Interconnect Network

A presentation by

Marjan Fariborz

UCDAVIS COMPUTER SCIENCE

DArchR DAVIS ARCHITECTURE RESEARCH

Extend and compile gem5

From gem5-bootcamp-env run:

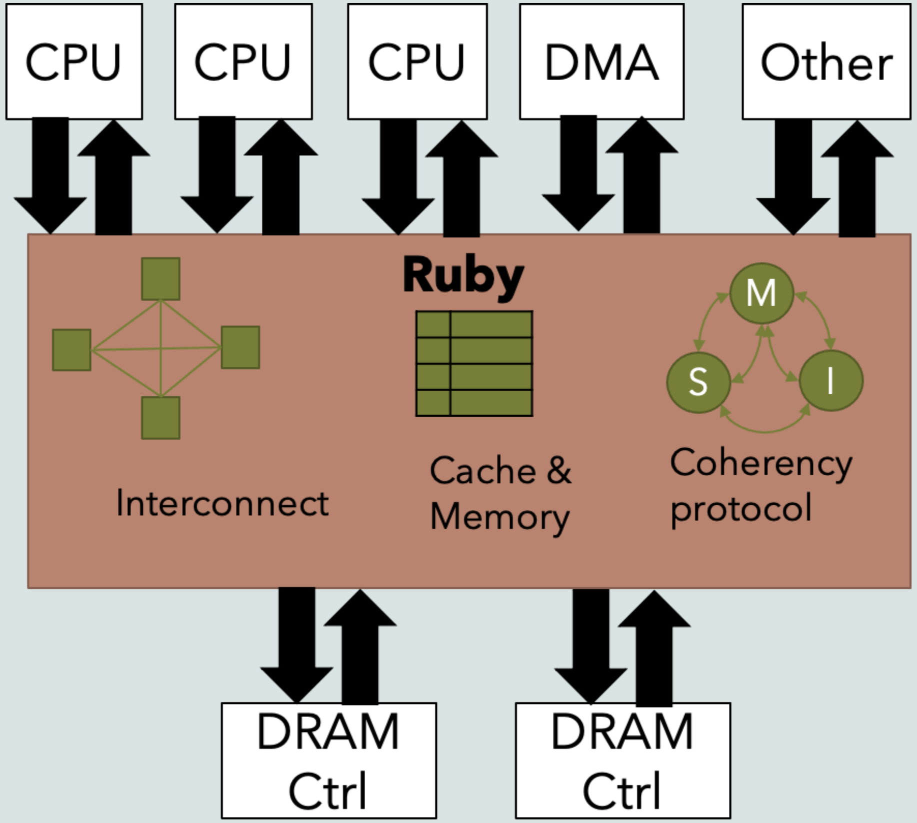
- “cp materials/developing-gem5-models/10-ruby-network/topologies/* gem5/src/python/gem5/components/cachehierarchies/ruby/topologies”
- “cp materials/developing-gem5-models/10-ruby-network/SConscript gem5/src/python”
- “cp materials/developing-gem5-models/10-ruby-network/mi_example_cache_network.py gem5/src/python/gem5/components/cachehierarchies/ruby”

From gem5-bootcamp-env/gem5/ run:

