On-Chip Networks from a Networking Perspective:

Congestion and Scalability in Many-Core Interconnects

George Nychis†, Chris Fallin†, Thomas Moscibroda☆, Onur Mutlu†, Srinivasan Seshan†

Carnegie Mellon University †
Microsoft Research Asia ☆
What is the On-Chip Network?

Multi-core Processor (9-core)

- Cores
- Cache Banks
- Accelerometers
- Memory Controllers
What is the On-Chip Network?
What is the On-Chip Network?
Networking Challenges

- Familiar discussion in the architecture community, e.g.:
  - How to reduce **congestion**
  - How to **scale** the network
  - Choosing an effective **topology**
  - Routing and buffer size

All historical problems in our field...
Can We Apply Traditional Solutions? (1)

1. **Different constraints:** unique network design
2. **Different workloads:** unique style of traffic and flow

**Routing**

*min. complexity:*

X-Y-Routing, low latency
Can We Apply Traditional Solutions? (1)

1. **Different constraints:** unique network design
2. **Different workloads:** unique style of traffic and flow

**On-Chip Network (3x3)**

**Routing**
- min. complexity:
- X-Y-Routing,
- low latency

**Coordination**
- global is often less expensive
Can We Apply Traditional Solutions? (1)

1. **Different constraints:** unique network design
2. **Different workloads:** unique style of traffic and flow

**Bufferless**

- Area: -60%
- Power: -40%

**On-Chip Network (3x3)**

**Routing**

- min. complexity:
- X-Y-Routing,
- low latency

**Coordination**

- global is often less expensive

**Links**

- links cannot be over-provisioned
Can We Apply Traditional Solutions? (2)

1. **Different constraints:** unique network design
2. **Different workloads:** unique style of traffic and flow

**Bufferless**
- Area: -60%
- Power: -40%

**Closed-Loop**
- Insn Win Limits
- In-Flight Traffic Per-Core

**Routing**
- Min. complexity: X-Y-Routing,
  - low latency

**Coordination**
- Global is often less expensive

**Links**
- Links cannot be over-provisioned
Injection only when output link is free

On-Chip Network

S1

S2

D

age is initialized
**Injection** only when output link is free
Traffic and Congestion

- Injection only when output link is free
Traffic and Congestion

- Injection only when output link is free
Traffic and Congestion

- Injection only when output link is free
- Arbitration: oldest pkt first (dead/live-lock free)

Manifestation of Congestion

1. Deflection: arbitration causing non-optimal hop

contending for top port, oldest first, newest deflected
Traffic and Congestion

- **Injection** only when output link is free
- **Arbitration:** oldest pkt first (dead/live-lock free)

**Manifestation of Congestion**

1. **Deflection:** arbitration causing non-optimal hop

On-Chip Network

Can’t inject packet without a free output port
Traffic and Congestion

- **Injection** only when output link is free
- **Arbitration:** oldest packet first (dead/live-lock free)

**Manifestation of Congestion**

1. **Deflection:** arbitration causing non-optimal hop
2. **Starvation:** when a core cannot inject (no loss)

**Definition:** Starvation rate is a fraction of starved cycles

*Can’t inject packet without a free output port*
Outline

- Bufferless On-Chip Networks: Congestion & Scalability
  - Study of congestion at network and application layers
  - Impact of congestion on scalability

- Novel application-aware congestion control mechanism

- Evaluation of congestion control mechanism
  - Able to effectively scale the network
  - Improve system throughput up to 27%
Congestion and Scalability Study

• **Prior work**: moderate intensity workloads, small on-chip net
  - Energy and area benefits of going bufferless
  - Throughput comparable to buffered

• **Study**: high intensity workloads & large network (4096 cores)
  - Still comparable throughput with the benefits of bufferless?

• Use real application workloads (e.g., matlab, gcc, bzip2, perl)
  - Simulate the bufferless network and system components
  - Simulator used to publish in ISCA, MICRO, HPCA, NoCs…
Congestion at the Network Level

- Evaluate 700 different appl. mixes in 16-core system
- **Finding:** net latency remains stable with congestion/deflects
  - Unlike traditional networks
- What about starvation rate?
- Starvation increases significantly with congestion
- **Finding:** starvation likely to impact performance; indicator of congestion

![Graph showing 25% Increase in avg. net latency](image1)

![Graph showing 700% Increase in avg. starvation rate](image2)
Congestion at the Application Level

- Define system throughput as the sum of instructions-per-cycle (IPC) of all applications in the system.
- Unthrottle apps in single worker.
- **Finding 1:** Throughput decreases under congestion.
- **Finding 2:** Self-throttling of cores prevents collapse.
- **Finding 3:** Static throttling can provide some gain (e.g., 14%), but we will show up to 27% gain with app-aware throttling.

