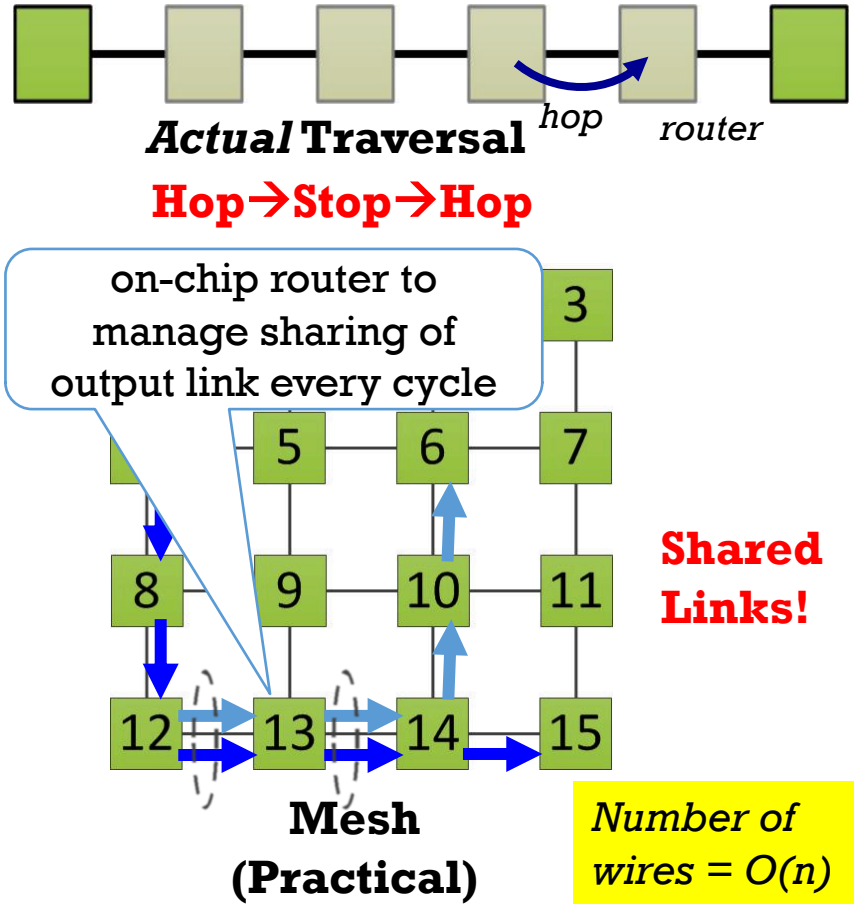
What limits us from designing a 1-cycle network?

Is it the wire delay? No!



Number of wires = $O(n^2)$

Fully-connected (Impractical)



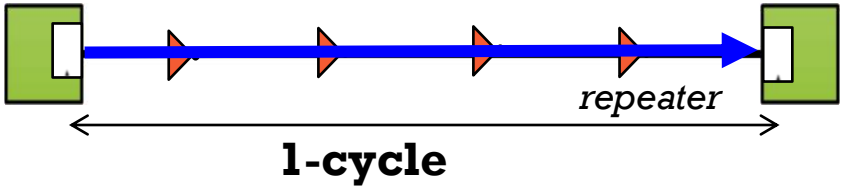
Mesh (Practical)

Number of wires = $O(n)$

DESIGNING A 1-CYCLE NETWORK

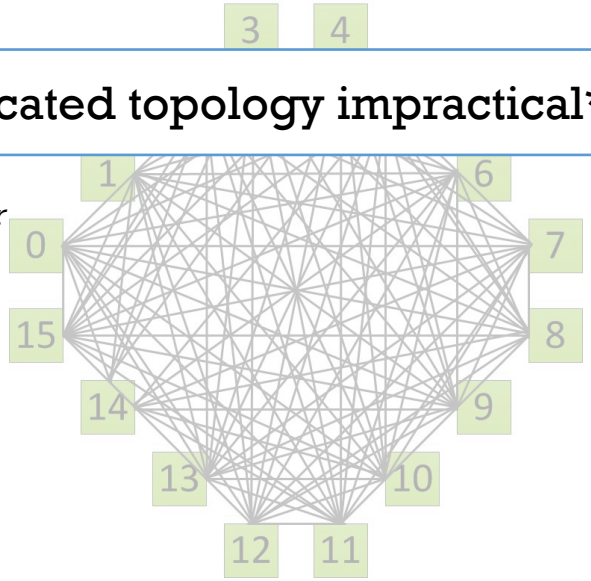
What limits us from designing a 1-cycle network?

Is it the wire delay? No!



Dedicated topology impractical*

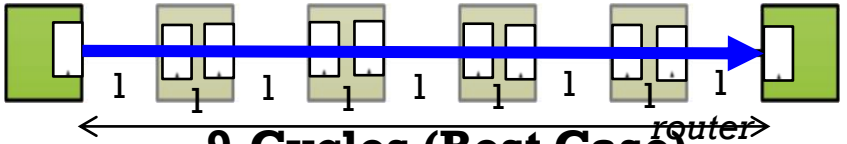
**unless we design a chip for a specific application*



Number of wires = $O(n^2)$

Fully-connected (Impractical)

Is it the routers? Yes!

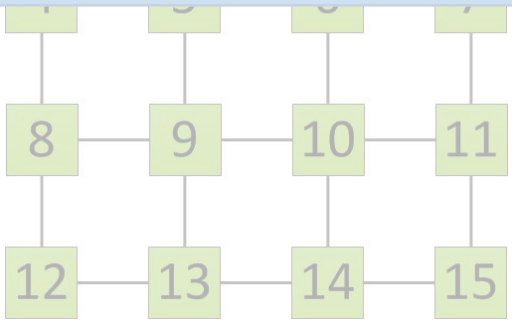


Hop → Stop → Hop

(single-cycle router, no other traffic!)

Routers required to share links

Note: this is at *every hop*



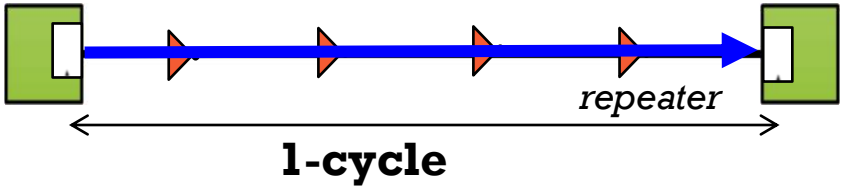
Mesh (Practical)

Number of wires = O(n)

DESIGNING A 1-CYCLE NETWORK

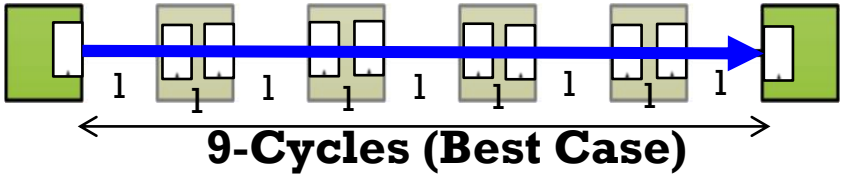
What limits us from designing a 1-cycle network?

Is it the wire delay? No!



Dedicated topology impractical*

Is it the routers? Yes!



(single-cycle router, no other traffic!)

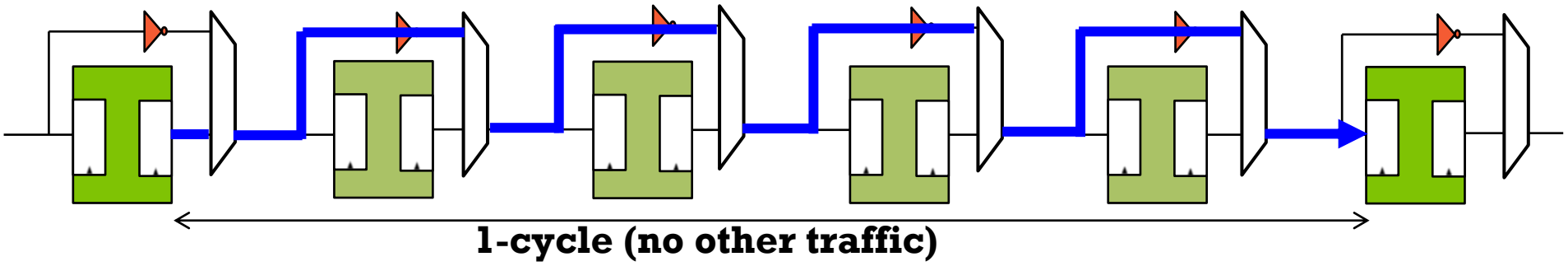
Routers required to share links

**unless we design a chip for a specific application*

Can we get both? Yes!

SMART: achieve the performance of ***dedicated*** connections over a network of ***shared*** links

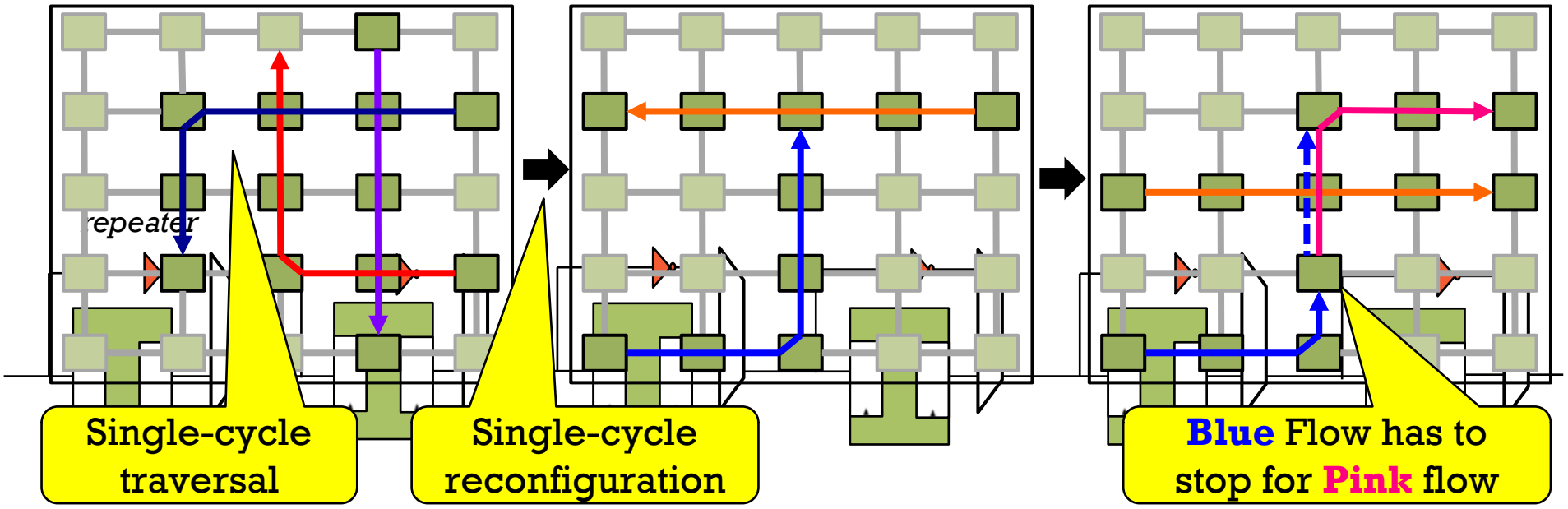
***S**ingle-cycle*
***M**ulti-hop*
***A**synchronous*
***R**epeated*
***T**raversal*



A "SMART" NETWORK-ON-CHIP

- **Microarchitecture**
  - *Bypass path* with *clockless* repeater at each router
- **Flow Control**
  - Compete for and reserve a *sequence of shared links* cycle-by-cycle

Dynamically create repeated links ("SMART paths") between any two routers



A "SMART" NETWORK-ON-CHIP

- **Microarchitecture**
  - *Bypass path* with *clockless* repeater at each router
- **Flow Control**
  - Compete for and reserve a *sequence of shared links* cycle-by-cycle

> Dynamically create repeated links ("SMART paths") between any two routers

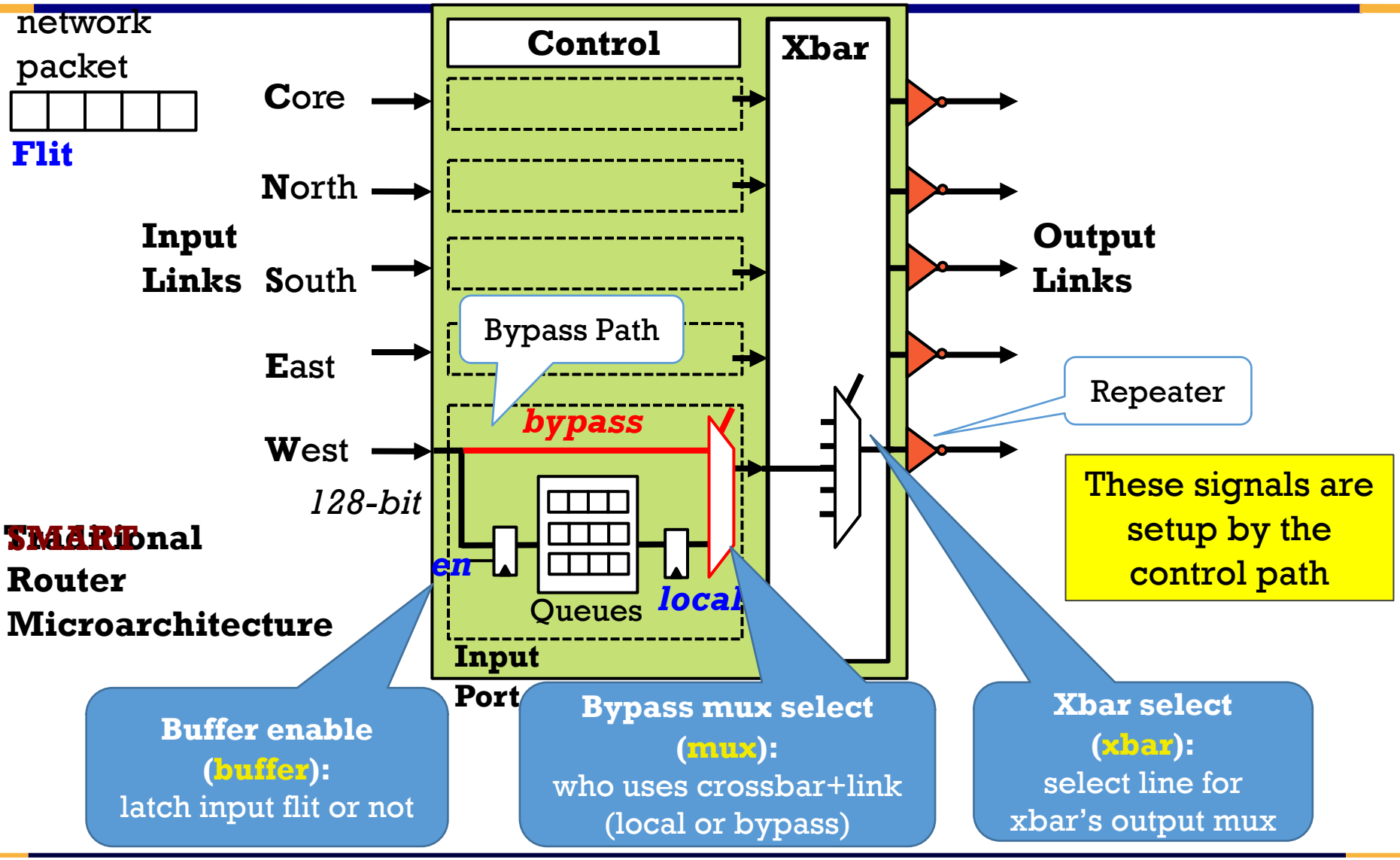
- **How well does SMART perform?**
  - 88-90% of the performance of an $O(n^2)$ wire fully-connected (dream) topology with an O(n) wire SMART NoC
  - Baseline Mesh needs to be clocked 5.4 times faster to match SMART

(64-core full-system simulation with real applications)

Microarchitecture and Flow Control details next!

