dist-gem5 Architecture

Illinois: Mohammad Alian, Daehoon Kim, Prof. Nam Sung Kim
ARM: Gabor Dozsa, Stephan Diestelhorst, Nikos Nikoleris, Radhika Jagtap

Tutorial at International Symposium on Computer Architecture (ISCA), Toronto, Canada
25 June 2017

Distributed Computer Systems

- Definition
  - A cluster of computers that communicate and interact with each other by passing messages over the network to process given tasks.
- Examples
  - Datacenters, supercomputers

A Google datacenter

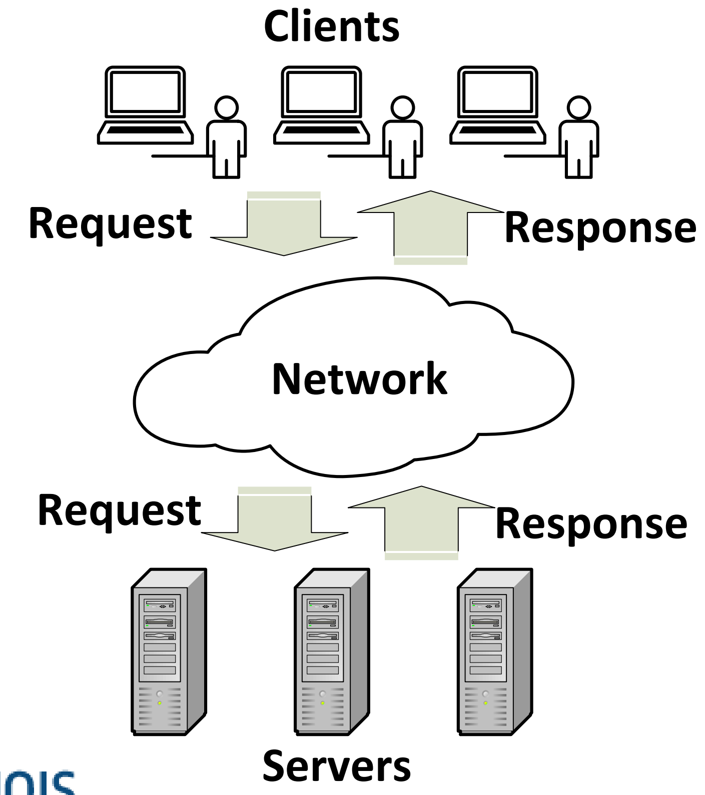
The IBM Blue Gene/P supercomputer "Intrepid" at Argonne National Laboratory runs 164,000 processor cores in 40 racks/cabinets connected by a high-speed 3-D torus network.

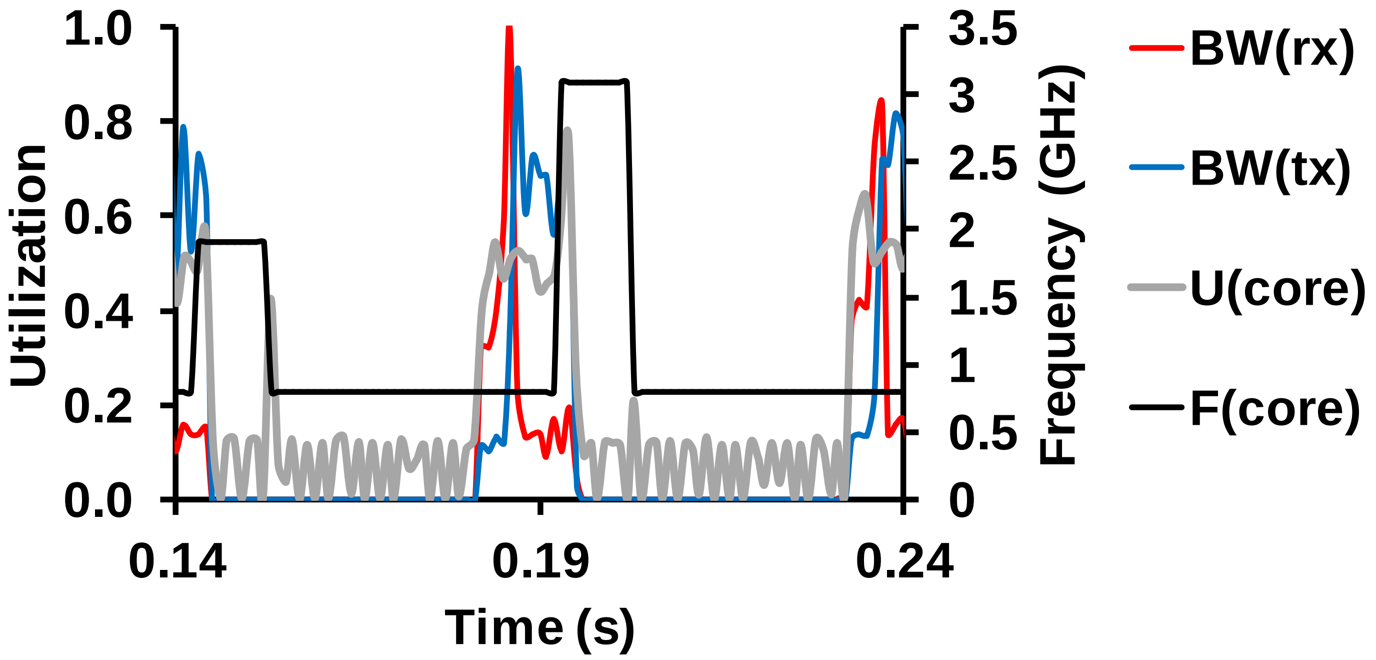
ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Exploring and Optimizing Distributed Computer Systems

- To maximize performance and/or energy-efficiency, we must capture the intricate interplay amongst computers and their HW/SW sub-systems, especially due to communications and interactions w/ each other by passing messages over the network





Past Methods Exploring Distributed Computer Systems [1]

Using physical computers

- Advantage
  - Fast evaluations for large-scale distributed computer systems
- Disadvantage
  - Limited design space exploration (unable to explore distributed computer systems based on future processor and computer sub-systems architectures that have not been developed yet)

Using queuing-theoretic models

- Advantage
  - Simple and fast evaluations for large-scale distributed computer systems
- Disadvantage
  - Inaccurate/misleading evaluations (unable to capture complex interplay b/w HW/SW sub-systems of computers)

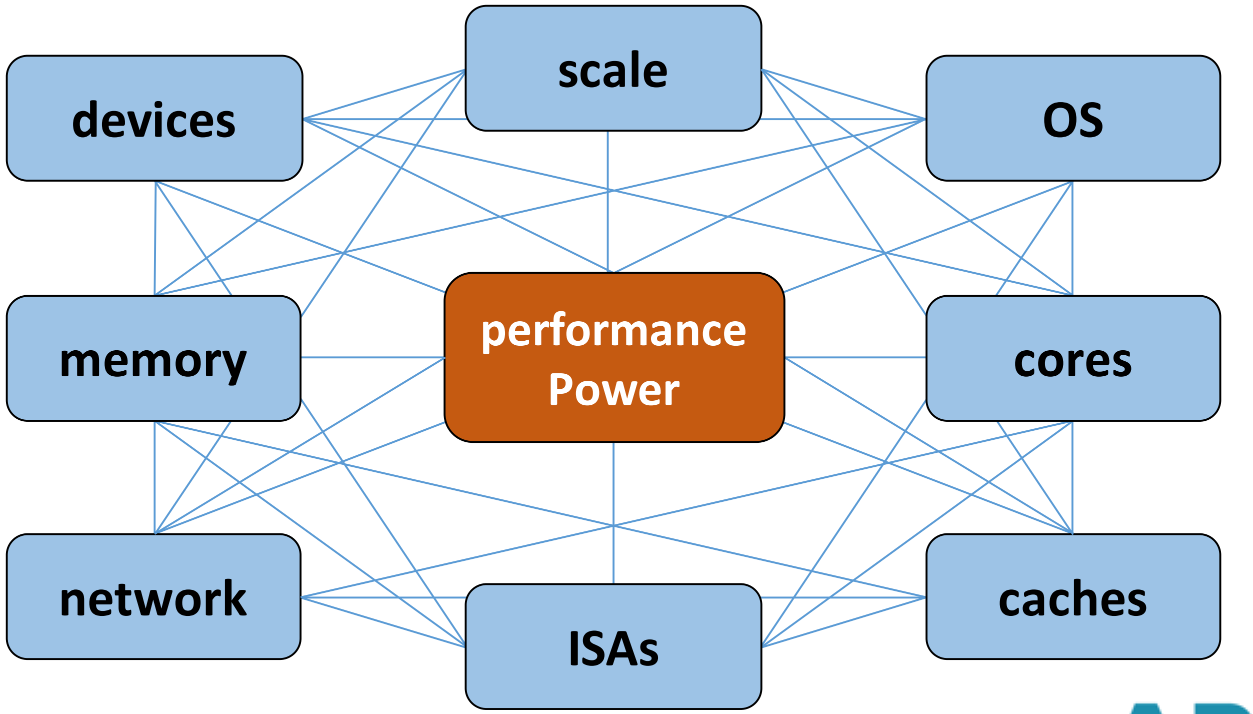
Past Methods Exploring Distributed Computer Systems [2]

Using existing (full-system) simulators

- Advantage
  - More flexible design space exploration than physical computer systems
  - More precise evaluation than queuing-theoretic models
- Disadvantage
  - gem5: limited scalability w/ slow evaluation (legacy gem5)
  - Not flexible (SST + gem5)
  - Proprietary and limited to x86 (COTSON)

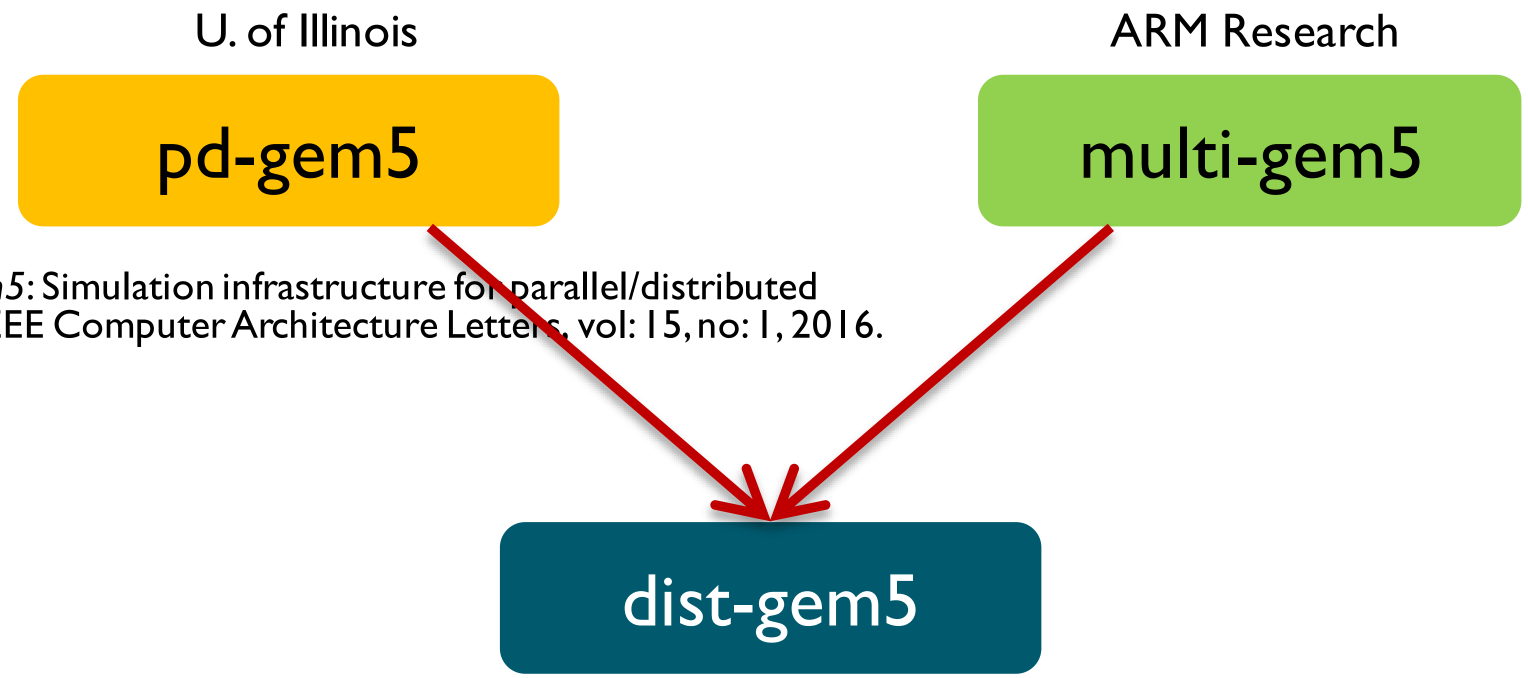
dist-gem5

- Evaluating performance and power dissipation of a distributed system
  - Complex interplay among system components at scale
- Demanding a full-system, cycle-level simulator which is fast enough to simulate a large-scale computer system
- Enabling distributed simulation:
  - Simulation of a distributed computer system w/ many simulation hosts



History of dist-gem5 Development

- Product of excellent synergistic collaboration b/w industry and academia
  - Integrating the best features of concurrently developed multi-gem5 from ARM and pd-gem5 from U. of Illinois for fast and deterministic simulations of distributed computer simulations



M. Alian, et al., *pd-gem5*: Simulation infrastructure for parallel/distributed computer systems. IEEE Computer Architecture Letters, vol: 15, no: 1, 2016.

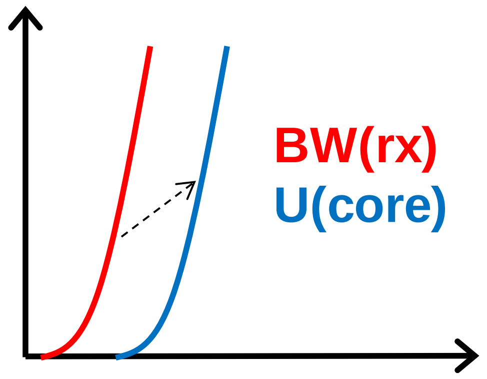
[Best Paper Finalist] M. Alian, et al., "dist-gem5: Distributed Simulation of Computer Clusters," IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), April 2017.

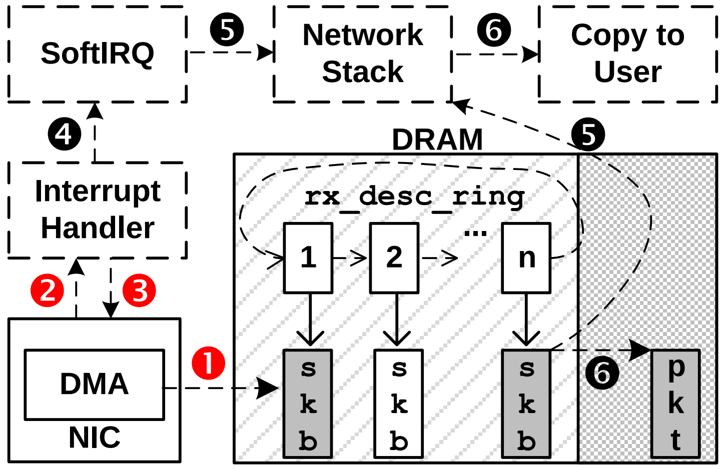
Example of Research w/ dist-gem5

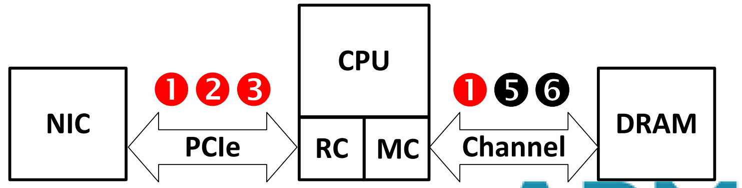
Datacenter power management algorithm

- **Desired P/C-state governor**
  - react to change in core utilization in a timely manner
- **Approaches**
  - predict changes in core utilization
  - core utilization is highly **correlated** w/ network activity
- Hide P/C-state transition latency
  - overlap P/C-state transition w/ **packet reception and processing**

[Nominated for the Best Paper Award] M. Alian, et al. "NCAP: Network-Driven, Packet Context-Aware Power Management for client-server architecture. IEEE International Symposium on High-Performance Computer Architecture (HPCA), February 2017.

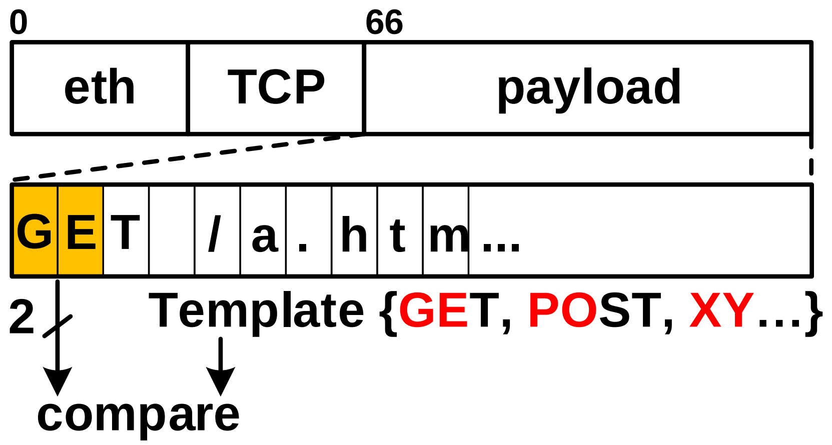






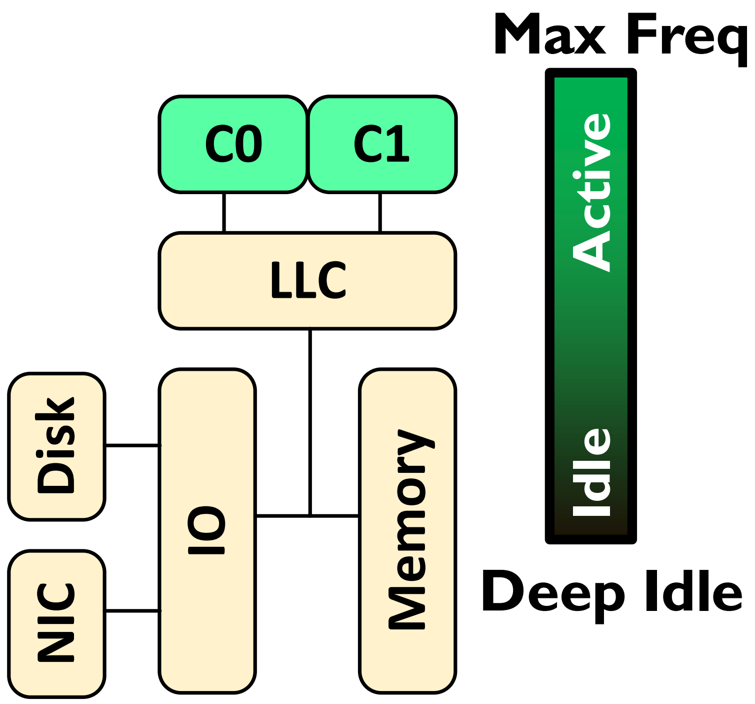
NCAP power management – BW(Rx) surge

- **Detect high rate of "RX" latency-critical packets w/ simple HW in NIC**



- NIC will notify CPU by sending an interrupt to:
  - activate cores
  - boost frequency
  - disable **`menu`** governor

overlap P/C state transition time with packet reception and processing



Other Promising Research Directions

- Exploring HW/SW cross-layer approaches for datacenter computers and their sub-systems
  - Exploiting information from network HW/SW layers as hints for efficient management of computer resource management (e.g., prefetching pages from slow to fast memory in hybrid memory system)
  - Off-loading simple data-intensive operations to network interface cards (NICs)
- Developing efficient evaluation methodologies for large-scale distributed computer systems
  - Exploring systematic hybrid evaluation approaches judiciously mixing queuing-theoretic modeling and dist-gem5-based simulation approaches for efficiently evaluating a VERY large-scale distributed computer systems (e.g., obtaining detailed parameters for queuing-theoretic analytical model using dist-gem5)

Programme

- Introduction (15min)
- Overview of gem5 (45 min)
- *— 15 min Break —*
- dist-gem5 deep-dive (60 min)
  - Packet forwarding
  - Synchronisation
  - Checkpointing
  - Deterministic execution
- *— 15 min Break —*
- Evaluation (30 min)
  - Validation and simulation scalability
  - Demo

What is gem5?

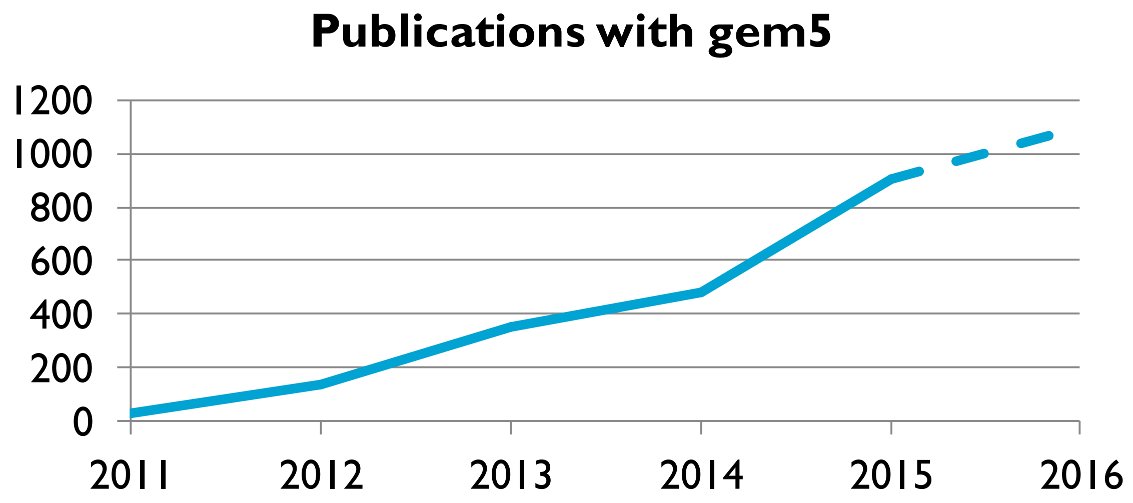
Michigan m5 + Wisconsin GEMS = gem5

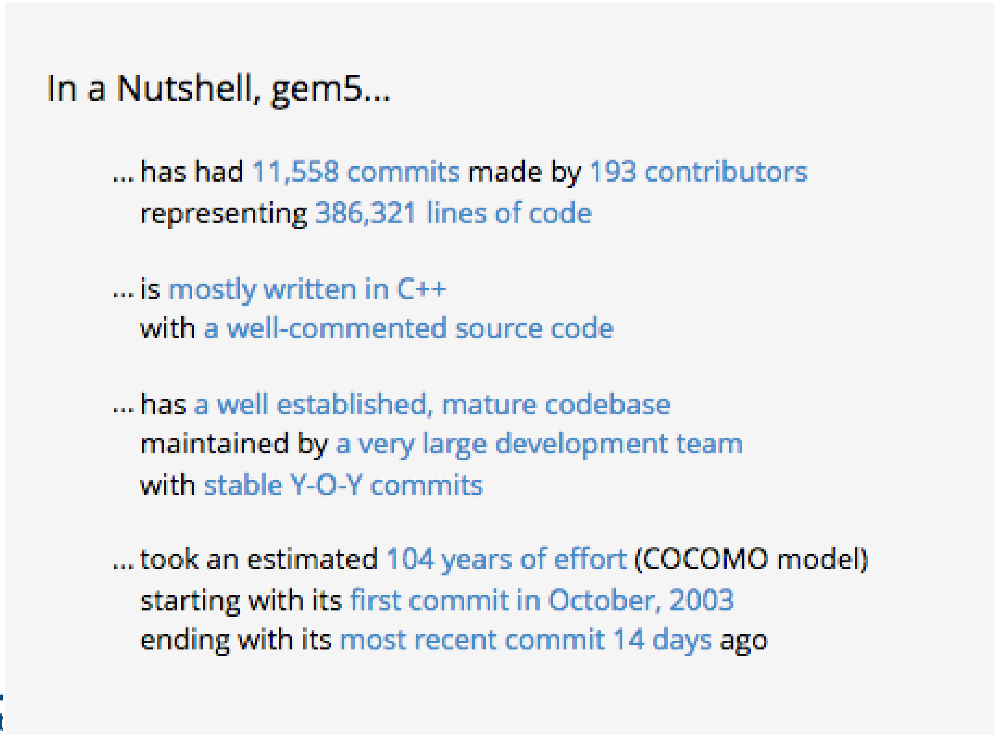
"The gem5 simulator is a modular platform for computer-system architecture research, encompassing system-level architecture as well as processor microarchitecture."

