Garnet2.0: A Detailed On-Chip Network Model Inside a Full-System Simulator

Tushar Krishna

Assistant Professor
School of ECE and CS
Georgia Institute of Technology

tushar@ece.gatech.edu

gem5 workshop
ARM Research Summit
September 11, 2017

http://synergy.ece.gatech.edu/tools/garnet

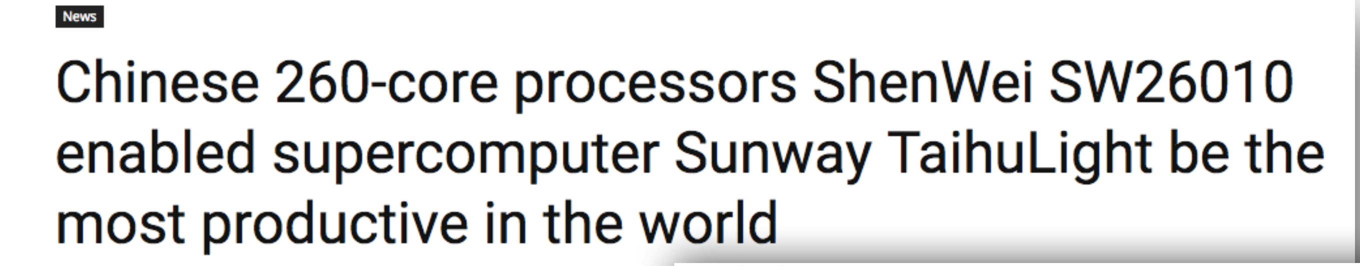
Overview

- What are Networks-On-Chip (NoCs)?
- Why model NoCs accurately
- Garnet2.0
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - System Integration
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - Output Stats
- Extensions and FAQs

Overview

- **What are Networks-On-Chip (NoCs)?**
- Why model NoCs accurately
- Garnet2.0
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - System Integration
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - Output Stats
- Extensions and FAQs

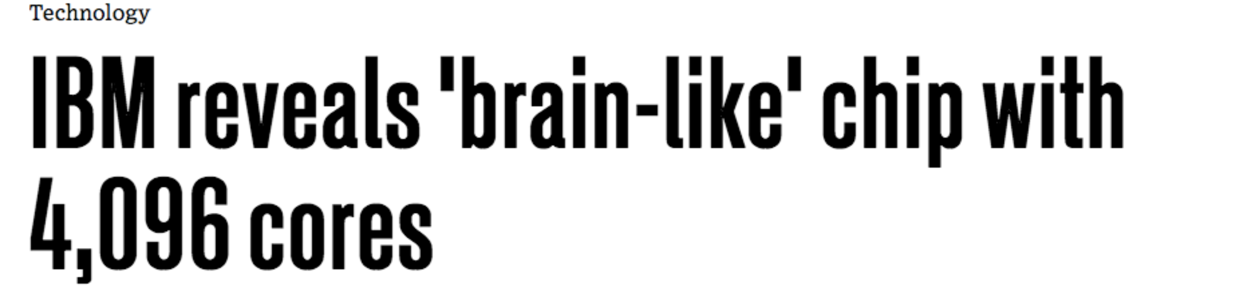
Networks-on-Chip



News

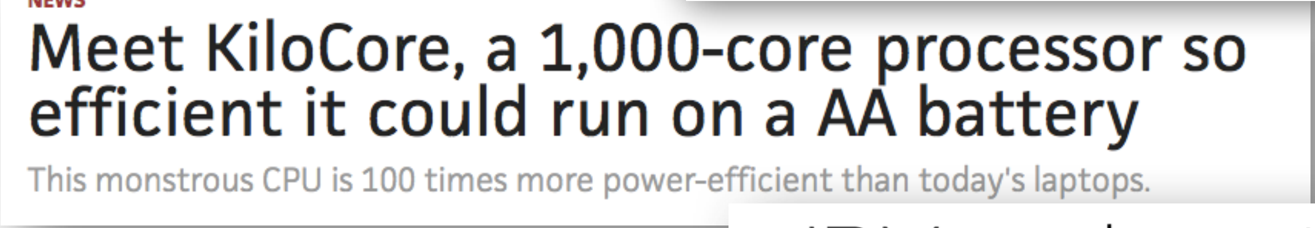
Chinese 260-core processors ShenWei SW26010 enabled supercomputer Sunway TaihuLight be the most productive in the world

June 21, 2016 331 0



Technology

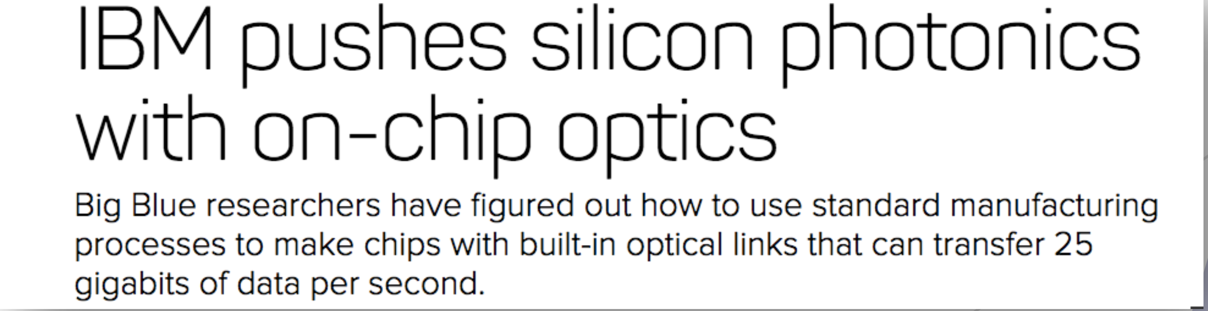
IBM reveals 'brain-like' chip with 4,096 cores



NEWS

Meet KiloCore, a 1,000-core processor so efficient it could run on a AA battery

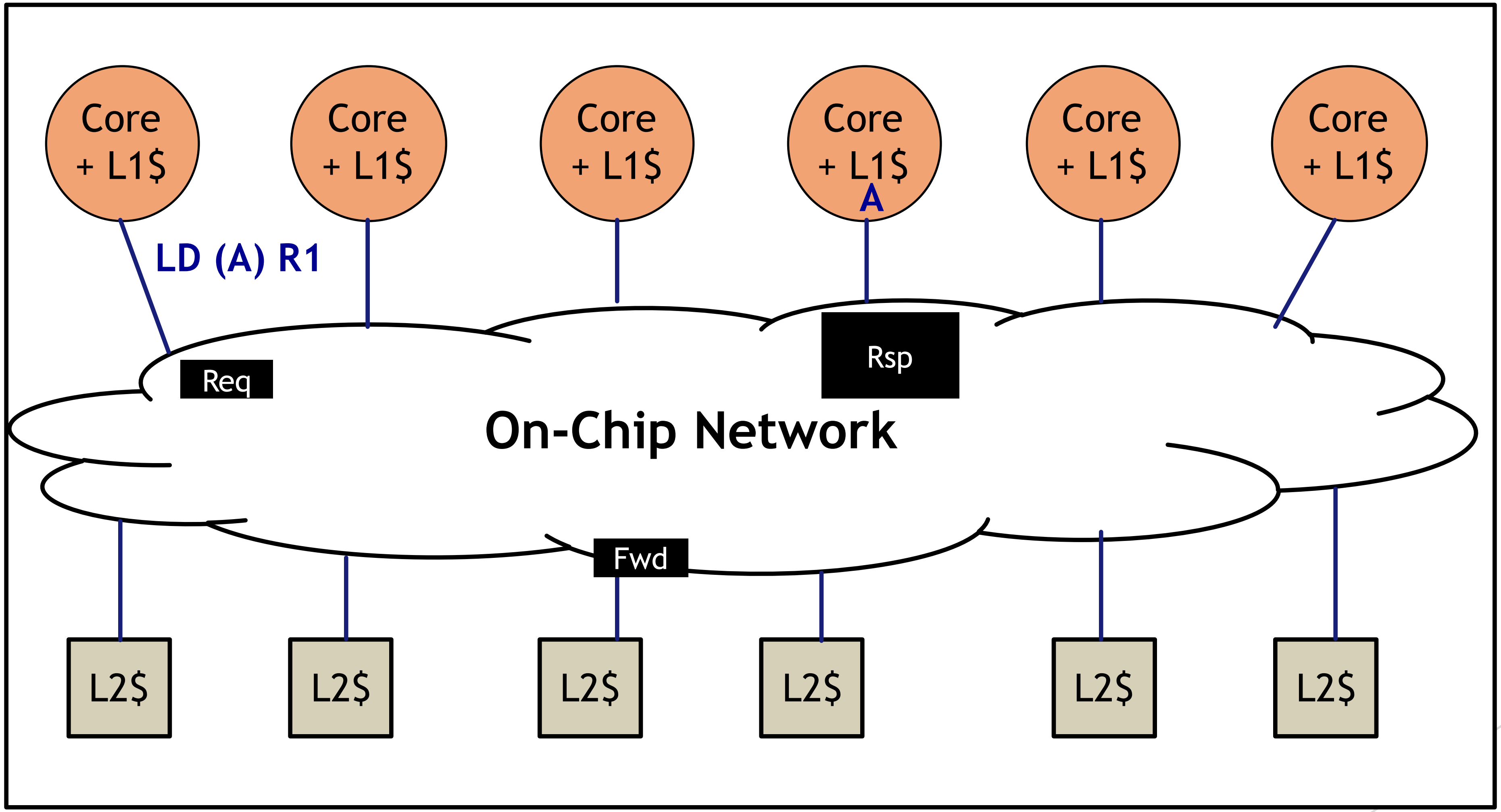
This monstrous CPU is 100 times more power-efficient than today's laptops.