T. Krishna et al.
HPCA 2013
IEEE Computer 2013
IEEE Micro Top Picks 2014
NOCS 2014 (Best Paper Award)
H Kwon et al., ISPASS 2017

MICROARCHITECTURE: DATA PATH



MICROARCHITECTURE: CONTROL PATH

Dedicated repeated links from every router to help setup a SMART path

Length = HPC_{max} hops
Width = $\log_2(1+HPC_{max})$ bits

HPC_{max} *(max Hops Per Cycle): maximum number of "hops" that the underlying wire allows the flit to traverse within a clock cycle*

e.g., HPC_{max} = 10-16 @45nm, 1GHz, 1mm hop

Let HPC_{max} = 3

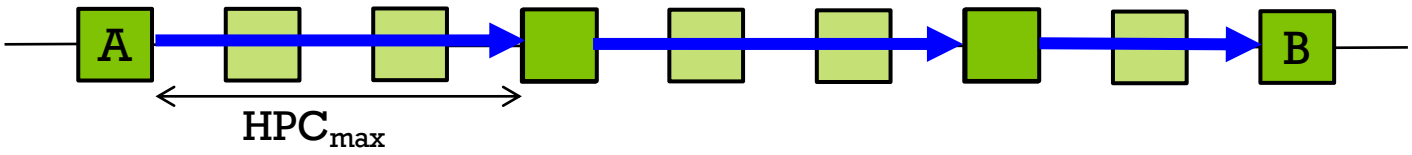
SSR for E direction

SMART FLOW CONTROL

Assume HPC_{max} = 3 (max Hops Per Cycle)

- **Request a path of *desired length (in hops)* over the SSR wires**
  - Intermediate routers arbitrate between control requests from various routers and setup *buffer*, *mux* and *xbar* for the data path
  - No ACK has to be sent back!

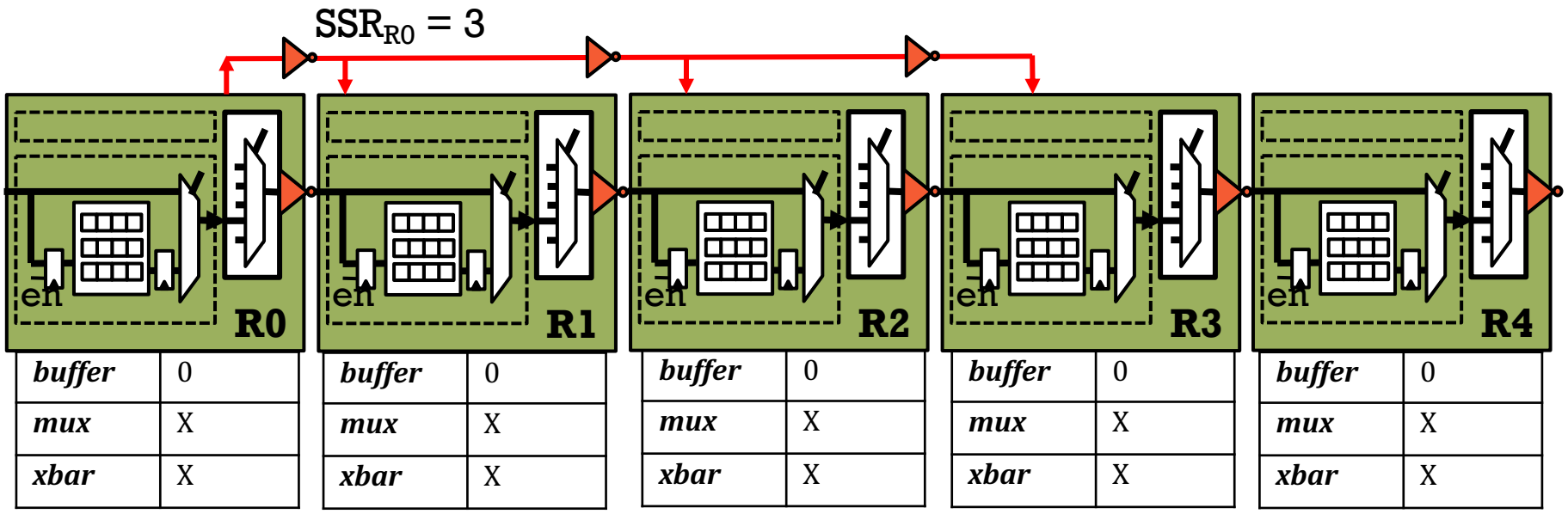
- **Send the flit on the data wires**
  - May get partial or full SMART path based on contention that cycle



TRAVERSAL EXAMPLE 1: R0 → R3

Cycle 1 (Ctrl): R0 sends SSR = 3 *(3-hop path request)*

Assume HPC_{max} = 3 (max Hops Per Cycle)



R0	
buffer	0
mux	X
xbar	X

R1	
buffer	0
mux	X
xbar	X

R2	
buffer	0
mux	X
xbar	X

R3	
buffer	0
mux	X
xbar	X

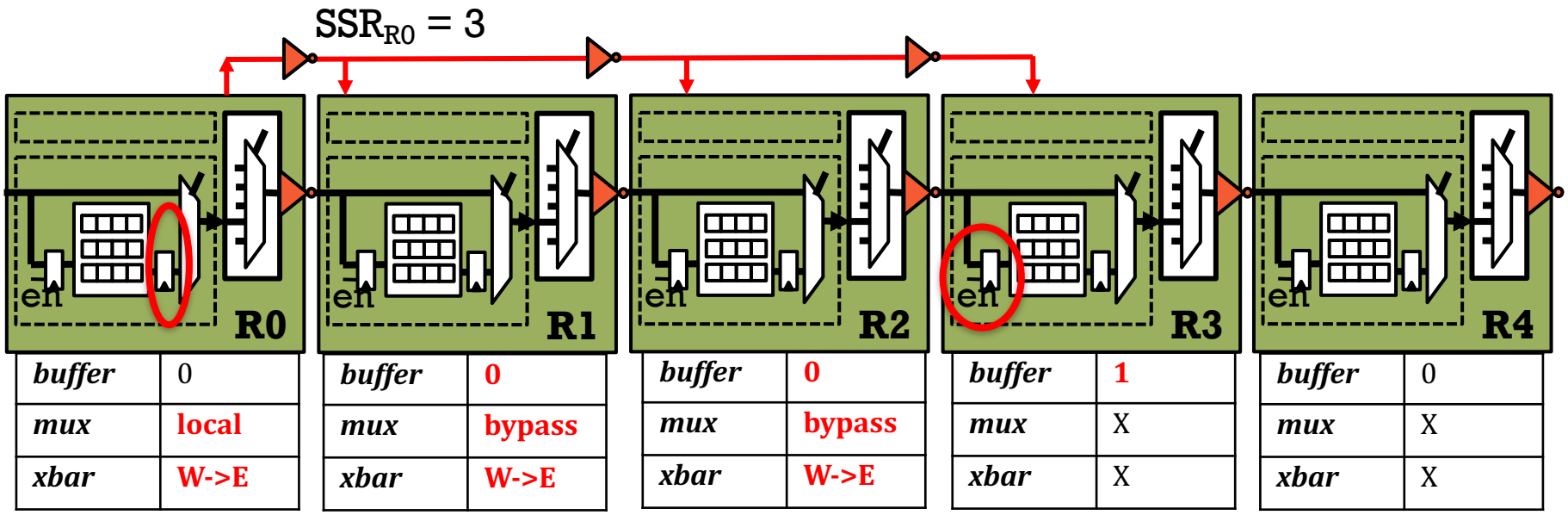
R4	
buffer	0
mux	X
xbar	X

TRAVERSAL EXAMPLE 1: R0 → R3

Cycle 1 (Ctrl): R0 sends SSR = 3 *(3-hop path request)*

All routers set *buffer*, *mux* and *xbar* for this request.

Assume HPC_{max} = 3 (max Hops Per Cycle)



R0	
buffer	0
mux	local
xbar	W->E

R1	
buffer	0
mux	bypass
xbar	W->E

R2	
buffer	0
mux	bypass
xbar	W->E

R3	
buffer	1
mux	X
xbar	X

R4	
buffer	0
mux	X
xbar	X

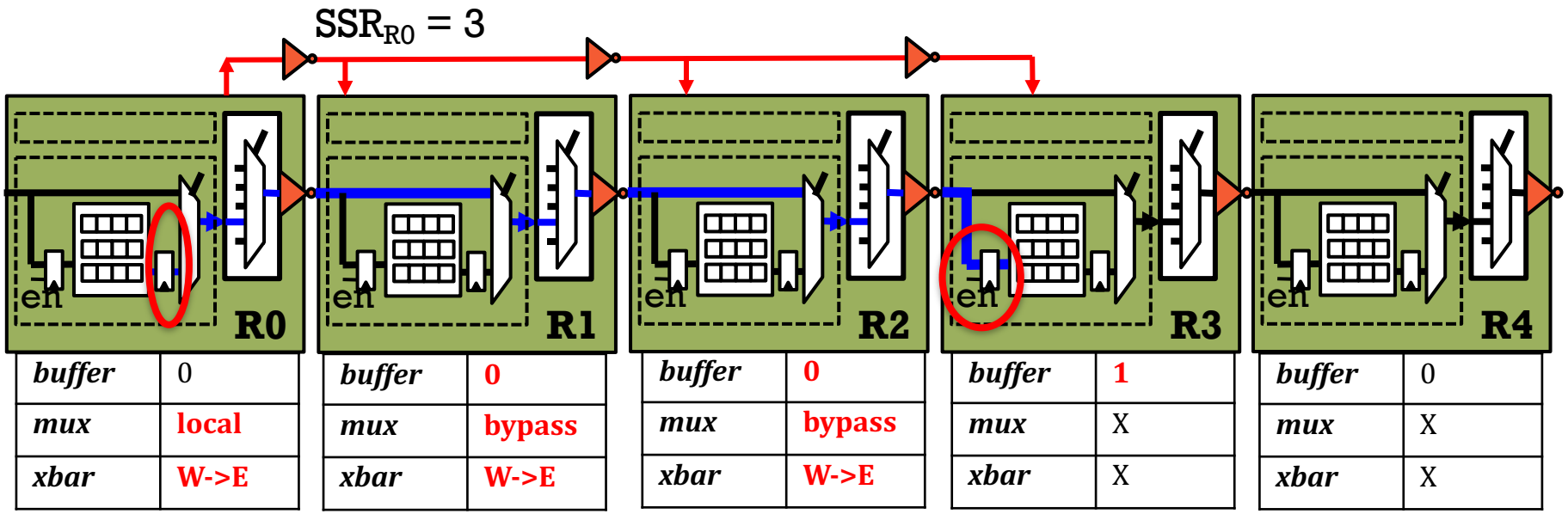
TRAVERSAL EXAMPLE 1: R0 → R3

Cycle 1 (Ctrl): R0 sends SSR = 3 *(3-hop path request)*

All routers set *buffer*, *mux* and *xbar* for this request.

Assume HPC_{max} = 3 (max Hops Per Cycle)

Cycle 2 (Data): R0 sends flit to R3



R0	
buffer	0
mux	local
xbar	W->E

R1	
buffer	0
mux	bypass
xbar	W->E

R2	
buffer	0
mux	bypass
xbar	W->E

R3	
buffer	1
mux	X
xbar	X

R4	
buffer	0
mux	X
xbar	X

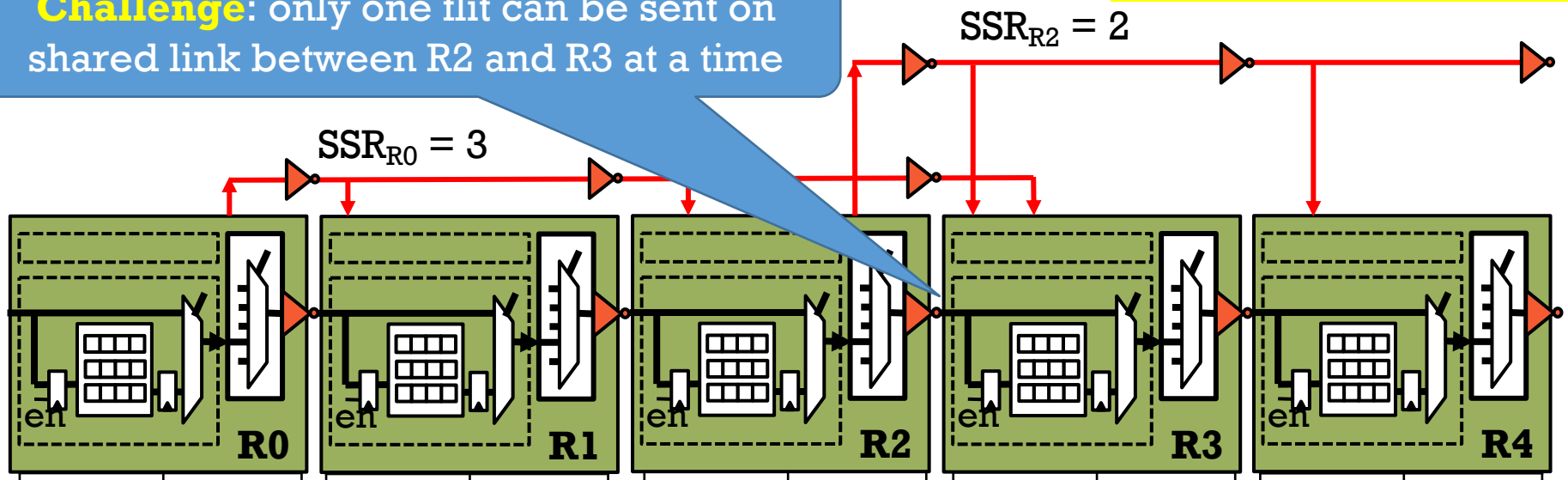
A SMART path is simply a combination of ***buffer***, ***mux*** and ***xbar*** at all the intermediate routers.