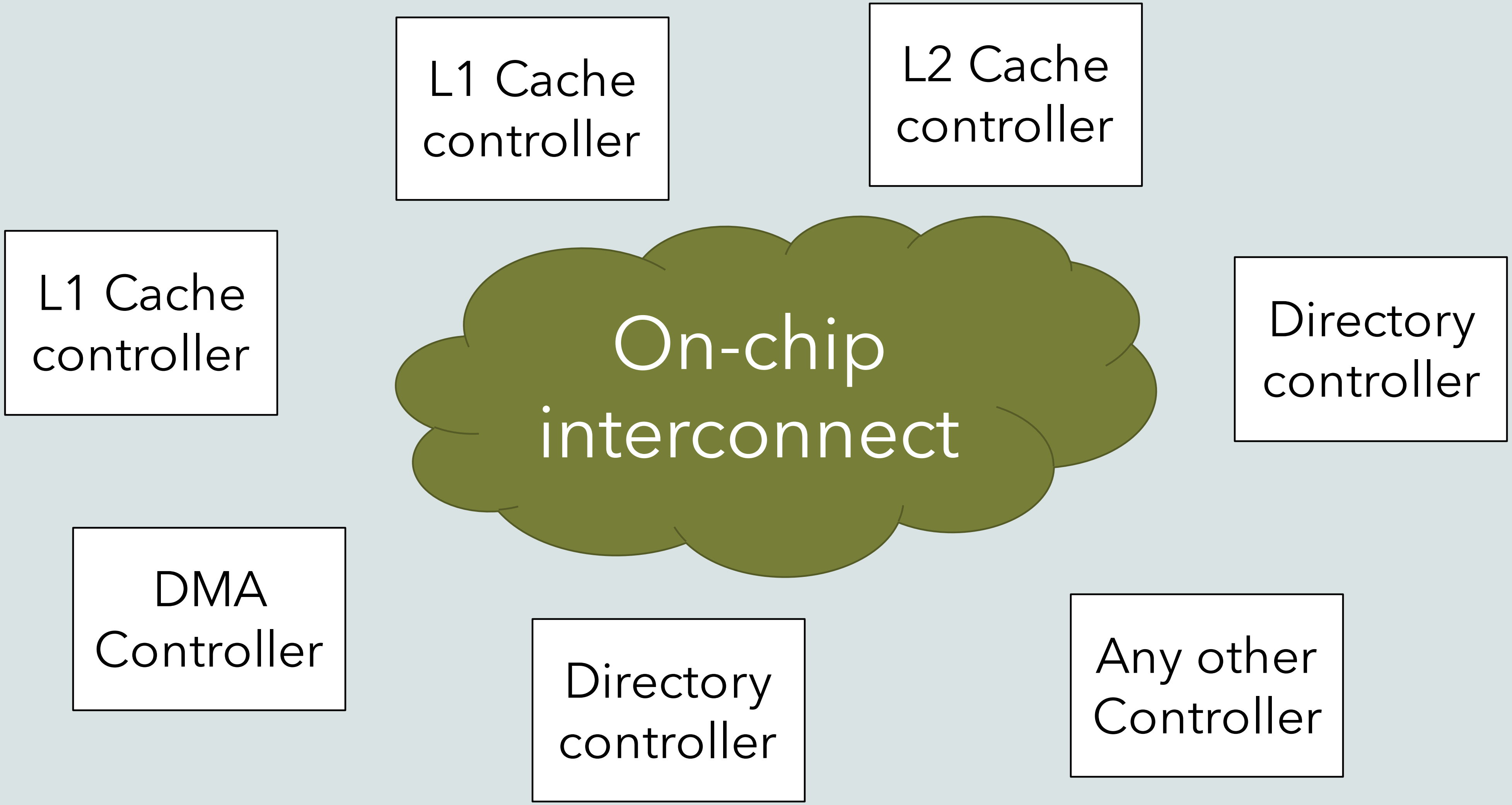
- “scons build/NULL/gem5.opt –j$(nproc)”

Review on Ruby

- **Controller models** (cache controller, directory controller)
- **Controller topology (Mesh, all-to-all, and etc.)**
- **Network models**
- **Interface** (classic ports)



Interconnect Network



Background

- As the number of on-chip cores increases, a scalable low-latency and high-bandwidth communication fabric to connect them becomes critically important
  - Crossbars
  - Buses
  - Network on chip

Background

- As the number of on-chip cores increases, a scalable low-latency and high-bandwidth communication fabric to connect them becomes critically important
  - **Crossbars**
  - **Buses**
  - Network on chip

Scale Poorly

Background

- As the number of on-chip cores increases, a scalable low-latency and high-bandwidth communication fabric to connect them becomes critically important
  - **Crossbars**
  - **Buses**