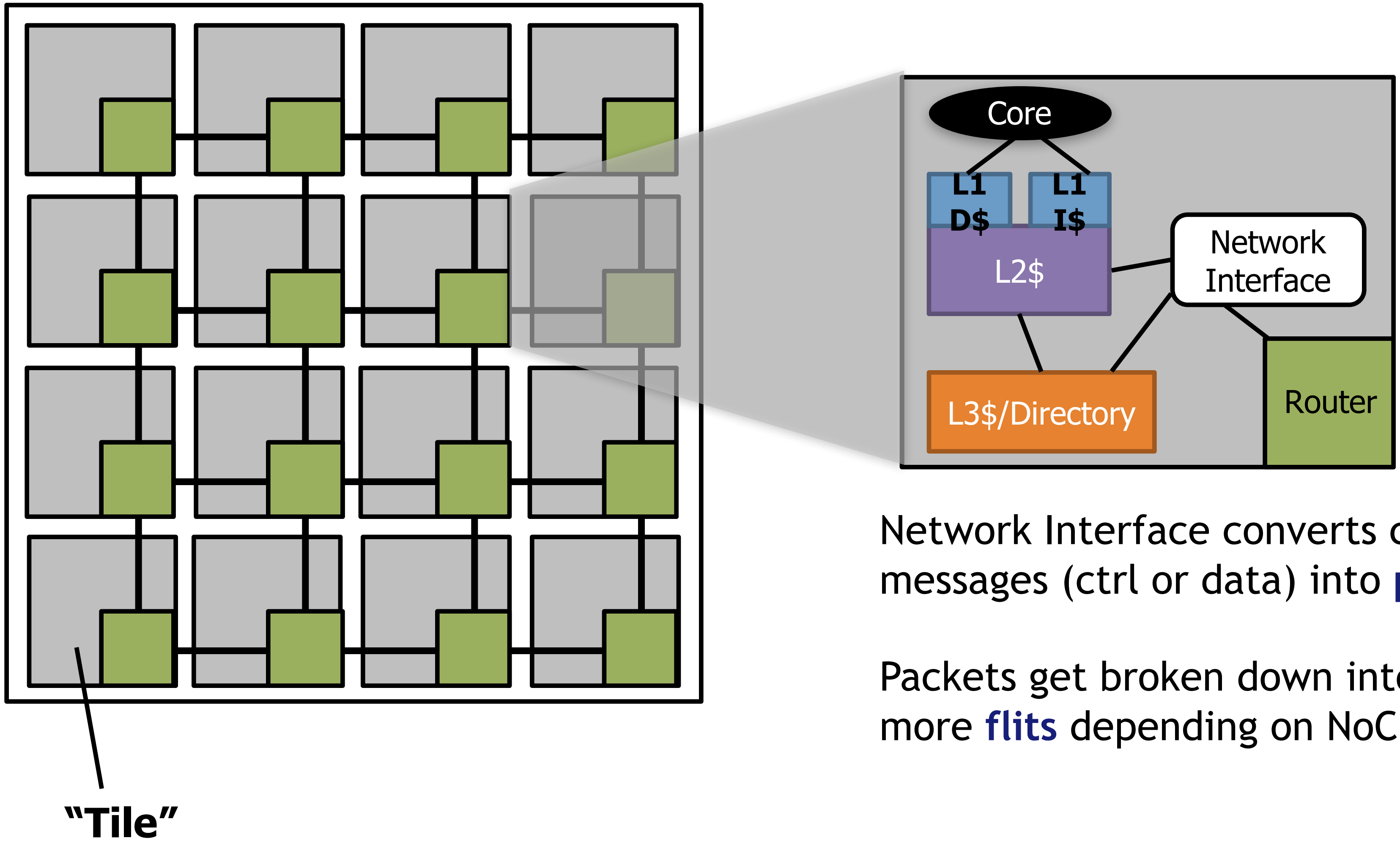
IBM pushes silicon photonics with on-chip optics

Big Blue researchers have figured out how to use standard manufacturing processes to make chips with built-in optical links that can transfer 25 gigabits of data per second.

Introduction to NoCs



Modern NoCs



Network Interface converts cache messages (ctrl or data) into **packets**.

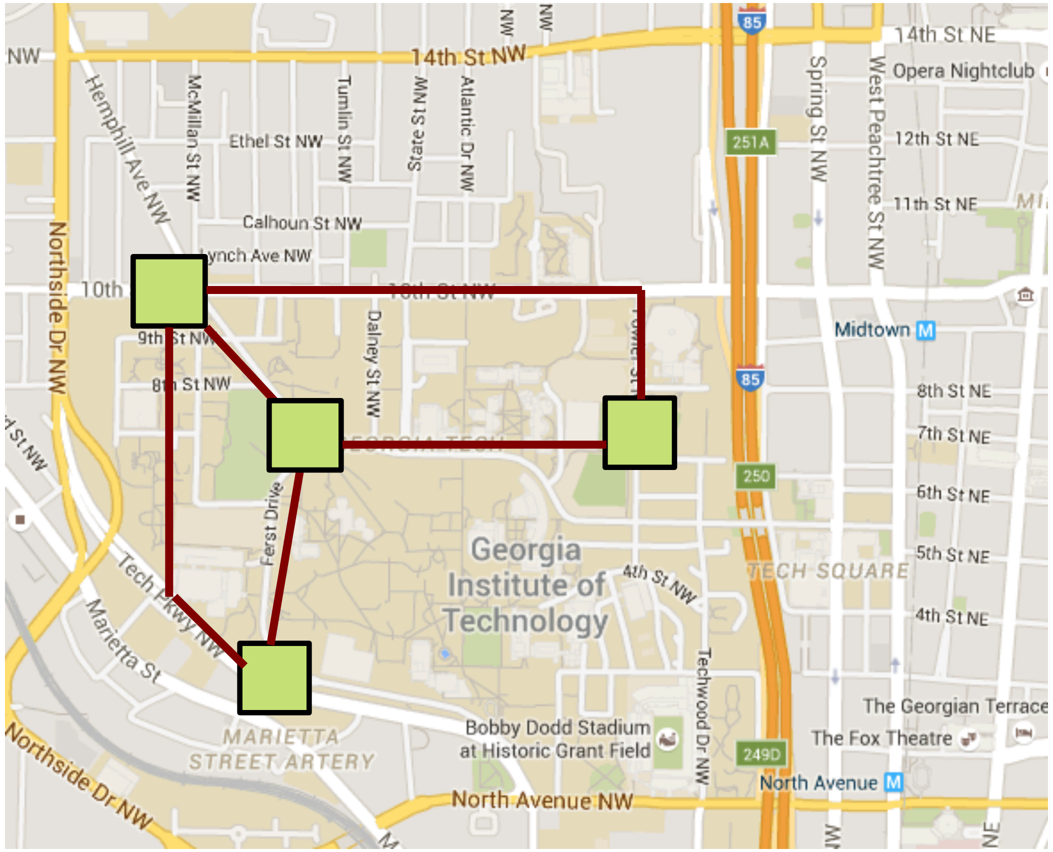
Packets get broken down into one or more **flits** depending on NoC link width

Network Architecture

- Topology
- Routing
- Flow Control
- Router Microarchitecture

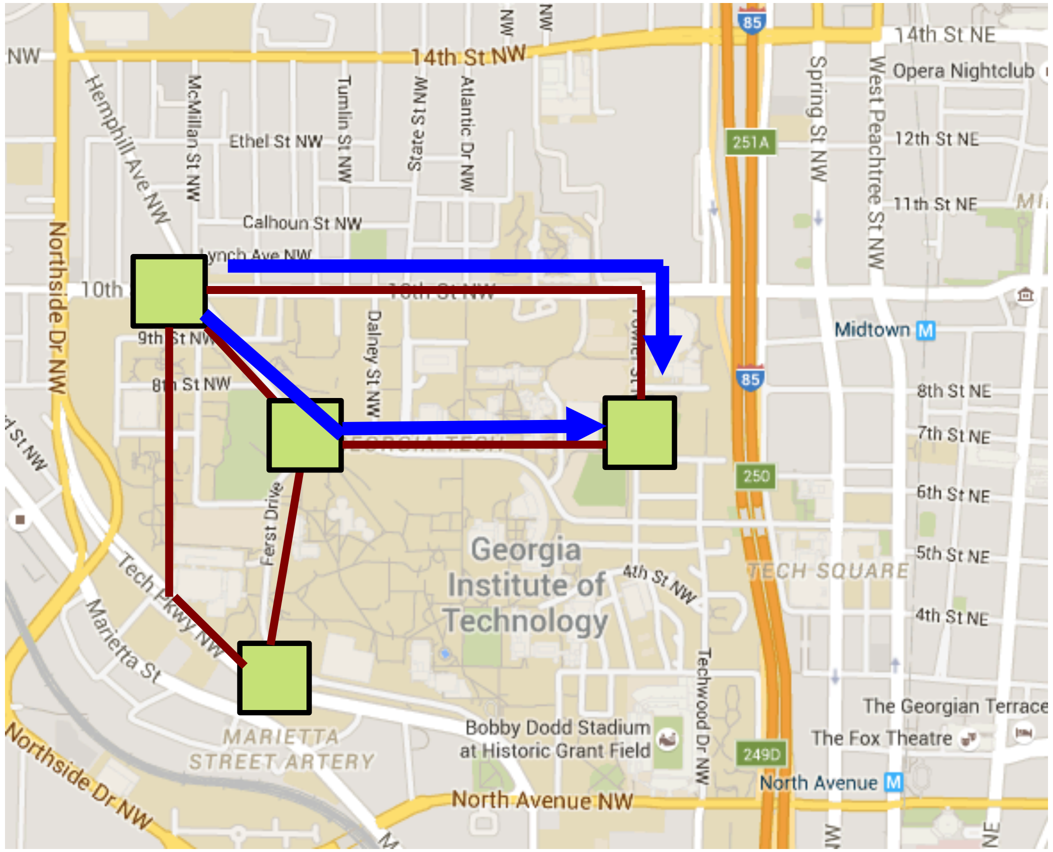
Topology: How to connect nodes with links

