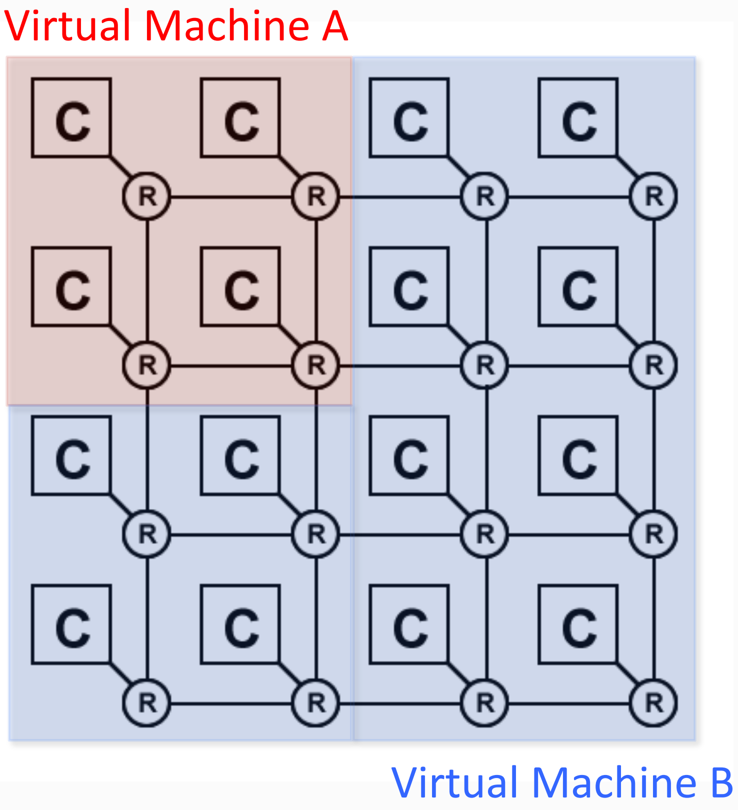
Efficient Timing Channel Protection for On-Chip Networks

Yao Wang and G. Edward Suh

Cornell University

On-Chip Networks are Shared Resources

- Future large-scale multi-cores will be shared among multiple applications / virtual machines



Problem: Timing Channels

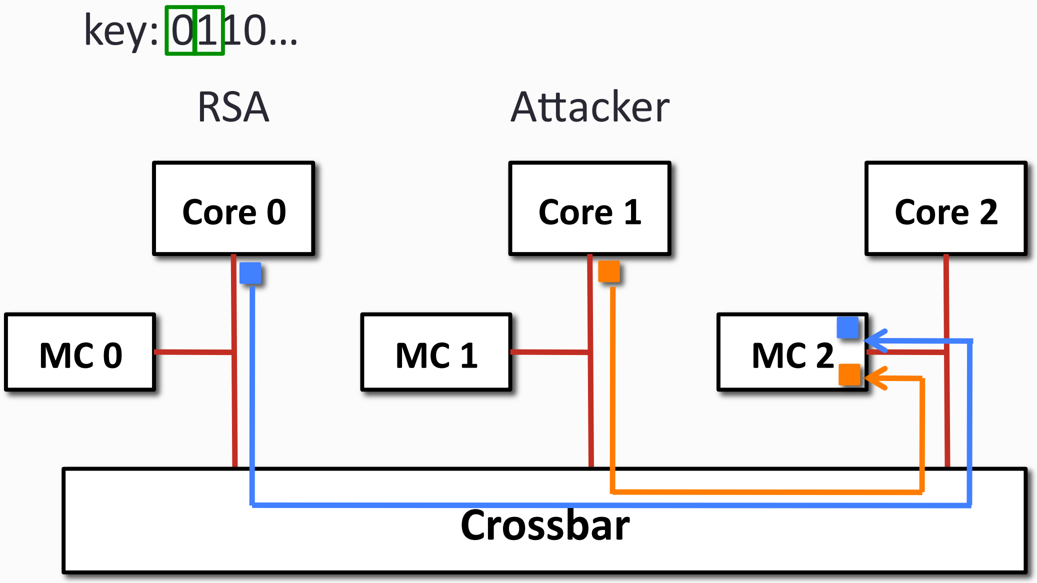
- Shared NoC causes interference

- Network interference introduces timing channels
  - Side channel
  - Covert channel

- High assurance systems requires security guarantee
  - Example: Corporate virtual machines on the cloud