EXAMPLE 2: R0 → R3 *AND* R2 → R4

Cycle 1 (Ctrl): R0 sends SSR = 3. R2 sends SSR = 2

Assume HPC_{max} = 3 (max Hops Per Cycle)

Challenge: only one flit can be sent on shared link between R2 and R3 at a time



buffer	0	*buffer*	0	*buffer*	0	*buffer*	0	*buffer*	0
mux	X	*mux*	X	*mux*	X	*mux*	X	*mux*	X
xbar	X	*xbar*	X	*xbar*	X	*xbar*	X	*xbar*	X

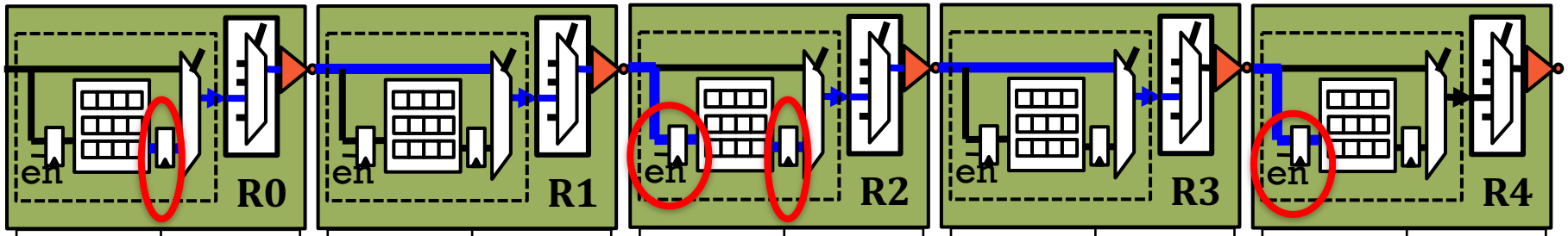
Solution: Routers prioritize between path requests based on distance

two alternate schemes:
Prio=Local and ***Prio=Bypass***

EXAMPLE 2: R0 → R3 *AND* R2 → R4

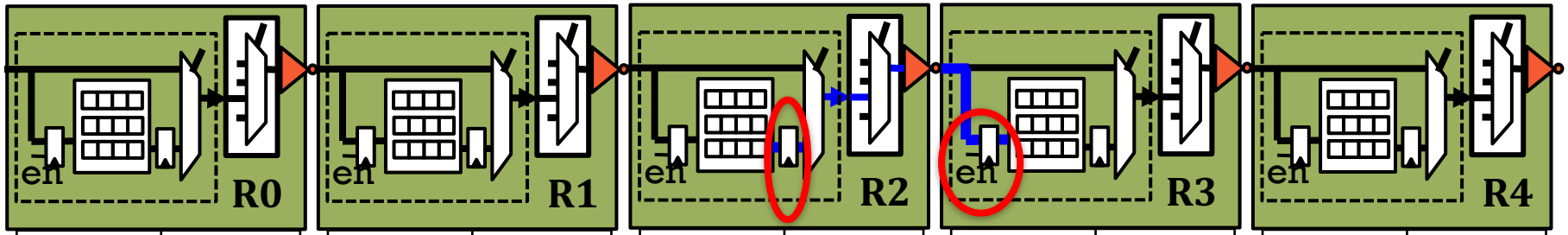
Prio = Local ➔ 0 hop > 1 hop > 2 hop … > HPC_{max} hop

Cycle 2 (Data): R2 sends flit to R4. R0's flit blocked at R2.



	R0	R1	R2	R3	R4
buffer	0	0	1	0	1
mux	local	bypass	local	bypass	X
xbar	W->E	W->E	W->E	W->E	X

Cycle 4+ (Data): R2 sends blocked flit to R3. *(after local arbitration + sending ctrl)*

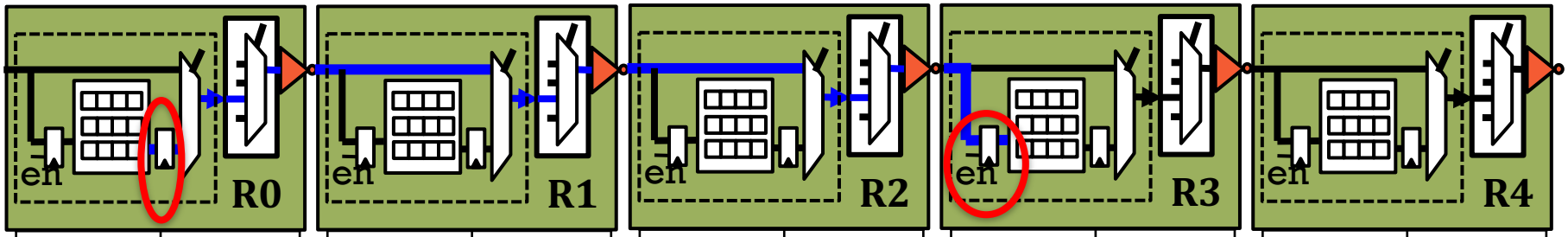


	R0	R1	R2	R3	R4
buffer	0	0	0	1	0
mux	X	X	local	X	X
xbar	X	X	W->E	X	X

EXAMPLE 2: R0 → R3 *AND* R2 → R4

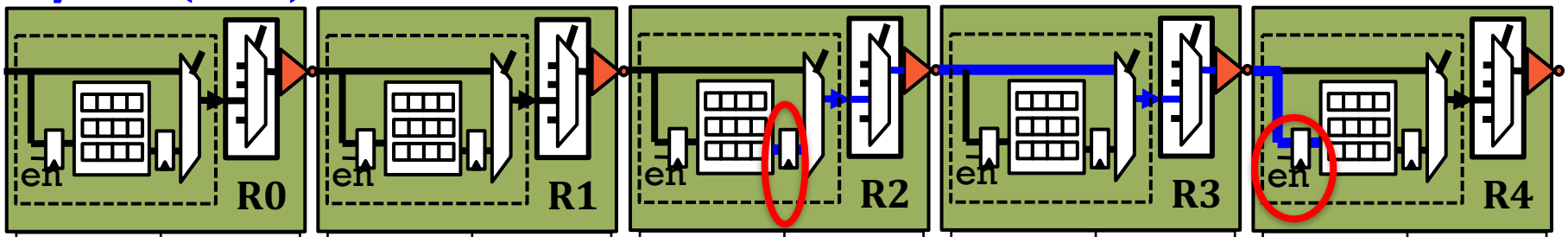
Prio = Bypass ➔ HPC_{max} hop > (HPC_{max} - 1) hop > … > 1 hop > 0 hop

Cycle 2 (Data): R0 sends flit to R3. R2's flit waits.



R0		R1		R2		R3		R4	
buffer	0	*buffer*	0	*buffer*	0	*buffer*	1	*buffer*	0
mux	local	*mux*	bypass	*mux*	bypass	*mux*	X	*mux*	X
xbar	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	X	*xbar*	X

Cycle 3 (Data): R2 sends flit to R4.



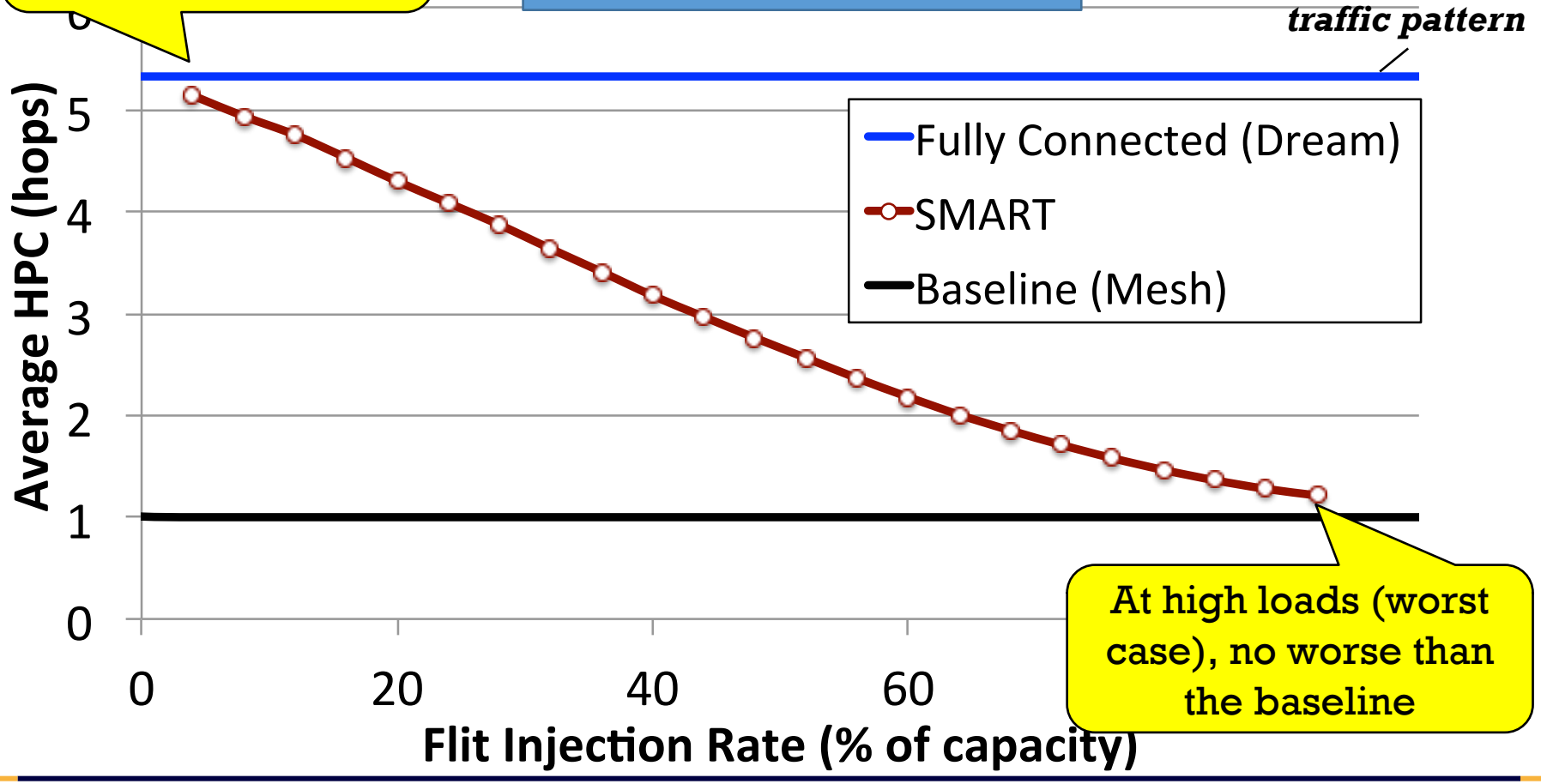
R0		R1		R2		R3		R4	
buffer	0	*buffer*	0	*buffer*	0	*buffer*	0	*buffer*	1
mux	X	*mux*	X	*mux*	local	*mux*	bypass	*mux*	X
xbar	X	*xbar*	X	*xbar*	W->E	*xbar*	W->E	*xbar*	X

ACHIEVABLE HOPS PER CYCLE (HPC)

SMART paths are opportunistic

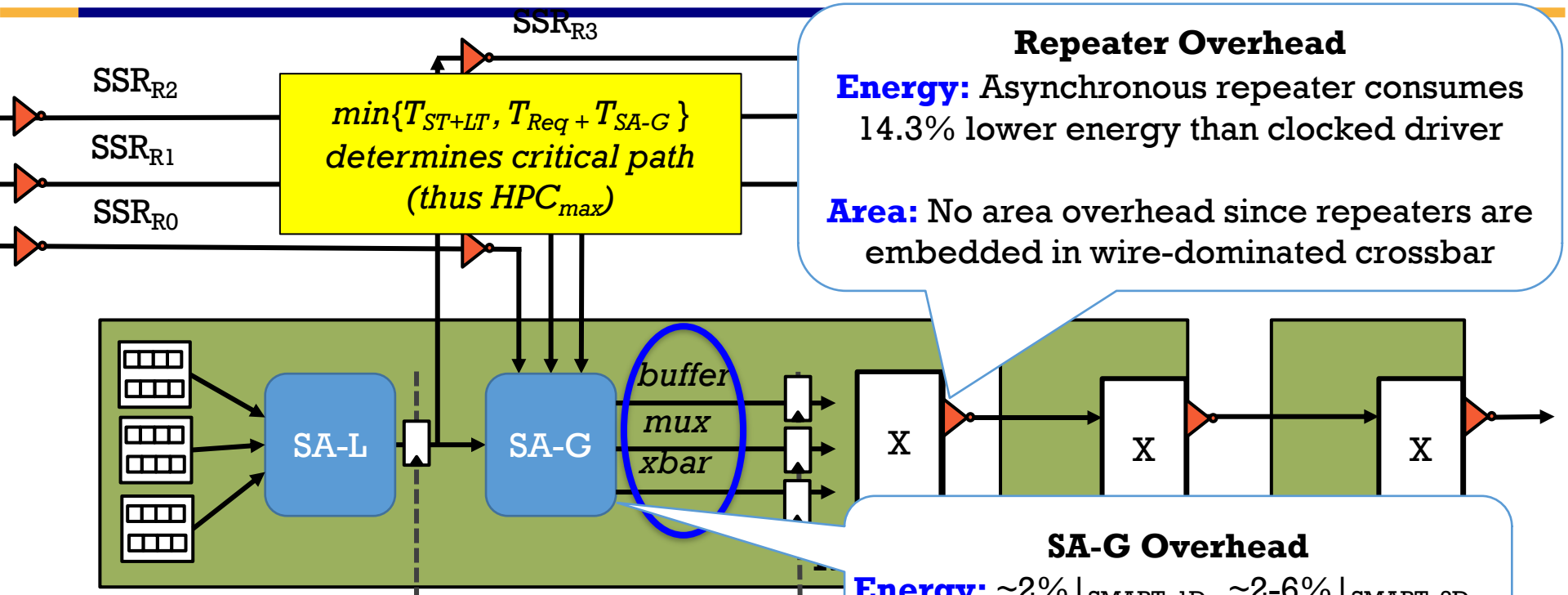
Achievable HPC depends on link contention