~Road Network



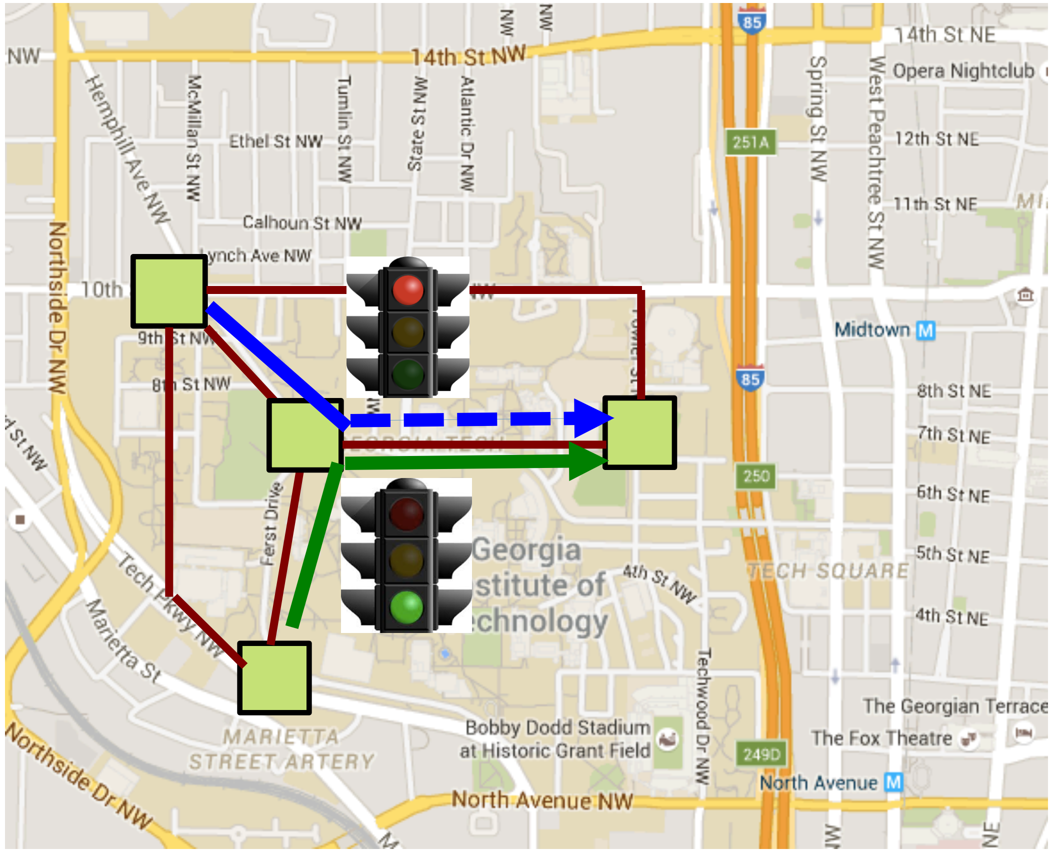
Routing: Which path should a message take

~Series of road segments from source to destination



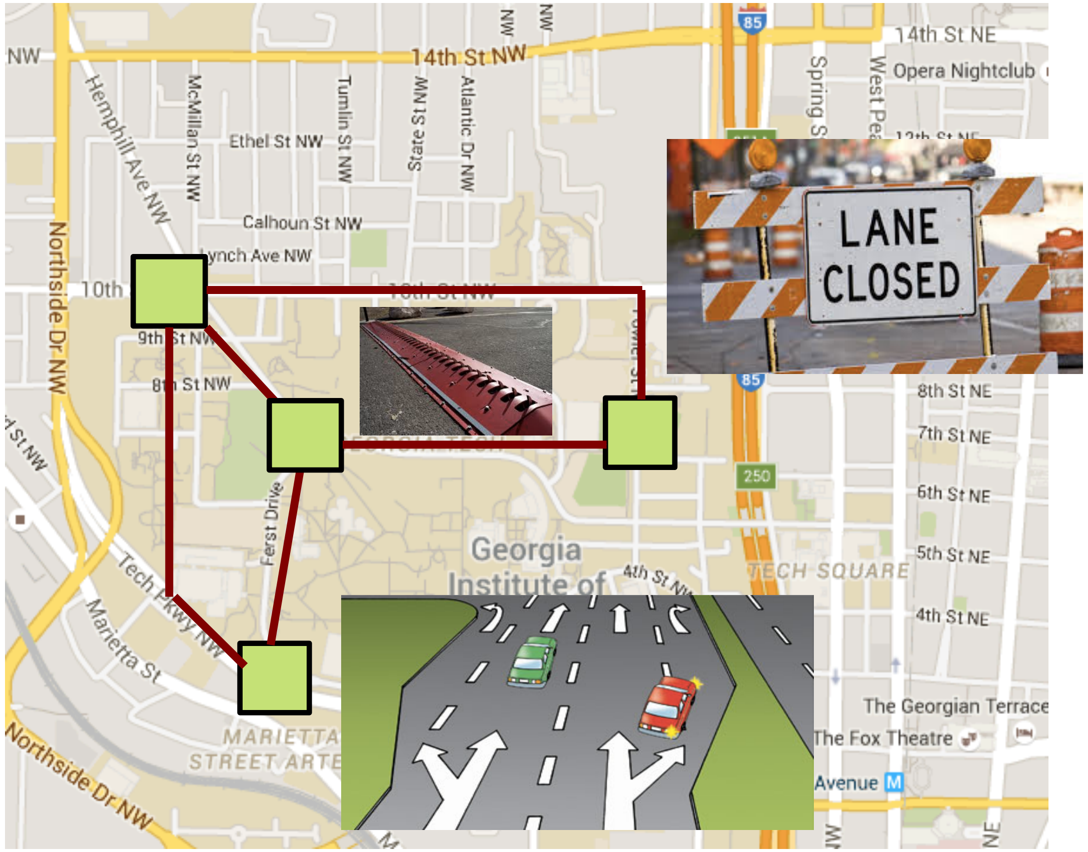
Flow Control: When does a message stop/proceed

~Traffic Signals / Stop signs at end of each road segment


14th St NW
Northside Dr NW
Georgia Institute of Technology
Midtown
Tech Square
North Avenue NW
Bobby Dodd Stadium at Historic Grant Field

Router Microarchitecture: How to build the routers

~Design of traffic intersection (number of lanes, algorithm for turning red/green)

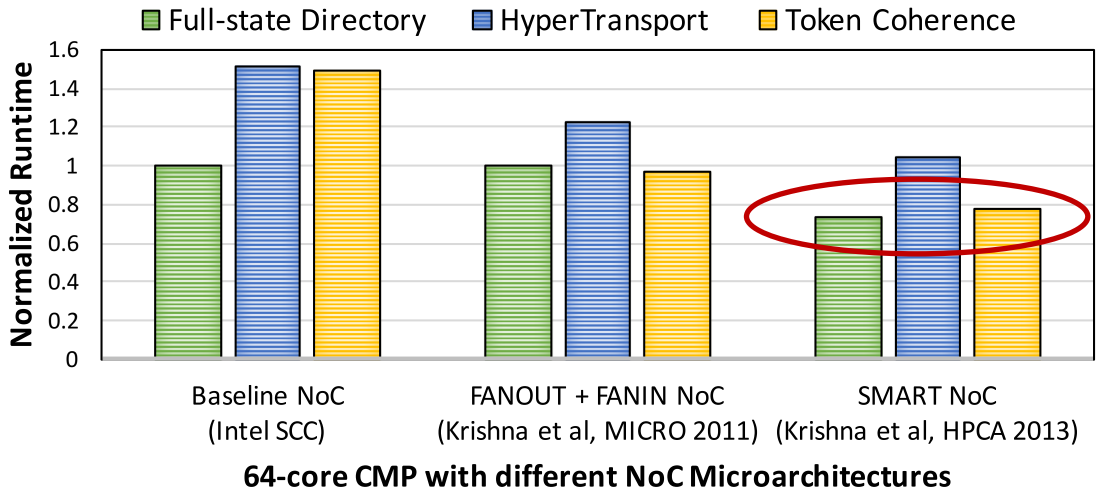


Overview

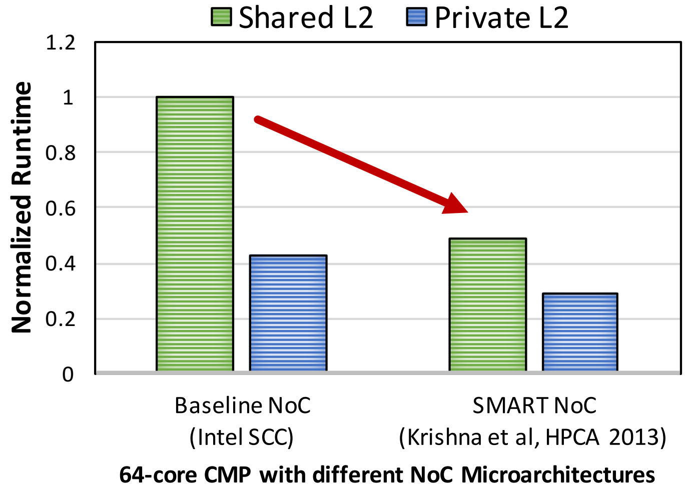
- What are Networks-On-Chip (NoCs)?
- **Why model NoCs accurately**
- Garnet2.0
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - System Integration
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - Output Stats
- Extensions and FAQs

Why model NoCs accurately?

Case Study I: Directory vs. Broadcast Protocols



Case Study II: Private vs. Shared L2



Different NoC Microarchitectures may lead to different microarchitectural decisions and new design optimization opportunities

64-core CMP running PARSEC workloads in full-system gem5. Average runtime plotted.