Scale Poorly

  - **Network on chip**
    - Topology
    - Routing
    - Flow control
    - Router microarchitecture
    - Link architecture

Types of network in gem5

Types of network:

- Simple network
  - Fast
  - Doesn’t have detailed parameters
    - Link Bandwidth and bandwidth
    - Router latency
- Garnet network
  - Detailed implementation of routers, links, and the flow control
  - More detailed statistics

Configuration

● **Cache/Dir controller**

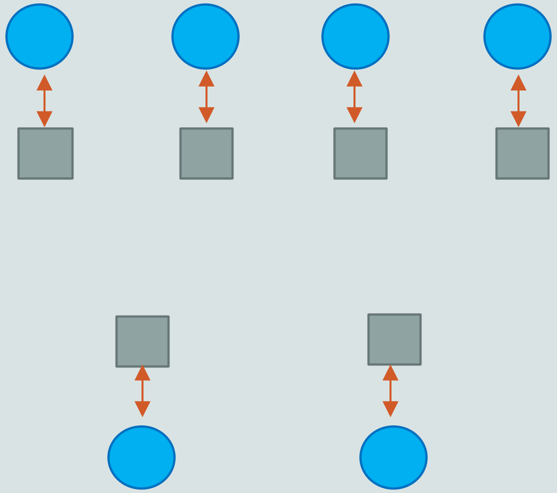
■ **Router**

↔ **External link (bi-directional)**

Connects each **Controller** to one **router** through an **External** link.

```
self.routers = [Switch(router_id = i) for i in range(len(controllers))]

self.ext_links = [SimpleExtLink(link_id=i, ext_node=c,
                                int_node=self.routers[i])
                  for i, c in enumerate(controllers)]
```



Configuration

● Cache/Dir controller

■ Router

↔ External link (bi-directional)

— Internal Link (single direction

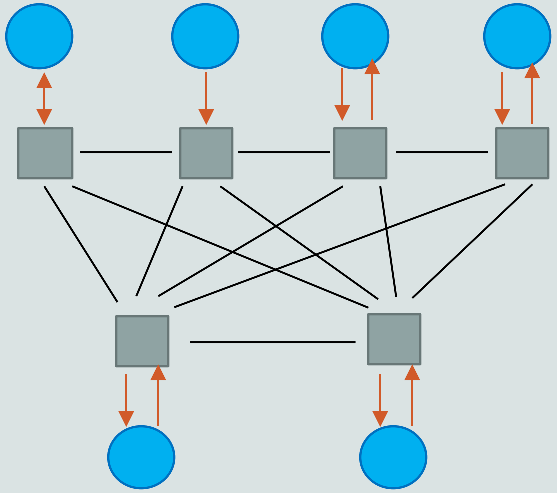
Connects each **Controller** to one **router** through an **External** link.

```
self.routers = [Switch(router_id = i) for i in range(len(controllers))]

self.ext_links = [SimpleExtLink(link_id=i, ext_node=c,
                                int_node=self.routers[i])
                  for i, c in enumerate(controllers)]
```

An **internal** link between each of the routers to every other router

```
self.int_links = []
for routeri in self.routers:
    for routerj in self.routers:
        if routeri == routerj : continue # Don't connect a router to itself!
self.int_links.append(SimpleIntLink(link_id = link_count,
                                    src_node = routeri,
                                    dst_node = routerj))
```



Configuration

● **Cache/Dir controller**

■ **Router**

↔ **External link (bi-directional)**

— **Internal Link (single direction**

Connects each **Controller** to one **router** through an **External** link.