N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. 2011. The gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2 (August 2011), 1-7. DOI=http://dx.doi.org/10.1145/2024716.2024718

Users and contributors

- Widely used in academia and industry
- Contributions from
  - ARM, AMD, Google,…
  - Wisconsin, Cambridge, Michigan, BSC, …



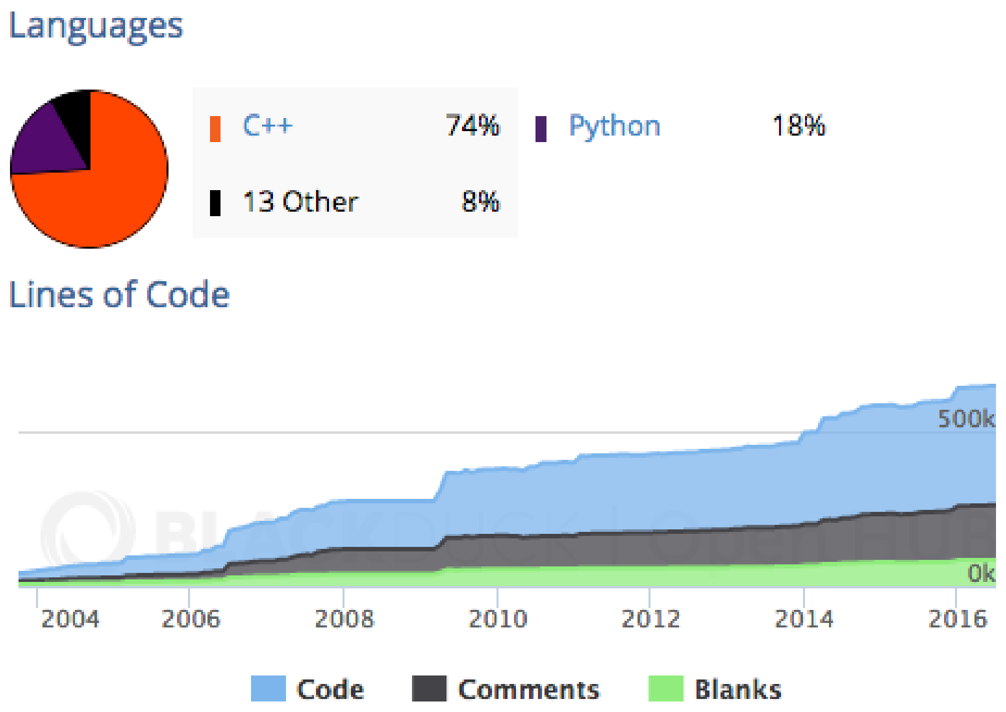

In a Nutshell, gem5...

... has had 11,558 commits made by 193 contributors representing 386,321 lines of code

... is mostly written in C++ with a well-commented source code

... has a well established, mature codebase maintained by a very large development team with stable Y-O-Y commits

... took an estimated 104 years of effort (COCOMO model) starting with its first commit in October, 2003 ending with its most recent commit 14 days ago

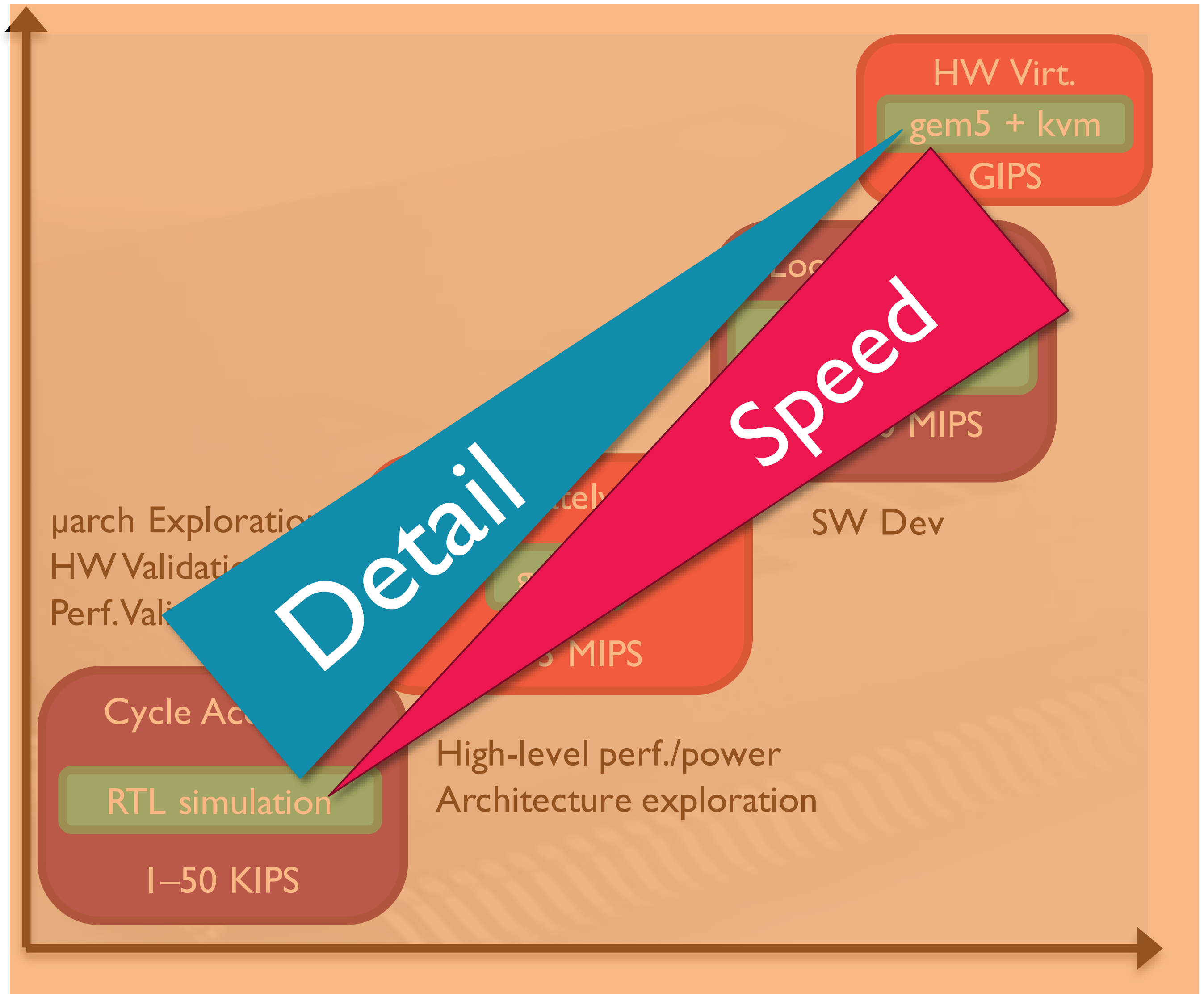

Languages

C++	74%	Python	18%
13 Other	8%		

Lines of Code

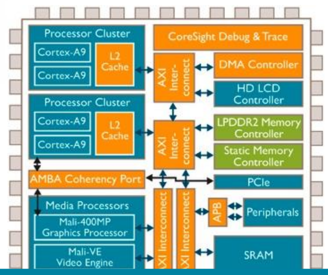
Level of detail

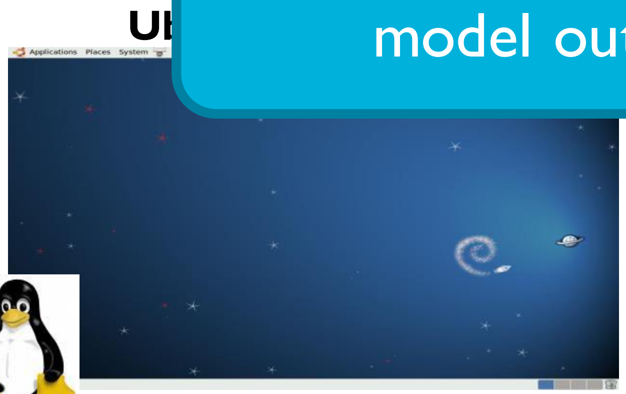
- HW Virtualization
  - Very no/limited timing
  - The same Host/Guest ISA
- Functional mode
  - No timing, chain basic blocks of instructions
  - Can add cache models for warming
- Timing mode
  - Single time for execute and memory lookup
- Detailed mode
  - Full out-of-order, in-order CPU models
  - Hit-under-miss, reodering, …

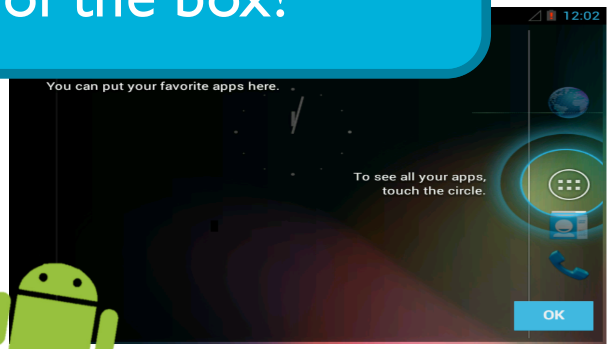


Why gem5?

- Runs real workloads
  - Analyze workloads that customers use and care about
  - … including complex workloads such as Android
- Comprehensive model library
  - Memory and I/O devices
  - Full OS, Web browsers
  - Clients and servers
- Rapid *early* prototyping
  - New ideas can be tested quickly
  - System-level impact can be quantified
- Can be wired to custom models
  - *Add detail where it matters, when it matt*






12:02
You can put your favorite apps here.
To see all your apps, touch the circle.
OK

But **not** a microarchitectural model out of the box!

When not to use gem5

- Performance validation
  - gem5 is not a (out of the box) cycle-accurate microarchitecture model!
  - This typically requires more accurate models such as RTL simulation.
  - Commercial products such as **ARM CycleModels** operate in this space.
- Core microarchitecture exploration
  - *Only* do this if you have a custom, detailed, CPU model!
  - gem5's core models were not designed to replace more accurate microarchitectural models.
- To validate functional correctness or test bleeding-edge ISA improvements
  - gem5 is not as rigorously tested as commercial products.
  - New (ARMv8.0+) or optional instructions are sometimes not implemented
  - Commercial products such as **ARM FastModels** offer better reliability in this space.

Getting Started with gem5

Building gem5

```
$ git clone http://repo.gem5.org/gem5
```

```
$ scons build/ARM/gem5.opt
```

- Guest architecture
- Several architectures in the source tree.
- Most common ones are:
  - **ARM**
  - **NULL** – Used for trace-drive simulation
  - **X86**

- Optimization level:
  - **debug**: Debug symbols, no/few optimizations
  - **opt**: Debug symbols + most optimizations
  - **fast**: No symbols + even more optimizations

ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Example disk images

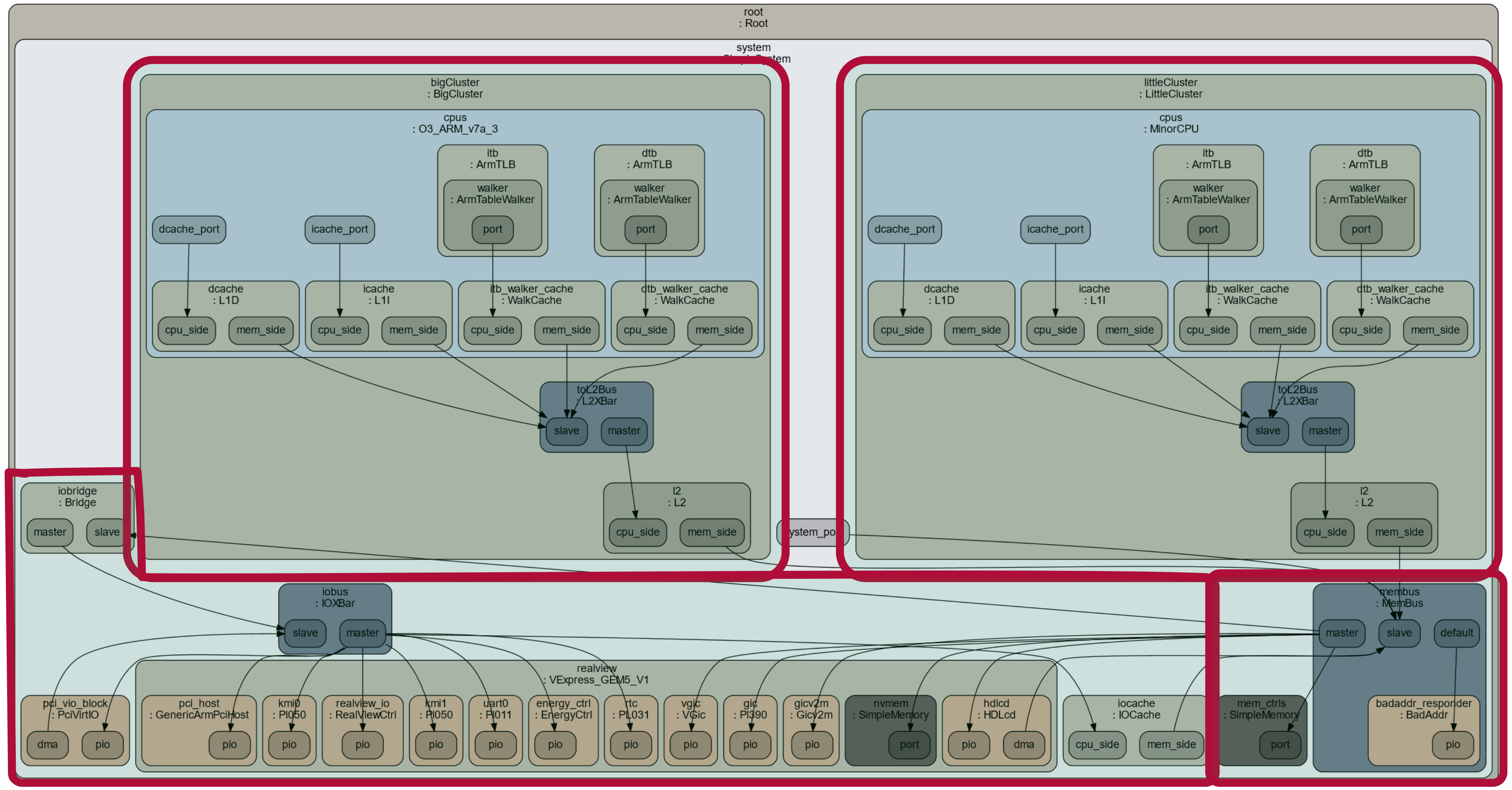
- Example kernels and disk images can be downloaded from gem5.org/Download
  - This includes pre-compiled boot loaders
  - Old but useful to get started
- For example download and extract:
  - `wget http://www.gem5.org/dist/current/arm/aarch-system-2014-10.tar.xz`
  - `mkdir dist; cd dist`
  - `tar xvf ../aarch-system-2014-10.tar.xz`
- Set the M5_PATH variable to point to this directory:
  - `export M5_PATH=/path/to/dist`
- Most example scripts try to find files using `M5_PATH`
  - Kernels/boot loaders/device trees in `${M5_PATH}/binaries`
  - Disk images in `${M5_PATH}/disks`

Running an example script

```
$ build/ARM/gem5.opt configs/example/arm/fs_bigLITTLE.py \
    --disk your_disk_image.img \
    --kernel path/to/vmlinux \
    --dtb $PWD/system/arm/dt/armv8_gem5_v1_big_little_1_1.dtb \
    --cpu-type timing
```

- Simulates a bL system with 1+1 cores
  - Using the 'timing' CPU type: an OoO + InO configuration
  - Alternative: 'atomic' - a functional 'atomic' CPU model

System Overview

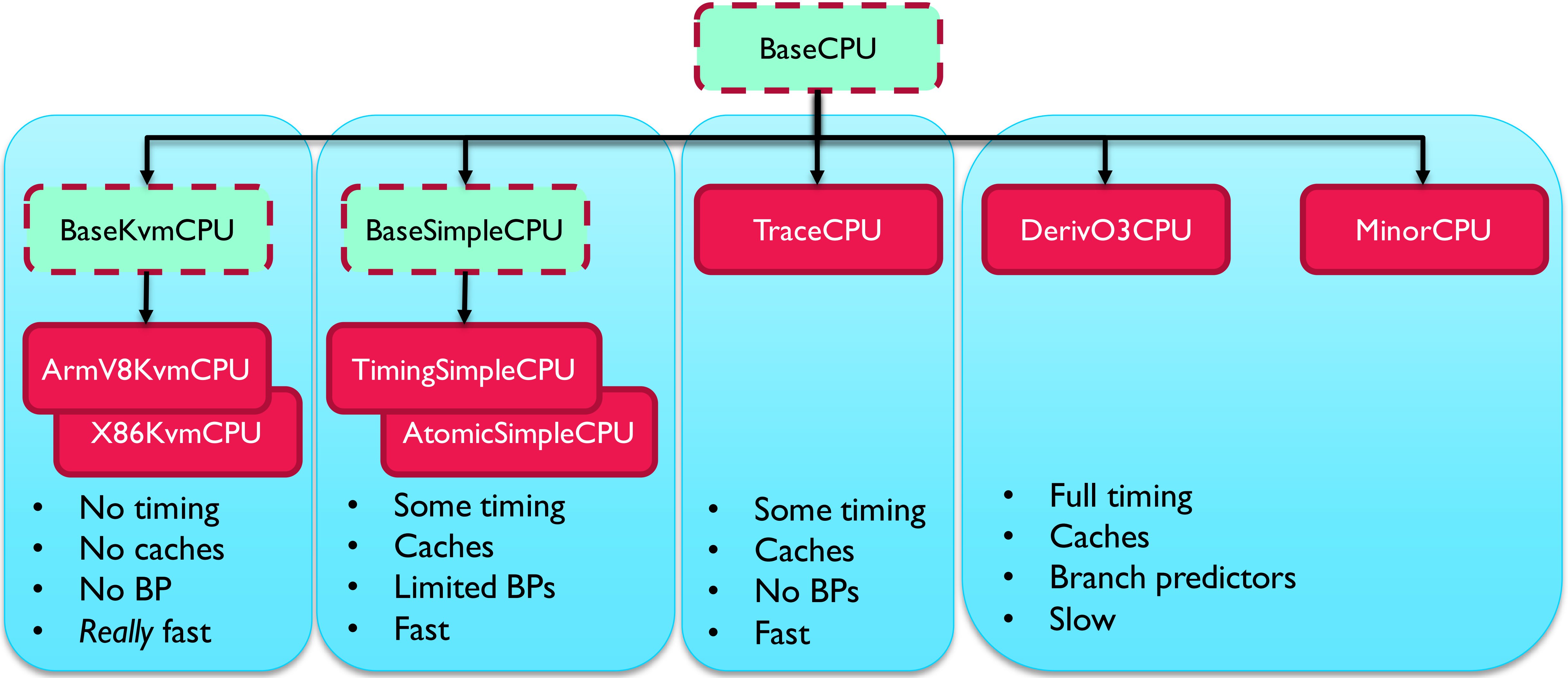


Basic models in gem5

ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

CPU models overview



- No timing
- No caches
- No BP
- *Really* fast

- Some timing
- Caches
- Limited BPs
- Fast

- Some timing
- Caches
- No BPs
- Fast

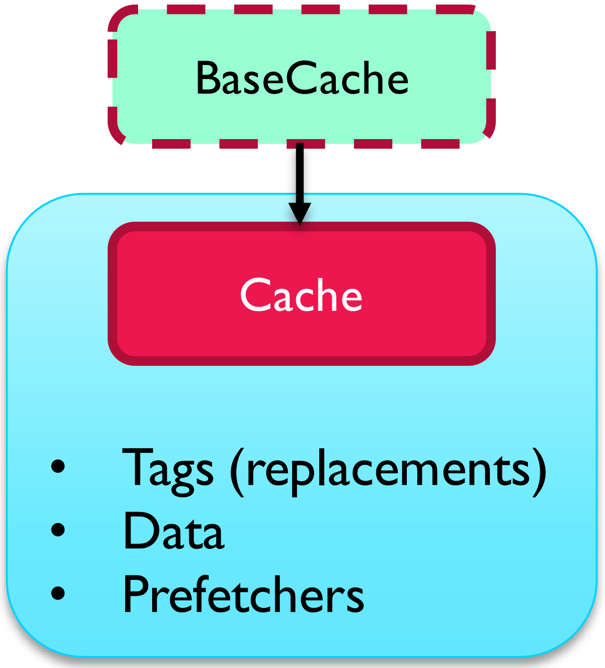
- Full timing
- Caches
- Branch predictors
- Slow

