

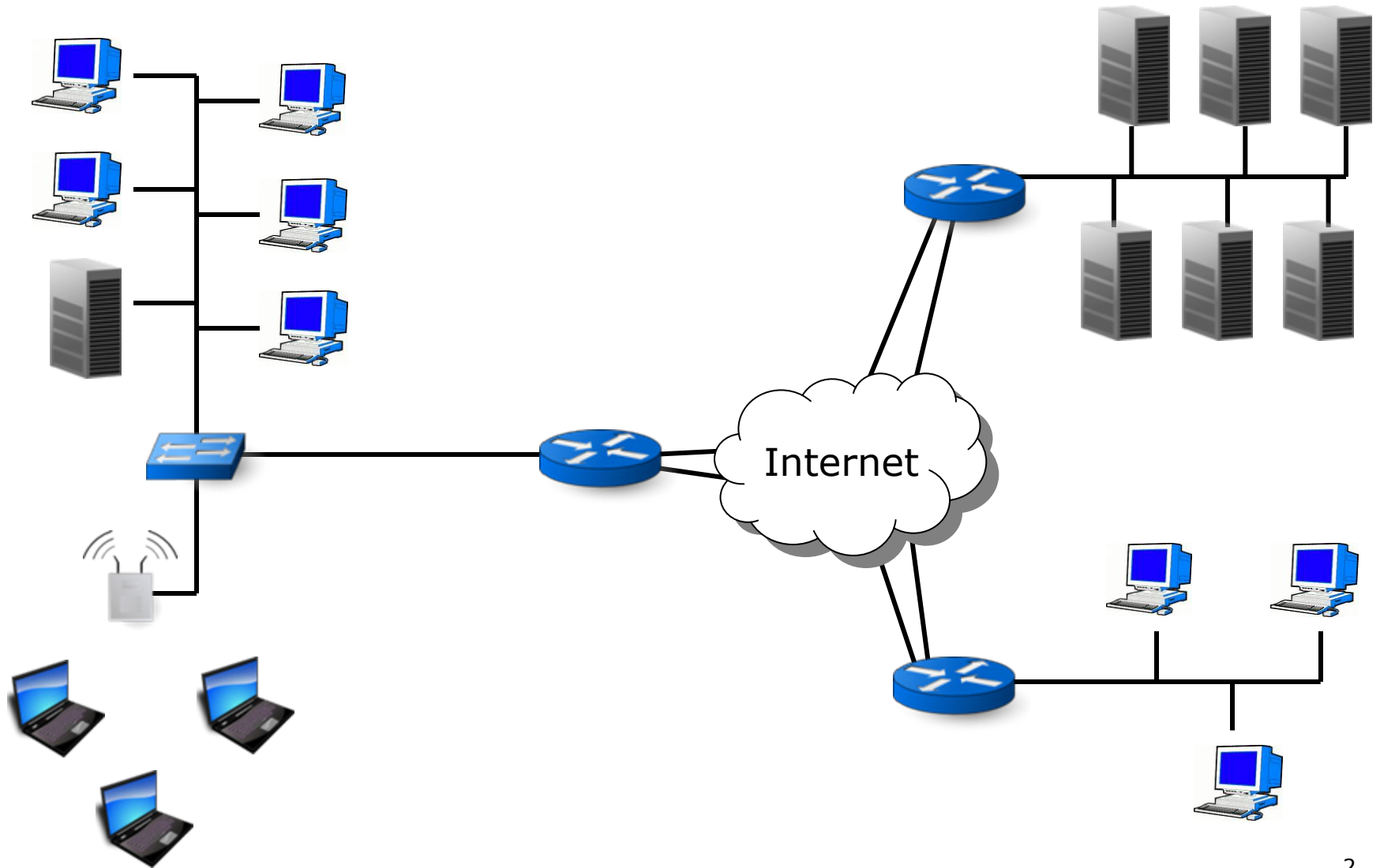
# Evaluating GPUs for Network Packet Signature Matching

Randy Smith, Neelam Goyal, Justin Ormont  