Overview

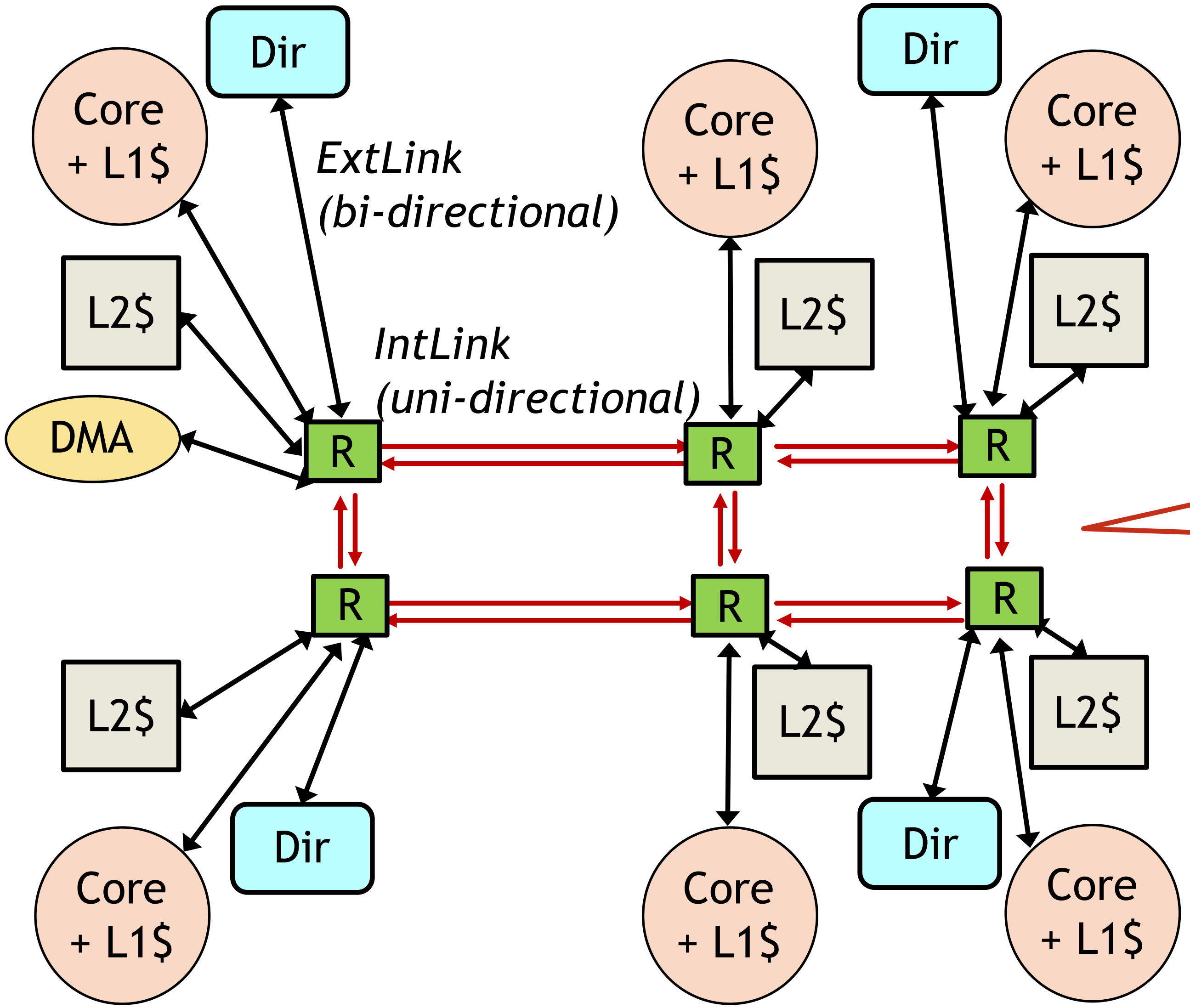
- What are Networks-On-Chip (NoCs)?
- Why model NoCs accurately
- **Garnet2.0**
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - System Integration
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - Output Stats
- Extensions and FAQs

Garnet2.0

- Detailed NoC Model
  - Currently part of Ruby Memory System in gem5
  - Original version (5-stage pipeline) released in 2009
    - Developed by Niket Agarwal (currently at Google) and myself
  - New version (1-stage pipeline, more configurability) released in 2016
- Resources
  - Source: `src/mem/ruby/network/garnet2.0`
  - gem5 wiki page: www.gem5.org/garnet2.0
  - Dev patches + practice labs: http://synergy.ece.gatech.edu/garnet

Topology

Each topology is a python file in `configs/topologies/`



Step 1: Instantiate routers and connect to controllers via “Ext Links”

Step 2: Instantiate “Int Links” and connect to routers in desired topology

This is an example of MeshDirCorners_XY.py

Topology Configurable Parameters

- **Router**
  - `router_latency` (in cycles)
    - Can be set per router
    - Defined in `src/mem/ruby/network/BasicRouter.py`
- **Link**
  - `link_latency` (in cycles)
    - Can be set per link
    - Defined in `src/mem/ruby/network/BasicLink.py`
  - `weight` (i.e., link weight)
    - To bias routing algorithm [later slides]
  - `src_outport` (string) and `dst_inport` (string)
    - Port direction (e.g., “East”)
    - Helps with readability of Config file + Adaptive routing algorithms
  - `bw_multiplier` (value)
    - Used by Ruby’s simple network model, NOT by Garnet
    - Link bandwidth is set inside Garnet via the `ni_flit_size` parameter [later slides]

Default Topologies Supported

- Pt2Pt
- Crossbar
- Mesh
  - Mesh_XY
  - Mesh_westfirst
  - MeshDirCorners_XY
- Cluster

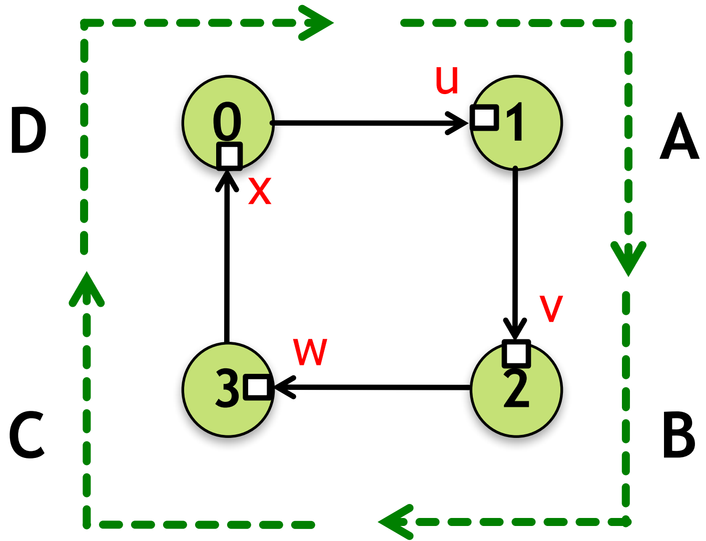
Routing

- Routing table
  - Automatically populated based on topology
  - All messages use shortest path
    - In case of multiple options, the path with the smaller weight is chosen
  - Deterministic Routing

- Custom
  - Users can leverage outport/inport direction names associated with each port to implement custom algorithms (say adaptive)

Deadlock Avoidance

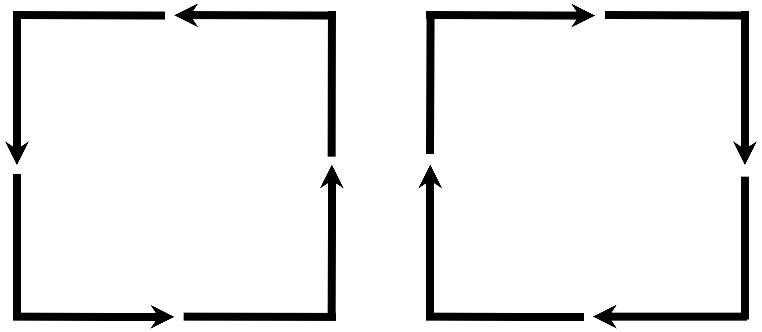
- **Deadlock:** A condition in which a set of **agents** wait indefinitely trying to acquire a set of **resources**



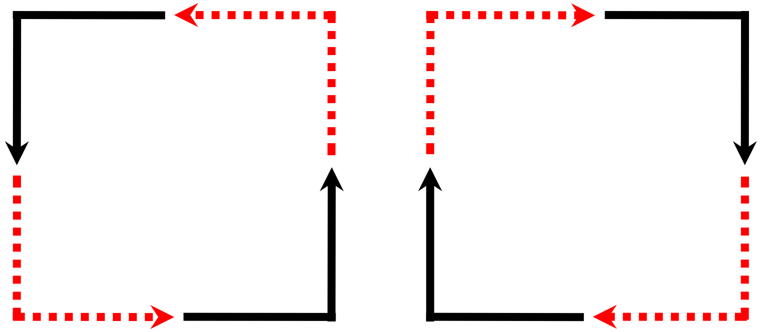
- Packet A holds buffer u (in 1) and wants buffer v (in 2)
- Packet B holds buffer v (in 2) and wants buffer w (in 3)
- Packet C holds buffer w (in 3) and wants buffer x (in 0)
- Packet D holds buffer x (in 0) and wants buffer u (in 1)

Deadlock-free Routing Algorithms

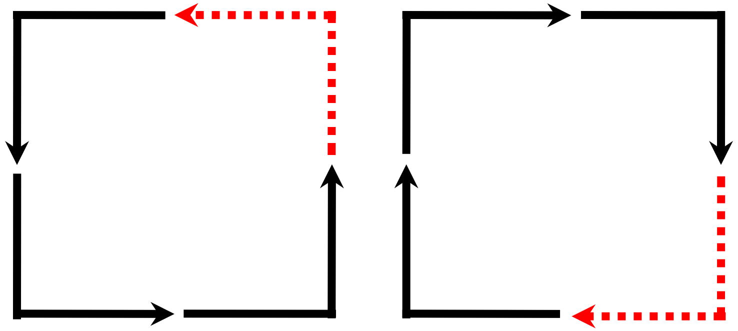
- Deadlocks may occur if the turns taken form a cycle