At low loads (best case), as good as dedicated wires



At high loads (worst case), no worse than the baseline

PIPELINE AND IMPLEMENTATION



Repeater Overhead

Energy: Asynchronous repeater consumes 14.3% lower energy than clocked driver

Area: No area overhead since repeaters are embedded in wire-dominated crossbar

SA-G Overhead

Energy: ~2% | SMART_1D, ~2-6% | SMART_2D

Area: < 1% | SMART_1D, 1-5% | SMART_2D

Cycle 0

Switch Allocation Local (SA-L)

(can be bypassed at low loads)

Cycle 1*

Req + Switch Allocation Global (SA-G)

HPC_{max} = 13 | SMART_1D

HPC_{max} = 9 | SMART_2D

Multi-hop Switch (crossbar) + Link Traversal (ST+LT)

(till it is stopped by some *buffer=1*)

HPC_{max} = 11

**Conventionally (i.e. in baseline), winners of SA-L go for Switch + Link Traversal*

THE DEVIL IS IN THE DETAILS

- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?
- **SMART_2D**
  - How can flits bypass routers at turns?
- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?
- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in

THE DEVIL IS IN THE DETAILS

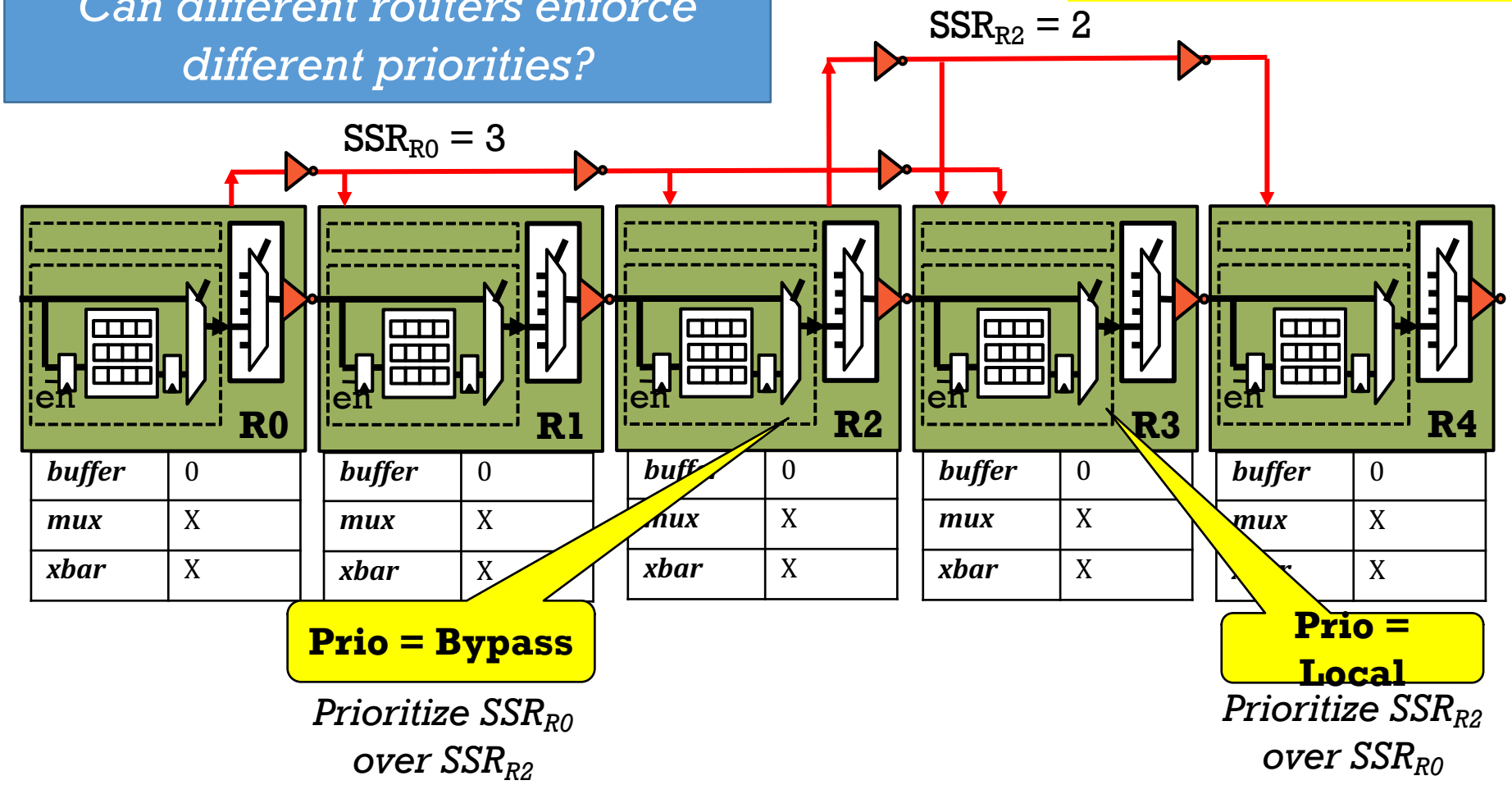
- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?
- **SMART_2D**
  - How can flits bypass routers at turns?
- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?
- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in

MANAGING DISTRIBUTED ARBITRATION

Cycle 1: R0 sends SSR = 3. R2 sends SSR = 2.

Assume HPC_{max} = 3 (max Hops Per Cycle)

Can different routers enforce different priorities?

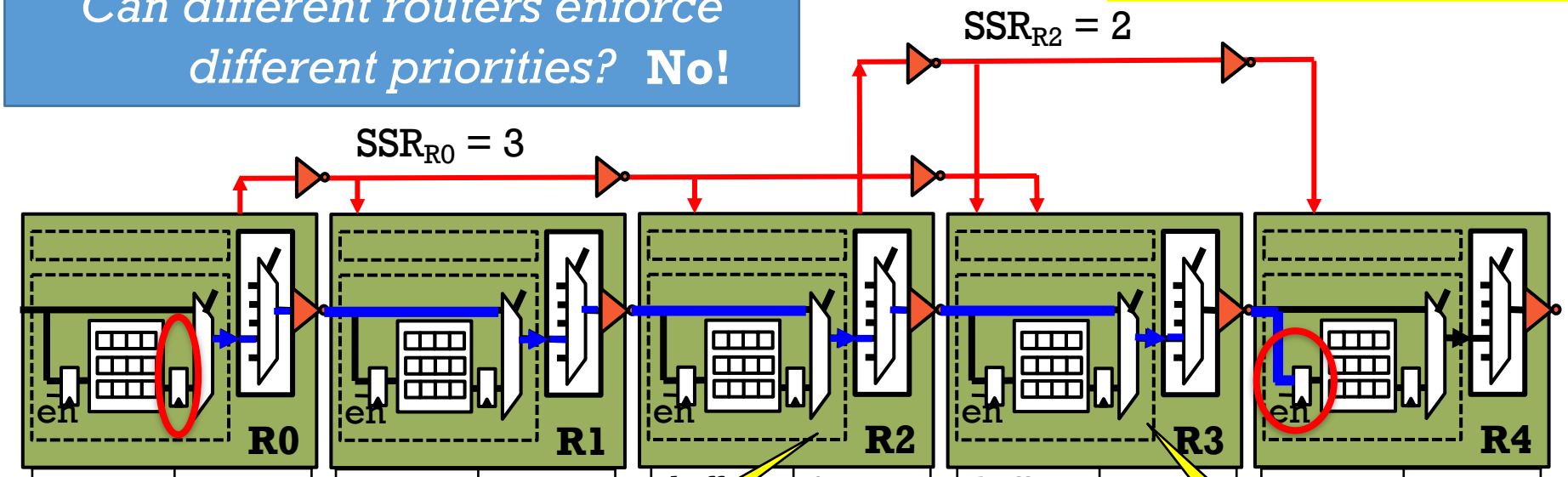


MANAGING DISTRIBUTED ARBITRATION

Cycle 1: R0 sends SSR = 3. R2 sends SSR = 2.

Assume HPC_{max} = 3 (max Hops Per Cycle)

Can different routers enforce different priorities? **No!**



R0		R1		R2		R3		R4	
buffer	0	*buffer*	0	*buffer*	0	*buffer*	0	*buffer*	1
mux	local	*mux*	bypass	*mux*	bypass	*mux*	bypass	*mux*	X
xbar	W→E	*xbar*	W→	*xbar*	W→E	*xbar*	W→E		X

Prio = Bypass

Prioritize SSR_{R0} over SSR_{R2}

R0's flit incorrectly reaches R4, instead of getting stopped at R3.

Prio = Local

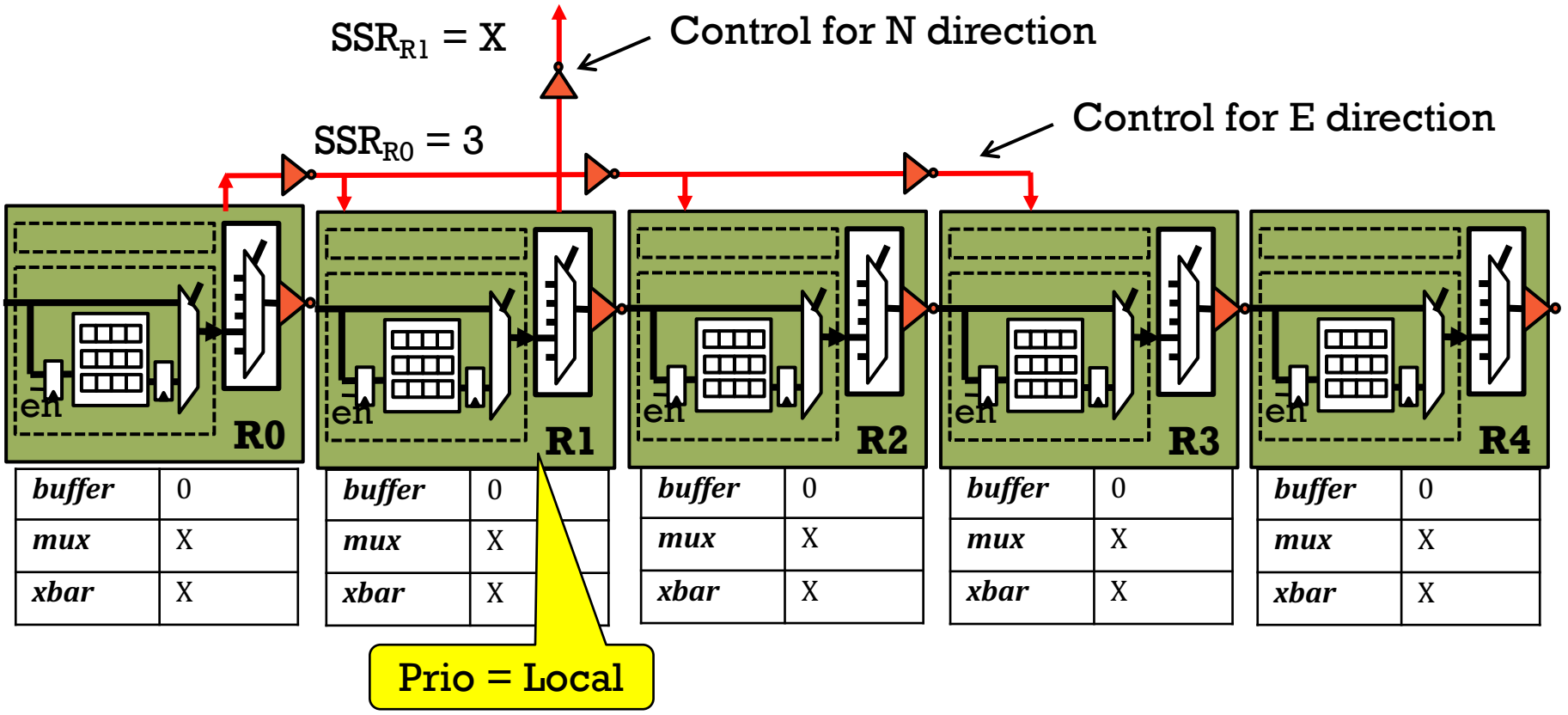
Prioritize SSR_{R2} over SSR_{R0}

MANAGING DISTRIBUTED ARBITRATION

- **Distributed Consensus:** All routers need to take the ***same decision*** about ***multiple contending flits*** in a ***distributed manner***
- **Solution:** All routers follow the same *static priority* between the path setup requests that they receive
  - **Prio = Local:** 0 hop > 1 hop > … (HPC_{max}-1) hop > HPC_{max} hop
  - **Prio = Bypass:** HPC_{max} hop > (HPC_{max}-1) hop > … 1 hop > 0 hop
- **Implication:** *a router will **not receive** a flit that it **does not expect***
- But *can a router **not receive** a flit that it **does expect**?*

MANAGING DISTRIBUTED ARBITRATION

Can a router *not receive* a flit that it *does expect*?



MANAGING DISTRIBUTED ARBITRATION

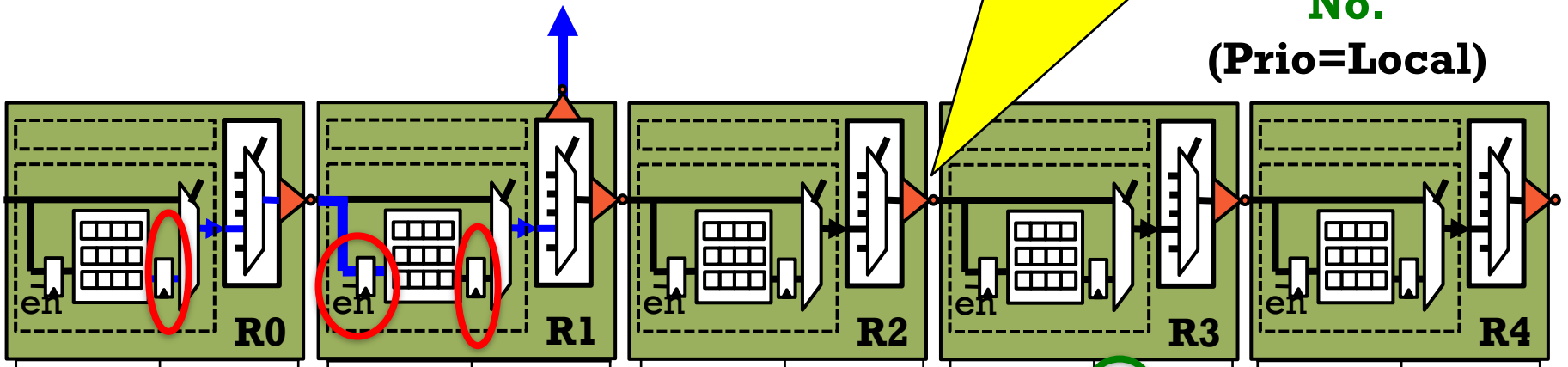
Can a router *not receive* a flit that it *does expect*? **Yes!**

Is there a performance loss?