ECE ILLINOIS
Department of Electrical and Computer Engineering

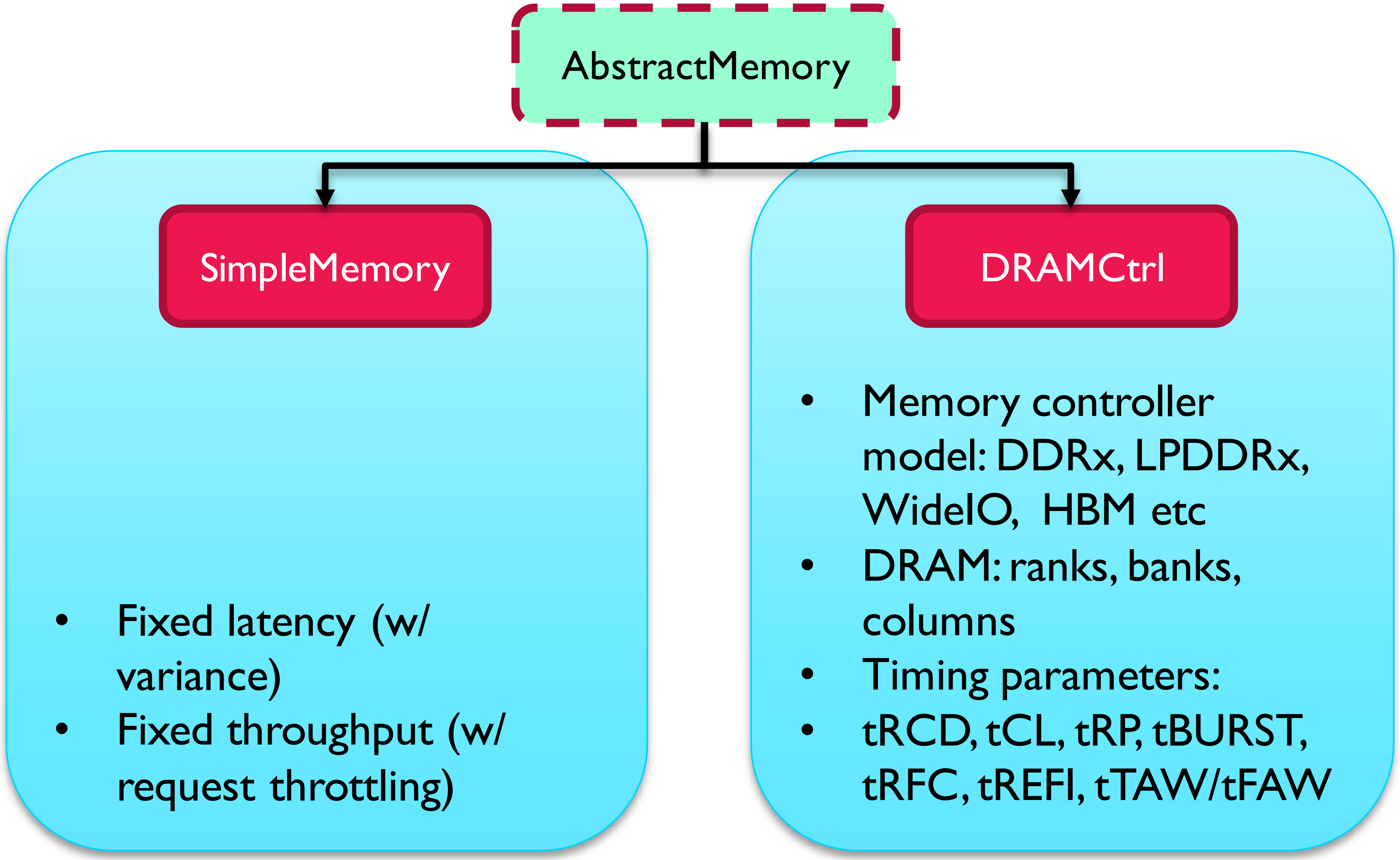
ARM

Caches

- Cache model with several components:
  - Cache: request processing, miss handling, coherence
  - Tags: data storage and replacement (LRU, Random, etc.)
  - Prefetcher: N-Block Ahead, Tagged Prefetching, Stride Prefetching
  - MSHR: track pending/outstanding requests
  - WriteQueue: track writebacks
  - Parameters: size, hit latency, block size, associativity, number of MSHRs (max outstanding requests)



Memory controllers

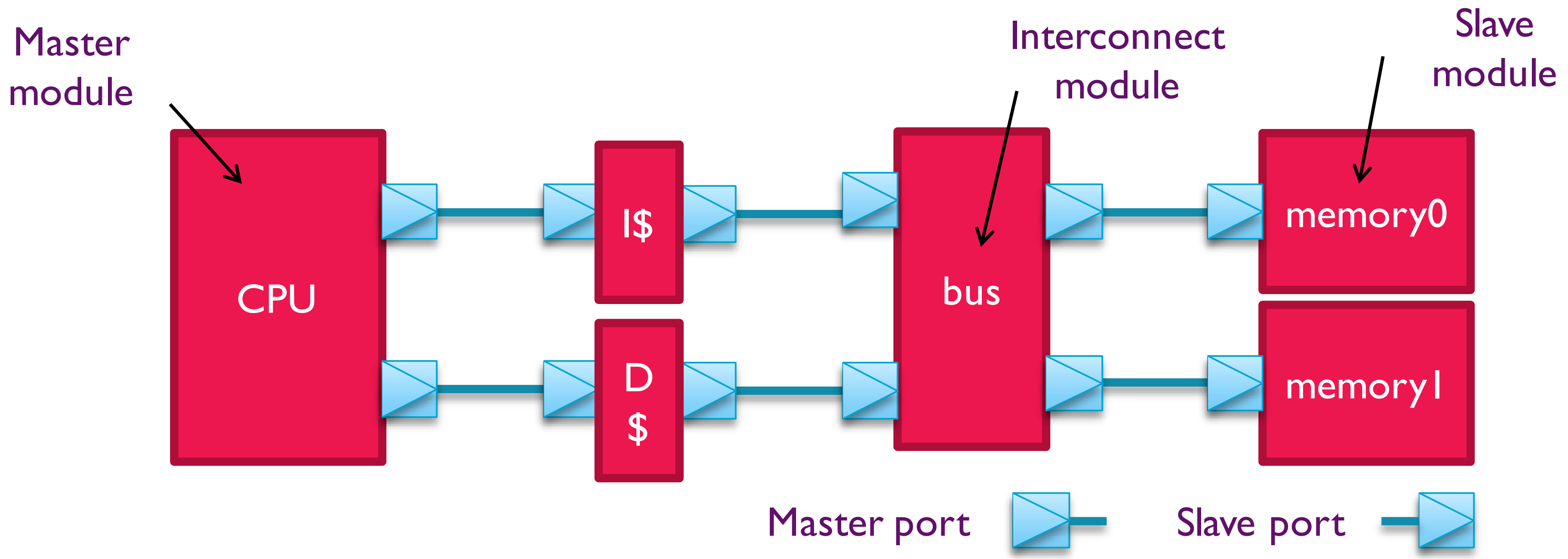


- Fixed latency (w/ variance)
- Fixed throughput (w/ request throttling)

- Memory controller model: DDRx, LPDDRx, WideIO, HBM etc
- DRAM: ranks, banks, columns
- Timing parameters:
- tRCD, tCL, tRP, tBURST, tRFC, tREFI, tTAW/tFAW

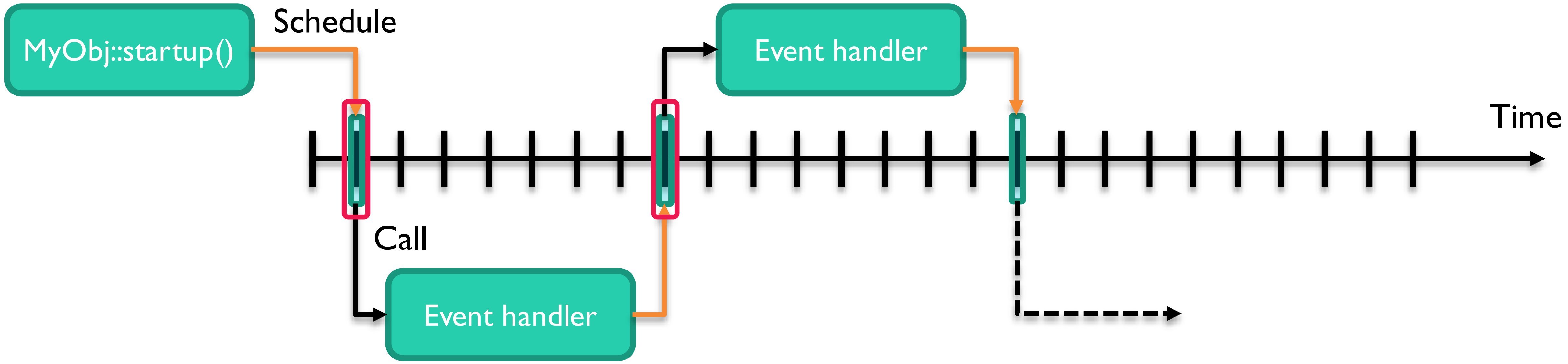
Ports, Masters and Slaves

- Components (MemObjects) are connected through master and slave ports
  - A master port always connects to a slave port (e.g. CPU's master port to cache's slave port)
  - An interconnect module at least one of each
  - Similar to TLM-2 notation



Background

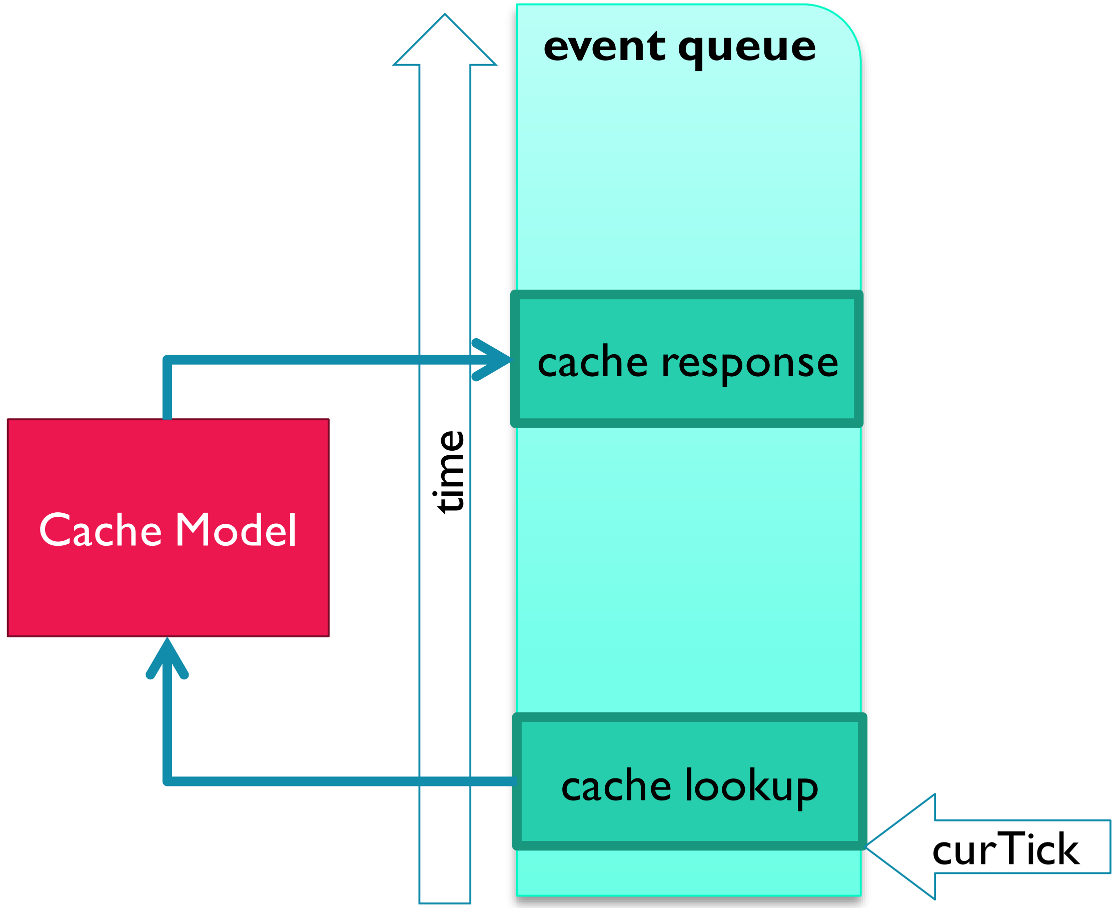
Discrete event based simulation



- Discrete: Handles time in discrete steps
  - Each step is a tick
  - Usually 1THz in gem5
- Simulator skips to the next event on the timeline

Example: Cache Reques

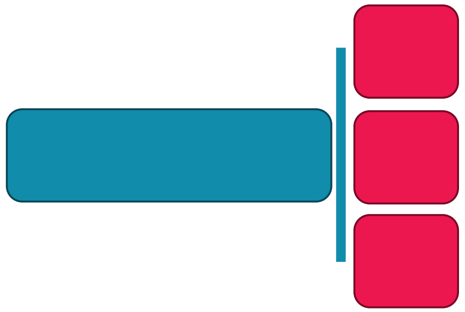
- Event-driven
  - no activity -> no clocking
  - event queue
- Deterministic
  - fixed random number seed
  - no dependence on host addresses
- Multi-Queue
  - multiple workers



Accelerating gem5

- Switching modes
  - (kvm +) functional + timing / detailed

- Checkpoints
  - boot Linux -> checkpoint
  - run multiple configurations in parallel
  - run multiple checkpoints in parallel



- Multi-threading
  - multiple queues
  - multiple workers execute events
  - data sharing and tight coupling limits speedup

- Multi-processed gem5
  - for design space explorations

Checkpointing

- Any simulation object with state, needs to be written to the checkpoint
- Checkpointing takes place on a *drained* simulator
  - Draining ensures that microarchitectural state is flushed
  - Models may need to flush pipelines and wait for outstanding requests to finish.

Creating a checkpoint

Trigger checkpointing

- Script call: **m5.checkpoint(“my.cpt”)**

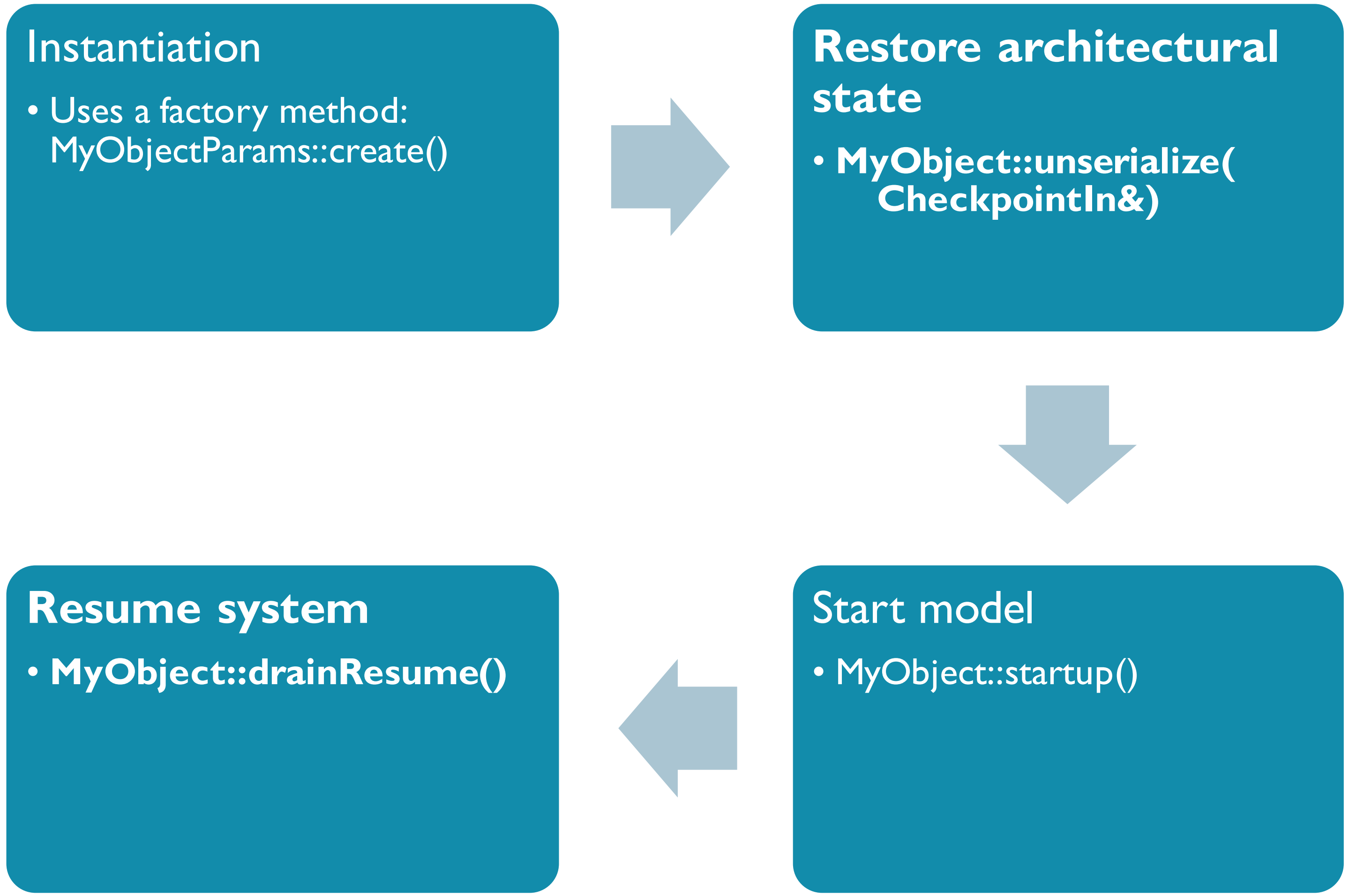
Drain the simulator

- Ensures a well-defined architectural state
- Flushes CPU pipelines
- Writes back caches

Serialize objects

- MyObject::serialize(
      CheckpointOut&)

Restoring from a checkpoint



Instantiation

- Uses a factory method: MyObjectParams::create()

Restore architectural state

- **MyObject::unserialize(CheckpointIn&)**

Start model

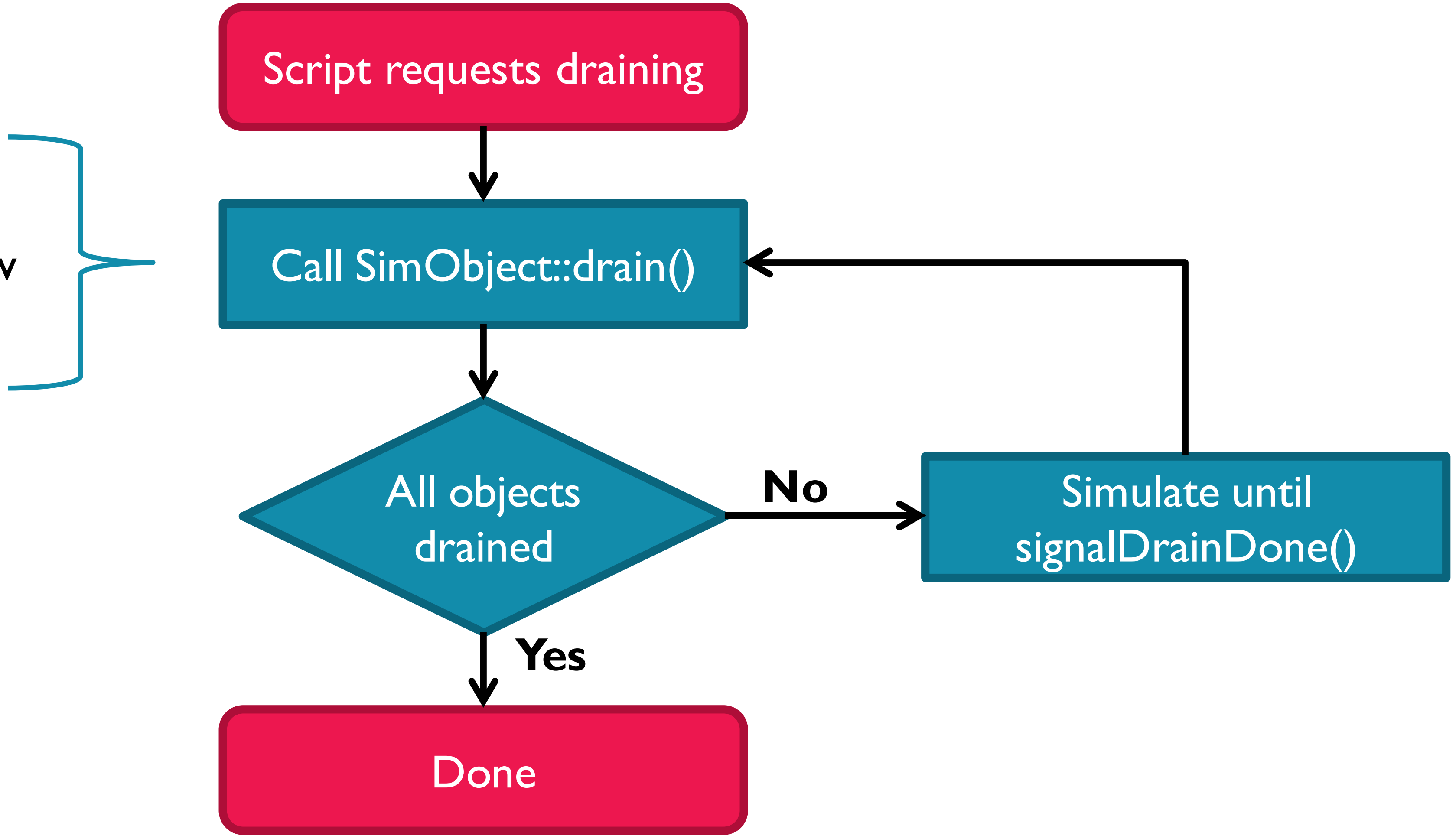
- MyObject::startup()

Resume system

- **MyObject::drainResume()**

Draining

- Flush internal state
- Stop producing new messages



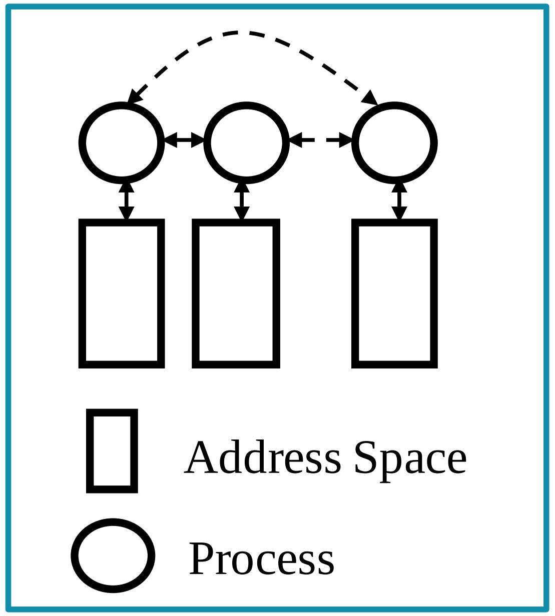
15 min break

Programme

- Introduction (15min)
- Overview of gem5 (45 min)
- *— 15 min Break —*
- dist-gem5 deep-dive (60 min)
  - Packet forwarding
  - Synchronisation
  - Checkpointing
  - Deterministic execution
- *— 15 min Break —*
- Evaluation (30 min)
  - Validation and simulation scalability
  - Demo

The Problem

- Design space exploration for future HPC systems requires simulators to cope with scalable benchmarks
  - e.g. MPI proxy apps from co-design centers (Lulesh, CoMD,…)
- Scale out efficiency related research questions
  - What would be the performance implications of using better/worse network links, NICs, etc. ?
  - What would be the optimal end-to-end latency of the system for a particular parallel application ?
- **Enable gem5 to simulate distributed memory systems on real clusters**