  - Removing some turns can make the routing algorithm deadlock free



XY Model

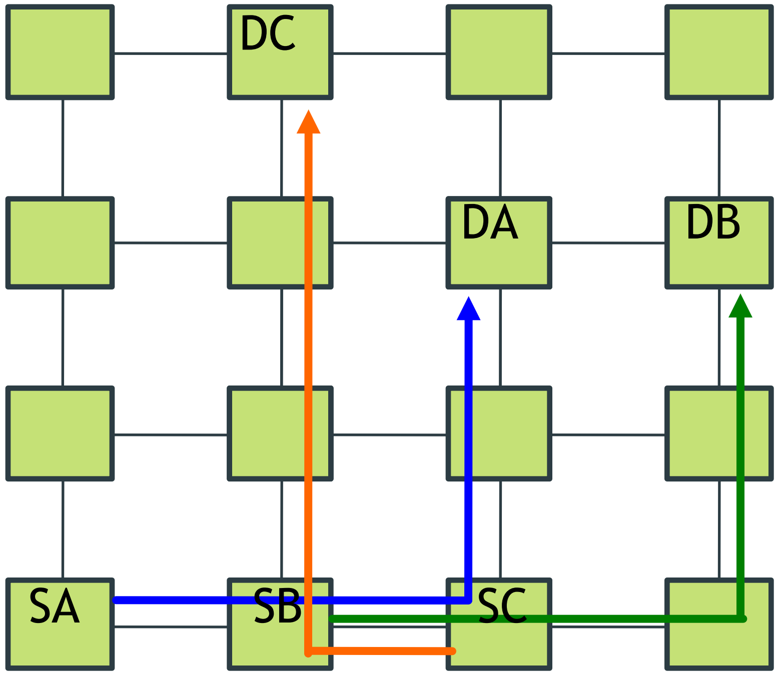


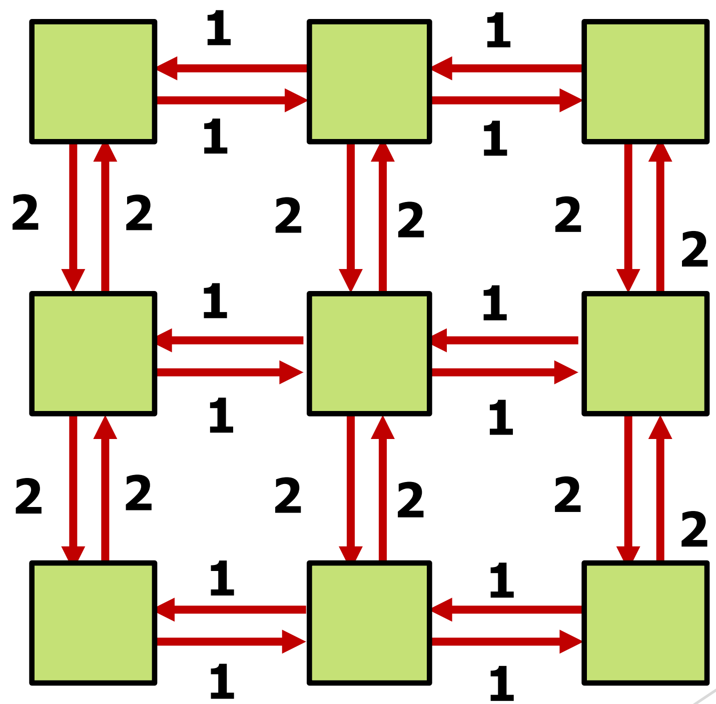
West-First Turn Model

Deadlock-free Routing Algorithms

- Assign weights to bias which links used first (to ensure no cyclic dependence)

XY Routing: Always go X first, then Y



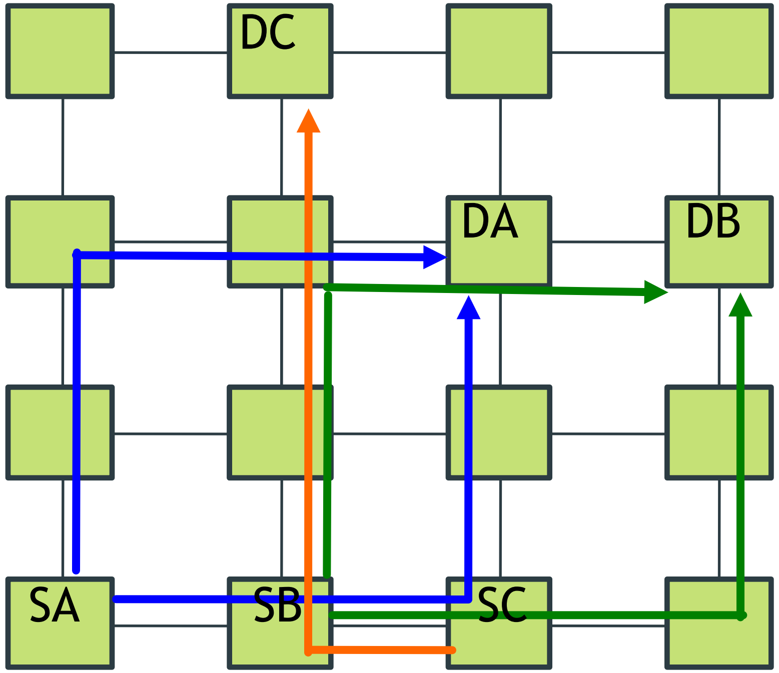


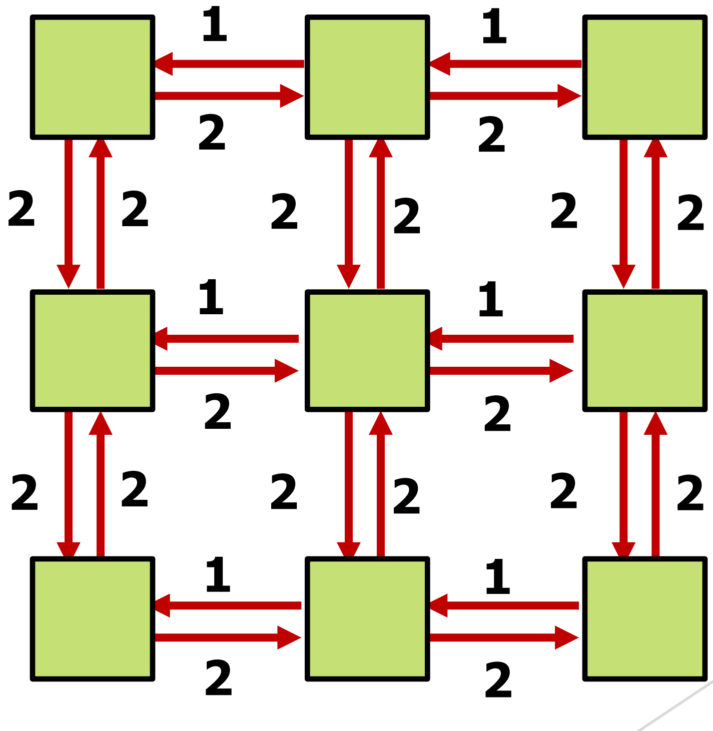
Mesh_XY

Deadlock-free Routing Algorithms

- Assign weights to bias which links used first (to ensure no cyclic dependence)

West-first Routing: Go W first, then N/S



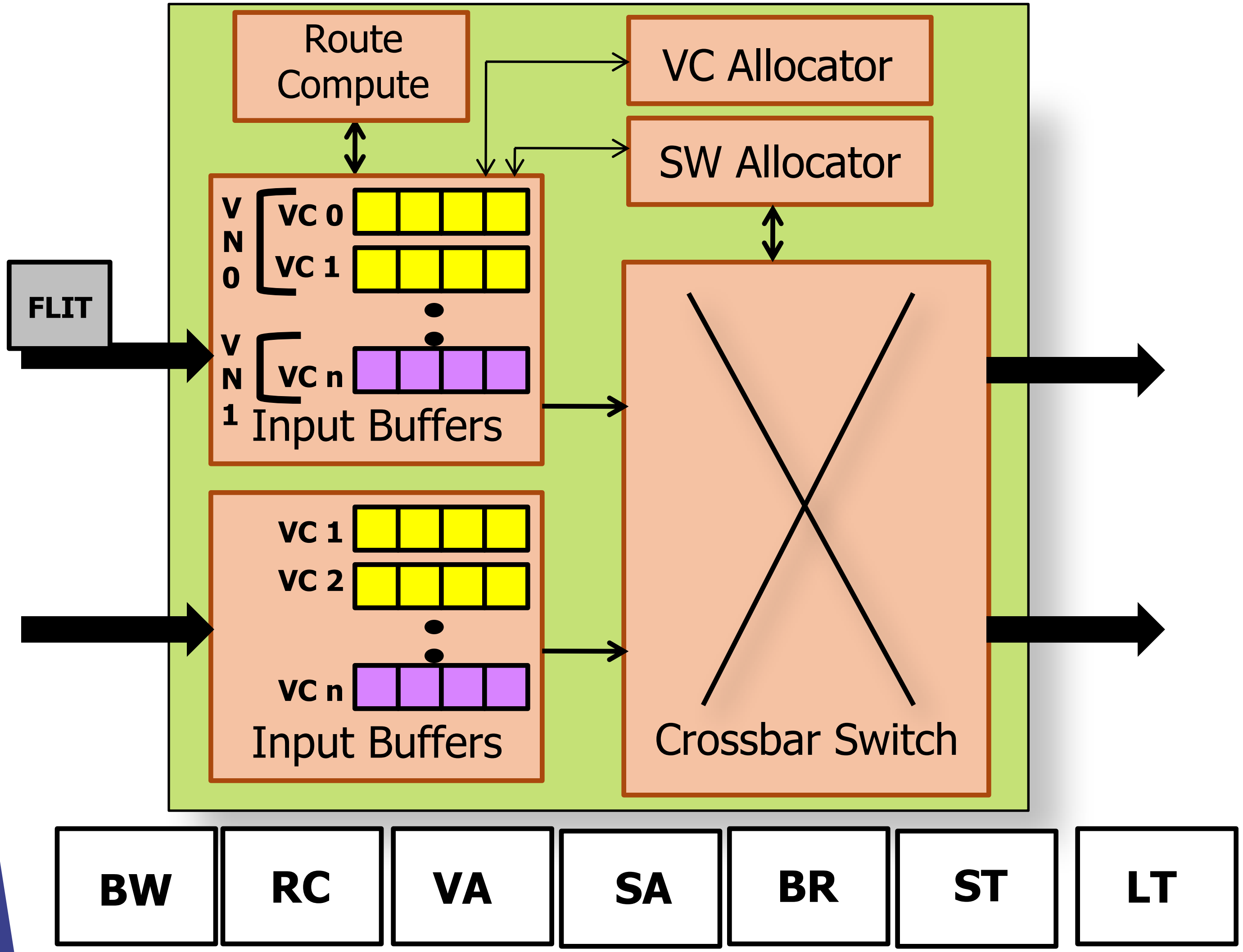


Mesh_westfirst

Flow Control

- **Virtual Channels**
  - Coherence Protocol requires certain number of virtual networks / message classes to avoid protocol deadlocks
    - This is the minimum number of VCs required
  - Within each vnet, there can be more than one VC for boosting network performance
    - In Garnet, only one packet can use a VC inside a router at a time
    - VCs in vnets carrying control messages are 1-flit deep
    - VCs in vnets carrying data (cacheline) messages are 4-5 flit deep
- **Credits**
  - Each VC conveys its buffer availability by sending credits to its upstream router