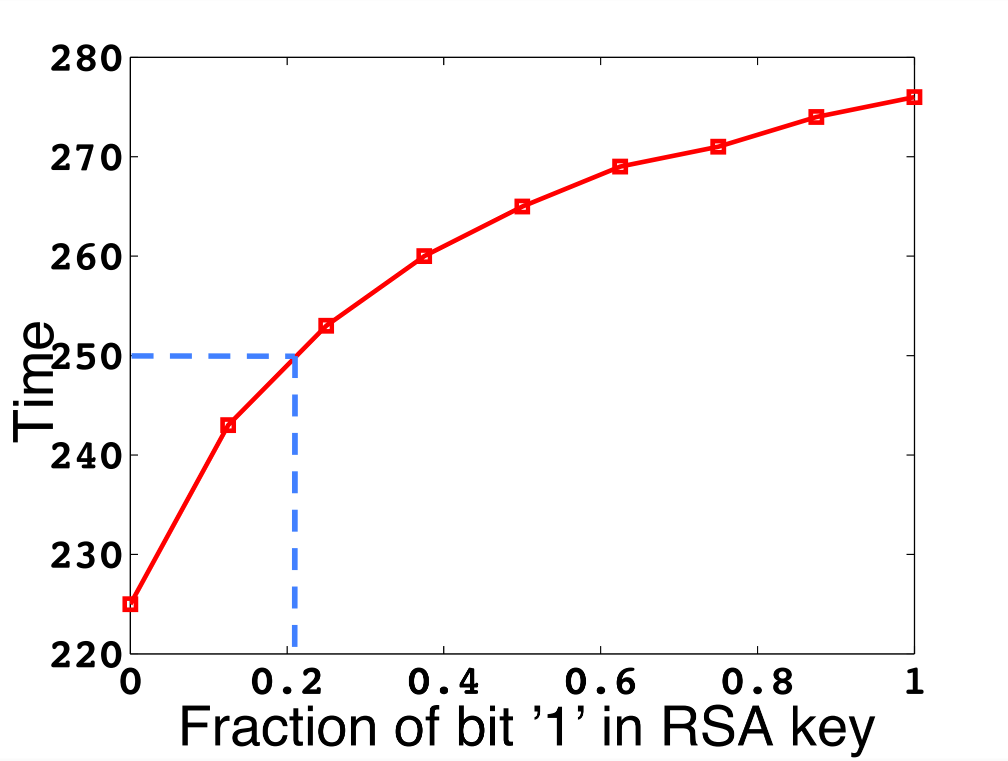
RSA Example

- RSA : a public key cryptographic algorithm
  - Prone to timing channel attacks



RSA Example

- RSA : a public key cryptographic algorithm
  - Prone to timing channel attacks



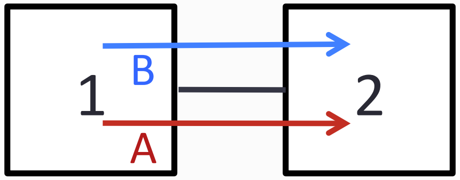
Outline

- Objective: Eliminate timing channels through the shared on-chip networks
  - Completely eliminate information leakage
  - Low performance overhead

- Rest of the talk
  - Potential approaches
  - Our solution
  - Evaluation
  - Related work
  - Conclusion

Use Quality-of-Service?

- QoS techniques provide performance isolation to different network flows
- QoS techniques are not enough for security
  - A flow can use bandwidth beyond its allocation
  - Bandwidth utilization reveals the flow demand



Bandwidth allocation

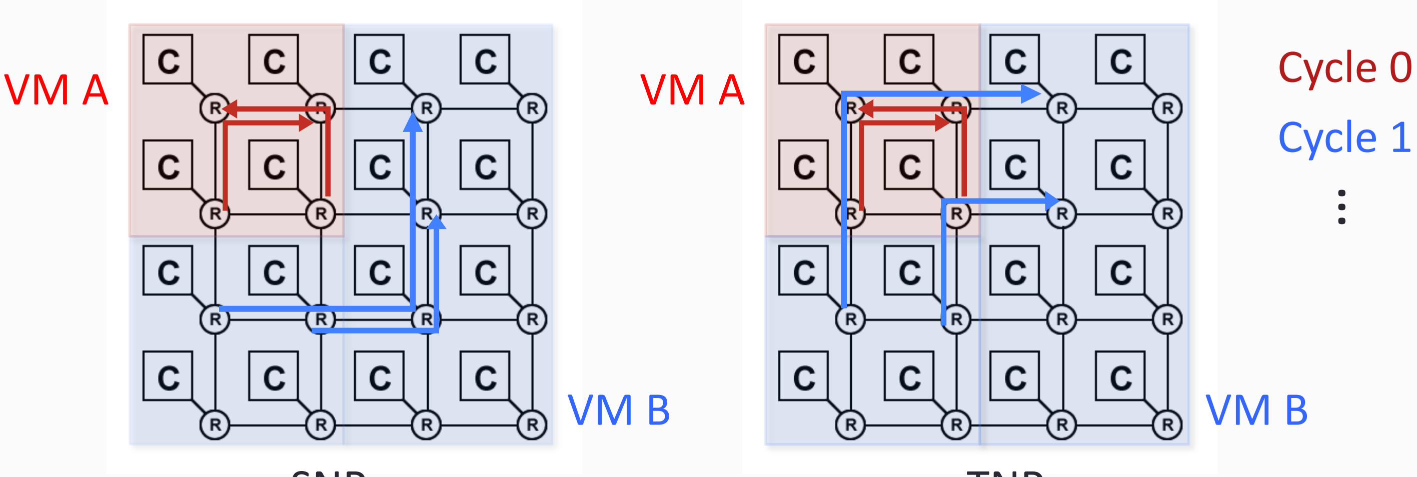
A: 50%

B: 50%

Flow A Demand	Flow B Demand	Flow A BW utilization
100%	100%	50%
100%	0%	100%

Static Partitioning

- To eliminate timing channels, resource allocation cannot depend on run-time demands
- Static partitioning
  - Spatial Network Partitioning (SNP)
  - Temporal Network Partitioning (TNP)