The R2→R3 link was granted for this cycle, but went unused. What if some other flit wanted to use it?

No.

(Prio=Local)



R0	
buffer	0
mux	local
xbar	W->E

R1	
buffer	1
mux	local
xbar	W->N

R2	
buffer	0
mux	bypass
xbar	W->E

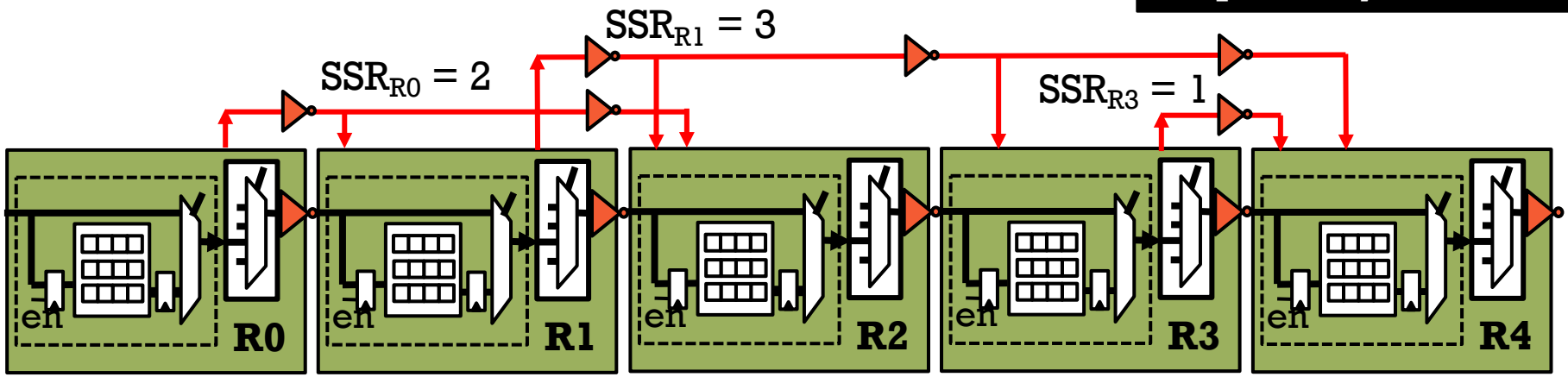
R3	
buffer	1
mux	X
xbar	X

R4	
buffer	0
mux	X
xbar	X

IMPACT OF FALSE NEGATIVES

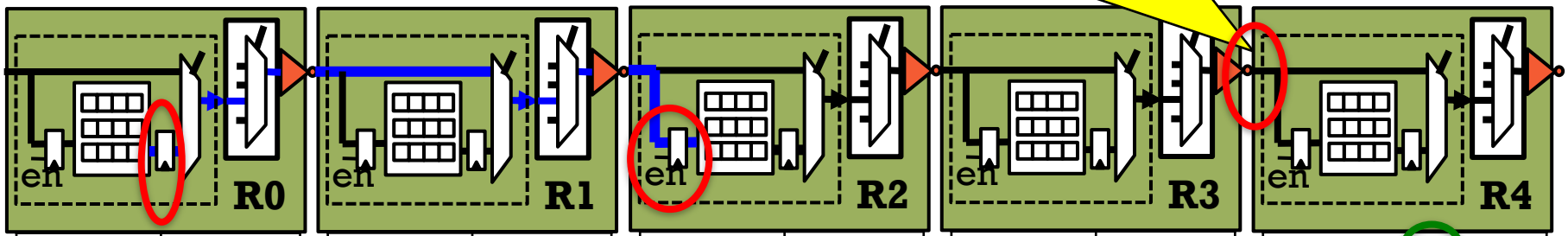
Cycle 1 (Ctrl Req): R0→R2, R1→R4, R3→R4.

Req Priority =



Cycle 2: Flit: R0→R2.

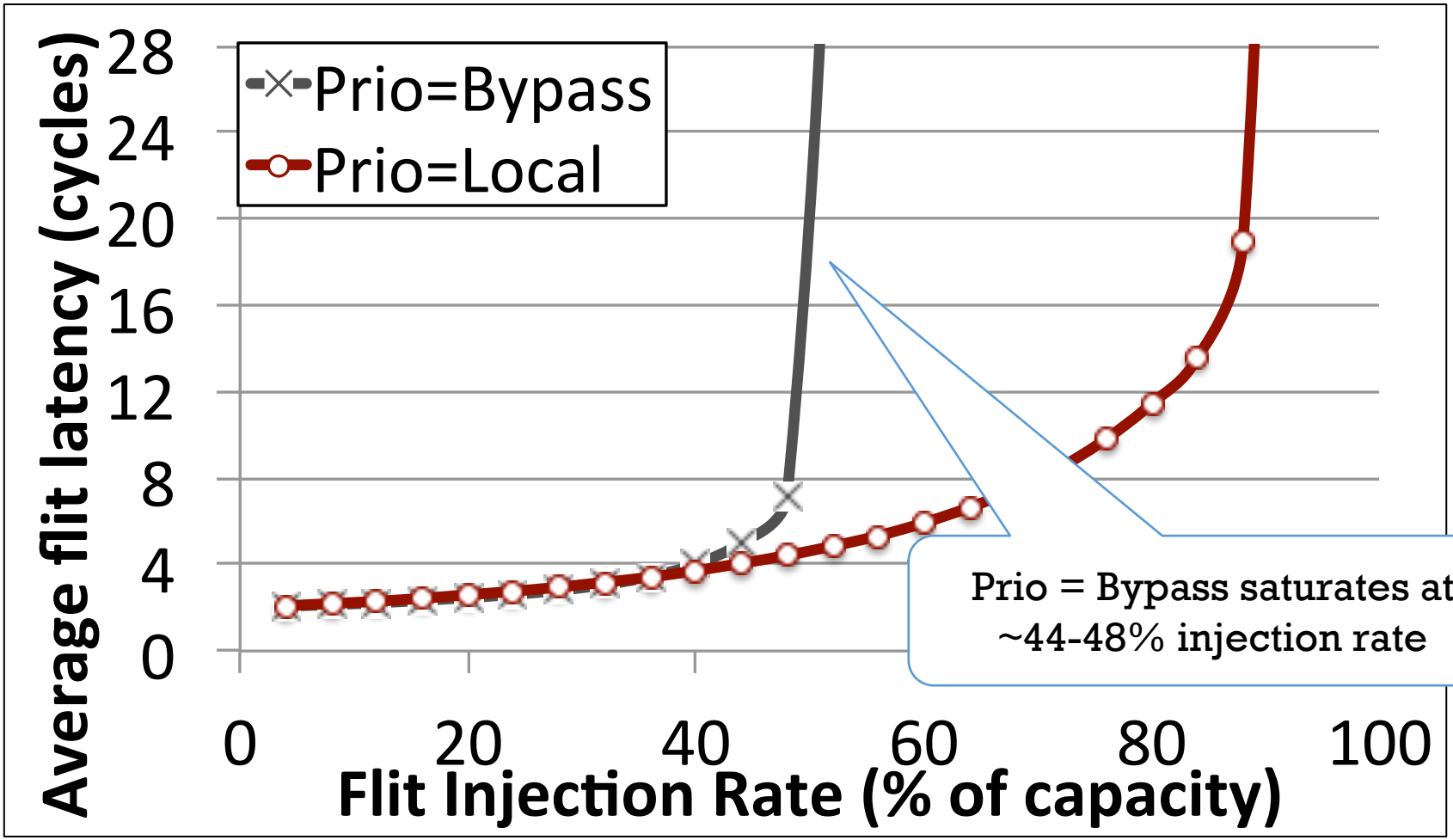
forced *starvation* and throughput loss



buffer	0	*buffer*	0	*buffer*	1	*buffer*	0	*buffer*	1
mux	local	*mux*	bypass	*mux*	local	*mux*	bypass	*mux*	X
xbar	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	X

IMPACT OF FALSE NEGATIVES

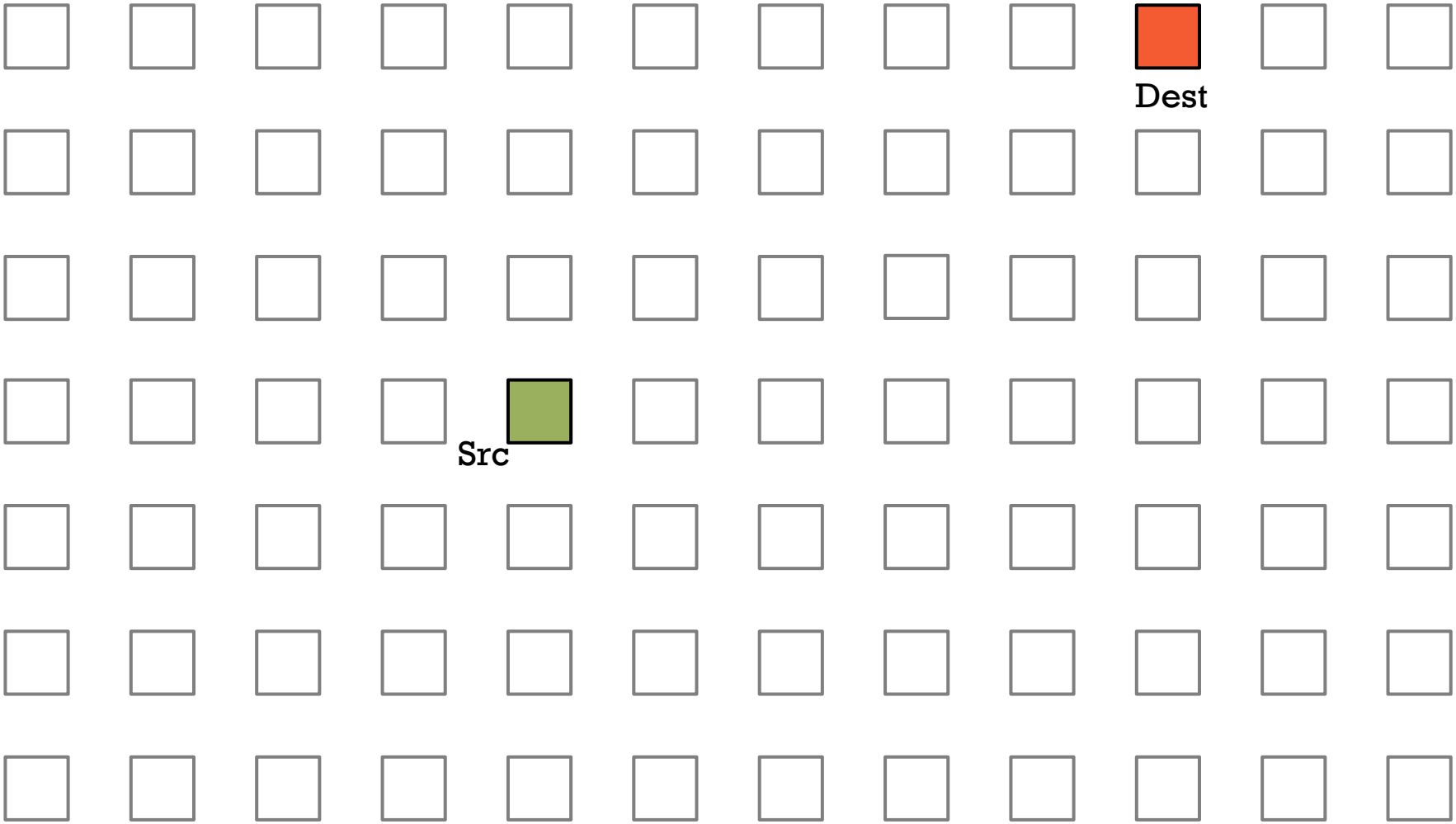
Prio = Bypass increases false negatives at high-loads



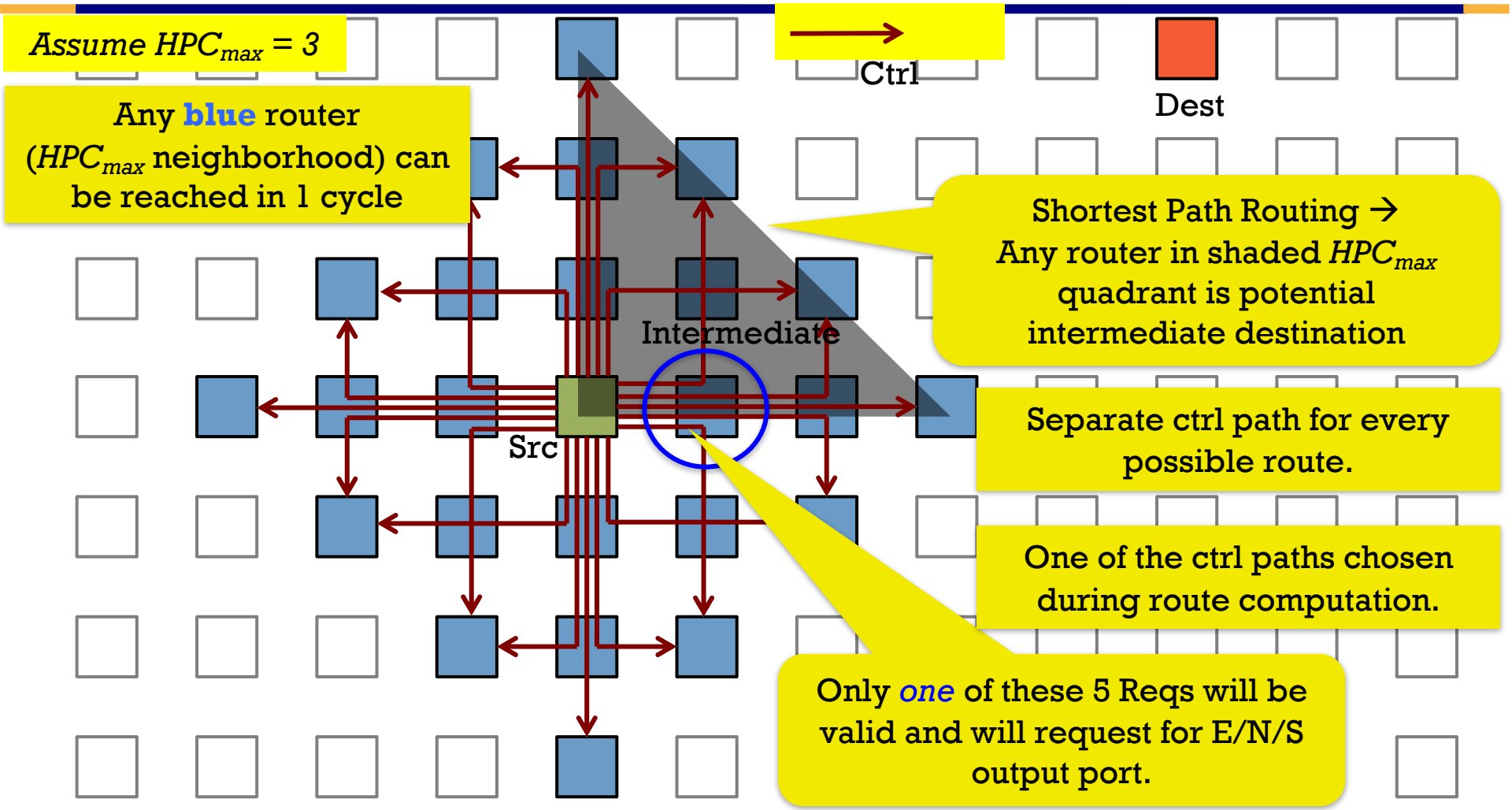
THE DEVIL IS IN THE DETAILS

- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?
- **SMART_2D**
  - How can flits bypass routers at turns?
- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?
- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in