Conventional VC Router Microarch



BW: Buffer Write

RC: Route Compute

VA: VC Allocation
Input VCs arbitrate for "output" VCs (Input VCs at next router)

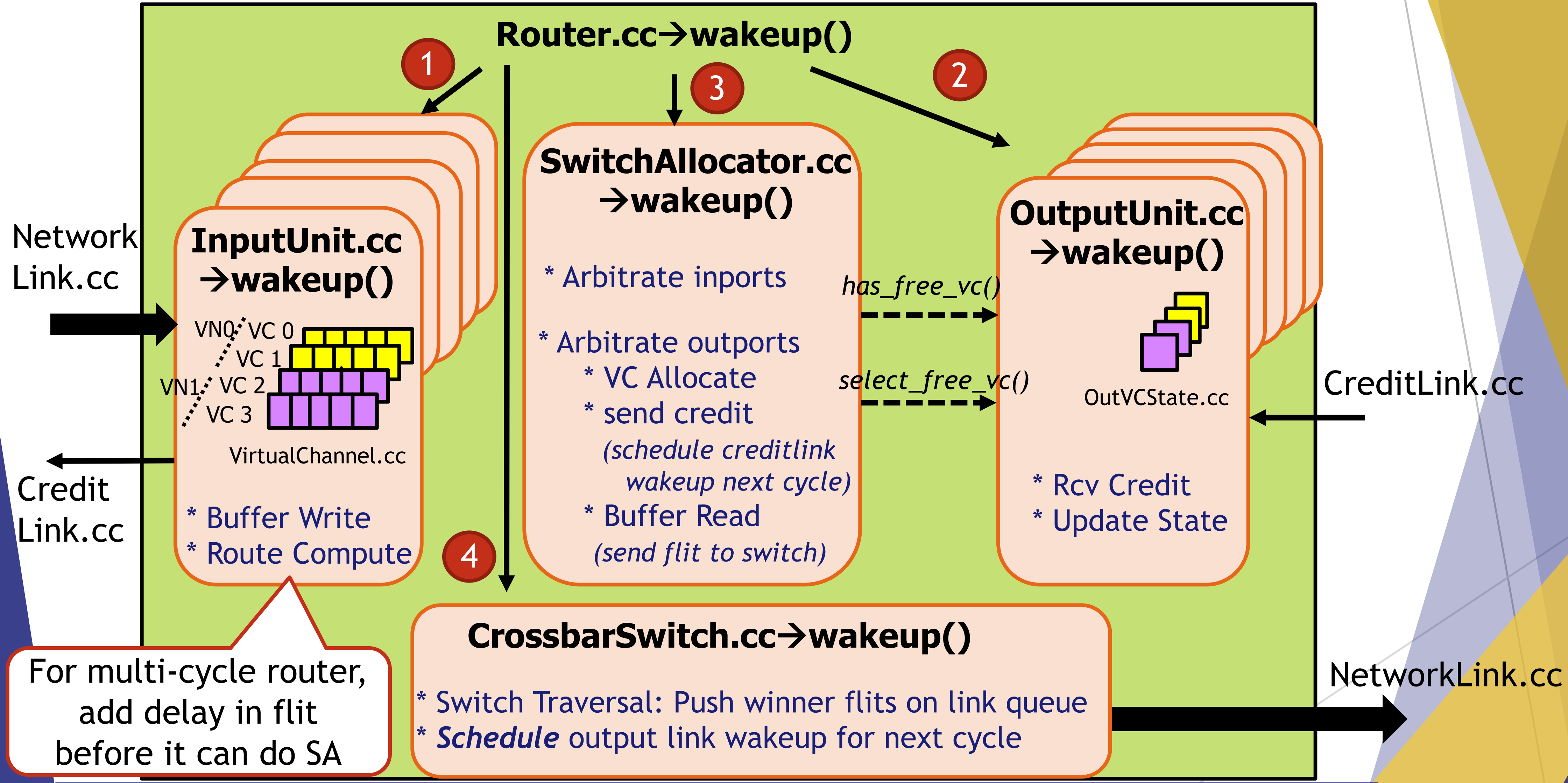
SA: Switch Allocation
Input ports arbitrate for output ports