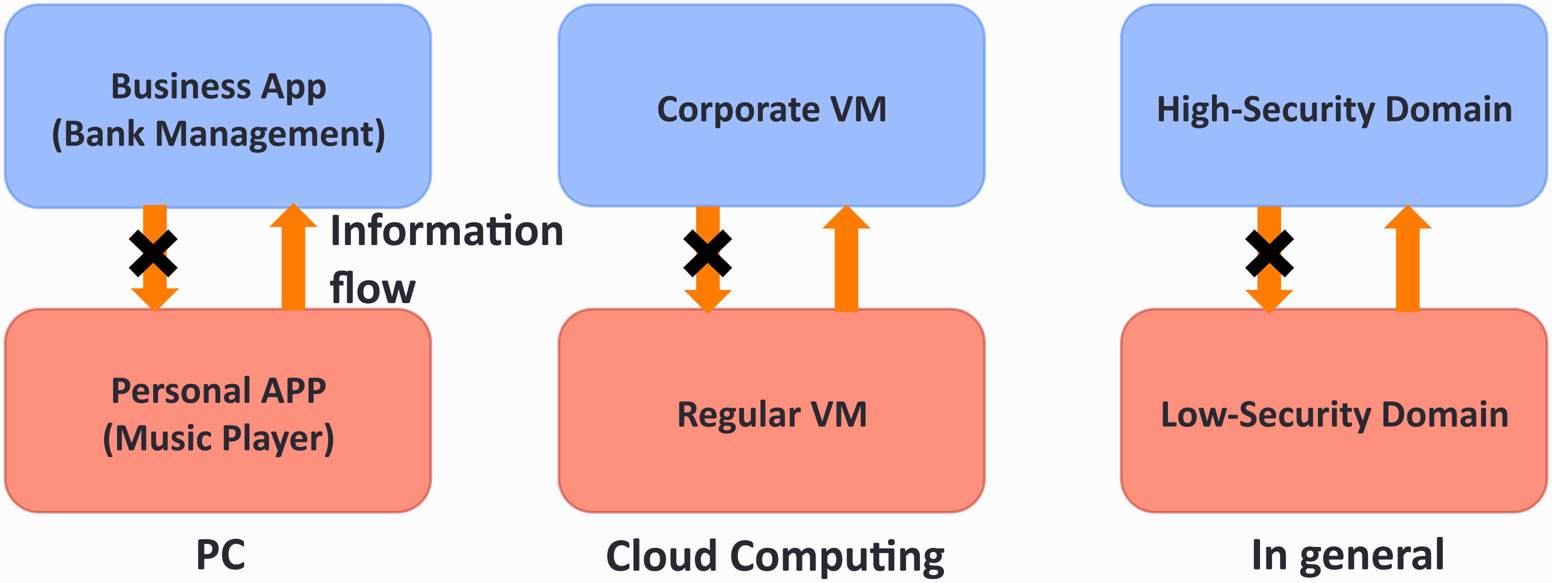
SNP

TNP

- Completely eliminate the timing channels
  - High performance overhead

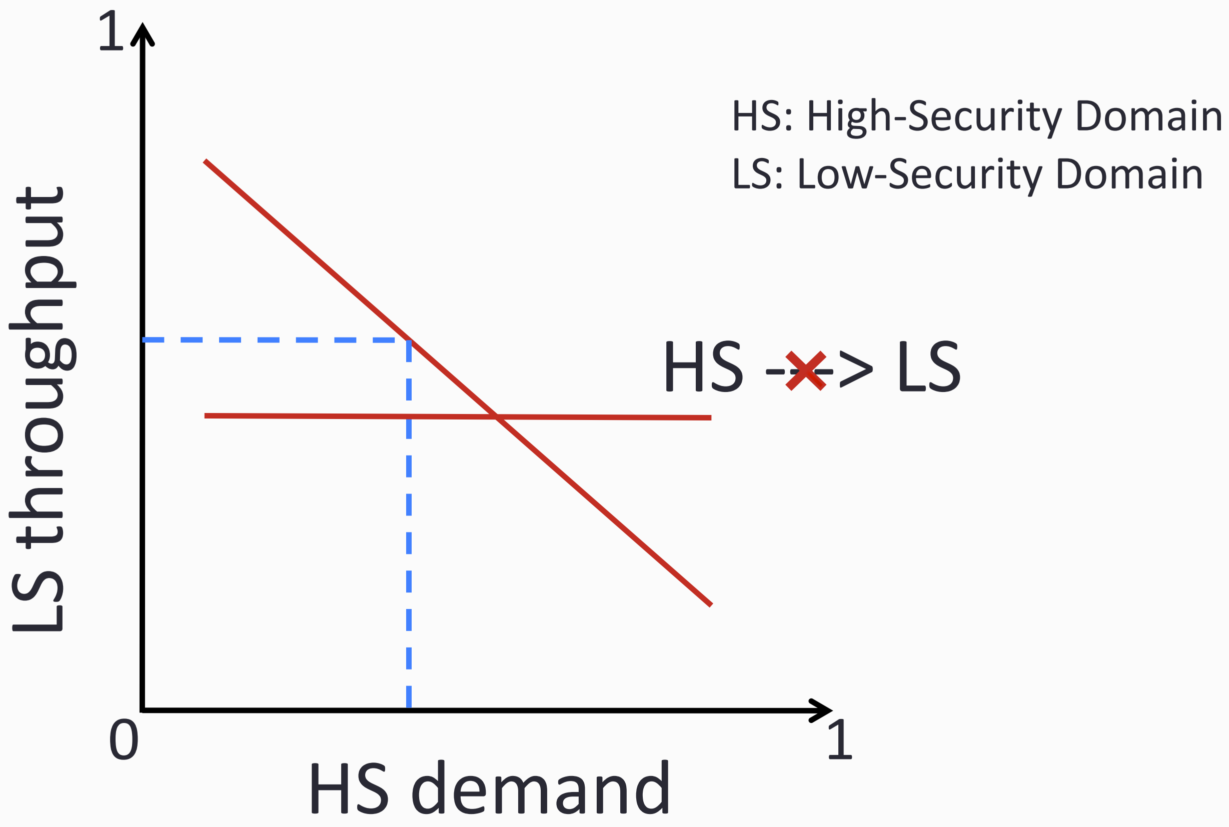
One-Way Information Leak Protection

- Usually only one-way information protection is needed
  - Multi-level security (MLS) model