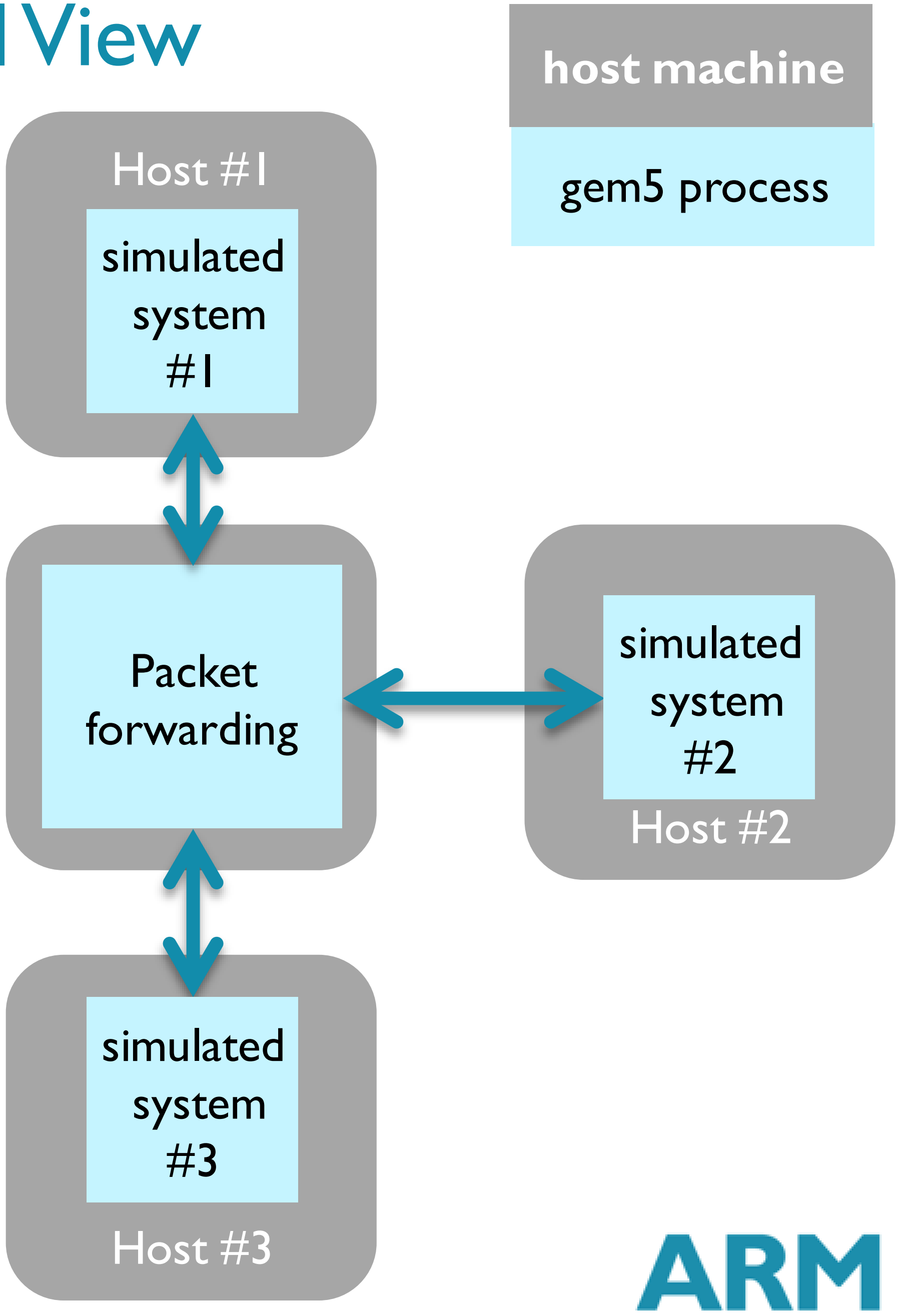


Distributed Memory

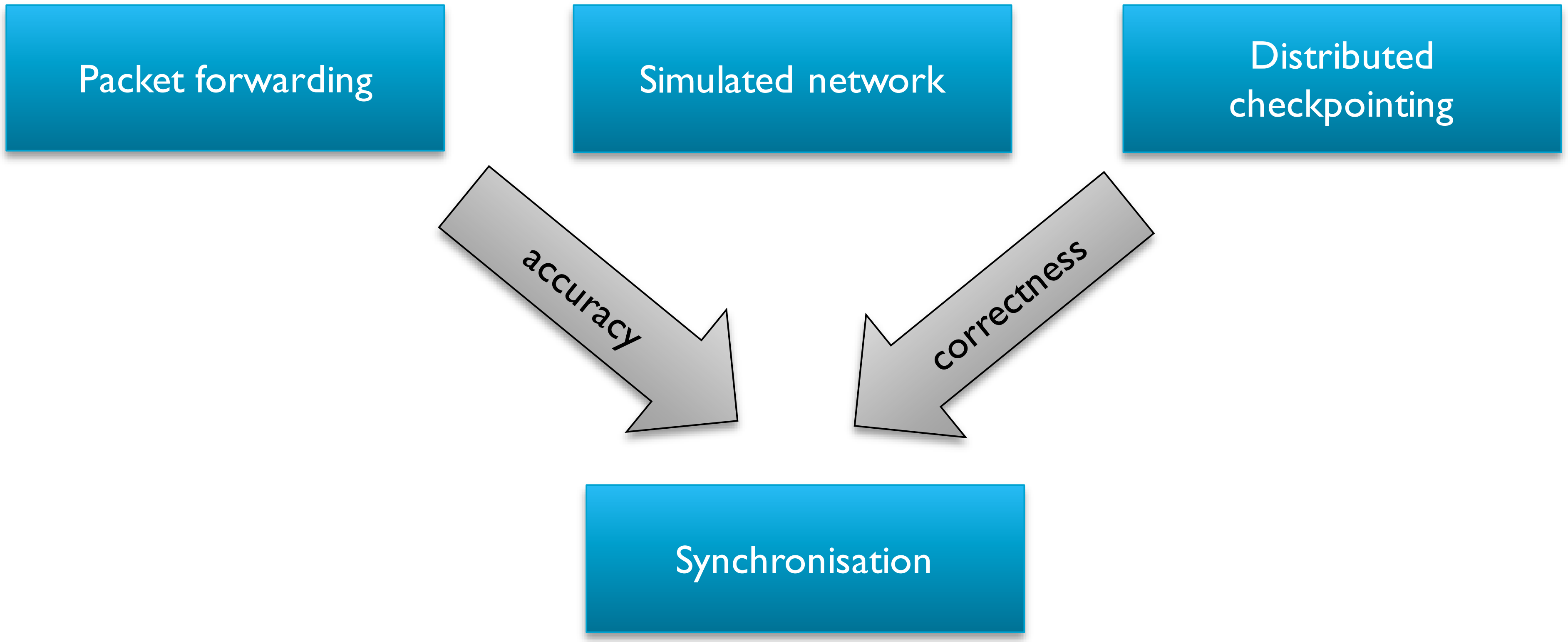
Message Passing

Distributed gem5 Simulation – High Level View

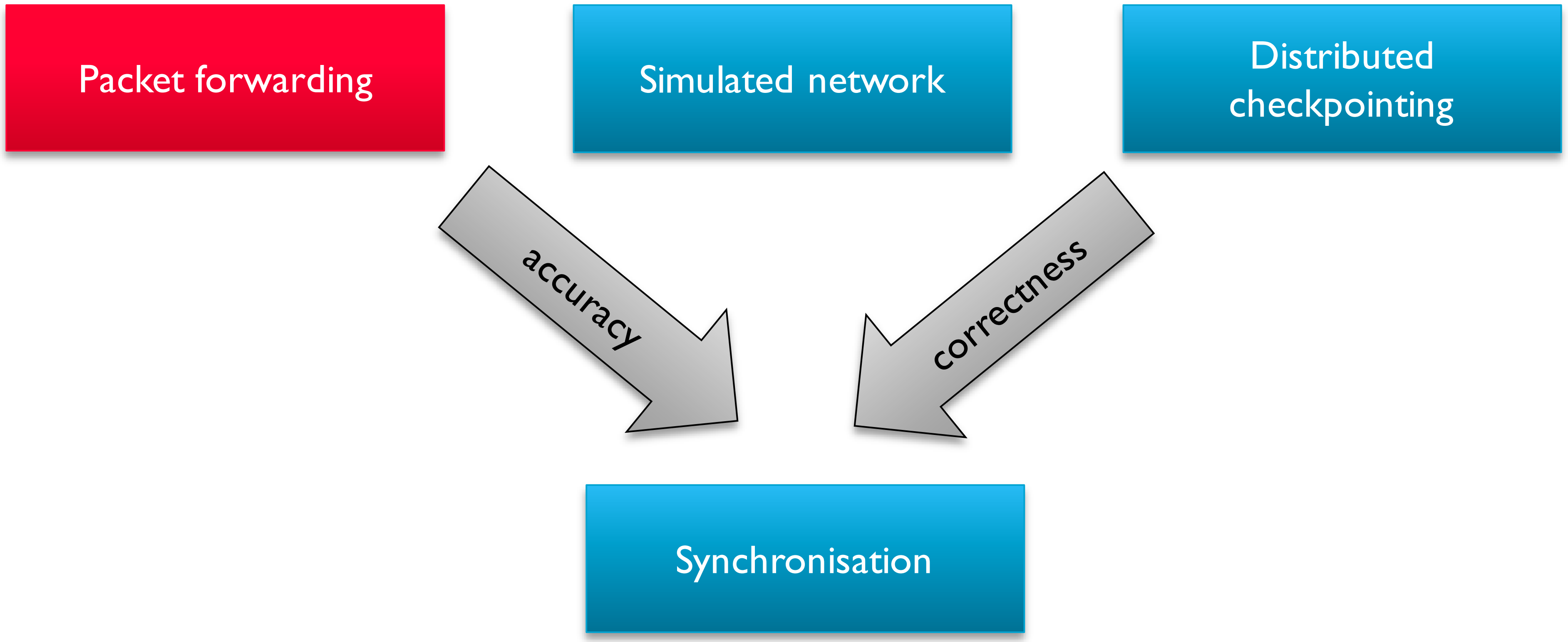
- gem5 processes modeling full systems run in parallel on a cluster of host machines
- Packet forwarding engine
  - Forward packets among the simulated systems
  - Synchronize the distributed simulation
  - Simulate network topology



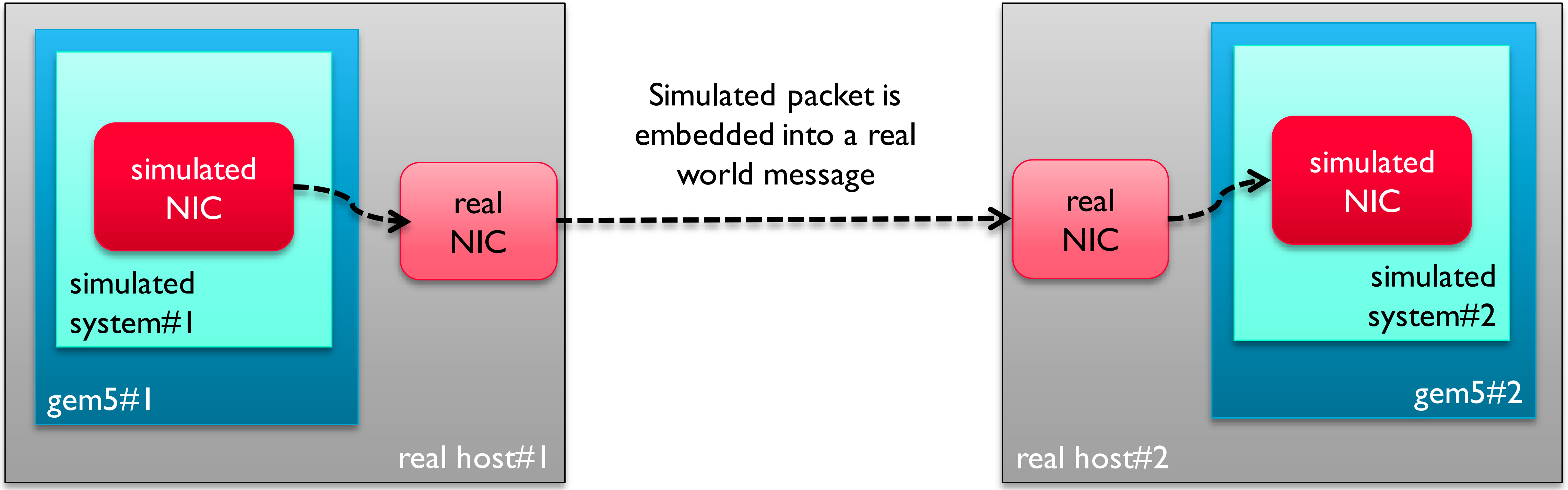
Core Components



Core Components

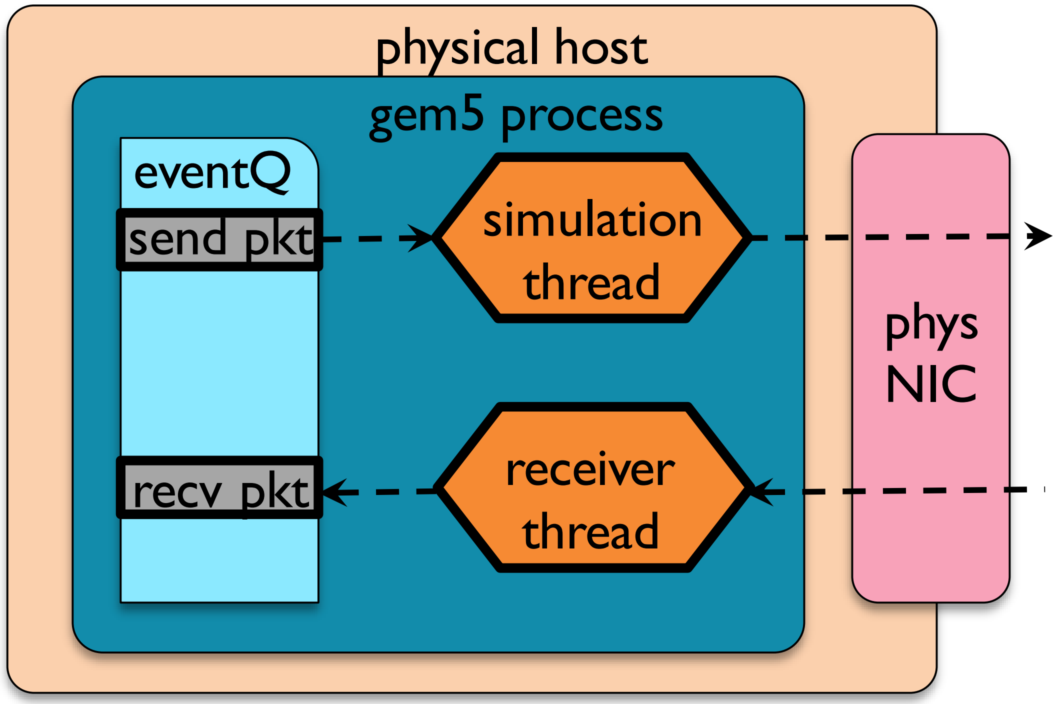


Packet Forwarding



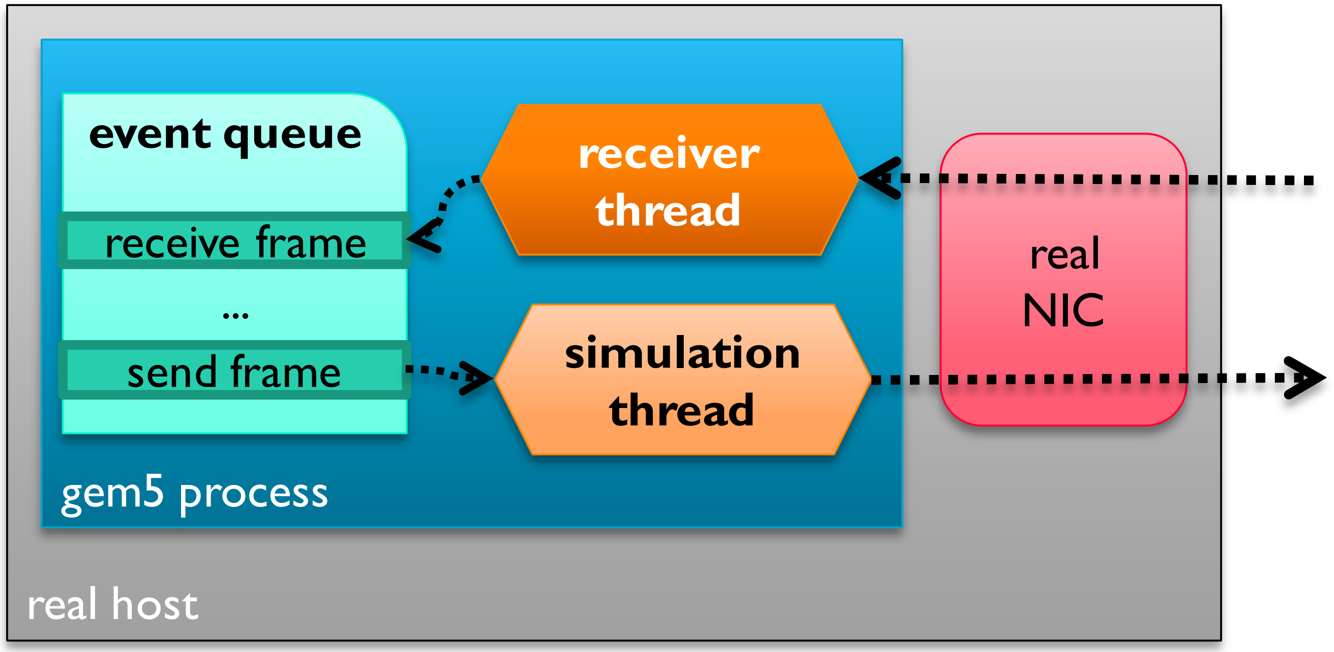
Asynchronous processing of incoming messages

- simulation thread (main thread)
  - process/insert events in the event queue
  - in case of send pkt event, encapsulate the simulated Ethernet packet in a message and send it out
- receiver thread
  - create for each gem5 process
  - waits for incoming packets
  - creates a recv pkt event and insert it to the event queue



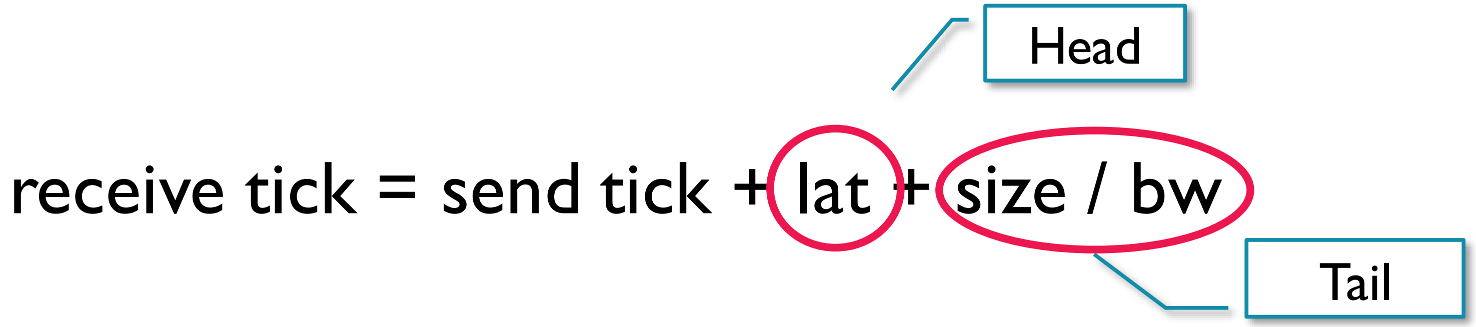
Asynchronous Processing of Incoming Messages

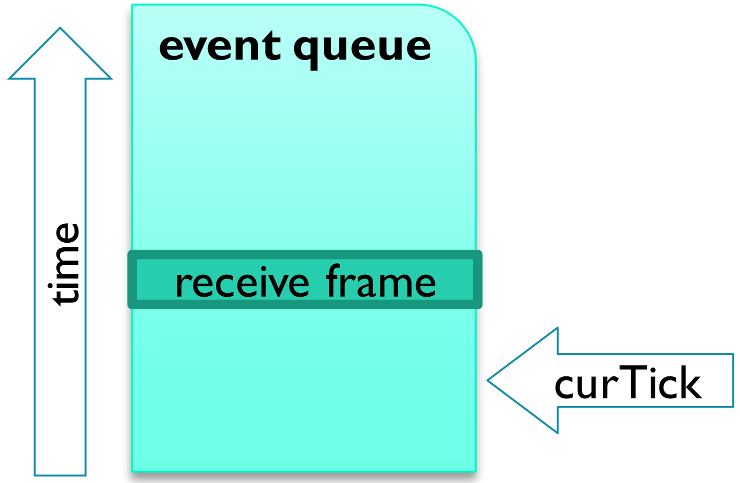
- Simulation thread (aka main())
  - Part of vanilla gem5
  - Process events in the event queue (and inserts new events in the queue)
  - In case of a 'send frame' event encapsulates the simulated Ethernet frame in a message and send it out
- Receiver thread
  - Created for each dist-gem5 process
  - Waits for incoming messages
  - Create a 'receive frame' event for each incoming message and insert it in the event queue



Simulation Accuracy and Packet Forwarding

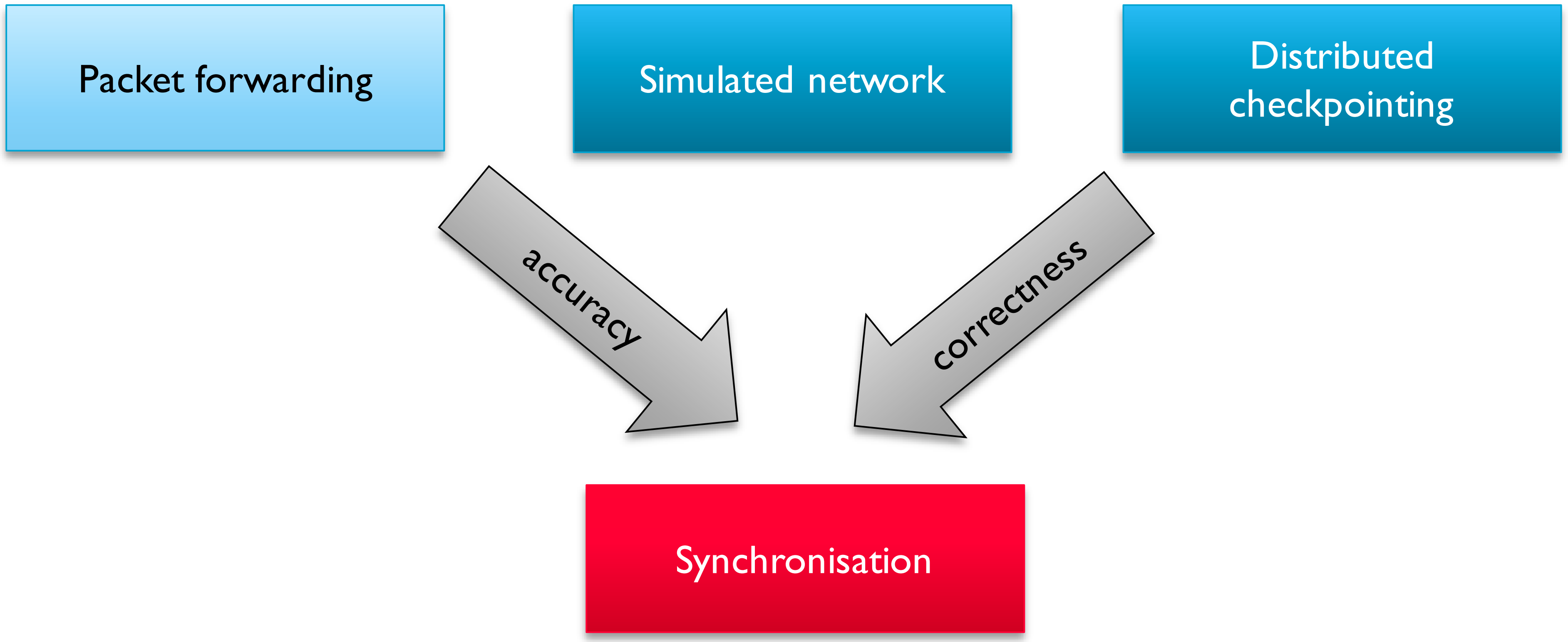
- What is the correct tick for the receive event?
  - *lat*: simulated link latency
  - *bw*: simulated link bandwidth (bytes/tick)
  - *size*: simulated packet size (bytes)