BR: Buffer Read

ST: Switch Traversal

LT: Link Traversal

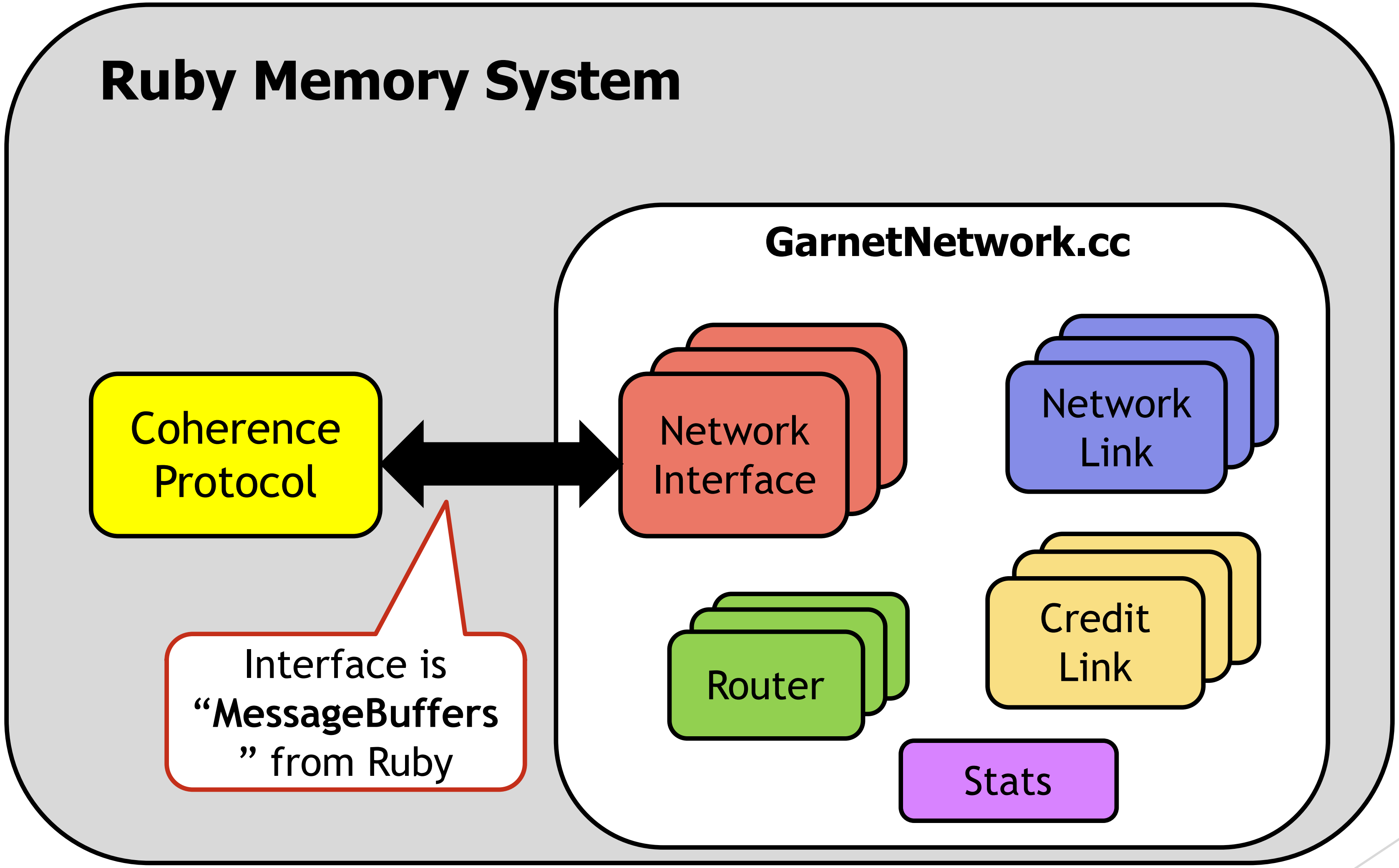
Single-Cycle Router Implementation



Overview

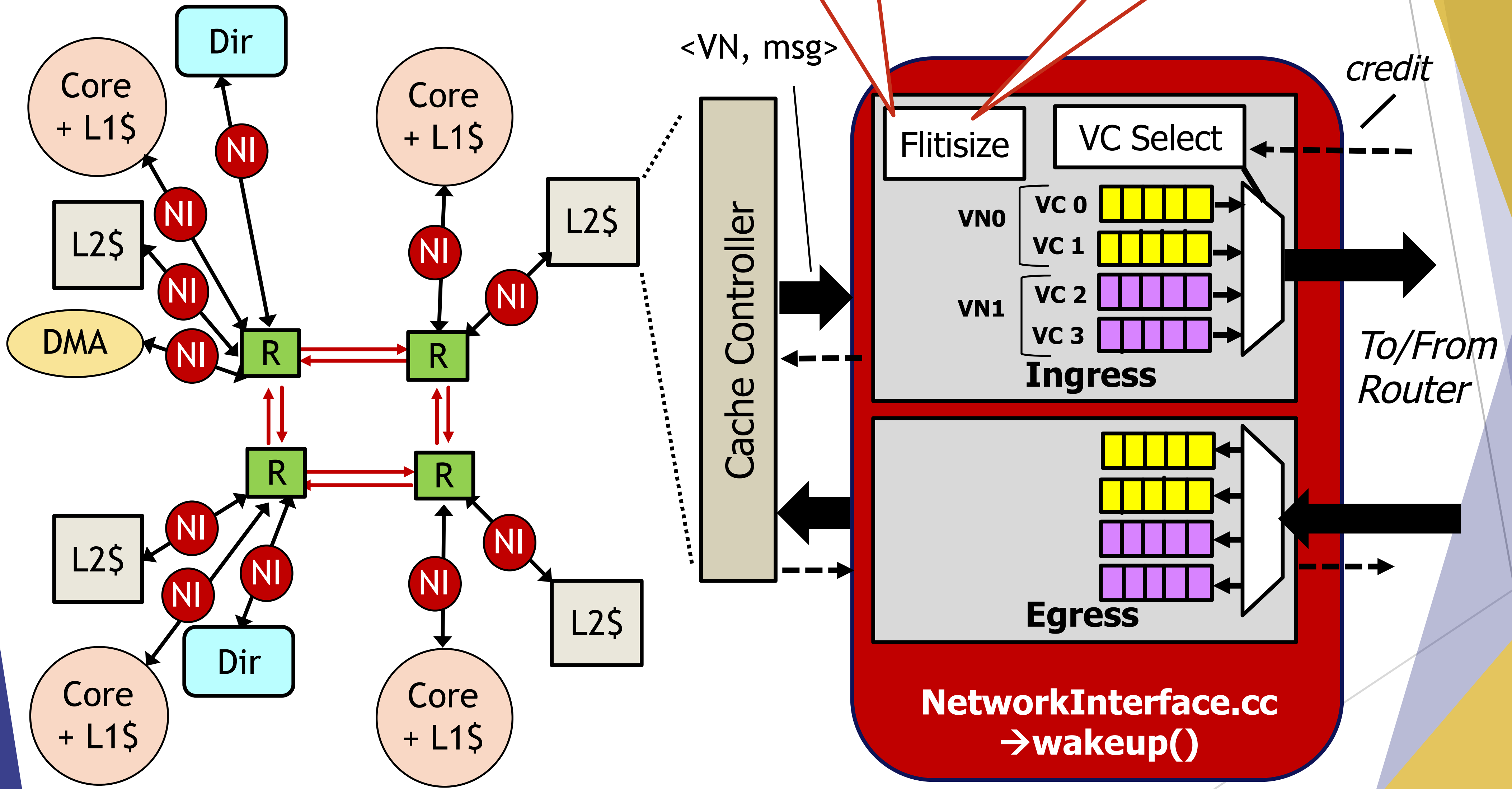
- What are Networks-On-Chip (NoCs)?
- Why model NoCs accurately
- **Garnet2.0**
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - **System Integration**
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - Output Stats
- Extensions and FAQs

System Organization



Interface is "**MessageBuffers**" from Ruby

Network Interface



Msg_size decides number of flits

Can add additional info in flit

NetworkInterface.cc →wakeup()

Garnet Configurable Parameters

- `ni_flit_size`
  - Default = 16B (128b) → 1-flit ctrl, 5-flit data
  - This sets the bandwidth of each physical link
- `vcs_per_vnet`
  - Total VCs in each inport = num_vnets * vcs_per_vnet
- `buffers_per_data_vc`
  - Default = 4
- `buffers_per_ctrl_vc`
  - Default = 1
- `routing_algorithm`
  - Weight-based table or custom
- **Defined in:** `src/mem/ruby/network/garnet2.0/GarnetNetwork.py`

Command Line Parameters

src/mem/ruby/network/

```
BasicRouter.py          Network.py
BasicLink.py            garnet2.0/GarnetNetwork.py
```

Definitions and Default Values

configs/

```
ruby/Ruby.py            common/Options.py
network/Network.py      example/garnet_synth_test.py
```

- Overrides default values in Ruby
- All parameters in in these .py files can be specified from command line

Running Garnet with Ruby

- Build Ruby Coherence Protocol

```
scons build/X86_MOESI_hammer/gem5.opt PROTOCOL=MOESI_hammer
```

  - Protocol determines number of message classes/virtual networks required

- Invoke garnet2.0 from command line with appropriate network parameters

```
 ./build/X86_MOESI_hammer/gem5.opt configs/example/fs.py
   --network=garnet2.0 --topology=Mesh_XY ...
```

Running Garnet Standalone

- Build the *Garnet_standalone* protocol

```
scons build/NULL/gem5.debug PROTOCOL=Garnet_Standalone
```

  - Dummy protocol just for traffic injection via Garnet Synthetic Traffic tester (next slide)
  - 3 Virtual Networks: vnet 0 and vnet 1 inject ctrl (1-flit) packets, vnet 2 injects data (5-flit) packets

- Run user-specified synthetic traffic

```
./build/NULL/gem5.debug
 configs/example/garnet_synth_traffic.py
  --network=garnet2.0 --topology=...  \
  --synthetic=uniform_random \
  --injectionrate=0.01 \
  --num-packets-max=100 \
```

Garnet Synthetic Traffic

- Dummy CPU Model [only works with *Garnet_standalone* protocol]
  - `src/cpu/testers/garnet_synthetic_traffic`
  - Inject packets for user-specified traffic pattern at user-specified injection rate
    - uniform_random
    - tornado
    - bit_complement
    - bit_reverse
    - bit_rotation
    - neighbor
    - shuffle
    - transpose
  - Ability to inject continuously or fixed number of packets, and/or for fixed time, and/or from fixed source and/or to fixed destination in one/more vnets
    - Heavy customization very useful for debugging
  - All parameters described here: http://www.gem5.org/Garnet_Synthetic_Traffic

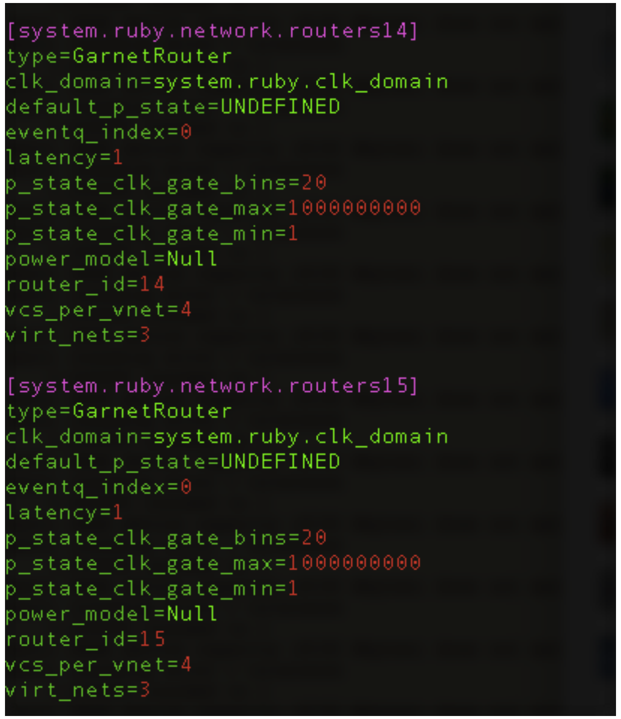
Overview

- What are Networks-On-Chip (NoCs)?
- Why model NoCs accurately
- **Garnet2.0**
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - System Integration
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - **Output Stats**
- Extensions and FAQs

Sample Simulation

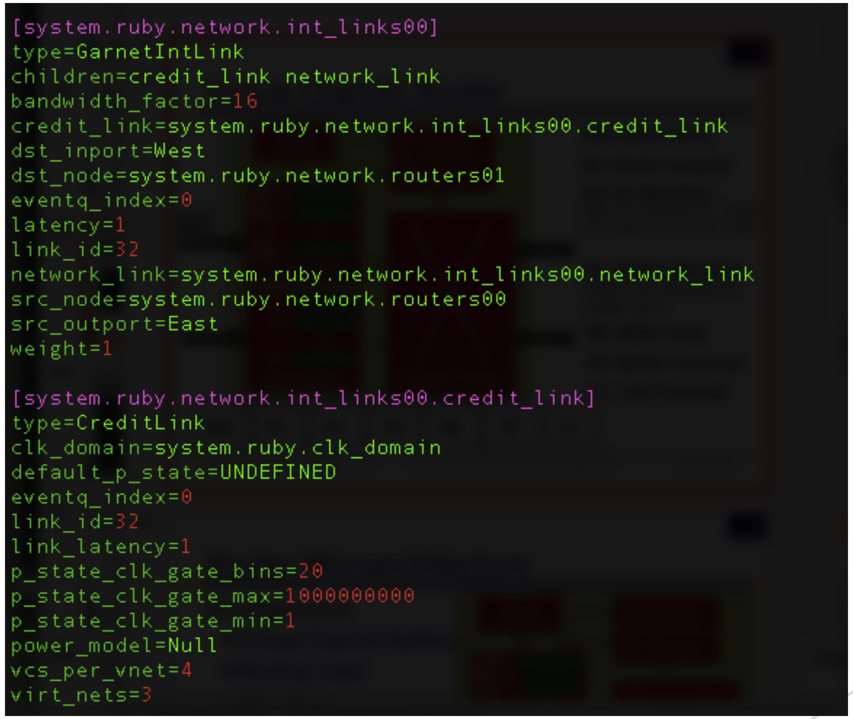
```
./build/NULL/gem5.debug configs/example/garnet_synth_traffic.py
--topology=Mesh_XY --num-cpus=16 --num-dirs=16 --mesh-rows=4
```

m5out/config.ini



```
[system.ruby.network.routers14]
type=GarnetRouter
clk_domain=system.ruby.clk_domain
default_p_state=UNDEFINED
eventq_index=0
latency=1
p_state_clk_gate_bins=20
p_state_clk_gate_max=1000000000
p_state_clk_gate_min=1
power_model=Null
router_id=14
vcs_per_vnet=4
virt_nets=3

[system.ruby.network.routers15]
type=GarnetRouter
clk_domain=system.ruby.clk_domain
default_p_state=UNDEFINED
eventq_index=0
latency=1
p_state_clk_gate_bins=20
p_state_clk_gate_max=1000000000
p_state_clk_gate_min=1
power_model=Null
router_id=15
vcs_per_vnet=4
virt_nets=3
```