Karu Sankaralingam, Cristian Estan

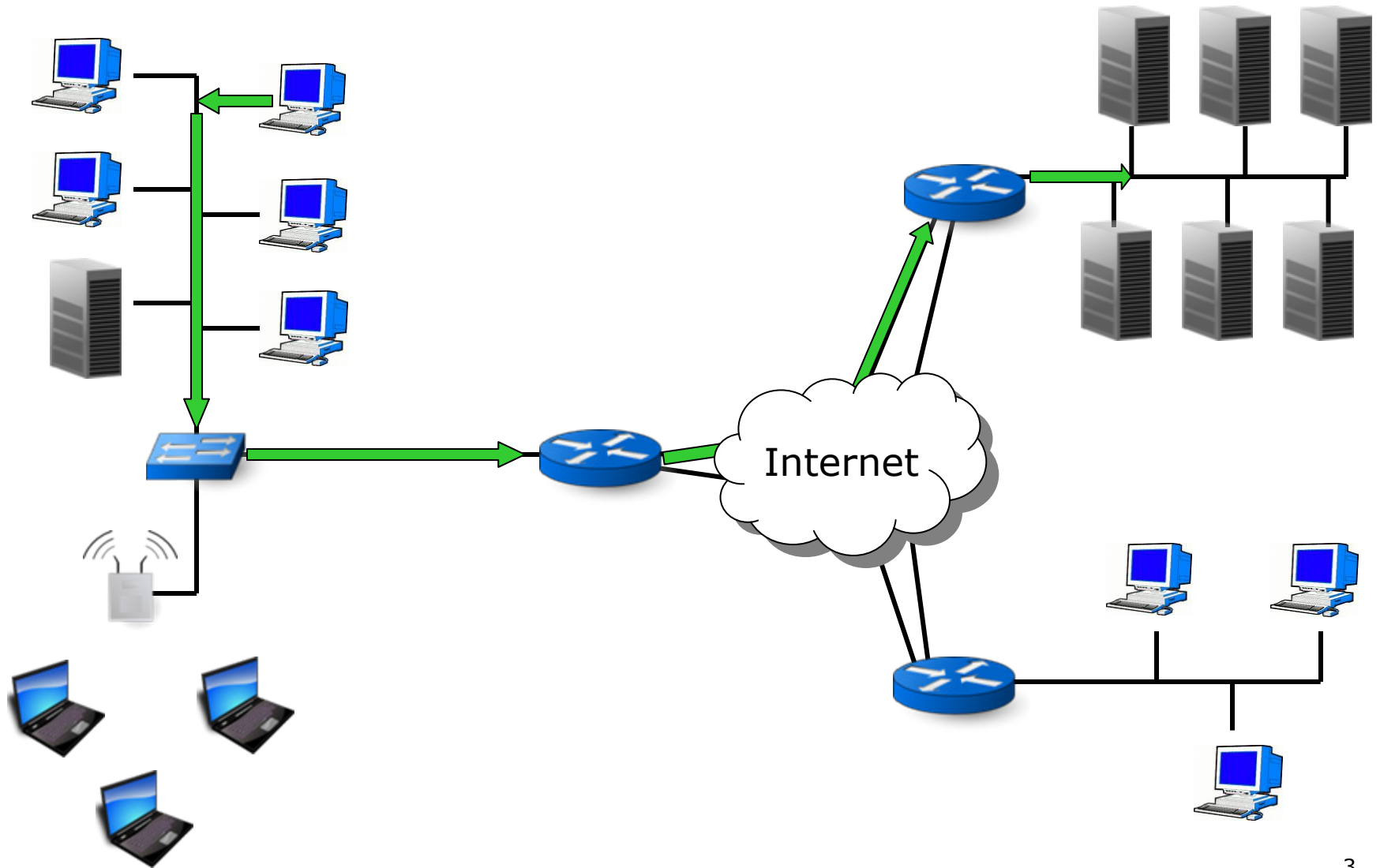
Department of Computer Sciences  