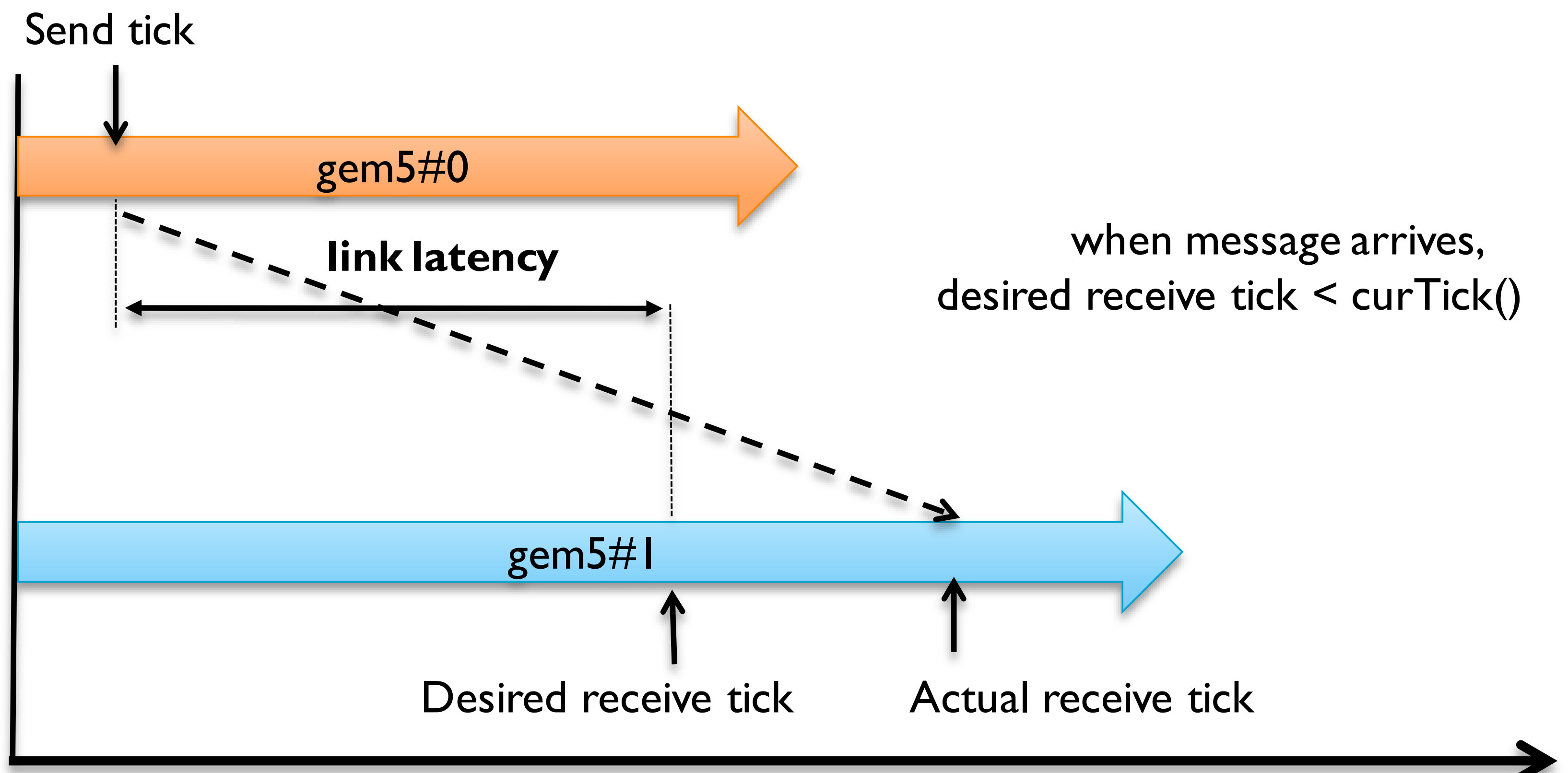
- For accurate simulation, we *must* have
  - **Receive tick >= curTick()** when the receiver gem5 gets the simulated packet
  - Receiver gem5 can schedule the receive event for the simulated NIC

Core Components



Synchronisation – why do we need this?

- Sender and receiver gem5 simulations progress independently of each other
  - Receiver may have less events to process => can run ahead of sender too much (in wall clock time)



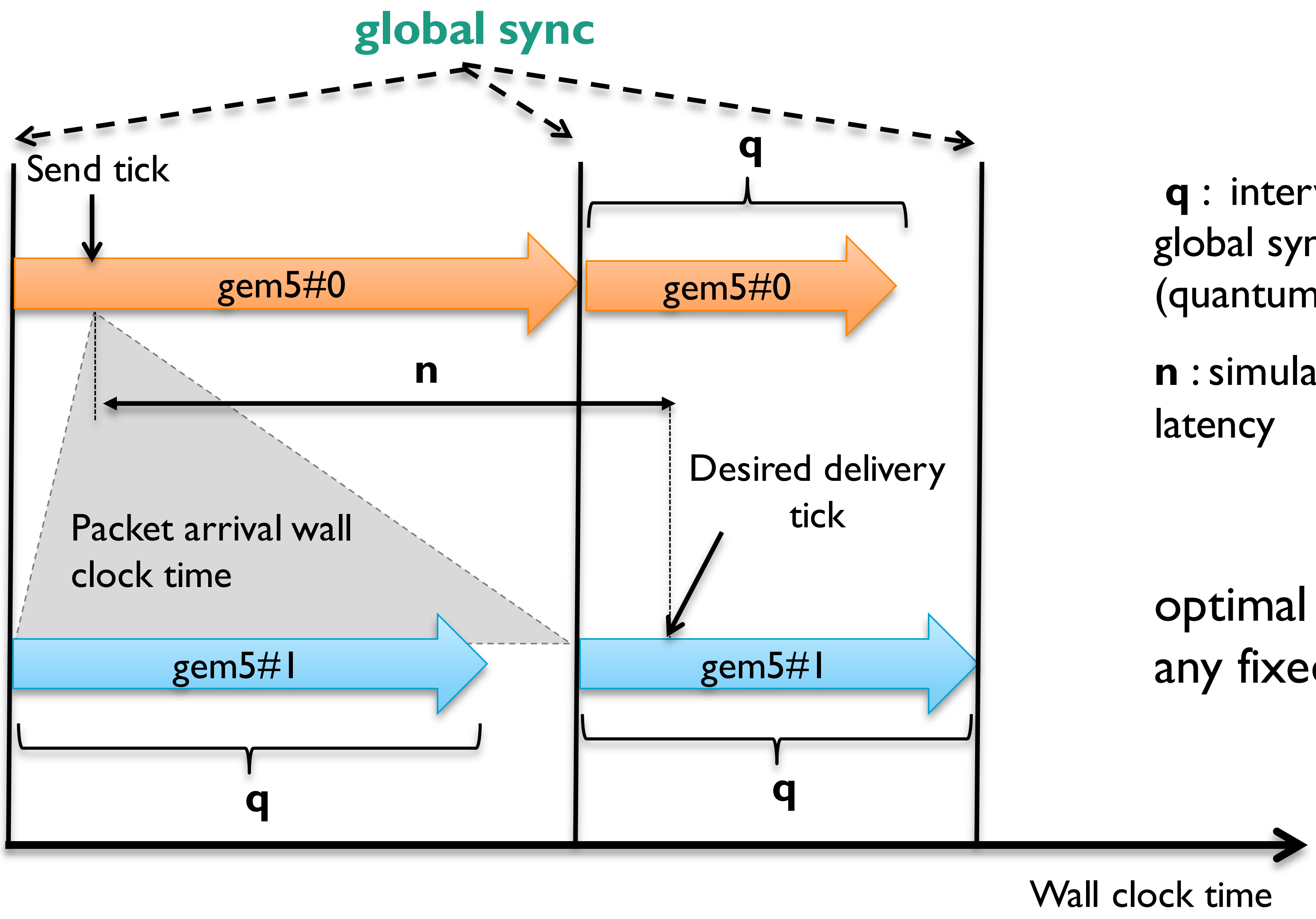
when message arrives,
desired receive tick < curTick()

Problem

Synchronisation – why do we need this?

- Sender and receiver gem5 simulations progress independently of each other
  - Receiver may have less events to process => can run ahead of sender too much (in wall clock time)
  - curTick() may already be larger than the desired receive tick when message arrives $\leftarrow$ Problem
- Synchronisation using a periodic "barrier" termed **global sync** event $\leftarrow$ Solution
  - Receiver and sender gem5 simulations wait for each other to complete global sync
  - curTick() in sender and receiver are kept "close enough" at any point in (wall clock) time
- Synchronisation incurs overhead
  - Try to do as few global sync as possible while still maintain accuracy

Accurate Packet Forwarding



q : interval for periodic global synchronisation (quantum)

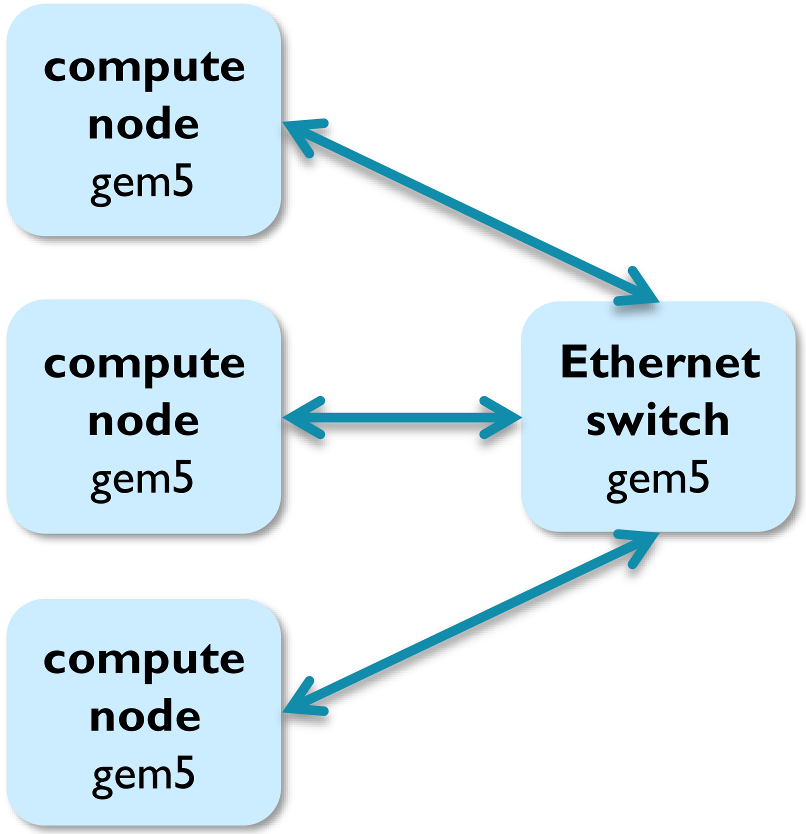
n : simulated network link latency

q ≤ n

optimal **q**: **q == n** for any fixed **n**

Compute Nodes, Switch and Synchronisation

- Simulation progress gets stopped at each sync event in each gem5 process
- Simulated compute node
  - Sends out ‘synq request’ message
  - Waits until ‘sync ack’ message comes back
- Simulated switch
  - Waits until it receives a ‘sync request’ message
  - Broadcasts out ‘sync ack’ message



The Global Sync Event

- A global sync event is scheduled every quantum (q ticks) in each gem5 process

- The process() method in a **compute node**
  - sends out ‘**sync request**’ messages for each simulated link
  - waits on a condition variable to get notified about completion by the receiver thread

- The process() method in a **switch**
  - waits for completion notification from the receiver thread
  - sends out ‘**sync ack**’ messages for each simulated link

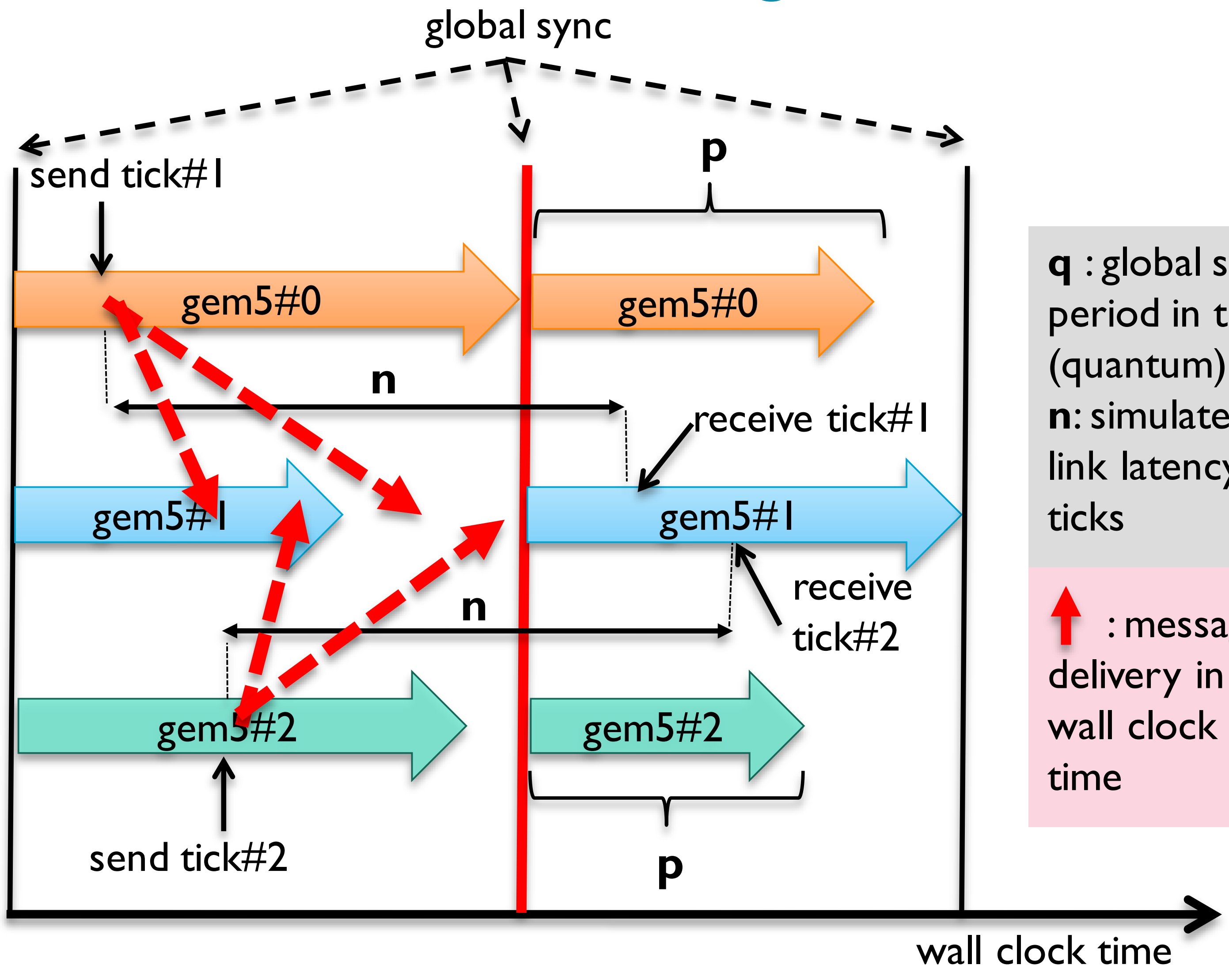
- Receiver thread keeps processing incoming messages while simulation thread is blocked
  - creates receive events in the event queue for simulated Ethernet frames
  - notifies blocked simulation thread when ‘**sync ack**’ messages arrive
  - notifies blocked simulation thread when ‘**sync request**’ messages arrive

Deterministic Execution Issues

- We assume that a single compute node gem5 simulation is deterministic
- Ordering and speed of dist-gem5 messages in real world
  - Speed of gem5 processes (relative to each other) may vary
  - Communication speed among gem5 process may vary
- Global sync guarantees deterministic packet forwarding
  - **sync quantum <= simulated link latency**
  - **global sync is a message barrier**

Global Sync and Deterministic Packet Forwarding

- Receive tick for a simulated packet may not fall within the same quantum which the message gets received in
- A message is always gets sent and received within a single quantum



q : global sync period in ticks (quantum)
n: simulated link latency in ticks

🡅 : message delivery in wall clock time

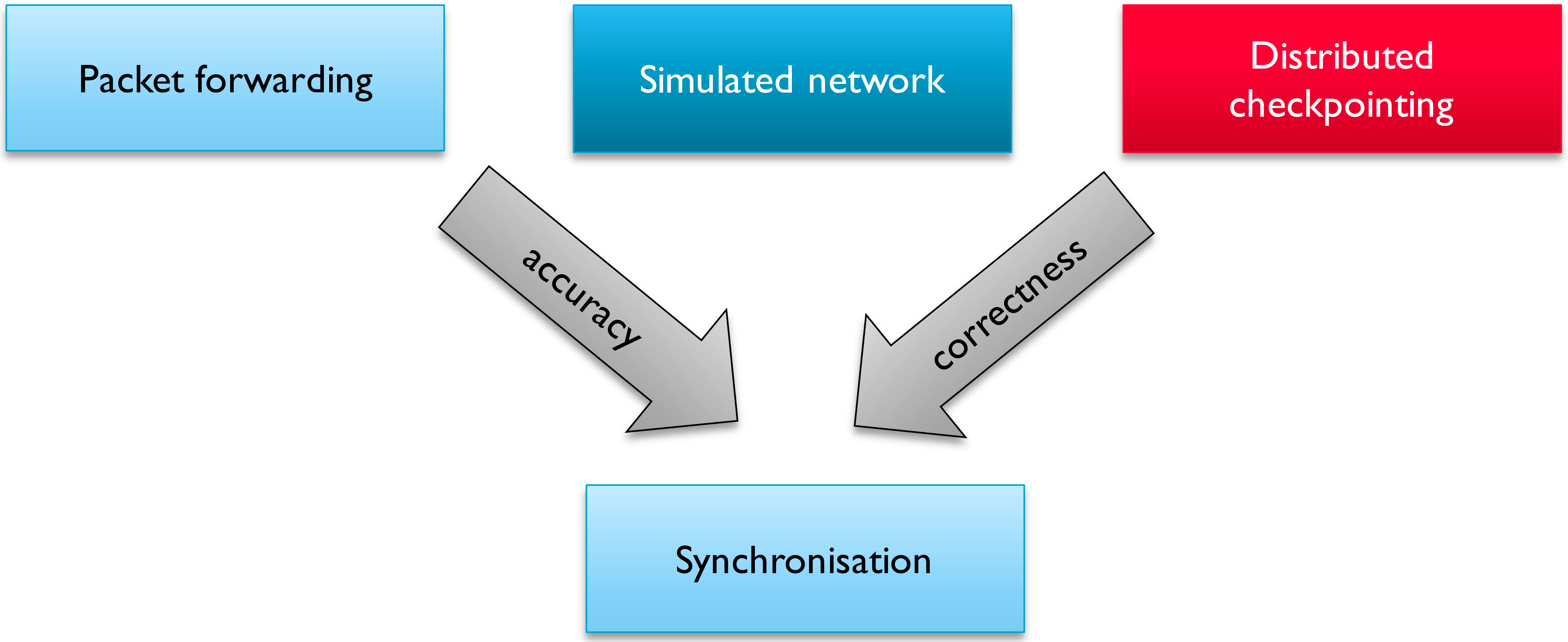
Global Sync and Deterministic Packet Forwarding (cont.)

Pre-condition	Invariant across multiple runs
quantum <= simulated link latency	**Receive order of messages within the same quantum does not matter**
	The sorted list of receive ticks falling within the active quantum will not change
global sync is a message barrier	**Each message will "happen" in exactly the same quantum across different runs**

ECE ILLINOIS
Department of Electrical and Computer Engineering

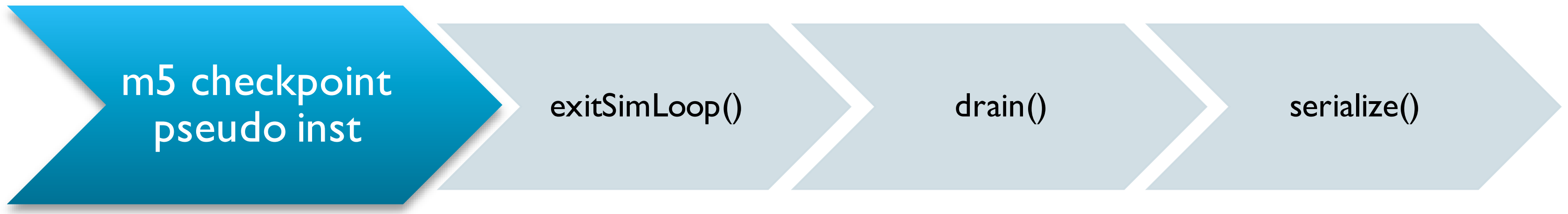
ARM

Core Components

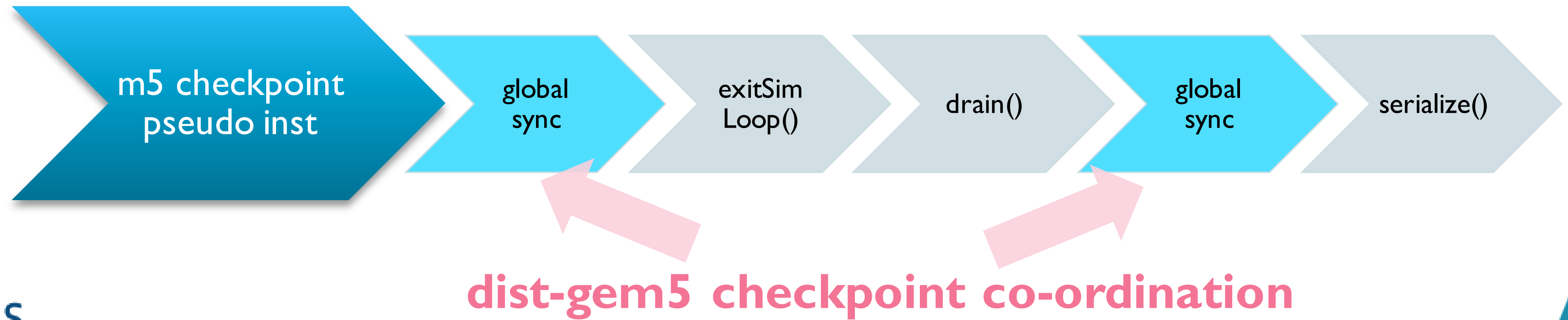


Distributed Checkpointing

- Checkpoint support for dist-gem5 relies on the mainline gem5 checkpoint support
  - Each gem5 process of a dist-gem5 run creates its own checkpoint



- dist-gem5 adds an extra co-ordination layer to ensure correctness
  - No in-flight message may exist among gem5 processes when the distributed checkpoint is taken



ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Distributed Checkpointing (cont.)