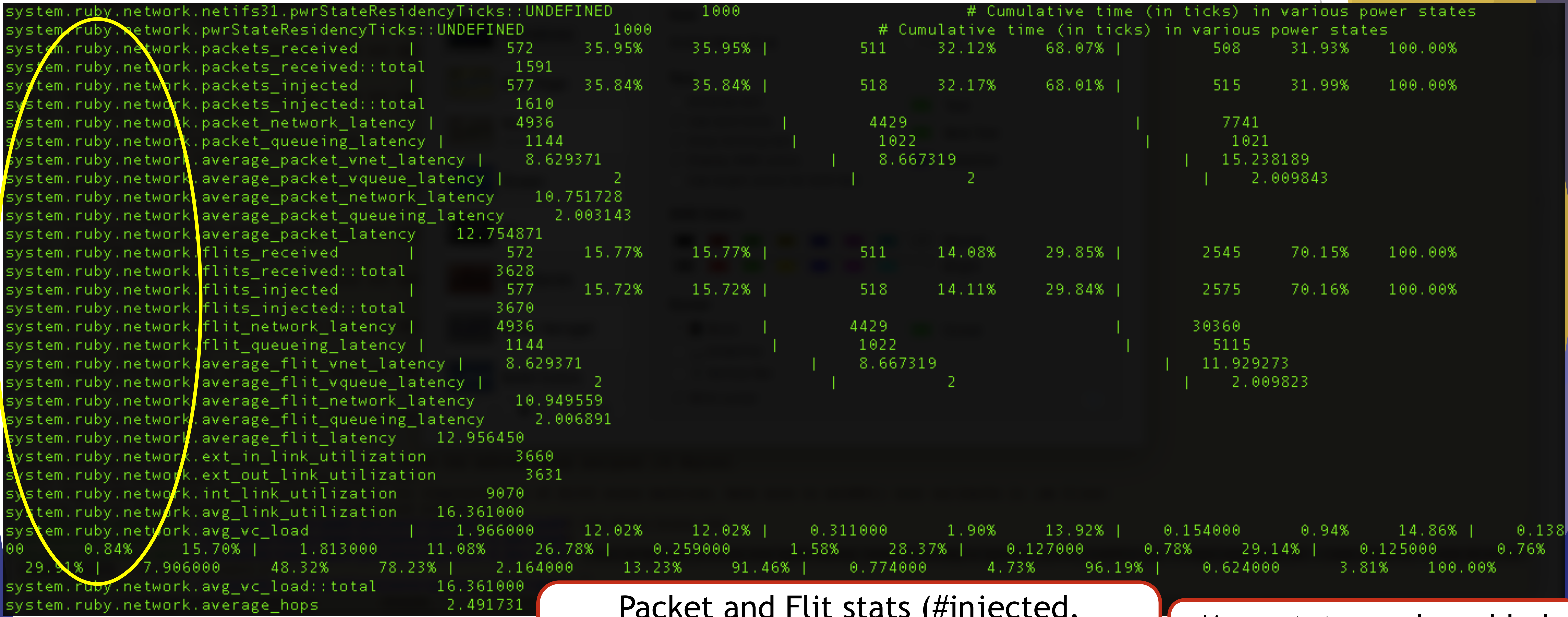


```
[system.ruby.network.int_links00]
type=GarnetIntLink
children=credit_link network_link
bandwidth_factor=16
credit_link=system.ruby.network.int_links00.credit_link
dst_inport=West
dst_node=system.ruby.network.routers01
eventq_index=0
latency=1
link_id=32
network_link=system.ruby.network.int_links00.network_link
src_node=system.ruby.network.routers00
src_outport=East
weight=1

[system.ruby.network.int_links00.credit_link]
type=CreditLink
clk_domain=system.ruby.clk_domain
default_p_state=UNDEFINED
eventq_index=0
link_id=32
link_latency=1
p_state_clk_gate_bins=20
p_state_clk_gate_max=1000000000
p_state_clk_gate_min=1
power_model=Null
vcs_per_vnet=4
virt_nets=3
```

Output Stats

m5out/stats.txt



```
system.ruby.network.netifs31.pwrStateResidencyTicks::UNDEFINED         1000                       # Cumulative time (in ticks) in various power states
system.ruby.network.pwrStateResidencyTicks::UNDEFINED         1000                       # Cumulative time (in ticks) in various power states
system.ruby.network.packets_received         |         572     35.95%     35.95% |         511     32.12%     68.07% |         508     31.93%    100.00%
system.ruby.network.packets_received::total          1591
system.ruby.network.packets_injected         |         577     35.84%     35.84% |         518     32.17%     68.01% |         515     31.99%    100.00%
system.ruby.network.packets_injected::total          1610
system.ruby.network.packet_network_latency |         4936                       |        4429                       |        7741
system.ruby.network.packet_queueing_latency |         1144                       |        1022                       |        1021
system.ruby.network.average_packet_vnet_latency |     8.629371                       |     8.667319                       |    15.238189
system.ruby.network.average_packet_vqueue_latency |            2                       |            2                       |     2.009843
system.ruby.network.average_packet_network_latency     10.751728
system.ruby.network.average_packet_queueing_latency     2.003143
system.ruby.network.average_packet_latency    12.754871
system.ruby.network.flits_received         |         572     15.77%     15.77% |         511     14.08%     29.85% |        2545     70.15%    100.00%
system.ruby.network.flits_received::total         3628
system.ruby.network.flits_injected         |         577     15.72%     15.72% |         518     14.11%     29.84% |        2575     70.16%    100.00%
system.ruby.network.flits_injected::total         3670
system.ruby.network.flit_network_latency |         4936                       |        4429                       |       30360
system.ruby.network.flit_queueing_latency |         1144                       |        1022                       |        5115
system.ruby.network.average_flit_vnet_latency |     8.629371                       |     8.667319                       |    11.929273
system.ruby.network.average_flit_vqueue_latency |            2                       |            2                       |     2.009823
system.ruby.network.average_flit_network_latency     10.949559
system.ruby.network.average_flit_queueing_latency     2.006891
system.ruby.network.average_flit_latency    12.956450
system.ruby.network.ext_in_link_utilization          3660
system.ruby.network.ext_out_link_utilization          3631
system.ruby.network.int_link_utilization         9070
system.ruby.network.avg_link_utilization    16.361000
system.ruby.network.avg_vc_load         |     1.966000     12.02%     12.02% |     0.311000      1.90%     13.92% |     0.154000      0.94%     14.86% |     0.138
00      0.84%     15.70% |     1.813000     11.08%     26.78% |     0.259000      1.58%     28.37% |     0.127000      0.78%     29.14% |     0.125000      0.76%
  29.91% |     7.906000     48.32%     78.23% |     2.164000     13.23%     91.46% |     0.774000      4.73%     96.19% |     0.624000      3.81%    100.00%
system.ruby.network.avg_vc_load::total    16.361000
system.ruby.network.average_hops          2.491731
```

Packet and Flit stats (#injected, #received, queueing latency at NIC, and network latency) per VN and overall

More stats can be added via GarnetNetwork.cc

Overview

- What are Networks-On-Chip (NoCs)?
- Why model NoCs accurately
- Garnet2.0
  - Configuration
    - Topology
    - Routing
    - Flow-Control
    - Router Microarchitecture
  - System Integration
    - Network Interface
    - Network Parameters
    - Running Garnet with Ruby
    - Ruby Garnet Standalone
  - Output Stats
- **Extensions and FAQs**

Extensions and FAQs

- **System Level Modeling**
  - **How can I integrate Garnet into my own simulator (or with the gem5 classic memory system)?**
    - *This should not be too hard - would just require some changes to the NI code on how it receives its inputs. If anyone wants to try it, I would love to give pointers.*
  - **How do I print a network trace?**
    - *You can add some code to the NI. I have a patch on my website for reference.*
  - **Can garnet read a network trace?**
    - *You can run Garnet in a standalone mode, and have it inject traffic from a trace instead of a fixed synthetic pattern. Alternately, try to use cpu/testers/traffic_gen*
  - **Does Garnet report NoC area and power numbers?**
    - *The output of Garnet can be fed into DSENT (Sun et al, NOCS 2012) which is present in `ext/dsent`. We will add an automatic stats.txt parser for it soon.*
  - **Can we model multiple clock domains?**
    - The entire Ruby memory system (including Garnet inside it) is one clock domain. To mimic a multi-clock domain design, you can schedule wakeups of slow routers intelligently at some multiples of the clock rather than every cycle.

Extensions and FAQs

- Topology
  - **How do we model multiple BW links in Garnet?**
    - *Inherently, that would require support in the router to manage multiple flit sizes. Instead, you can add multiple links between the same nodes if you want to model higher bandwidth. If they have the same weight, Garnet will randomly send over the two.*
  - **Can we model a heterogeneous CPU-GPU system?**
    - *Yes. The current AMD GPU model models a cluster of CPUs connected to a GPU.*
  - **Can we model indirect networks such as Clos?**
    - *Yes, there can be additional routers that are not connected to any controller and act purely as switches.*
  - **Can we model large-scale HPC networks?**
    - *Garnet can model any sized network. You can run 256-node standalone simulations easily. However, beyond that gem5 cannot instantiate more directories (which it uses as destination nodes). I have a patch to run 1024-node synthetic sims on my website. But these run quite slowly.*

Extensions and FAQs

- ## Routing
  - **How do we model an adaptive routing algorithm?**
    - *If you want to use internal NoC metrics (such as number of credits at an output port) for making routing decisions, do not use table-based routing. Instead, set the routing-algorithm to custom, and implement your own routing function inside RoutingUnit.cc. See outportComputeXY() for reference.*
- ## Flow Control
  - **Can we implement alternate deadlock avoidance schemes (such as escape VCs or dateline)?**
    - *You can update the vc selection scheme inside SwitchAllocator to control which VCs get allocated.*
- ## Microarchitecture
  - **Can we model variable number of VCs in each router?**
    - *Currently the codebase is very tied to the global vcs_per_vnet parameter. If you want to model variable number of VCs, one hack could be to have everyone instantiate the same number of VCs, but modify the VC select to never allocate certain VCs*

Conclusions

- Garnet2.0 is an open-source research vehicle
  - Use it and contribute to it!
  - Being actively maintained by the following people
    - Georgia Tech: Tushar Krishna
    - AMD Research: Brad Beckmann, Onur Kayiran, Jieming Yin, Matt Porembas
    - If you have any questions, email on gem5-users or gem5-dev mailing lists
  - Would love to have it integrated with the classic memory model

- Useful Resources:
  - www.gem5.org/Interconnection_Network
  - www.gem5.org/garnet2.0
  - http://synergy.ece.gatech.edu/garnet