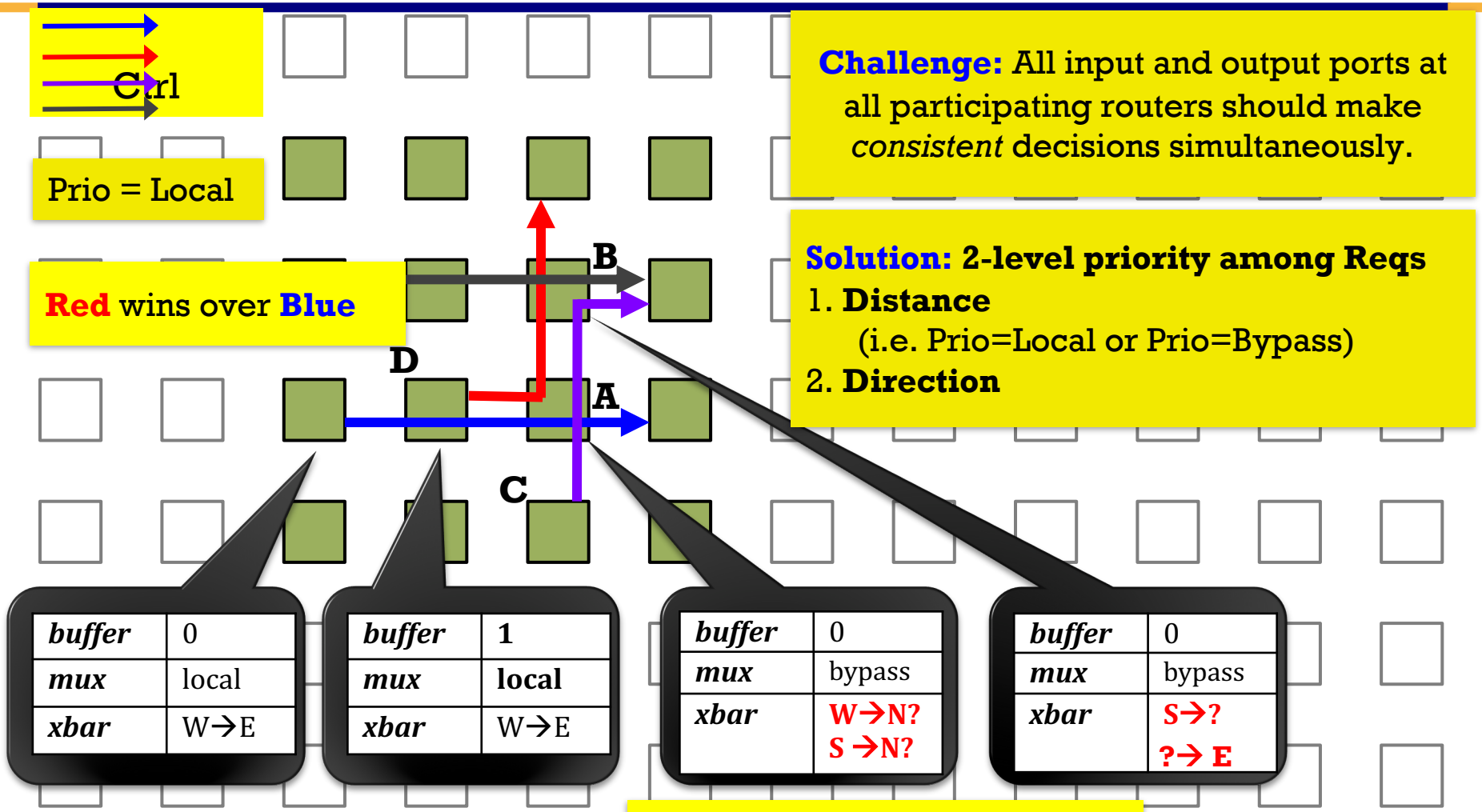
BYPASS AT TURNS



BYPASS AT TURNS



CONTROL ARBITRATION



Challenge: All input and output ports at all participating routers should make *consistent* decisions simultaneously.

Solution: 2-level priority among Reqs

1. **Distance**
   (i.e. Prio=Local or Prio=Bypass)
2. **Direction**

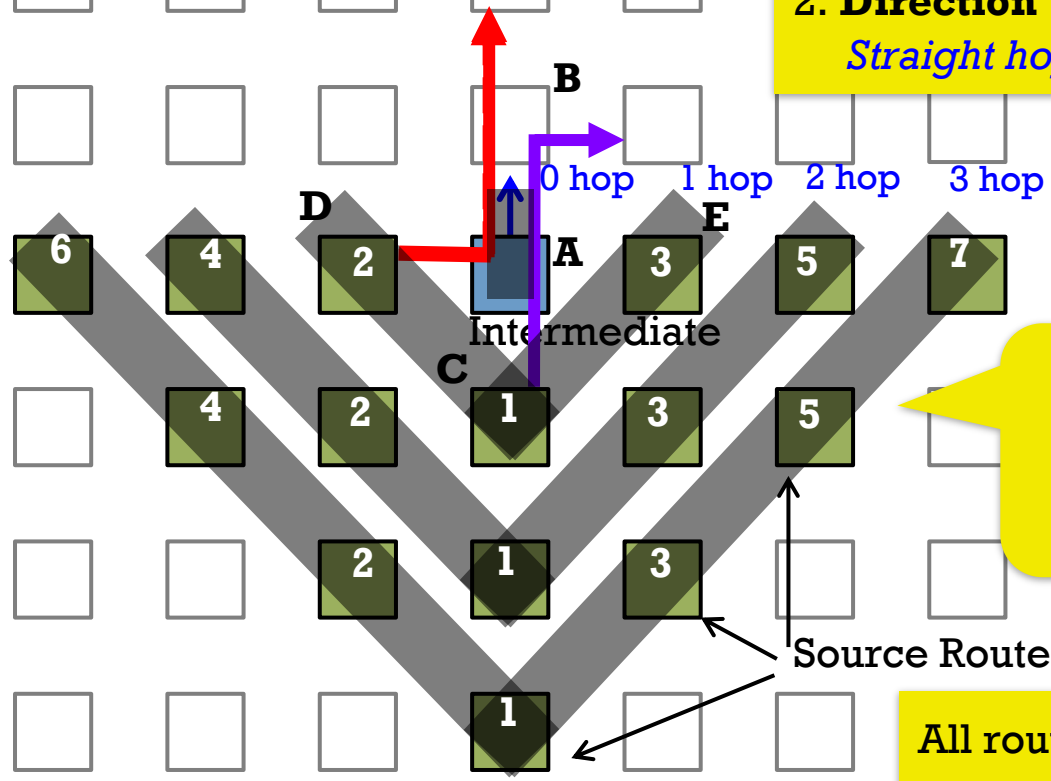
How do we choose between **Red** and **Purple**?

PRIORITY AT OUTPUT PORT

At **A:** **Purple** wins over **Red**

2-level Ctrl Req Priority

1. **Distance**
   0 hop > 1 hop > 2 hop … (Prio = Local)
2. **Direction**
   Straight hops > Left hops > Right hops



All **green** routers (sources) can request for the **North output port** at the **blue** (intermediate) router

Source Routers

All routers enforce same priority (to guarantee no false positives)

Assume HPC_{max} = 3

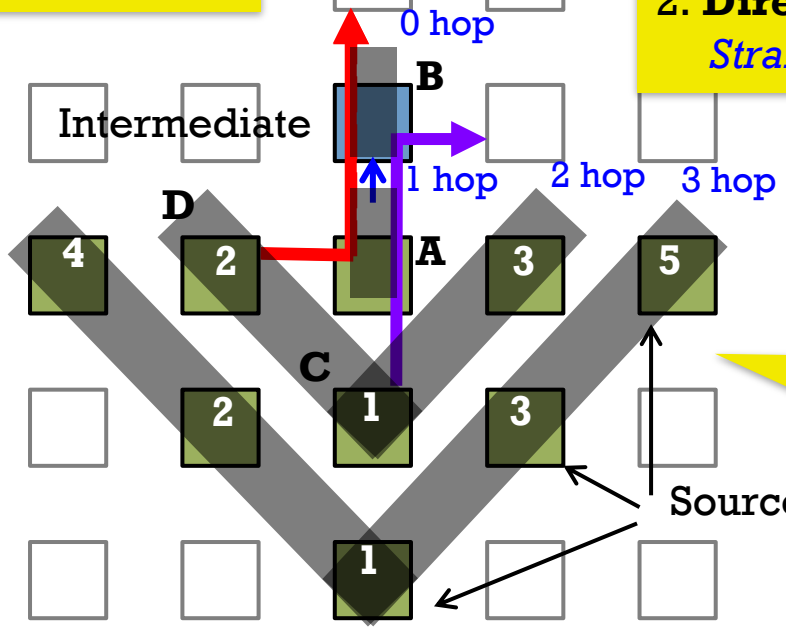
PRIORITY AT INPUT PORT

At **A:** **Purple** wins over **Red**

At **B:** **Purple** wins over **Red**

2-level Ctrl Req Priority

1. **Distance**
   0 hop > 1 hop > 2 hop … (Prio = Local)
2. **Direction**
   Straight hops > Left hops > Right hops



All **green** routers (sources) can request for the **South input port** at the **blue** (intermediate) router

Assume HPC_{max} = 3

THE DEVIL IS IN THE DETAILS

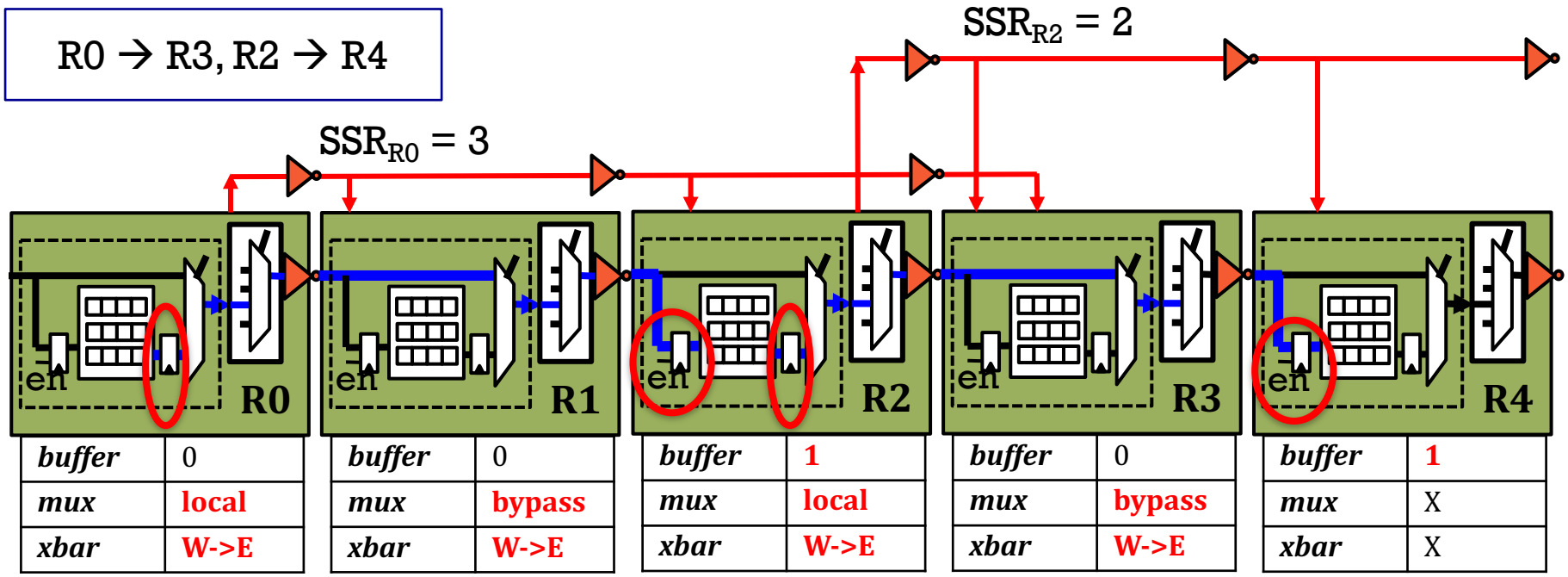
- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?
- **SMART_2D**
  - How can flits bypass routers at turns?
- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?
- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in

THE DEVIL IS IN THE DETAILS

- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?
- **SMART_2D**
  - How can flits bypass routers at turns?
- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?
- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in

BUFFER MANAGEMENT

R0 → R3, R2 → R4



buffer	0	*buffer*	0	*buffer*	1	*buffer*	0	*buffer*	1
mux	local	*mux*	bypass	*mux*	local	*mux*	bypass	*mux*	X
xbar	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	X

How do we guarantee R2 has a free buffer / VC for the flit from R0?

Every router has free VC information about its neighbor, just like the baseline.

R0 sends only if R1 has a free buffer

R1 lets the incoming flit bypass only if R2 has a free buffer, else latches it.