- One-way protection is the key for **efficient** timing channel protection

Timing Channel through NoC



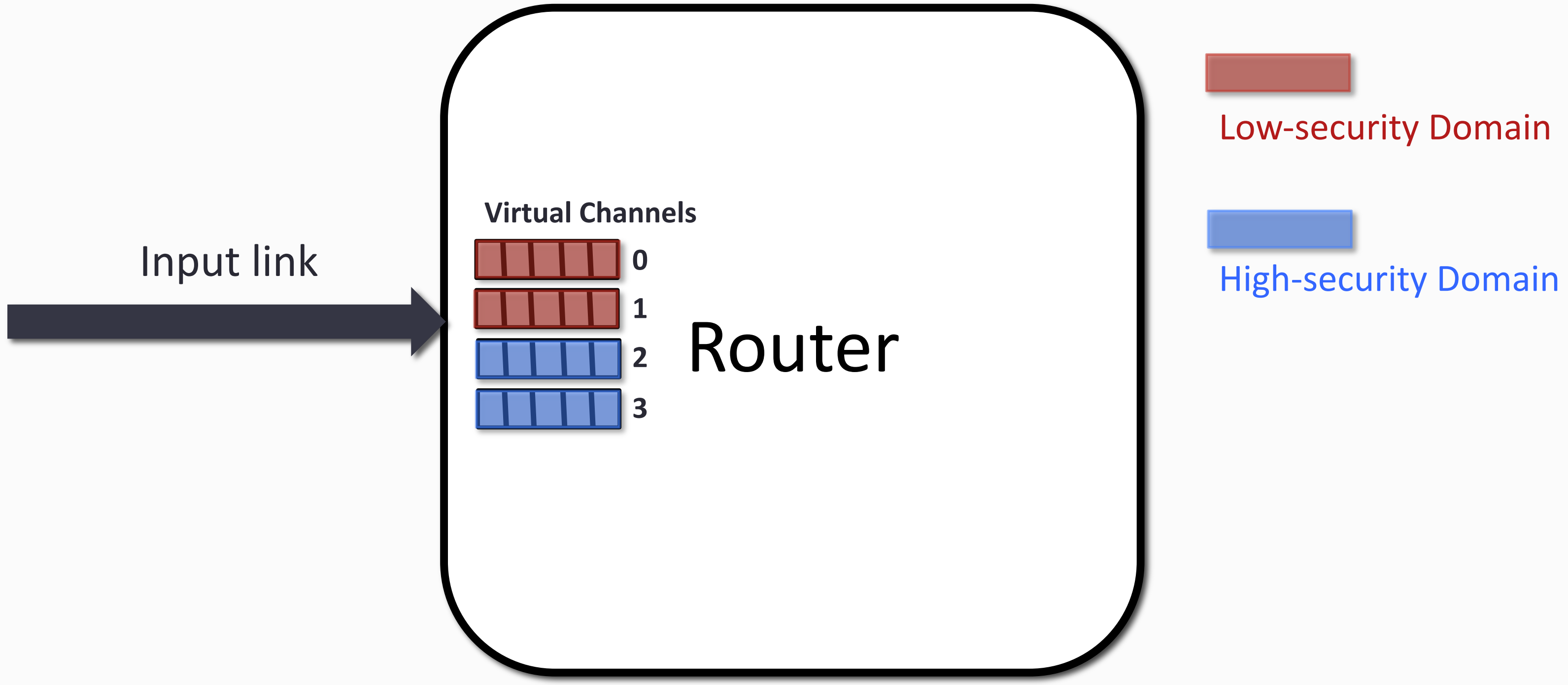
HS: High-Security Domain
LS: Low-Security Domain

Reversed Priority with Static Limits (RPSL)

- Reversed Priority
  - Assign high priority to low-security domain
  - The behavior (throughput, latency) of low-security domain is not affected by high-security domain
- Static Limits
  - Low-security domain could initialize Denial-of-Service (DoS) attack
  - Static limit controls the amount of traffic that low-security domain can send during a certain interval

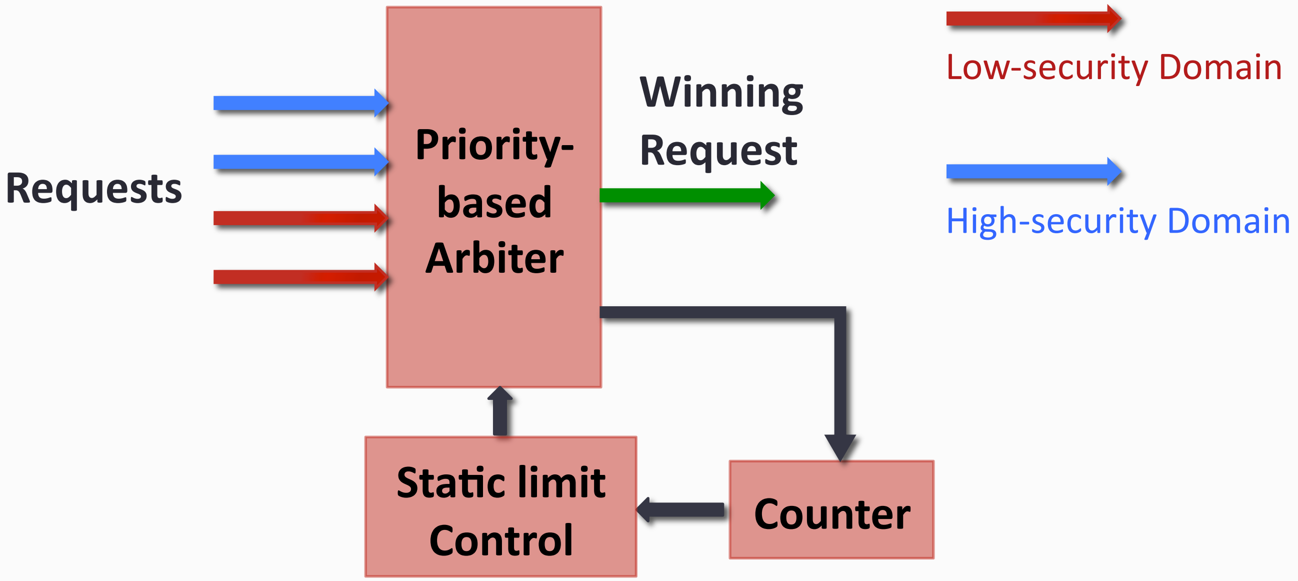
Implementation: Avoid Interference

- Priority-based separable allocator
  - Input arbiter & Output arbiter
- Static virtual channel allocation
  - Avoid head-of-line blocking