University of Wisconsin—Madison



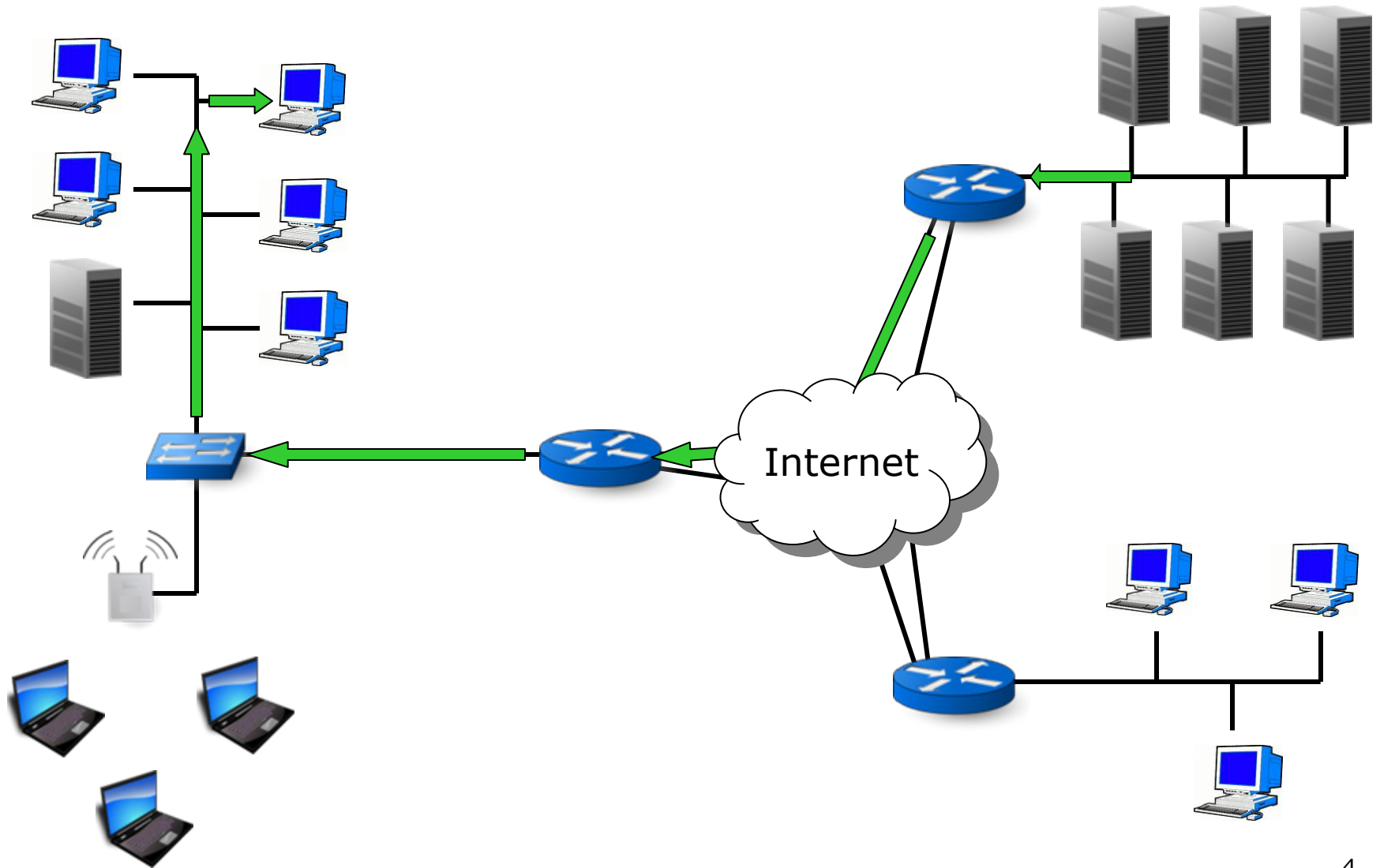
# Network Communication



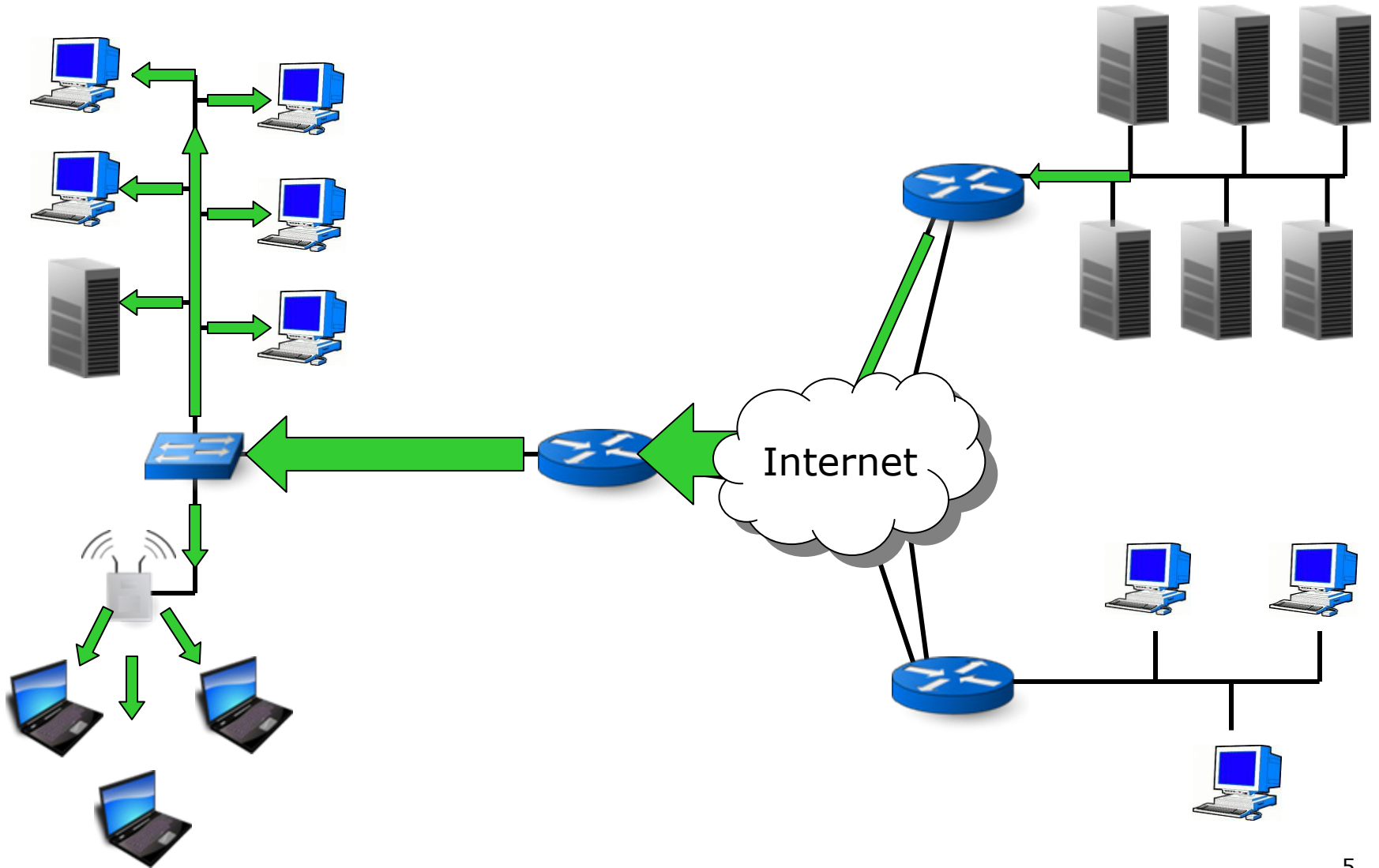