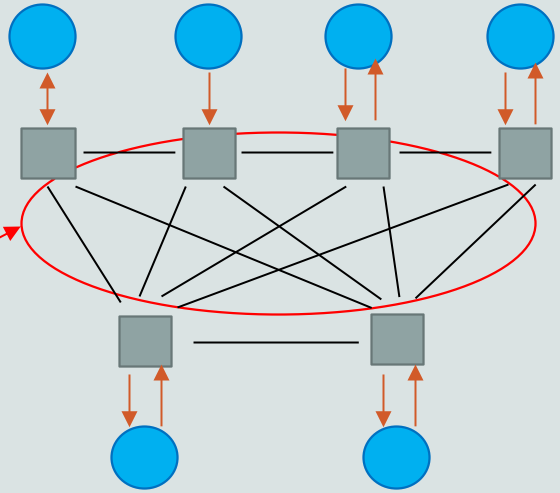
```
        self.routers = [Switch(router_id = i) for i in range(len(controllers))]

        self.ext_links = [SimpleExtLink(link_id=i, ext_node=c,
                                        int_node=self.routers[i])
                          for i, c in enumerate(controllers)]
```

An **internal** link between each of the routers to every other router

```
self.int_links = []
```

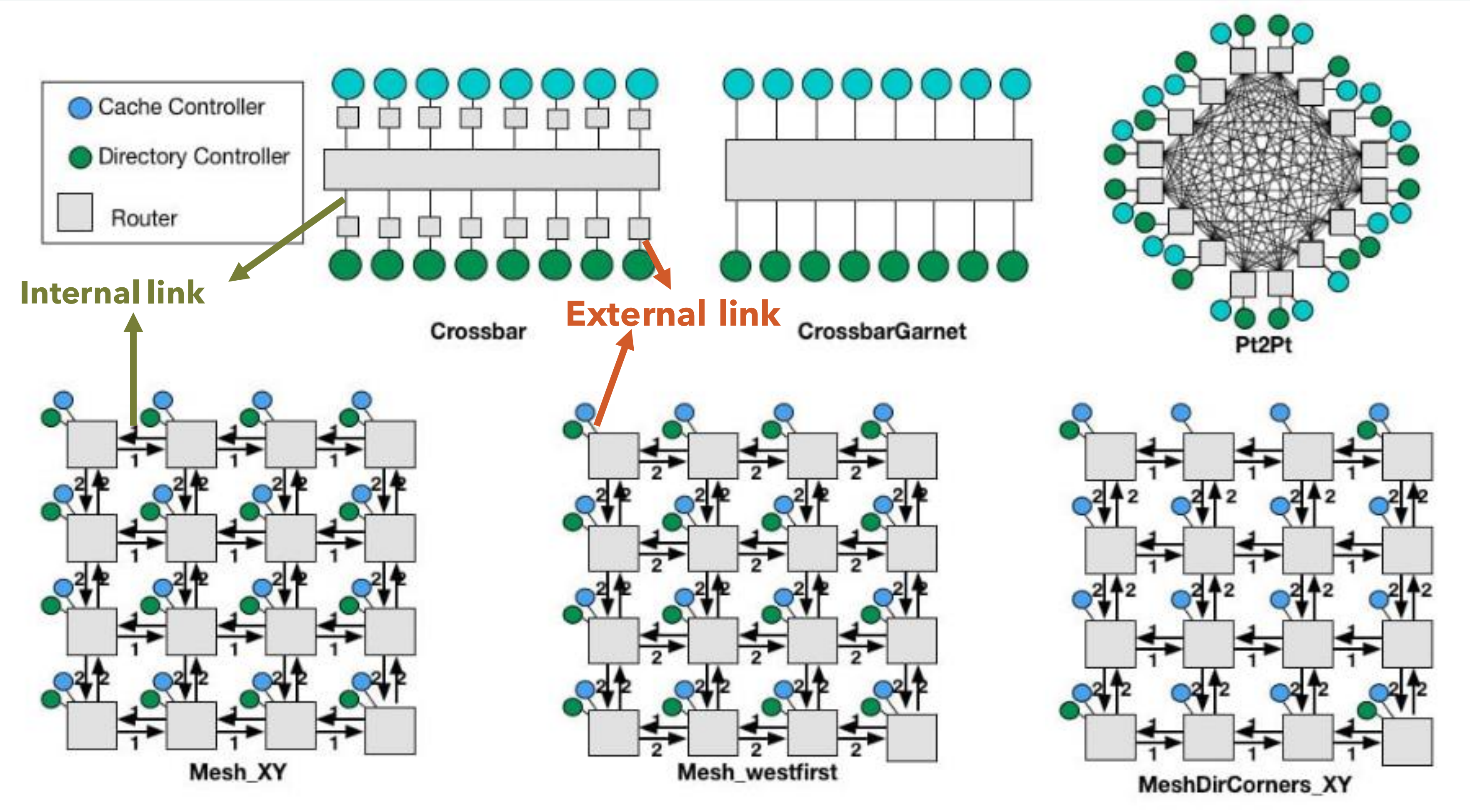
```
for routeri in self.routers:
    for routerj in self.routers:
        if routeri == routerj : continue # Don't connect a router to itself!
self.int_links.append(SimpleIntLink(link_id = link_count,
                                    src_node = routeri,
                                     dst_node = routerj))
```