Implementation: Avoid DoS

- Static limit control mechanism
  - Counter & Control logic



Requests

Priority-based Arbiter

Winning Request

Static limit Control

Counter

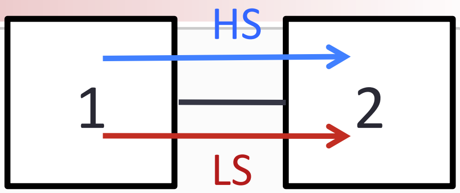
Low-security Domain

High-security Domain

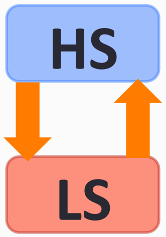
Input Arbiter

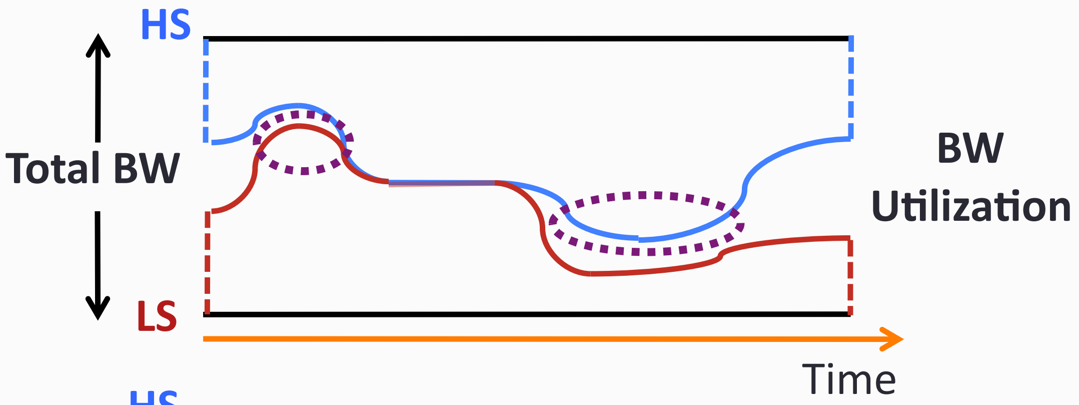
- Apply to both input and output arbiter