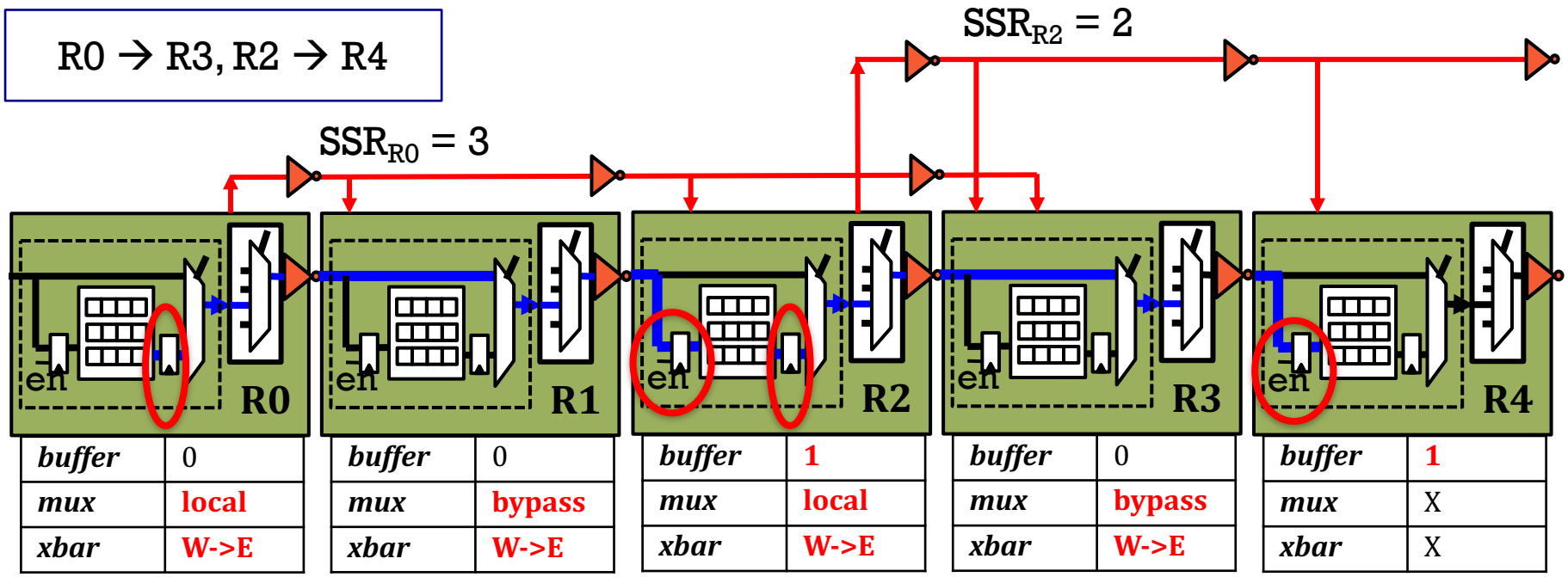
Corollary: VCid is allocated after it stops, rather than before starting, since router where it stops is not known.

THE DEVIL IS IN THE DETAILS

- **Managing Distributed Arbitration**
  - Could flits get misrouted?
  - Could flits not arrive when expected?
- **SMART_2D**
  - How can flits bypass routers at turns?
- **Buffer Management**
  - How is a flit guaranteed a buffer (and in the correct virtual channel) if it is stopped mid-way?
  - How is buffer availability conveyed?
- **Multi-flit packets**
  - How does SMART guarantee that flits (head, body, tail) of a packet do not get re-ordered?
  - How do flits know which VC to stop in

MULTI-FLIT PACKETS (1)

R0 → R3, R2 → R4



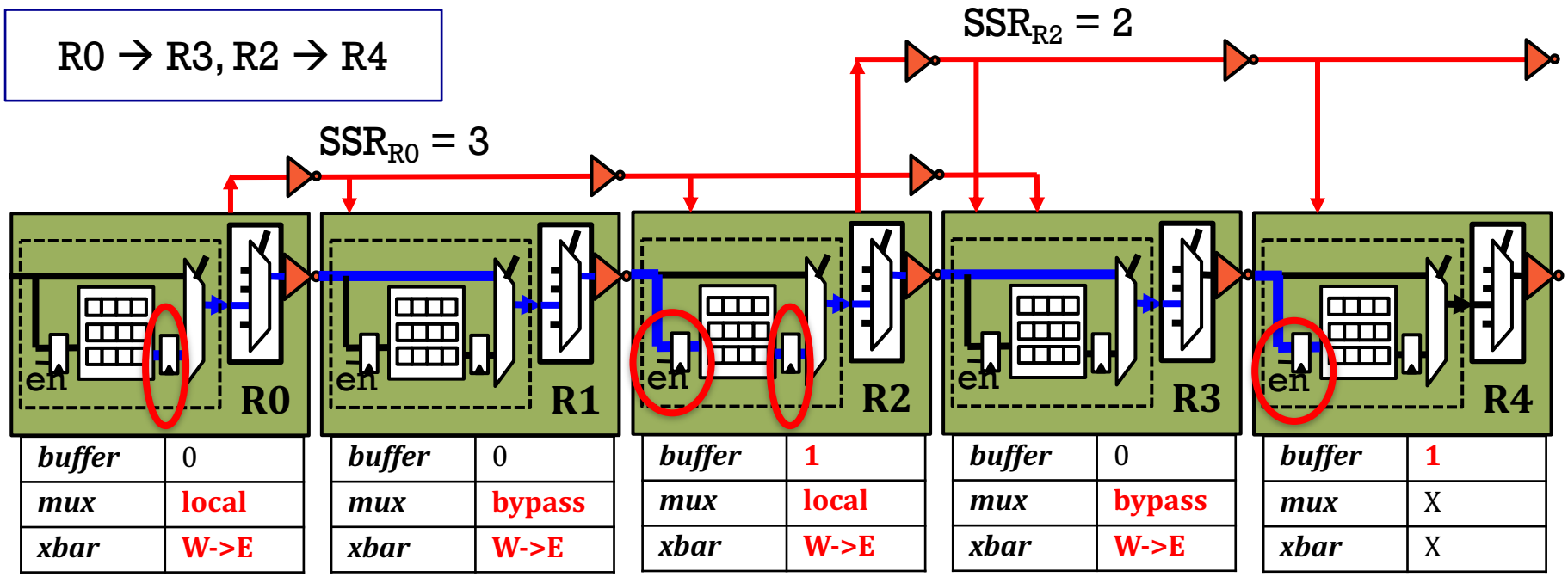
buffer	0	*buffer*	0	*buffer*	1	*buffer*	0	*buffer*	1
mux	local	*mux*	bypass	*mux*	local	*mux*	bypass	*mux*	X
xbar	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	X

How do we guarantee Body/Tail flits from R0 do not bypass R2 if Head was stopped?

If R2 sees a SSR requesting bypass from a router from where it has a buffered flit, it stops all subsequent flits from that router, and buffers them in the same VC

MULTI-FLIT PACKETS (2)

R0 → R3, R2 → R4



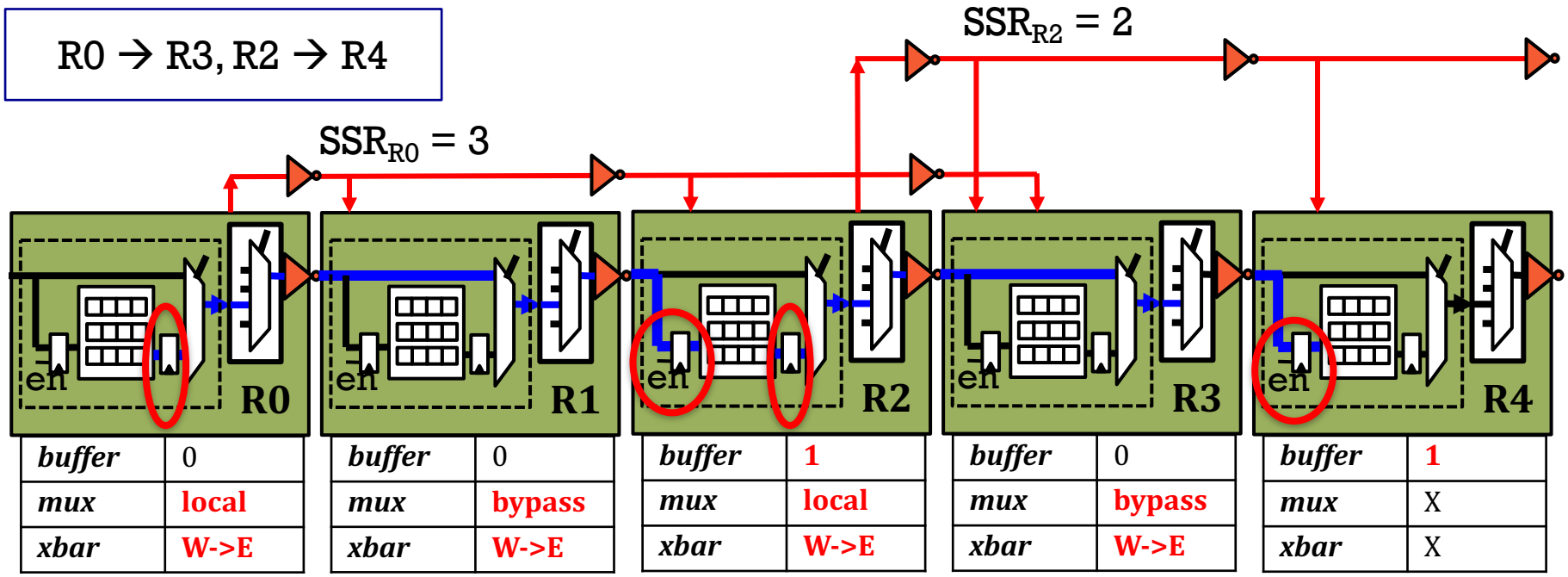
buffer	0	*buffer*	0	*buffer*	1	*buffer*	0	*buffer*	1
mux	local	*mux*	bypass	*mux*	local	*mux*	bypass	*mux*	X
xbar	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	W->E	*xbar*	X

How do the Body/Tail flits know which VC to stop in if VC is allocated at the router where Head stops?

Match using source_id of all the flits of the packet

MULTI-FLIT PACKETS (2)

R0 → R3, R2 → R4



	R0	R1	R2	R3	R4
buffer	0	0	1	0	1
mux	local	bypass	local	bypass	X
xbar	W->E	W->E	W->E	W->E	X

What if 2 flits from different packets from the same source arrive?

Virtual Cut-Through

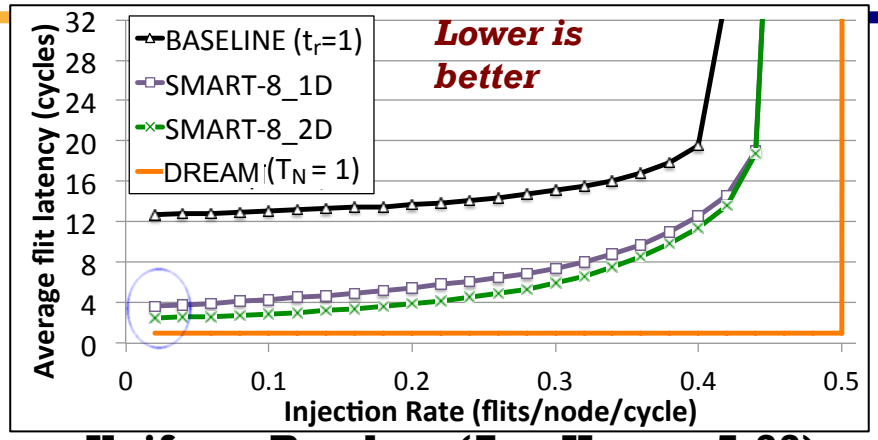
- enough Buffers to store all flits of a packet
- all flits of one packet should leave before flits of another packet

EVALUATIONS

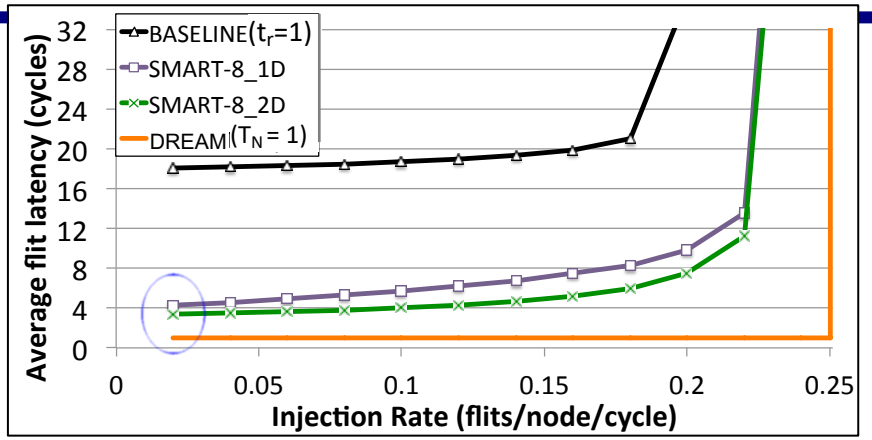
- **Simulation Infrastructure:** GEMS (Full-System) + Garnet (NoC)
- **System:** 64-core (8x8 Mesh)
- **Technology:** 45nm, 1GHz

- **Networks being compared**
  - **BASELINE (t_r=1):** Baseline Mesh with 1-cycle router at every hop
  - **SMART-<HPC$_{max}$>_1D:** Always stop at turning router
    - Best case delay: 4 cycles (Req_X → $Flit_X$ → Req_Y → $Flit_Y$)
  - **SMART-<HPC$_{max}$>_2D:** Bypass turning router
    - Best case delay: 2 cycles (Req → Flit)
  - **DREAM (T_N=1):** Contention-less 1-cycle network (fully-connected)

BREAKING THE LATENCY BARRIER

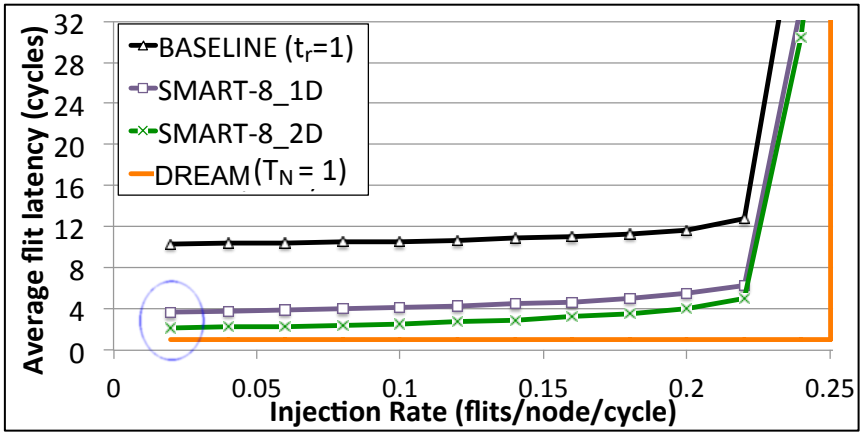


Uniform Random (Avg Hops = 5.33)

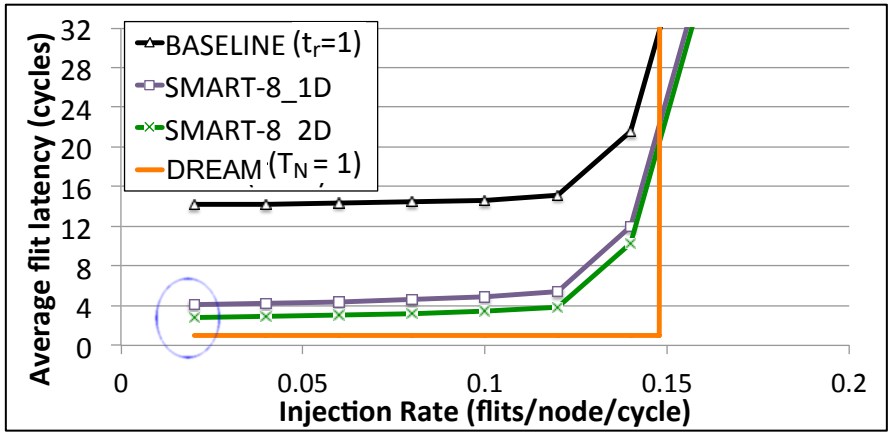


Bit Complement (Avg Hops = 8)

SMART reduces low-load latency to 2-4 cycles **across all** traffic patterns, ***independent*** of average hops.