# Network Communication



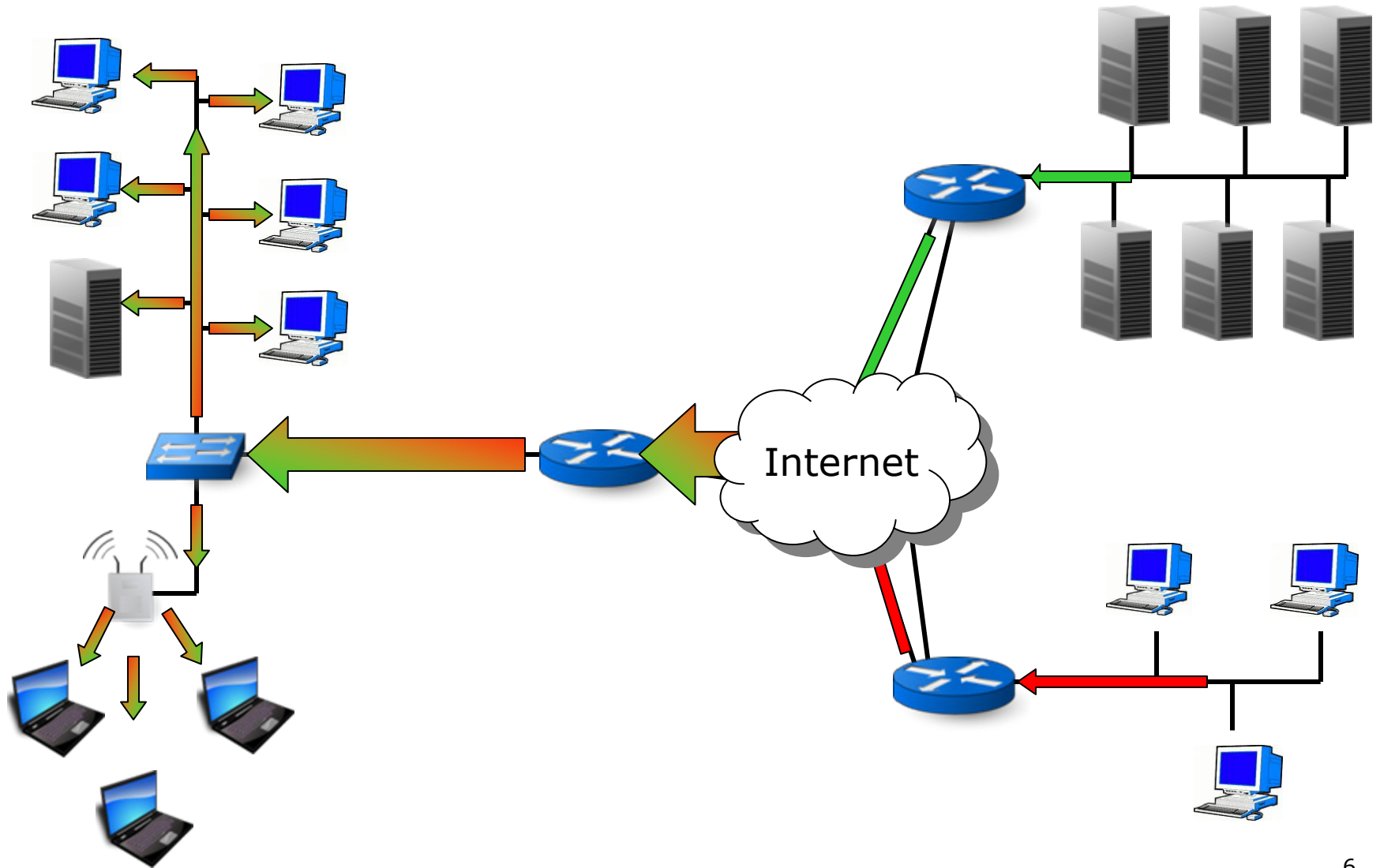
# Network Communication



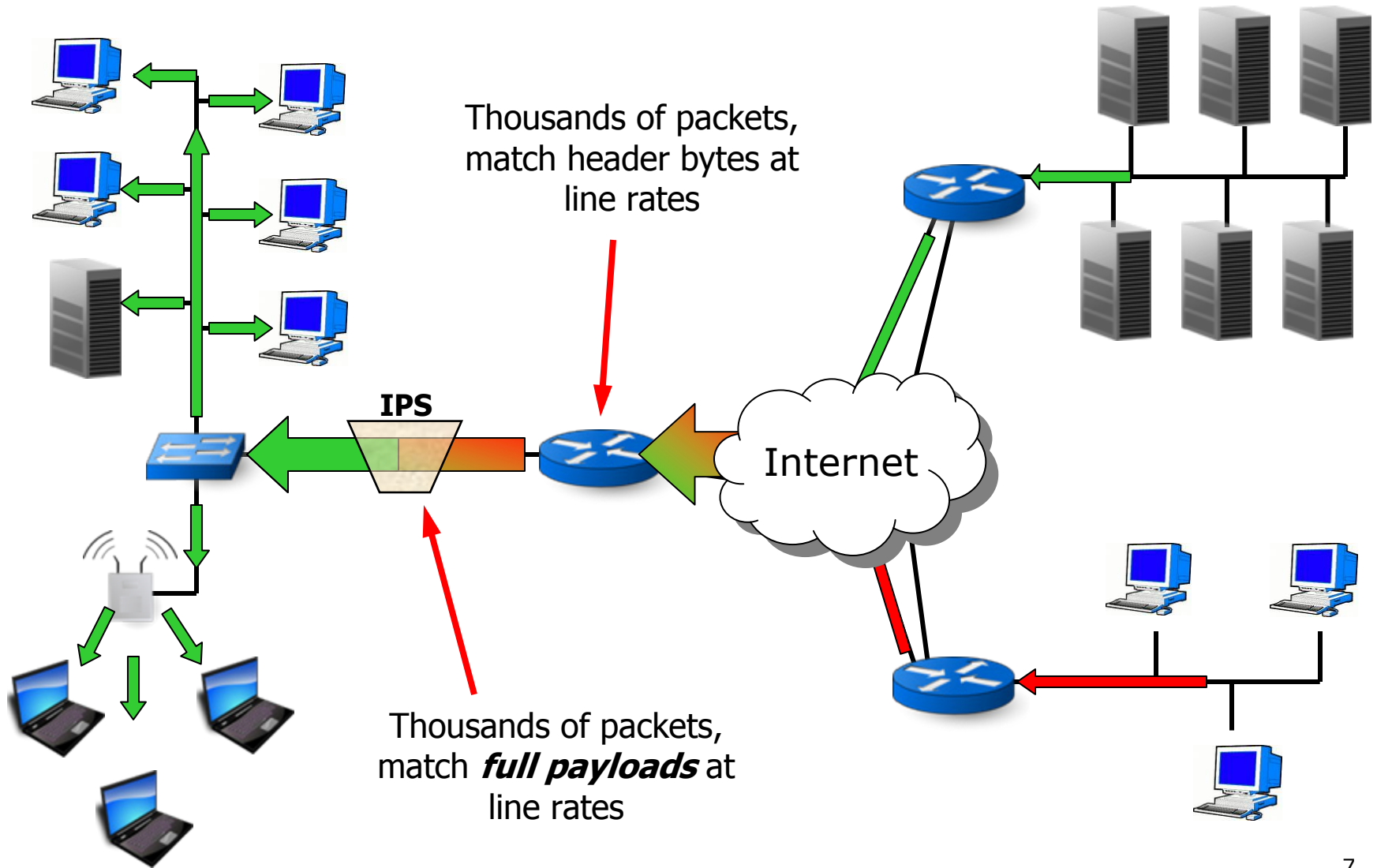