- Checkpoint can only be initiated at a periodic global sync
  - Simplifying implementation without sacrificing usability

Checkpoint flavour	collaborative checkpoint	immediate checkpoint
Condition	all compute nodes signal intent	at least one compute node signals intent
Example use case	Instrumented MPI application source code to take a checkpoint at the MPI_barrier() before ROI	Taking a checkpoint from the bootscript before starting an MPI application (i.e. before calling 'mpirun')

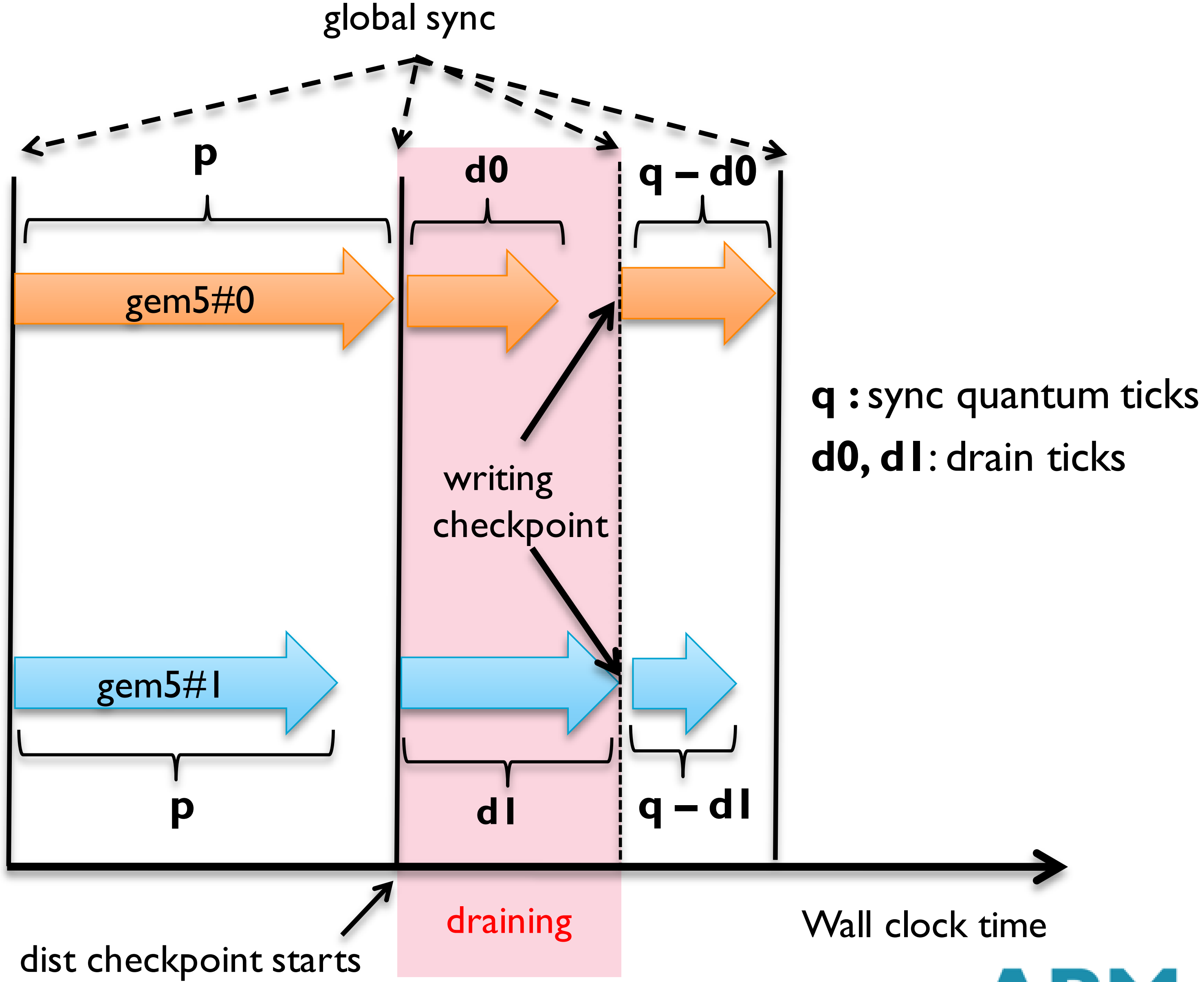
Distributed Checkpointing (cont.)

- Collaborative checkpoint
  - In practice the checkpoint is taken "near" an application barrier (e.g. MPI_Barrier() or mpirun)
  - When all processes hit the barrier in the application code => desired application *state is captured* even if we allow checkpoint writes only at global sync

- Immediate checkpoint
  - A compute node gem5 processes signals its intention to take a checkpoint
    - 'm5 checkpoint' pseudo instruction => 'need checkpoint' meta info in the next 'sync request' message
  - Switch gem5 process can "command" to write a checkpoint
    - 'write checkpoint' meta info in the 'sync ack' message => exitSimLoop() in all gem5 processes

Writing Checkpoint

- Distributed checkpoint can start only at a global sync
- Draining may require different number of ticks in each gem5
- After drain is complete, in-flight messages are flushed with an extra global sync
    - Global sync implements both an execution and a data (message) barrier



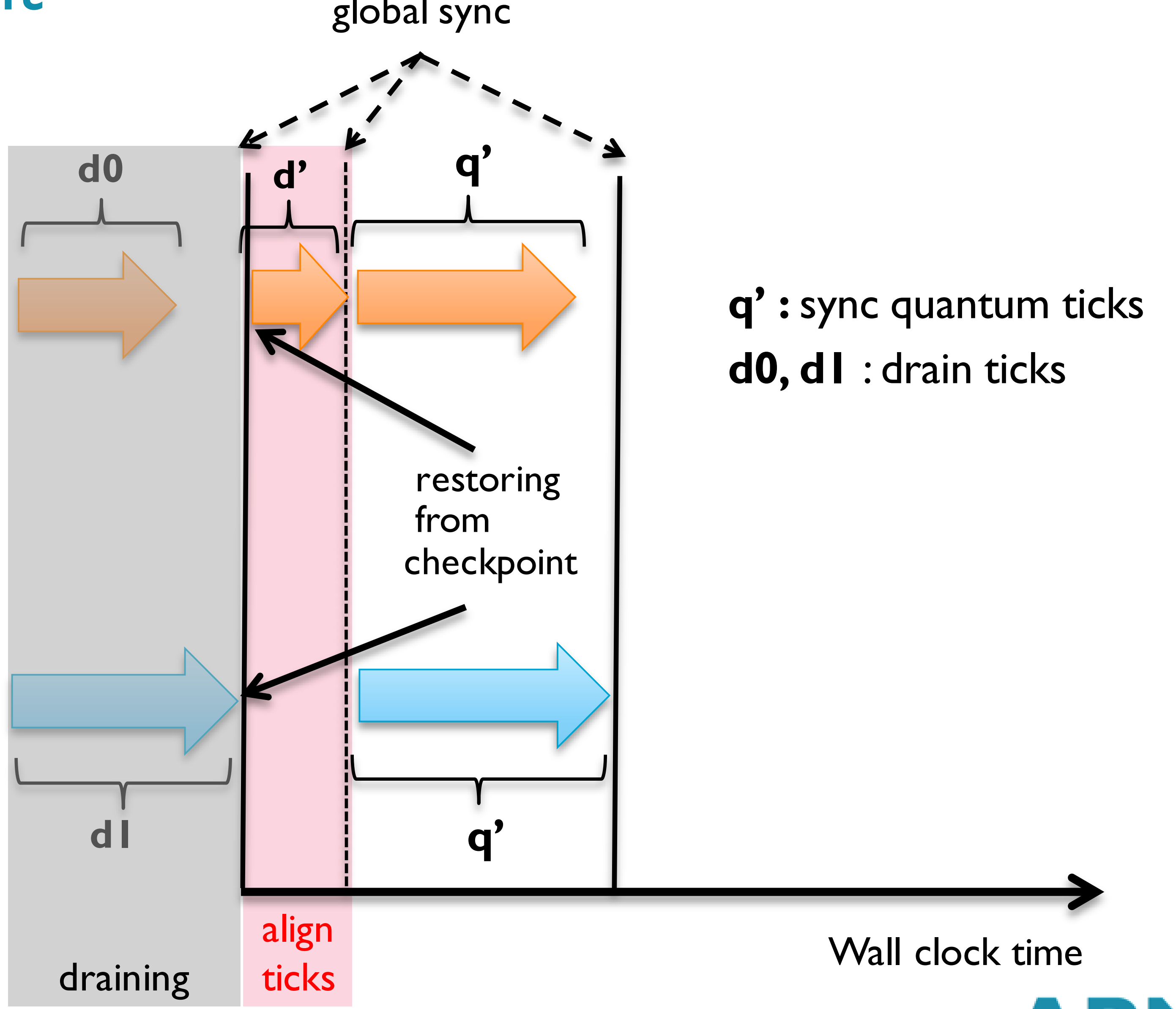
q : sync quantum ticks
d0, d1: drain ticks

Restoring from Checkpoint

- Checkpoint might be written at different ticks in different gem5 processes
- Extra global sync to align the ticks:

  d0 + d' = d1

  - Global sync delivers the max tick value to all gem5 processes
- Global sync period may change at restore
  - Same checkpoint can be used to explore different network link latency/bandwidth effects



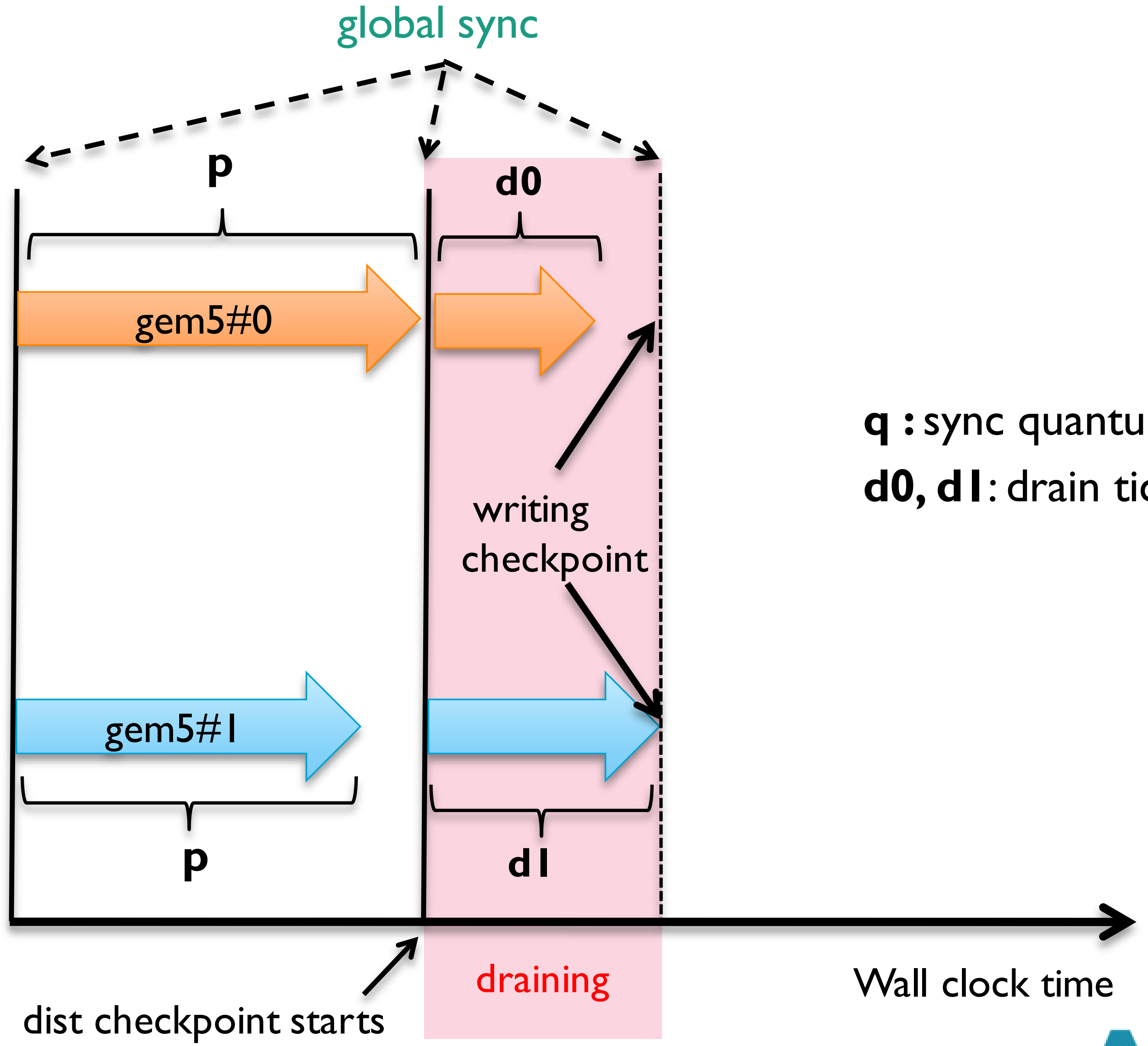
q' : sync quantum ticks
d0, d1 : drain ticks

ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Writing Checkpoint

- Distributed checkpoint can start only at a global sync
- Draining may require different number of ticks in each gem5
- After drain is complete, in-flight messages are flushed with an extra global sync
  - Global sync implements both an execution and a data (message) barrier

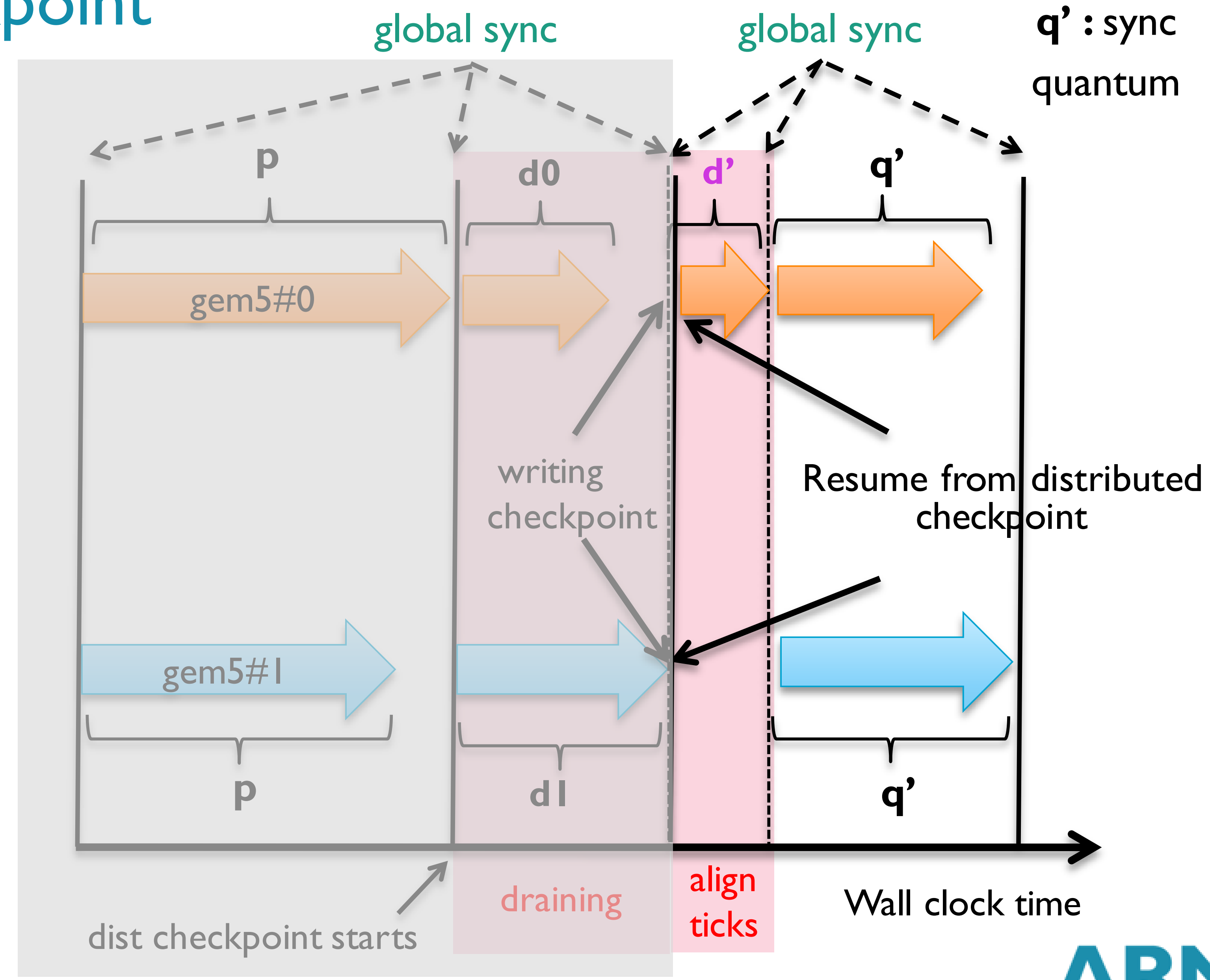


q : sync quantum ticks

d0, d1: drain ticks

Restoring from Checkpoint

- Checkpoint might be written at different ticks in different gem5 processes
- Extra global sync to align ticks:
  d' = d1- d0
  - Global sync delivers the max tick value to all gem5 processes
- Global sync period may change at restore
  - explore different network link latency/bandwidth effects



ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

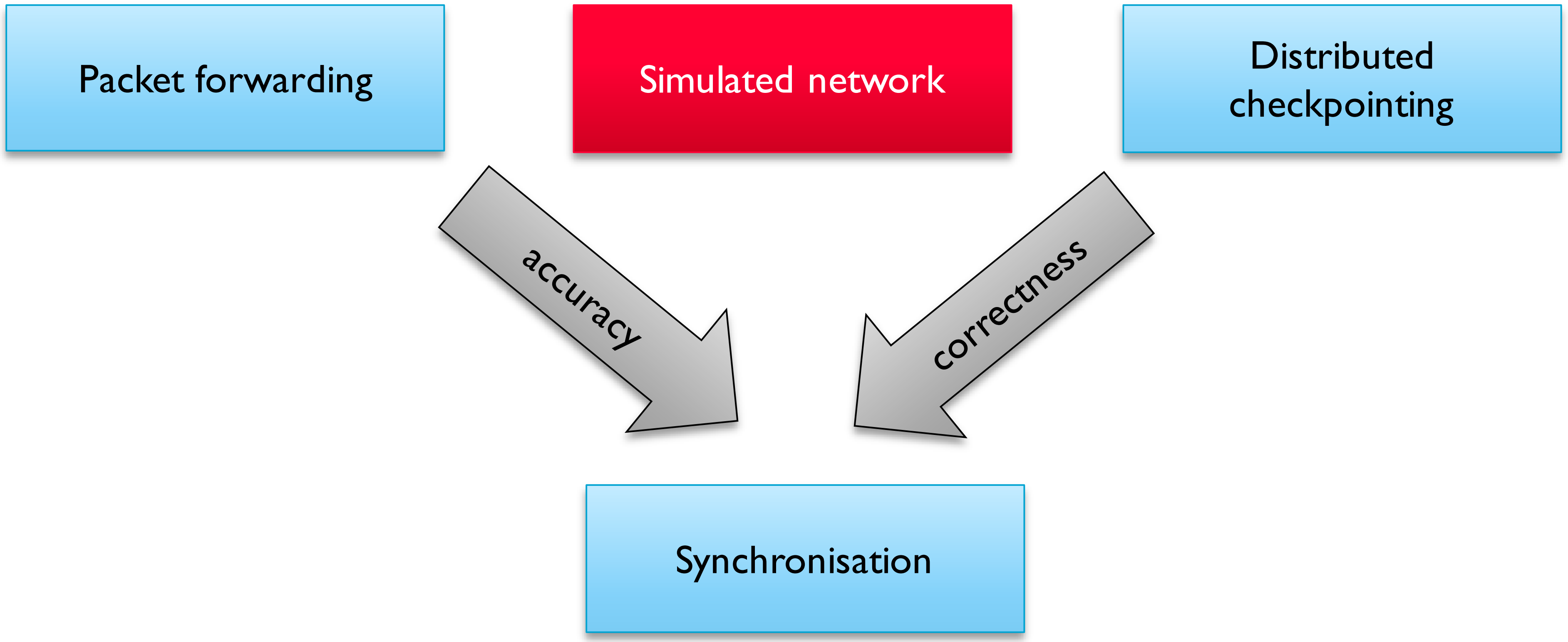
Restoring from Checkpoint (cont.)