Shuffle (Avg Hops = 4)



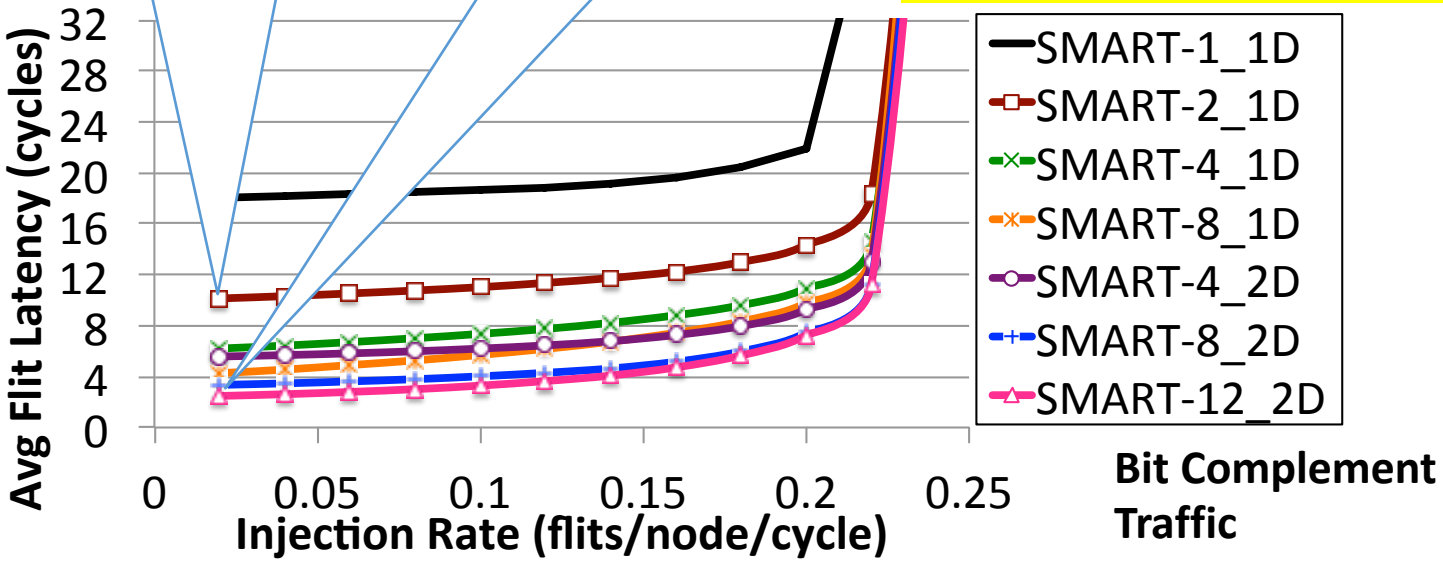
Transpose (Avg Hops = 6)

IMPACT OF HPC_{MAX}

HPC_{max} of 2 and 4 give 1.8X and 3X reduction in latency

HPC_{max} of 8 gives 5.4X reduction in latency

HPC_{max} *(max Hops Per Cycle): maximum number of "hops" that the underlying wire allows the flit to traverse within a clock cycle*



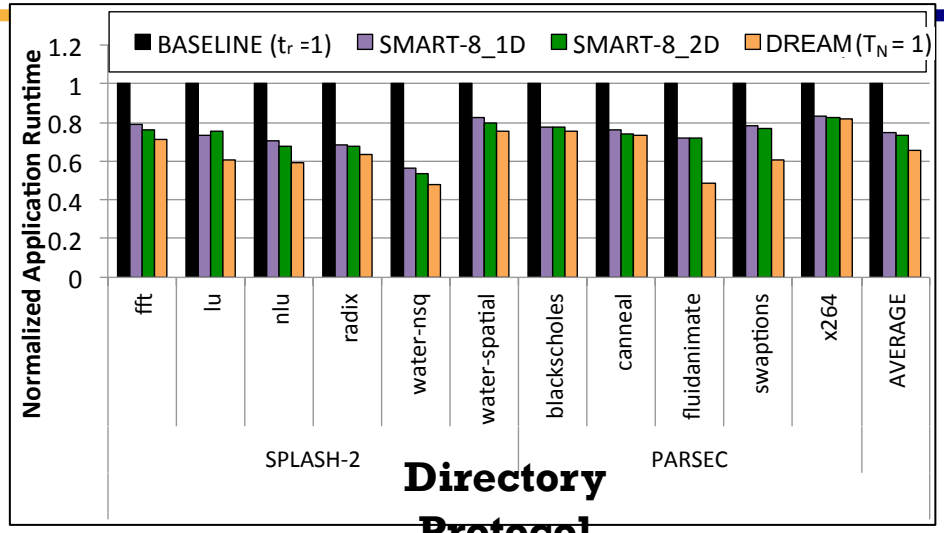
Bit Complement Traffic

SMART is a better design choice than a baseline 1-cycle router network even at larger tile size or higher frequency (i.e. lower HPC_{max})

A baseline mesh network needs to run at 5.4GHz to match the performance of a 1GHz SMART NoC

HPC_{max} will go up as technology scales! (smaller cores, similar frequency)

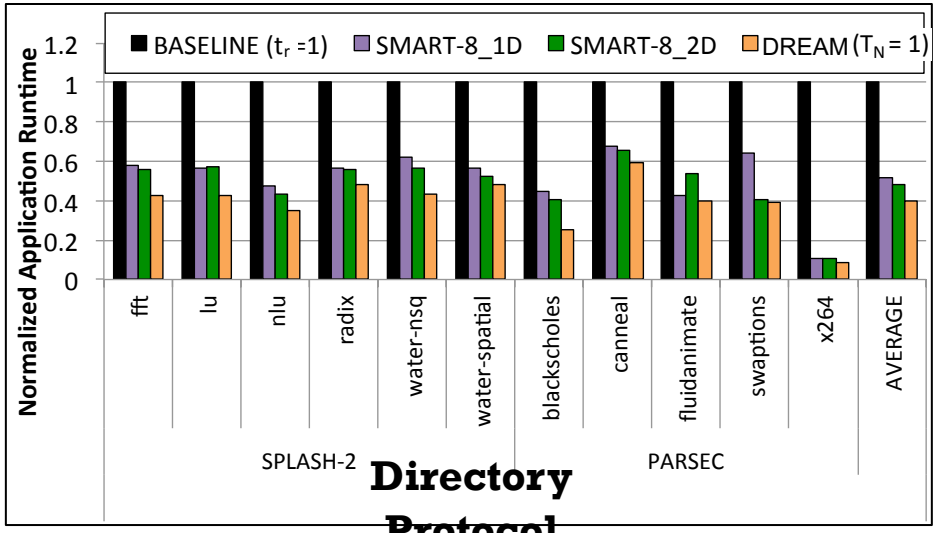
FULL-SYSTEM RUNTIME



Private L2/tile

SMART reduces runtime by 26-27% for Private L2

8% off a dream 1-cycle network



Shared L2/tile

SMART reduces runtime by 49-52% for Shared L2

9% off a dream 1-cycle network

PROJECT IDEAS USING SMART

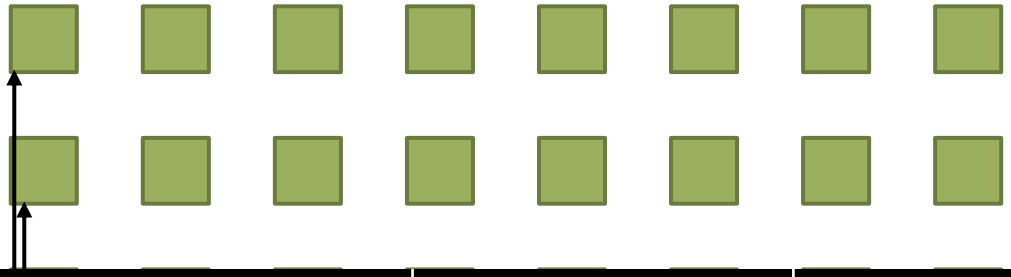
- **SMART NoCs**
  - SMART NoCs vs High-Radix topologies
  - SMART NoCs with non-minimal routes
  - SMART paths for latency guarantees
  - SMART NoC inside GPU
  - Reserve SMART paths using hints from cache/memory-controller/OS

BACKUPS

PROTOCOL-LEVEL ORDERING

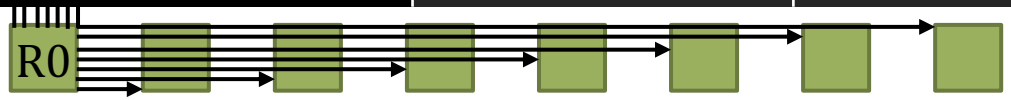
- If a virtual network requires protocol-level ordering
  - Only deterministic routing allowed within that virtual network.
  - Priority should be Local > Bypass
    - Guarantees that 2 flits from the same source do not overtake each other at any router.

SMART VS. FLATTENED BUTTERFLY



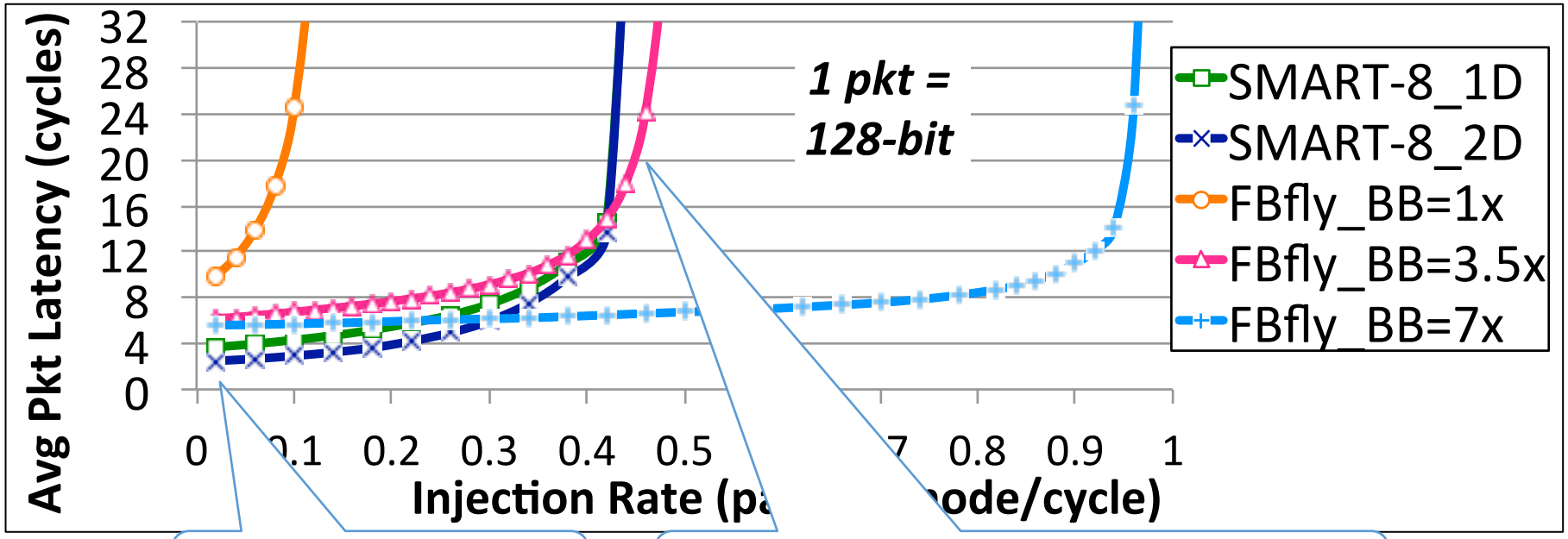
What if we exploit repeated wires and add explicit 1-cyle **physical express links**?

	SMART	Flattened Butterfly		
Ports	5-port router	15-port router		
Router Delay	1-2 cycle in router [SA-L, SSR+SA-G]	3-4 cycle in router [8-14:1 Arbiter, Multi-stage Xbar]		
Bisection Bandwidth (BB)	128x8 *1-flit req* *5-flit resp*	128x8 [**1x**] *7-flit req* *35-flit resp*	448x8 [**3.5x**] *2-flit req* *10-flit resp*	896x8 [**7x**] *1-flit req* *5-flit resp*
Dyn Power (mW)	24.5	23.1 [**~1x**]	37 [**1.5x**]	58 [**2.3x**]
Area (mm x mm)	0.27x0.27	0.47x0.47 [**3x**]	0.53x0.53 [**3.9x**]	0.7x0.7 [**6.7x**]



SMART VS. FLATTENED BUTTERFLY

Assume 1-cycle Flattened Butterfly Router: Highly optimistic assumption



SMART always beats FBfly in latency

FBfly matches SMART in throughput only with 3.5X more wires

Better to use SMART and reconfigure 1-cycle multi-hop paths based on traffic rather than use a **fixed** high-radix topology.