Benefits of One-Way Protection

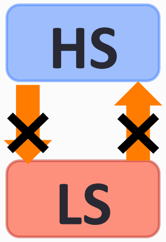


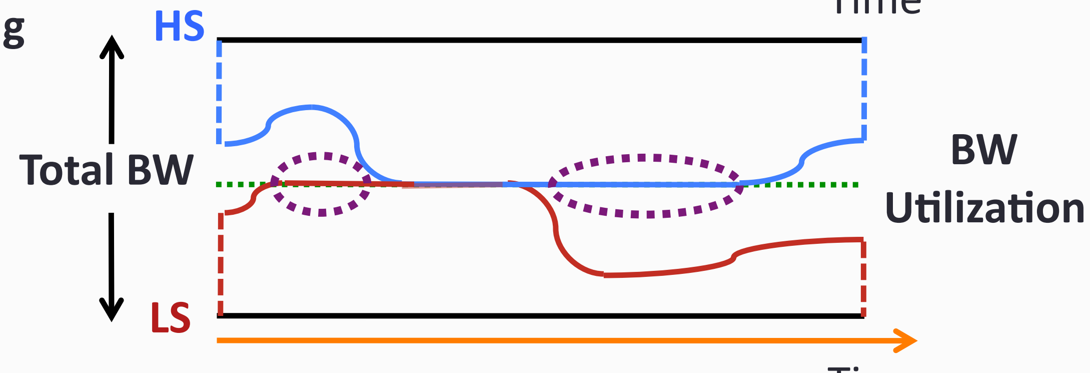
Round-robin Allocator



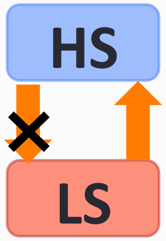


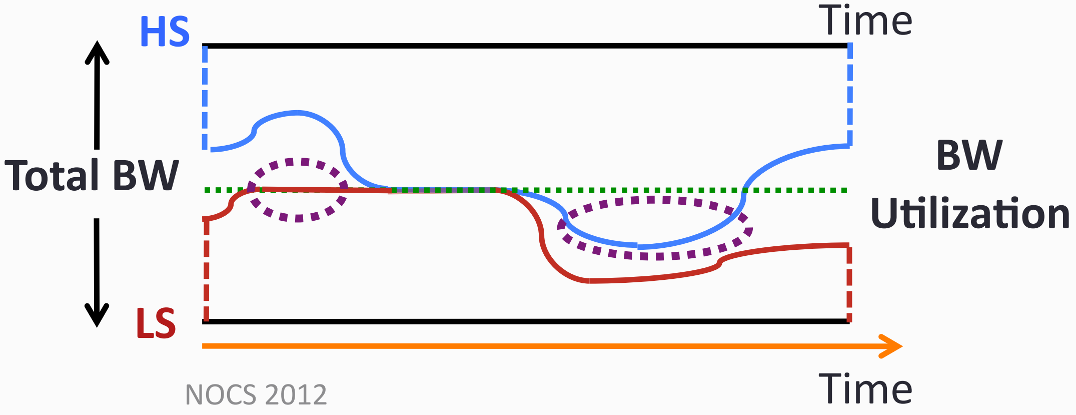
Temporal Network Partitioning





RPSL



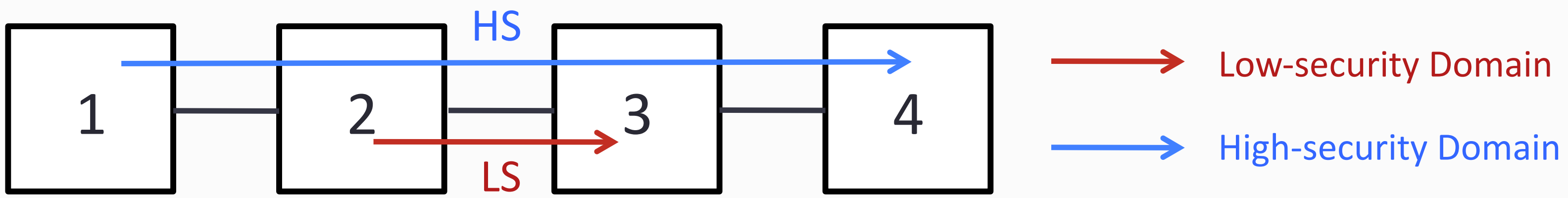


Experimental Setup

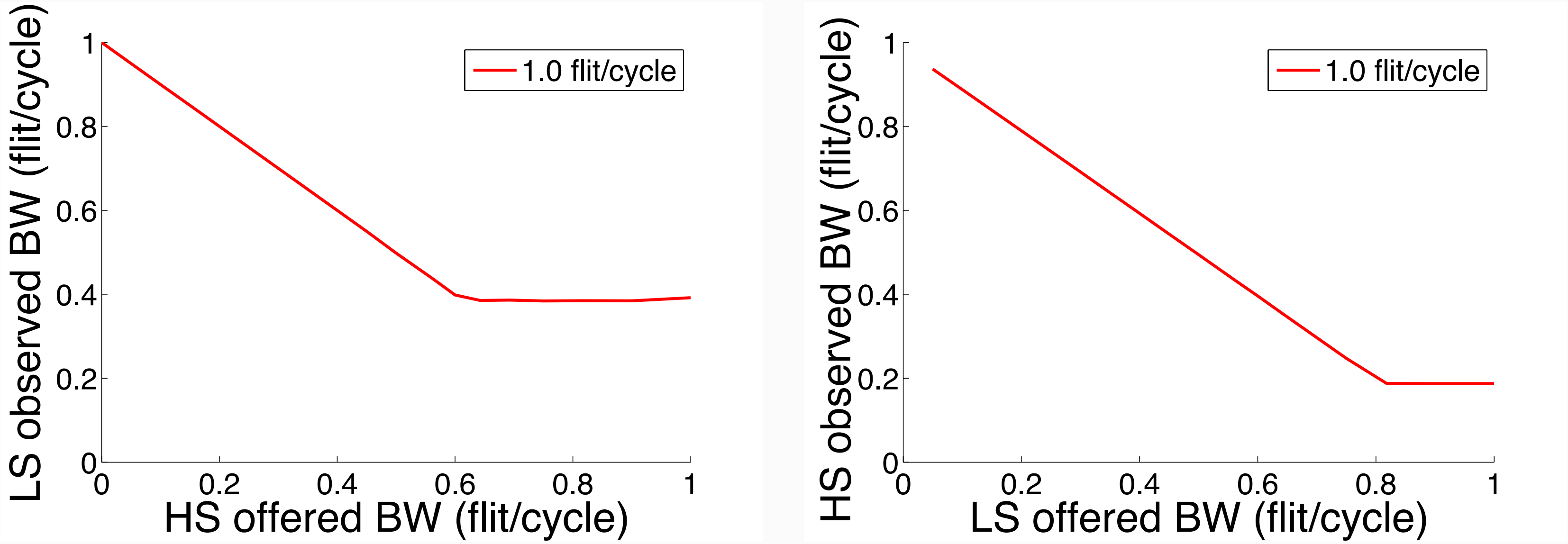
- Goals of experiments
  - Timing channel protection
  - DoS protection
  - Performance overhead
- Darsim: cycle-level NoC simulator
- Comparison of three schemes
  - Round-robin allocator (ISLIP)
  - Temporal Network Partitioning (TNP)
  - Reversed Priority with Static Limits (RPSL)