# Network Communication



# Network Communication



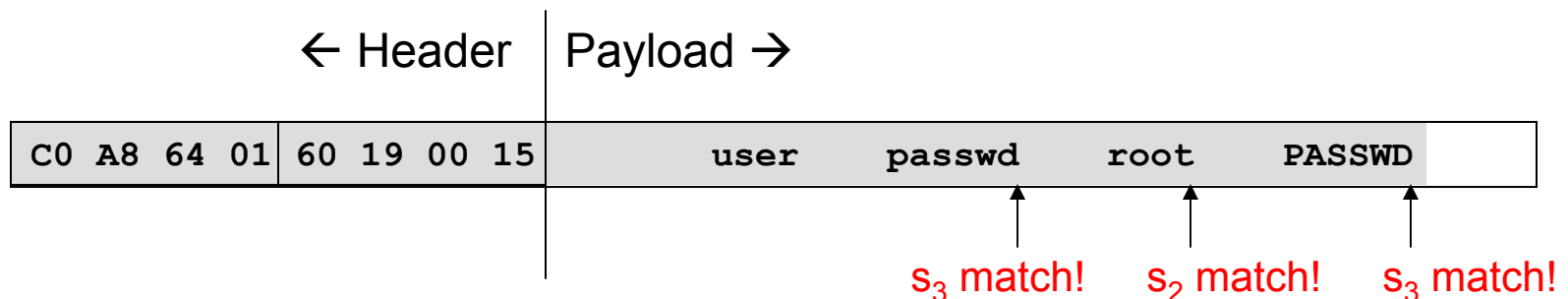
# Network Communication