Topology

Topology

How the routers are connected to each other



Router Microarchitecture

- Switch -> Simple network:
  - Router latency
  - Number of virtual networks
- Garnet Router -> Garnet network:
  - Number of virtual channels
  - Number of virtual networks
  - Size of network interface flits (flow control units)

Link Microarchitecture

- Simple network:
    - Just specifies the interface and bandwidth factor
  - Garnet network
    - separate links for data link and flow control links: Network and credit links
    - Supports clock domain crossing
    - Serialization and deserialization
    - Width of the link

Routing

- **Table-based Routing**
  - Shortest path
  - Chooses the route with minimum number of link traversals
  - Link weight impacts routing
- **Custom Routing algorithms**

Example: Garnet

- *Ruby- MI_Example coherency protocol*
- *4 cores (traffic generators)*
- *4 Private L1 cache*
- *1 Memory controller*
- *All-to-all topology*
- *USE STANDARD LIBRARY*

Garnet

From gem5-bootcamp-env run:

- “gem5/build/NULL/gem5.opt –re –outdir=results/Granet materials/developing-gem5-models/10-ruby-network/network_config.py 4 GarnetPt2Pt 512MiB”

- “gem5/build/NULL/gem5.opt –re –outdir=results/Simple materials/developing-gem5-models/10-ruby-network/network_config.py 4 SimplePt2Pt 512MiB”

Example: Garnet with Mesh topology

- *Ruby- MI_Example coherency protocol*
- *4 cores (traffic generators)*
- *4 Private L1 cache*
- *1 Memory controller*
- *2 Rows*

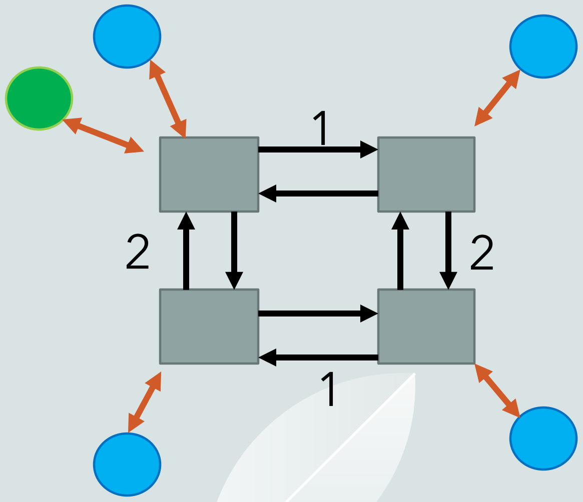
● **Directory controller**

● **Cache controller**

■ **Router**

↔ **External link (bi-directional)**

— **Internal Link (single direction**



Garnet

- From gem5-bootcamp-env run:
- “gem5/build/NULL/gem5.opt materials/developing-gem5-models/10-ruby-network/network_config.py 8 GarnetMesh 512MiB”