- User is allowed to change simulated link parameters when restoring from a checkpoint
  - Same checkpoint can be used to explore different network link latency/bandwidth effects

$\Downarrow$

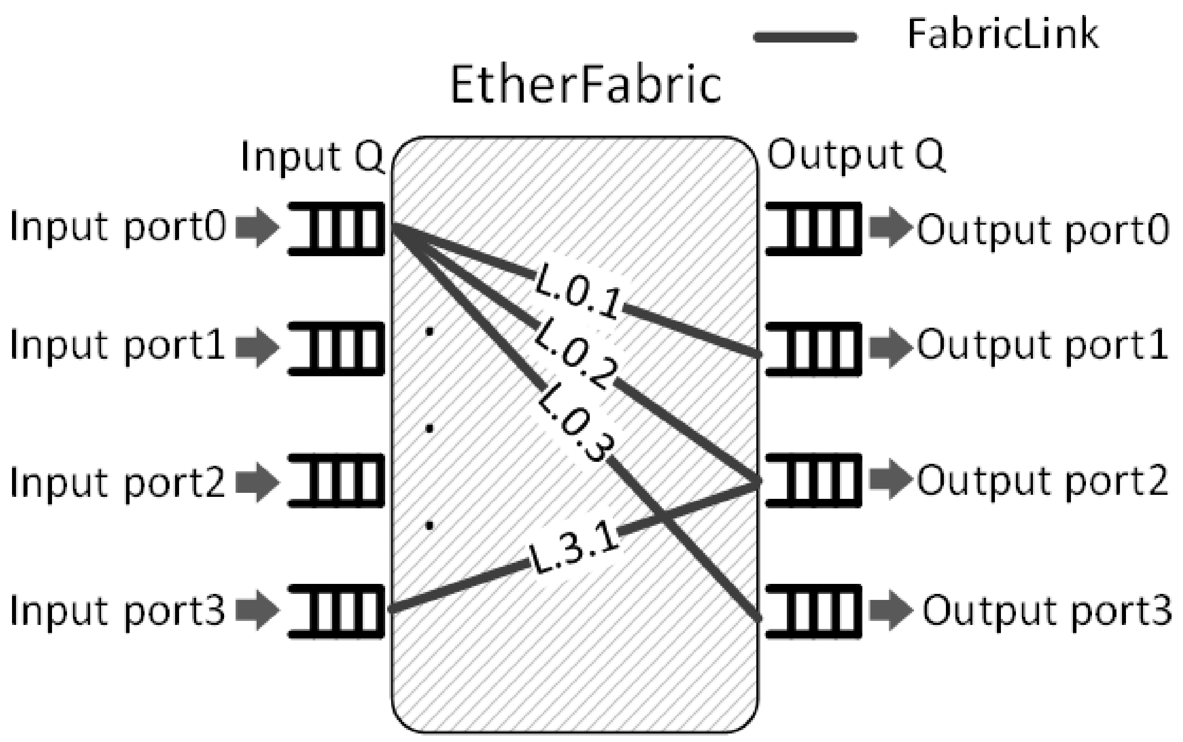
- Global sync period may change at restore (if the simulated link latency change)
  - Checkpoint may contain simulated packets to get received in the future
  - Receive ticks for such packets are adjusted to reflect the change of the simulated link parameters

Core Components

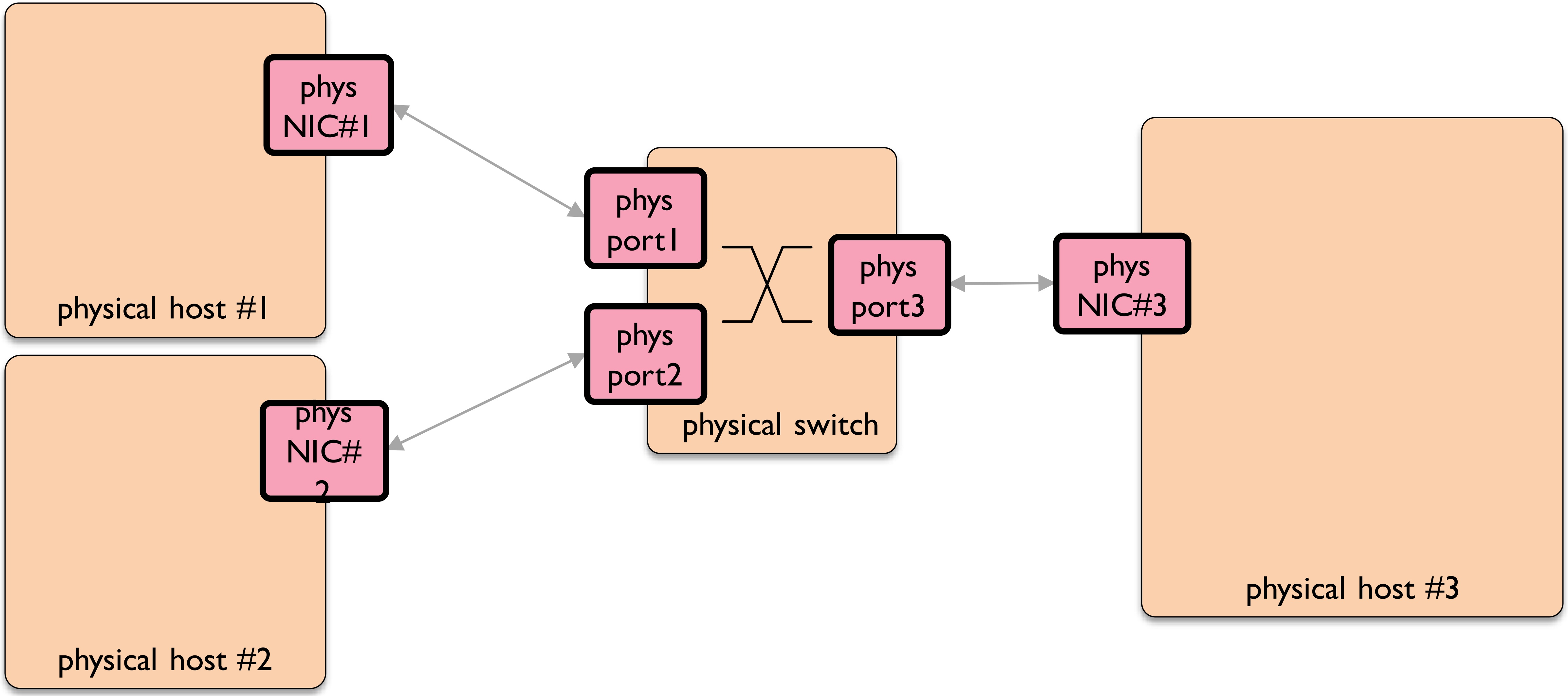


Architecture of the Simulated Ethernet Switch

- Interface (per port)
  - Input and output packet queues
  - Connects to DistEtherLink (or EtherLink)
- EtherFabric
  - Models a crossbar connecting input and output ports
- ForwardEngine
  - Moves packets from input queues to output queues
  - Schedules new attempt in the future in case of contention
  - Has a map of MAC addresses to ports



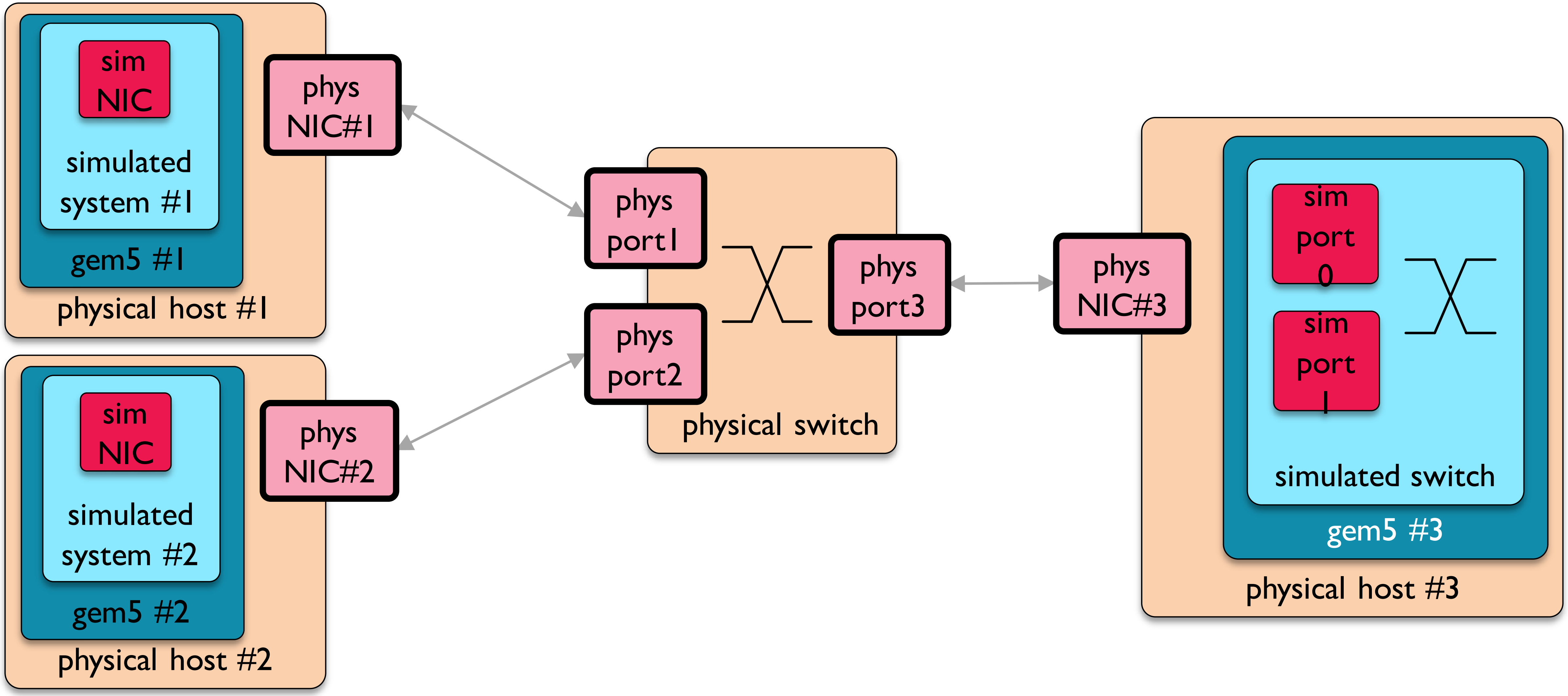
dist-gem5 architecture – packet forwarding



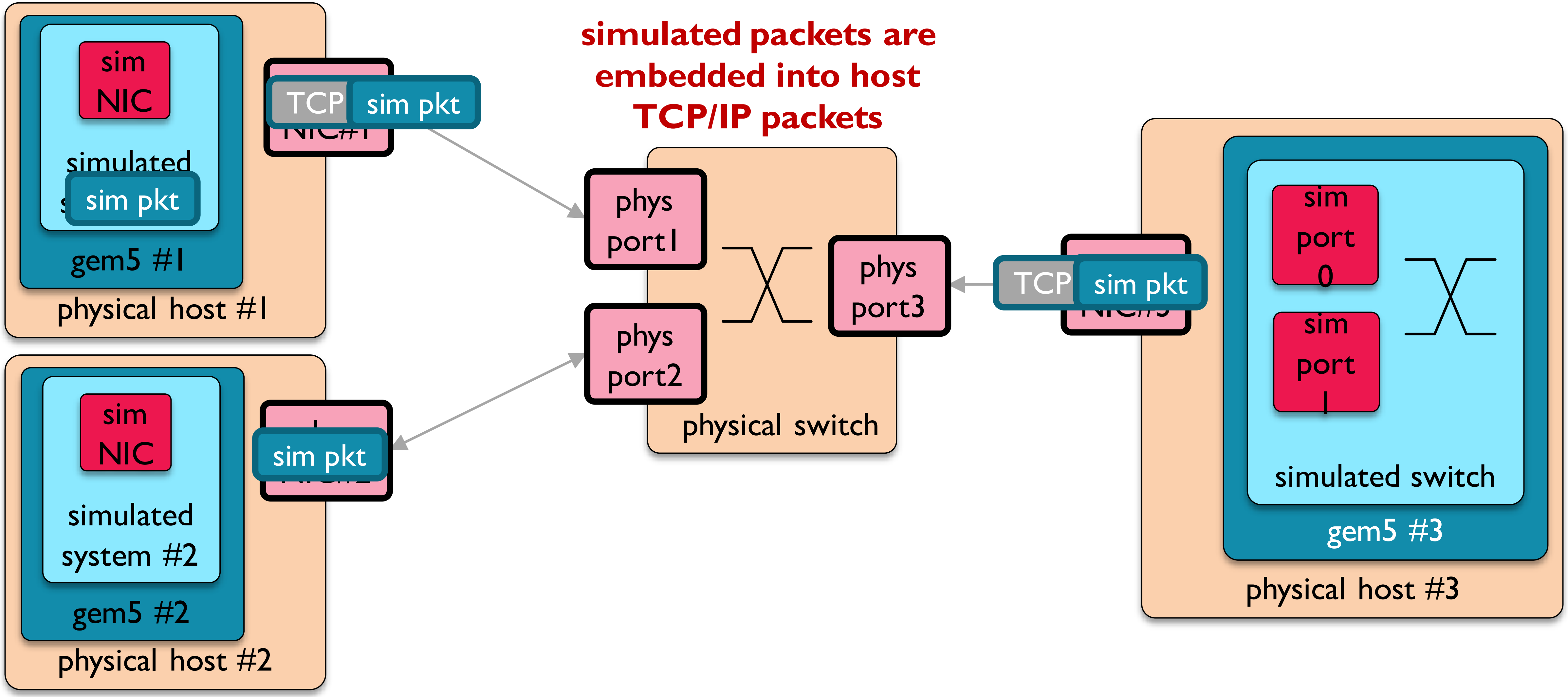
ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

dist-gem5 architecture – packet forwarding



dist-gem5 architecture – packet forwarding



ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

15 min break

Programme

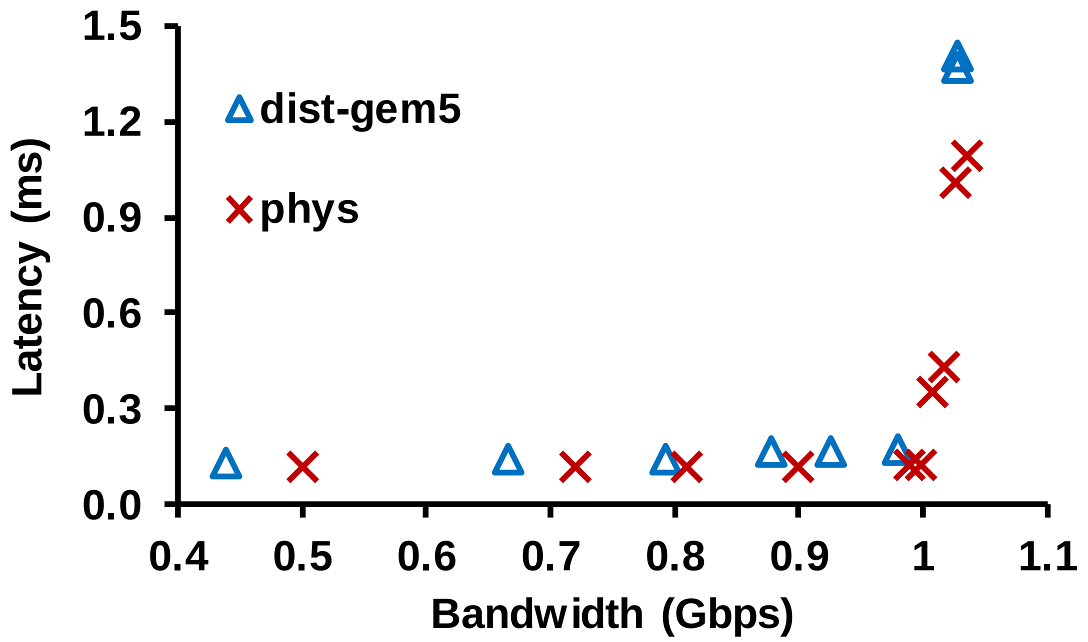
- Introduction (15min)
- Overview of gem5 (45 min)
- *— 15 min Break —*
- dist-gem5 deep-dive (60 min)
  - Packet forwarding
  - Synchronisation
  - Checkpointing
  - Deterministic execution
- *— 15 min Break —*
- Evaluation (30 min)
  - Validation and simulation scalability
  - Demo

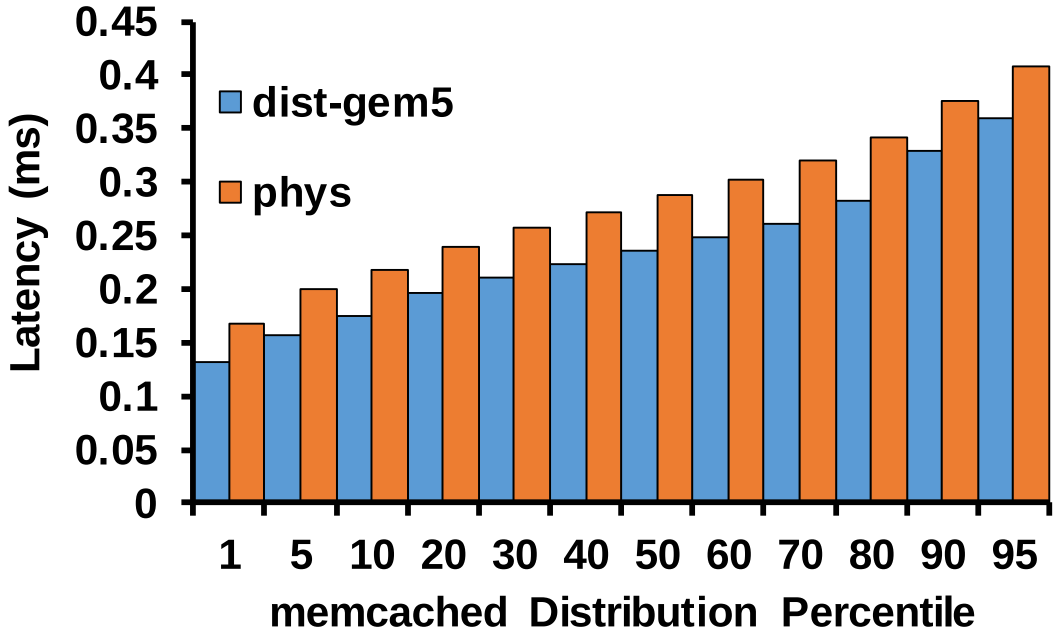
ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Validation – network latency and bandwidth

- iperf (left) and memcahed (right)
- follows the behavior of physical setup
- 17.5% lower response time for memcached





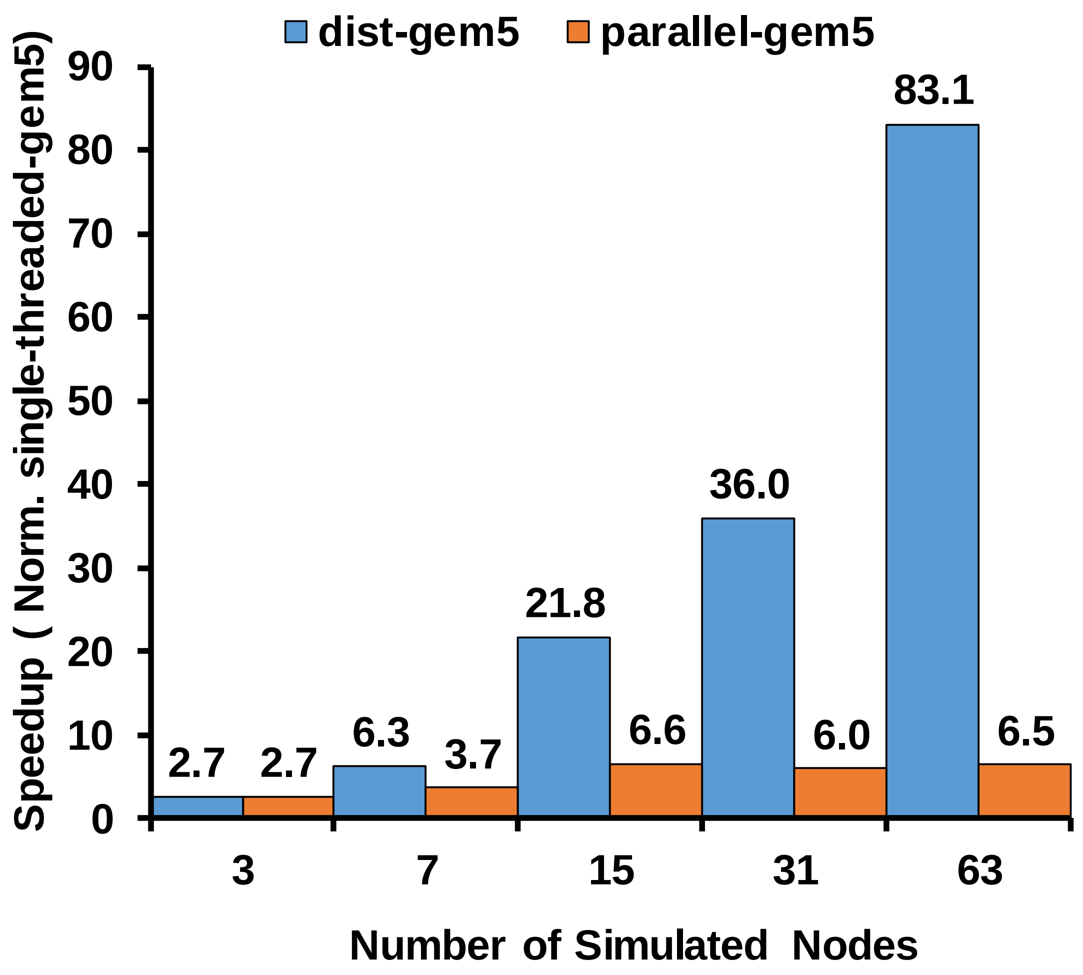
ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Speedup – simulation time reduction

- running httperf on each simulated node sending fixed number of requests to a unique simulated node (apache server)
- compared with single-threaded-gem5
- dist-gem5 simulating 63 nodes on 16 physical hosts is
  - 83.1× faster than single-threaded-gem5
  - 12.8× faster than parallel-gem5

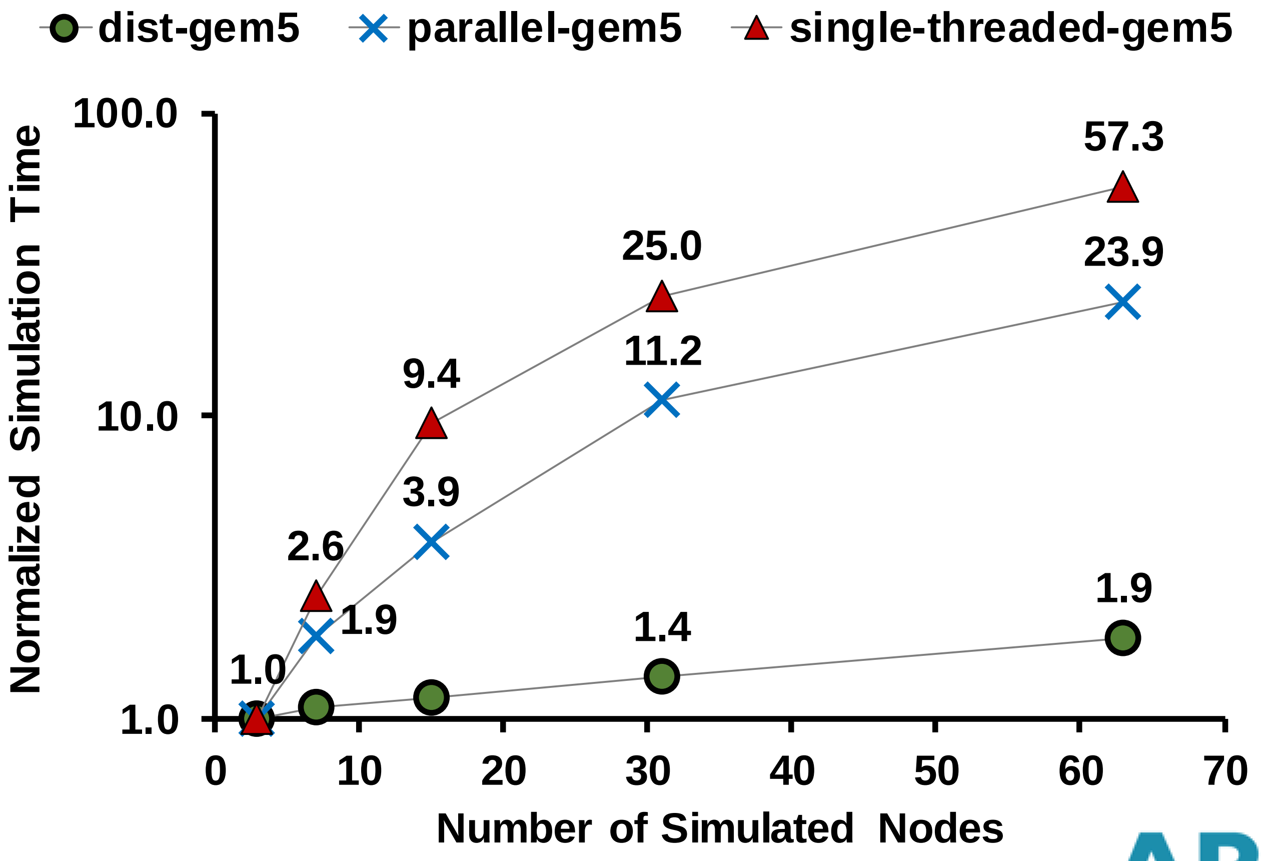
speedup of parallel-gem5 saturates!