Timing Channel: No Protection

- Simple network

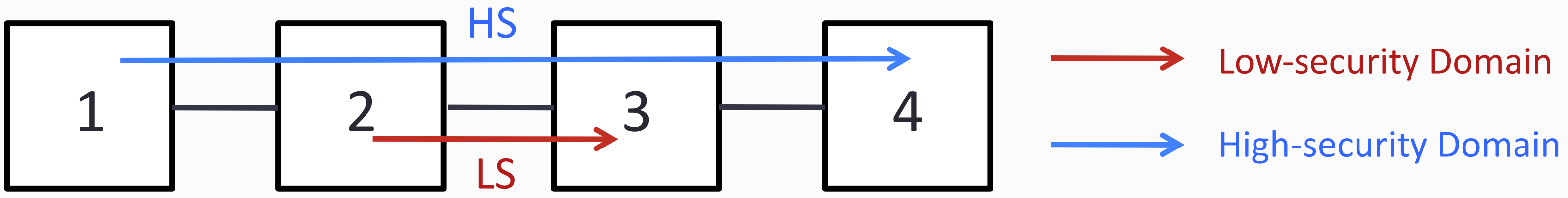


- Round-robin allocator

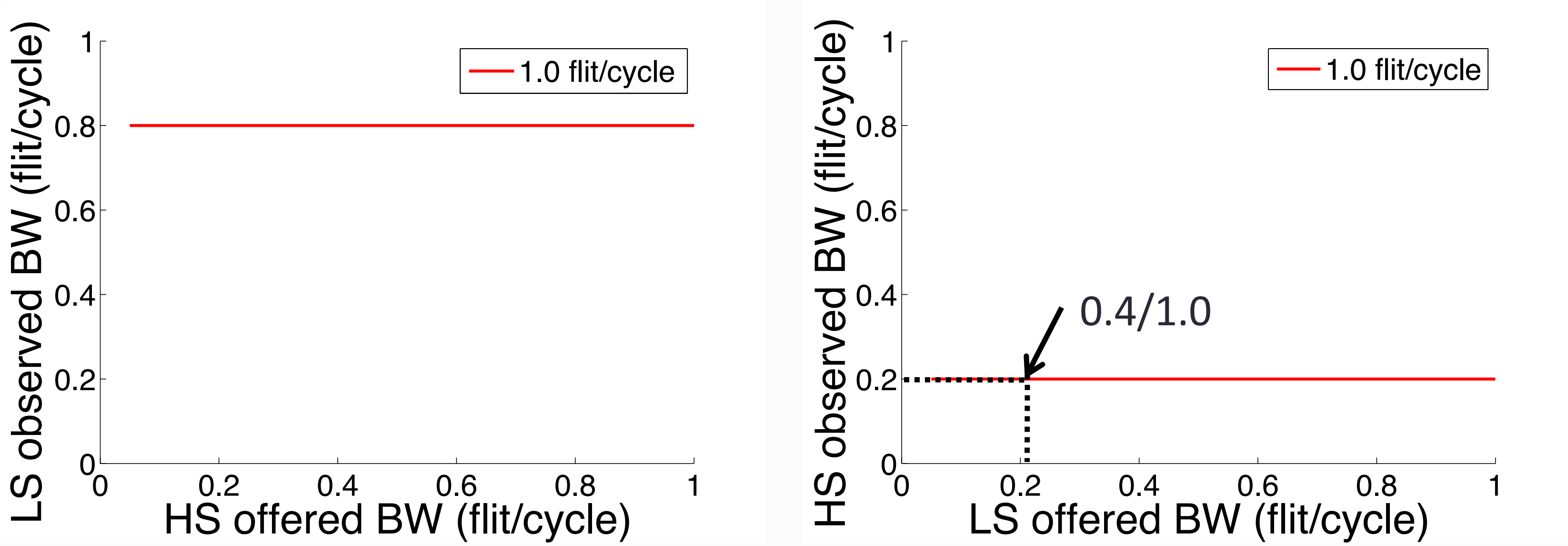


Timing Channel: Two-way Protection

- Simple network

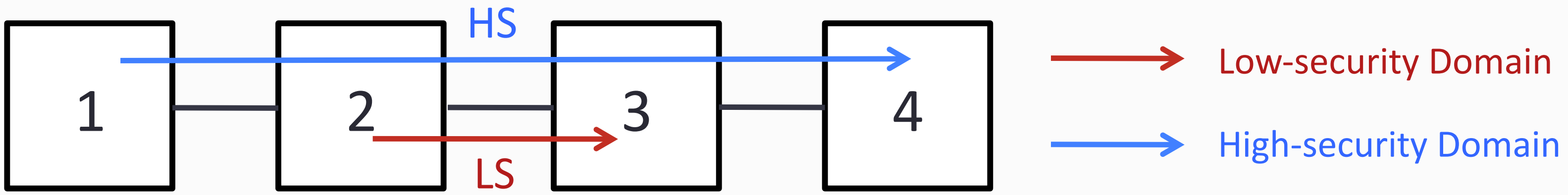


- Temporal Network Partitioning

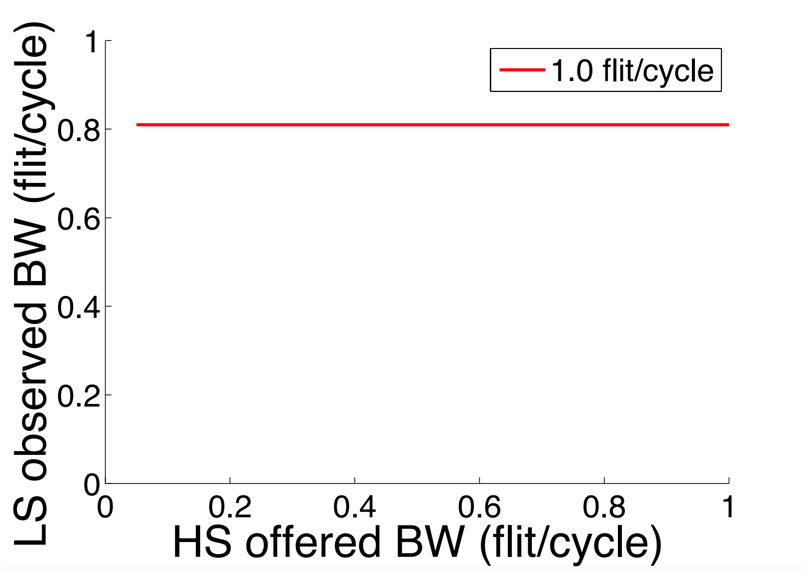


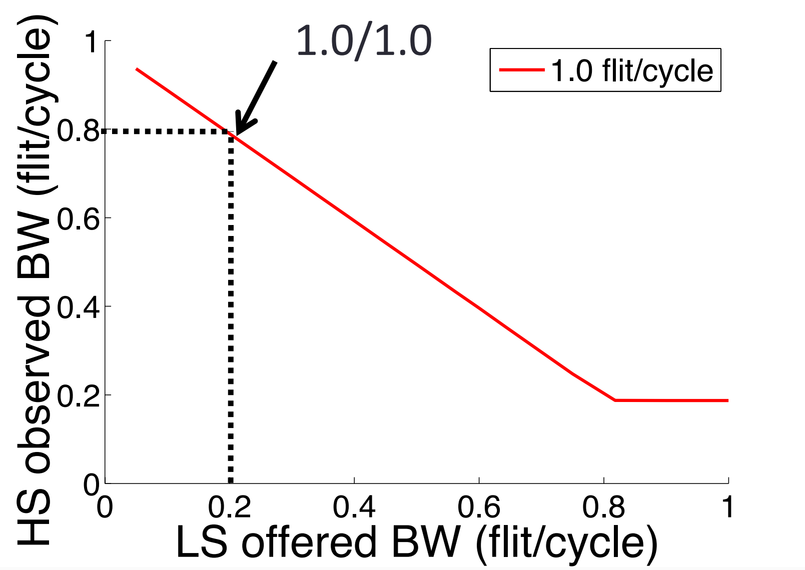
Timing Channel: One-way Protection

- Simple network

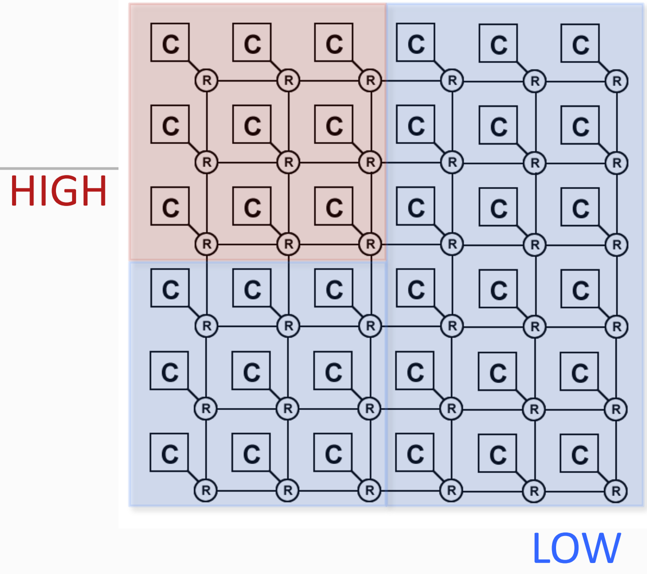


- Reversed Priority with Static Limits (Static limit = 0.8)

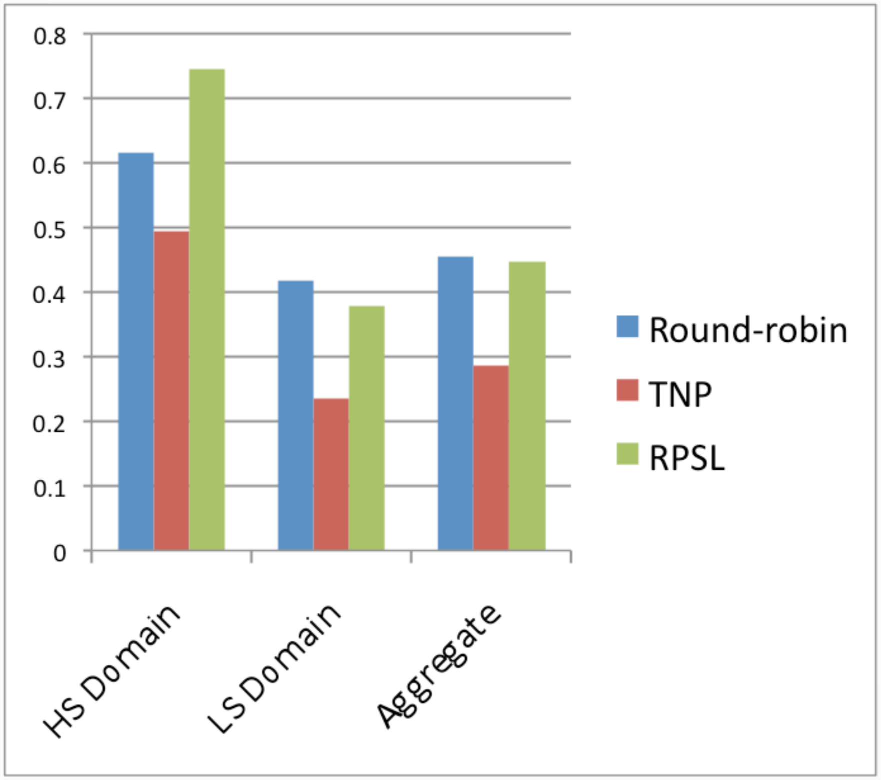




Performance



- Applications show bursty traffic



- RPSL is efficient for bursty traffic

Related Work

- Side-channel protection
  - Shared resources are prone to side-channel attacks, e.g. shared caches, branch prediction
  - Cannot be applied to NoC
- QoS schemes
  - Allows resource usage beyond allocation
  - Insufficient to prevent timing channel attacks
- Composability
  - Remove interference between applications for fast integration
  - Require bi-directional non-interference, incur high performance overhead

Conclusion

- Shared on-chip networks introduce timing channels
  - Prevent effective sharing of large-scale NoC in high assurance systems
- One-way timing channel protection is sufficient in many situations
- RPSL provides efficient one-way timing channel protection
  - Incurs low performance overhead