Instruction Throughput = $\sum_{i}^{N} IPC_i$

![Graph showing unthrottling applications with throttled and unthrottled states over average network utilization and instruction throughput.]
Impact of Congestion on Scalability

- Prior work: 16-64 cores
  Our work: up to 4096 cores

- As we increase system’s size:
  - Starvation rate increases
    - A core can be starved up to 37% of all cycles!
  - Per-node throughput decreases with system’s size
    - Up to 38% reduction
Summary of Congestion Study

• Network congestion limits scalability and performance
  - Due to *starvation rate*, *not* increased network latency
  - Starvation rate is the indicator of congestion in on-chip net

• Self-throttling nature of cores *prevent congestion collapse*

• Throttling: reduced congestion, improved performance
  - Motivation for *congestion control*

*Congestion control must be application-aware*
Outline

- Bufferless On-Chip Networks: Congestion & Scalability
  - Study of congestion at network and application layers
  - Impact of congestion on scalability

- Novel application-aware congestion control mechanism

- Evaluation of congestion control mechanism
  - Able to effectively scale the network
  - Improve system throughput up to 27%
Developing a Congestion Controller

- Traditional congestion controllers designed to:
  - Improve network efficiency
  - Maintain fairness of network access
  - Provide stability (and avoid collapse)
  - Operate in a distributed manner

- A controller for on-chip networks must:
  - Have minimal complexity
  - Be area-efficient
  - We show: be application-aware

When Considering:

On-Chip Network
Developing a Congestion Controller

- Traditional congestion controllers designed to:
  - Improve network **efficiency**
  - Maintain **fairness** of network access

When Considering:

- A controller for on-chip networks must:
  - Have **minimal complexity**
  - Be **area-efficient**
  - We show: be **application-aware**

... in paper: global and simple controller
Need For Application Awareness

- Throttling reduces congestion, improves system throughput
  - *Under congestion, what core should be throttled?*
- Use 16-core system, alternate 90% throttle rate to applications
- **Finding 1**: the app that is throttled impacts system performance
- **Finding 2**: application throughput does not dictate who to throttle
- **Finding 3**: different applications respond differently to an increase in network throughput (unlike gromacs, mcf barely gains)
Instructions-Per-Packet (IPP)

Key Insight: Not all packets are created equal
- more L1 misses \implies more traffic to progress

Instructions-Per-Packet (IPP) = I/P

IPP only depends on the L1 miss rate
- independent of the level of congestion & execution rate
- low value: need many flits to retire insns, shift window
- provides the application-layer insight needed

Since L1 miss rate varies over execution, IPP is dynamic
Instructions-Per-Packet (IPP)

- Since L1 miss rate varies over execution, IPP is dynamic
  - Throttling during a “high” IPP phase will hurt performance

Phase behavior on the order of millions of cycles
Instructions-Per-Packet (IPP)

- Since L1 miss rate varies over execution, IPP is dynamic
  - Throttling during a “high” IPP phase will hurt performance

- IPP provides application-layer insight in who to throttle
  - Dynamic IPP → Throttling must be dynamic

- When congested: throttle applications with low IPP

- Fairness: scale throttling rate by application’s IPP
  - Details in paper show that fairness in throttling
Outline

- Bufferless On-Chip Network: Congestion & Scalability
  - Study of congestion at network and application layers
  - Impact of congestion on scalability

- Novel application-aware congestion control mechanism

- Evaluation of congestion control mechanism
  - Able to effectively scale the network
  - Improve system throughput up to 27%
Evaluation of Improved Efficiency

- Evaluate with 875 real workloads (700 16-core, 175 64-core)
  - generate balanced set of CMP workloads (cloud computing)
- Improvement up to 27% under congested workloads
- Does not degrade non-congested workloads
- Only 4/875 workloads have perform. reduced > 0.5%
- Do not unfairly throttle applications down (in paper)
Evaluation of Improved Scalability

- Comparison points…
  - **Baseline bufferless**: doesn’t scale
  - **Buffered**: area/power expensive

- **Contribution**: keep area and power benefits of bufferless, while achieving comparable performance
  - Application-aware throttling
  - Overall reduction in congestion

- Power consumption reduced through increase in net efficiency

- Many other results in paper, e.g., Fairness, starvation, latency…
Summary of Study, Results, and Conclusions

• Highlighted a traditional networking problem in a new context
  - Unique design requires novel solution

• We showed congestion limited efficiency and scalability, and that self-throttling nature of cores prevents collapse

• Study showed congestion control would require app-awareness

• Our application-aware congestion controller provided:
  - A more efficient network-layer (reduced latency)
  - Improvements in system throughput (up to 27%)
  - Effectively scale the CMP (shown for up to 4096 cores)
Discussion

• Congestion is just one of many similarities, *discussion in paper*, e.g.,
  - **Traffic Engineering**: multi-threaded workloads w/ “hotspots”
  - **Data Centers**: similar topology, dynamic routing & computation
  - **Coding**: “XOR’s In-The-Air” adapted to the on-chip network:
    • i.e., instead of deflecting 1 of 2 packets, XOR the packets and forward the combination over the optimal hop