Scalability – simulation time vs. simulated cluster size

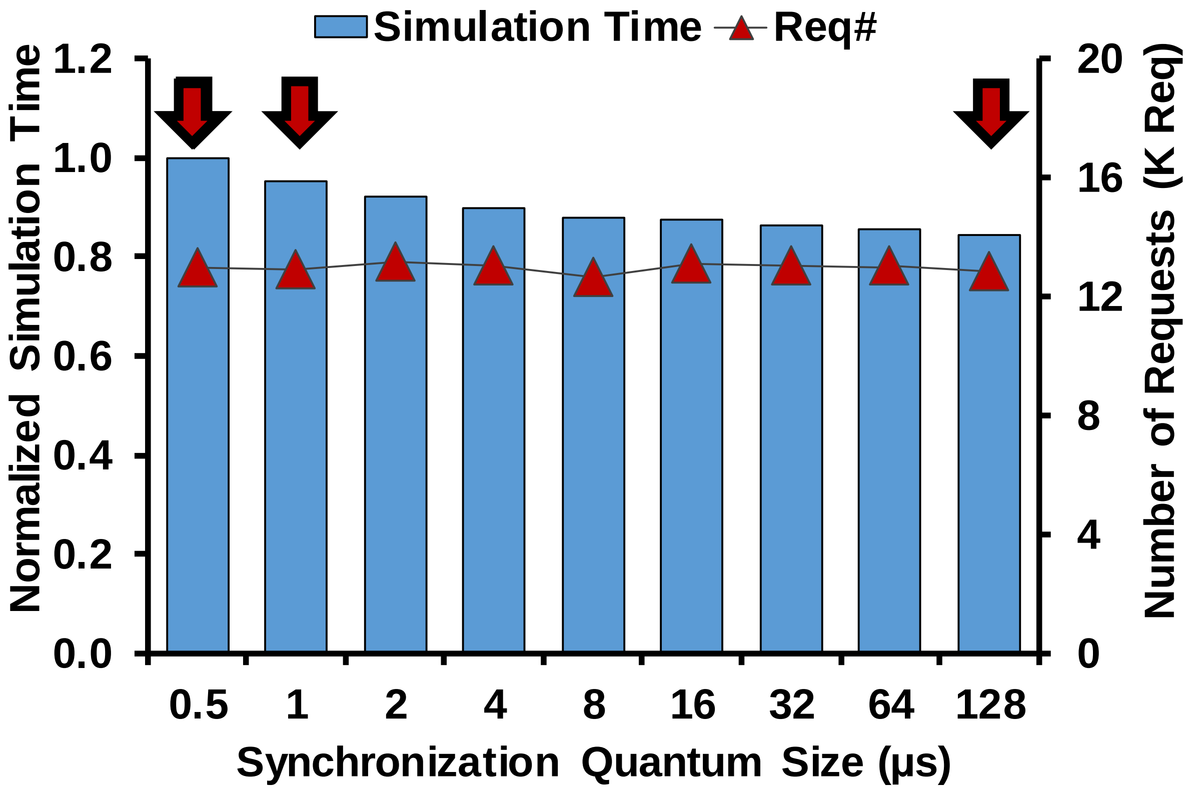
- simulation time increase for simulating 64 vs. 3 nodes:
  - 57.3× for Single-threaded-gem5
  - 23.9× for parallel-gem5
  - 1.9× for dist-gem5

dist-gem5 scales well!



Synchronization overhead

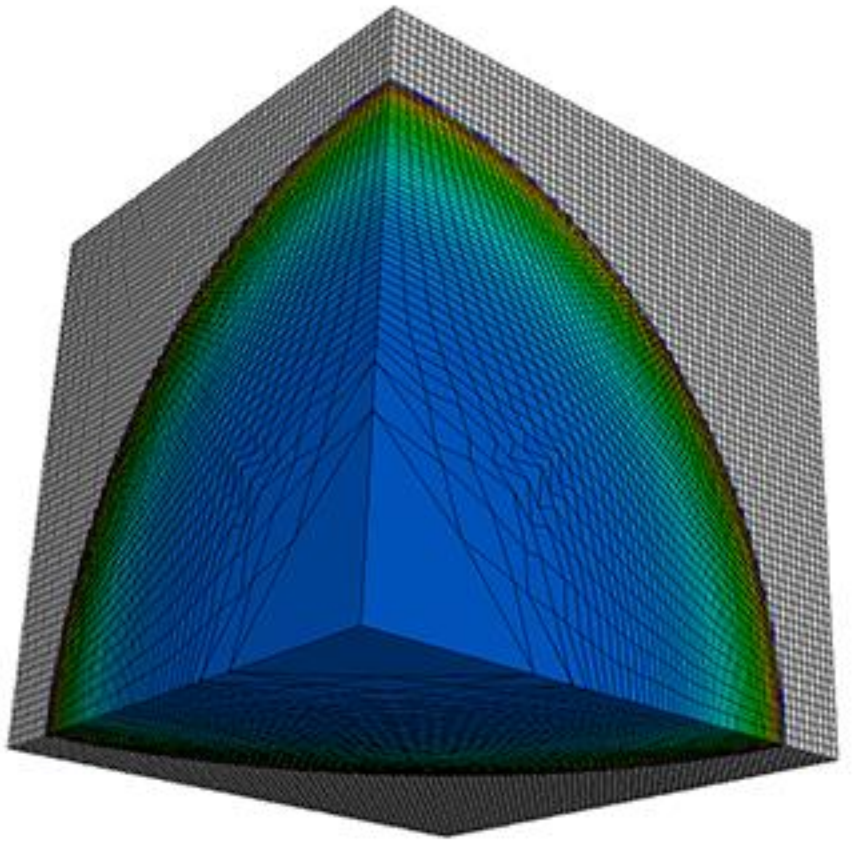
- sweep synchronization quantum size
- # of http req remains near constants
  - maximum 2.6% variance
  - almost the same amount of work done at each quantum size
- simulation time improvement
  - 4.9% from 0.5 µs to 1 µs
  - 15.7% from 0.5 µs to 128 µs



dist-gem5 synchronization is efficient!

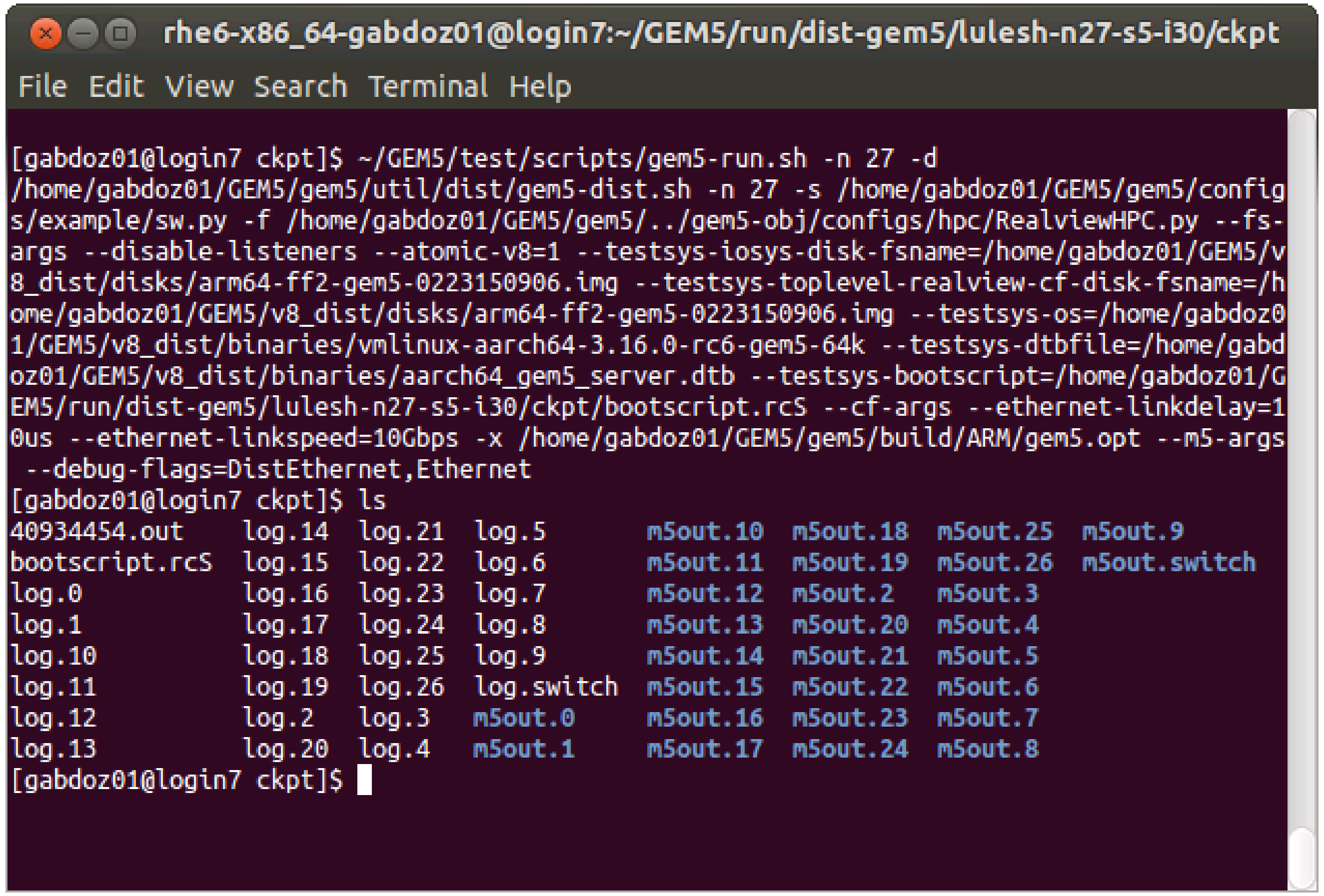
Case study : Network sensitivity of LULESH

- What is LULESH?
  - Livermore Unstructured Lagrange Explicit Shock Hydrodynamics
  - A widely studied proxy application in DOE co-design efforts for exascale
  - Modeling hydrodynamics, which describes the motion of materials relative to each other when subject to forces
  - Highly simplified application that represents a typical hydrocode
  - Ported to a number of programming models (MPI, OpenMP, CUDA, Chapel, Charm++, etc.)



Running LULESH on distributed gem5

- Compute node config
  - ARMv8 single core CPU @ 1GHz, 2 GB DRAM
  - Ethernet NIC
- Switch config
  - 27-port Ethernet xbar
  - 1KiB input/output buffer per port
- LULESH command line
  - mpirun –n 27 lulesh-mpi –s 5 –i 30
    - -s : input data size per MPI process
    - -i : number of iterations in the main compute loop


rhe6-x86_64-gabdoz01@login7:~/GEM5/run/dist-gem5/lulesh-n27-s5-i30/ckpt
File Edit View Search Terminal Help

```
[gabdoz01@login7 ckpt]$ ~/GEM5/test/scripts/gem5-run.sh -n 27 -d
/home/gabdoz01/GEM5/gem5/util/dist/gem5-dist.sh -n 27 -s /home/gabdoz01/GEM5/gem5/config
s/example/sw.py -f /home/gabdoz01/GEM5/gem5/../gem5-obj/configs/hpc/RealviewHPC.py --fs-
args --disable-listeners --atomic-v8=1 --testsys-iosys-disk-fsname=/home/gabdoz01/GEM5/v
8_dist/disks/arm64-ff2-gem5-0223150906.img --testsys-toplevel-realview-cf-disk-fsname=/h
ome/gabdoz01/GEM5/v8_dist/disks/arm64-ff2-gem5-0223150906.img --testsys-os=/home/gabdoz0
1/GEM5/v8_dist/binaries/vmlinux-aarch64-3.16.0-rc6-gem5-64k --testsys-dtbfile=/home/gabd
oz01/GEM5/v8_dist/binaries/aarch64_gem5_server.dtb --testsys-bootscript=/home/gabdoz01/G
EM5/run/dist-gem5/lulesh-n27-s5-i30/ckpt/bootscript.rcS --cf-args --ethernet-linkdelay=1
0us --ethernet-linkspeed=10Gbps -x /home/gabdoz01/GEM5/gem5/build/ARM/gem5.opt --m5-args
 --debug-flags=DistEthernet,Ethernet
[gabdoz01@login7 ckpt]$ ls
40934454.out    log.14  log.21  log.5       m5out.10  m5out.18  m5out.25  m5out.9
bootscript.rcS  log.15  log.22  log.6       m5out.11  m5out.19  m5out.26  m5out.switch
log.0           log.16  log.23  log.7       m5out.12  m5out.2   m5out.3
log.1           log.17  log.24  log.8       m5out.13  m5out.20  m5out.4
log.10          log.18  log.25  log.9       m5out.14  m5out.21  m5out.5
log.11          log.19  log.26  log.switch  m5out.15  m5out.22  m5out.6
log.12          log.2   log.3   m5out.0     m5out.16  m5out.23  m5out.7
log.13          log.20  log.4   m5out.1     m5out.17  m5out.24  m5out.8
[gabdoz01@login7 ckpt]$
```

Running LULESH on distributed gem5 (cont.)

- Source code instrumentation to capture ROI
  - ‘m5 checkpoint’ pseudo instruction was inserted before main compute loop
  - ‘m5 exit’ pseudo instruction was inserted after the main compute loop
  - ‘checkpoint’ and ’exit’ instructions can be collaborative : action is only taken when all participating gem5 processes complete the pseudo instruction

- Simulation runs
  1. Fast forwarding (atomic CPU) until the MPI_Barrier (before the ROI) was hit in all 27 processes
  2. Executing ROI in detailed (O3 CPU) mode by restoring from checkpoint
  - Change Ethernet link parameters at resume to explore latency/bandwith sensitivity

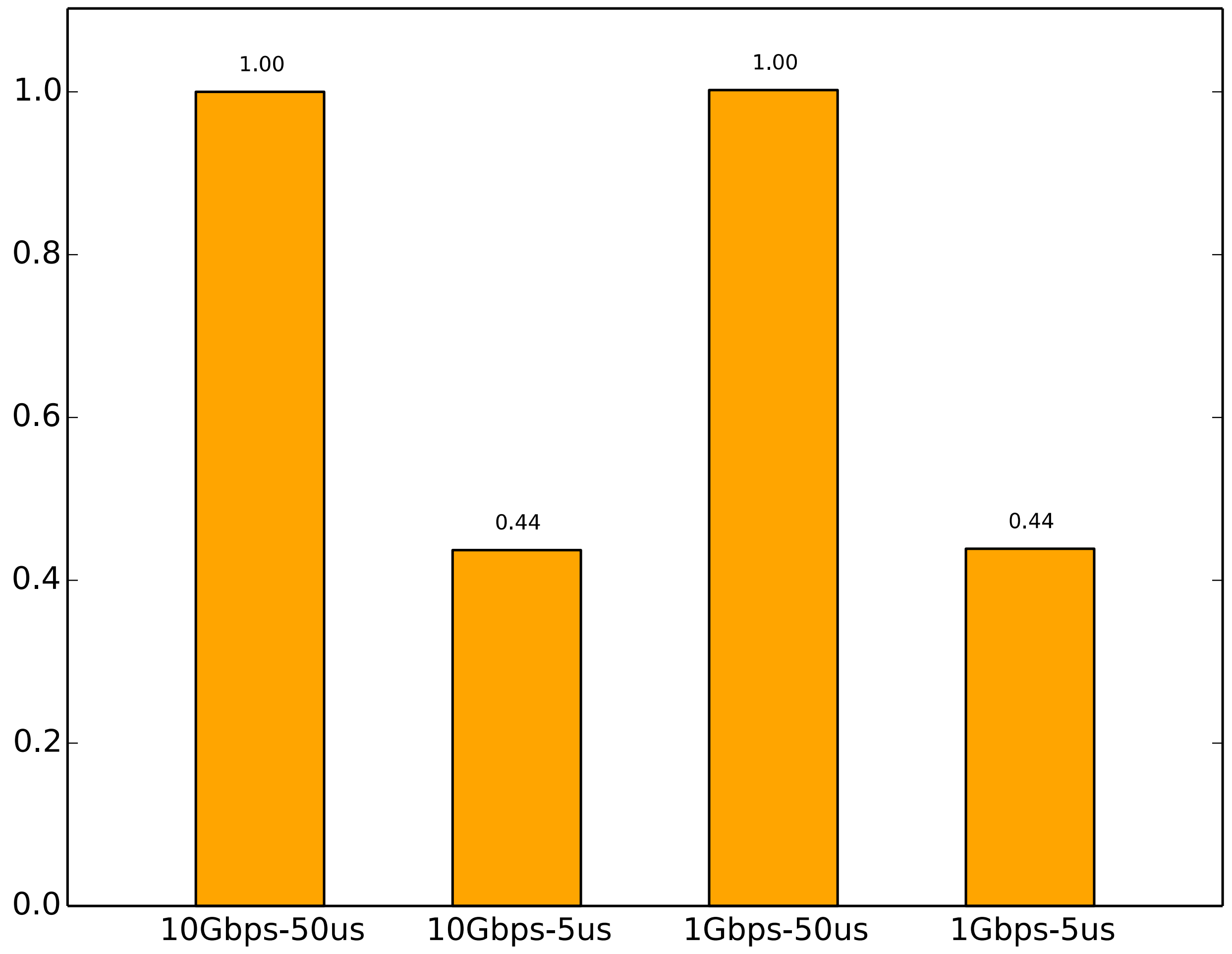
Ethernet link config	latency (us)	bandwidth (Gbps)
1.	50	10
2.	5	10
3.	50	1
4.	5	1

ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

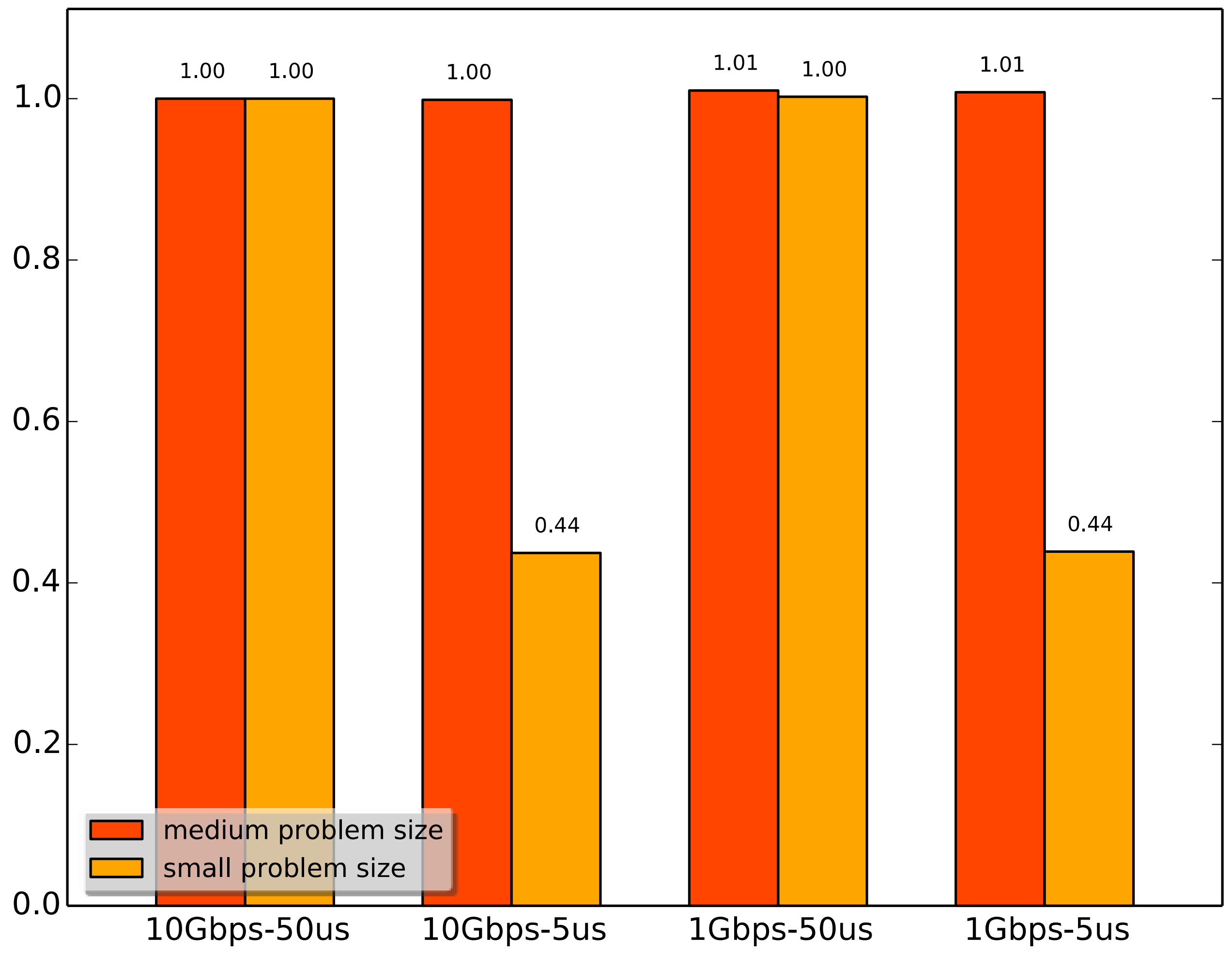
LULESH performance results – small input data size

- Performance is measured as run time of ROI
  - number of cycles from gem5 stats
  - max of the 27 compute nodes
- Results are normalized to the 1st config
  - 10Gbps bandwidth and 50us latency
- 5us link latency reduces run time by 55%



LULESH performance results – large vs. small input data size

- Results are normalized to the 1st config for both sets(10Gbps bandwidth and 50us latency)
- Sensitivity for link latency diminishes for large input data size
  - LULESH can overlap computation and communication



ECE ILLINOIS
Department of Electrical and Computer Engineering

ARM

Conclusions

- Distributed gem5 enables scalable simulations of distributed systems
- Integrated part of the gem5 simulator

- Collaboration between ARM Research and University of Illinois (ex-Wisconsin)
  - Prof. Nam Sung Kim (nskim@illinois.edu)
  - Mohammad Alian (malian2@illinois.edu)
  - Gabor Dozsa (gabor.dozsa@arm.com)
  - Stephan Diestelhorst (stephan.diestelhorst@arm.com)

ARM
Research Summit

11-13 September 2017
Robinson College, Cambridge, UK

Submission deadline - 30 April 2017
Early-bird discount ends - 30 June 2017