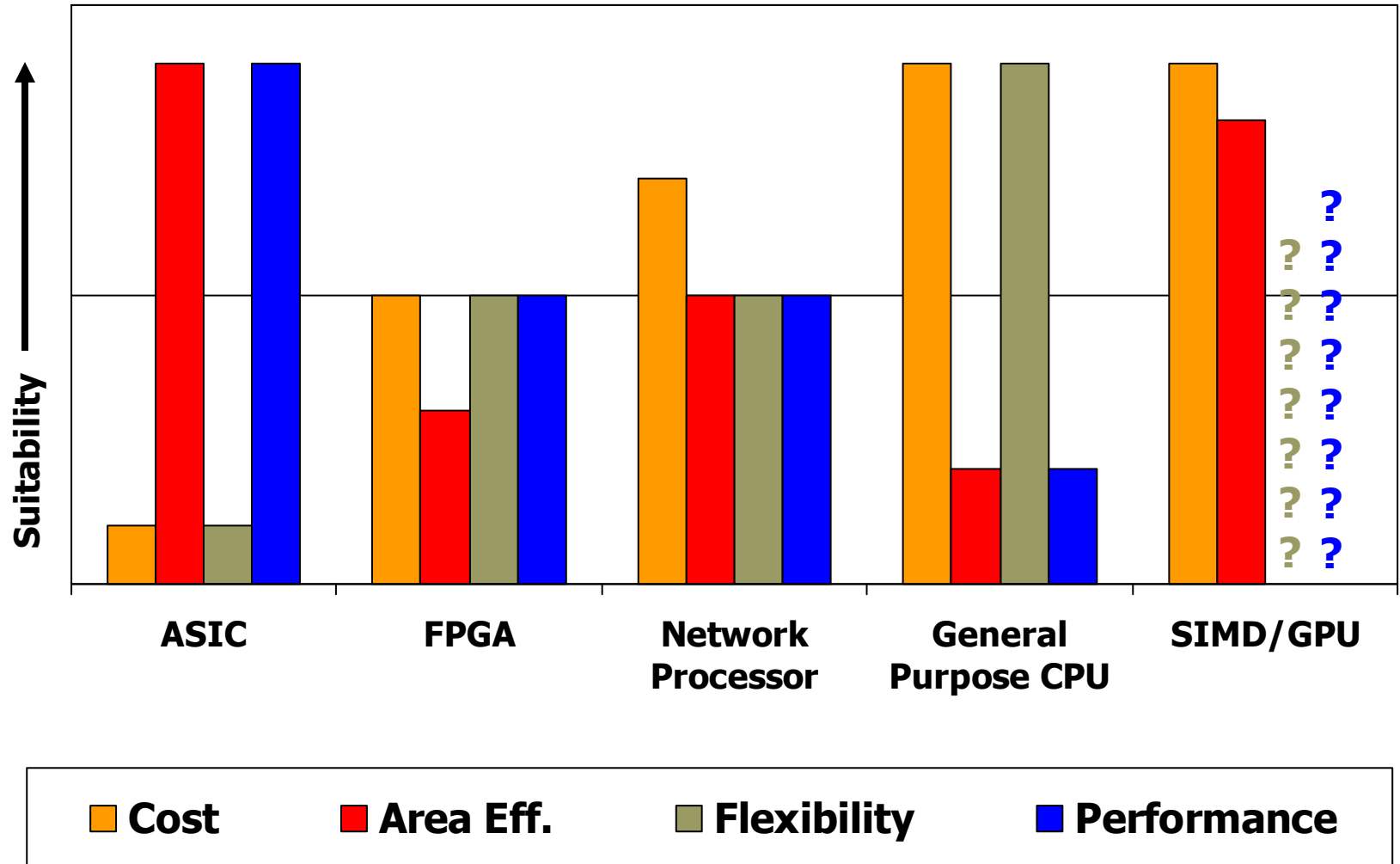
# Signature Matching Mechanics

- An intrusion prevention system is a filter
  - Functionality defined by user-supplied signatures
- Problem: find all matching signature occurrences up to the currently scanned byte

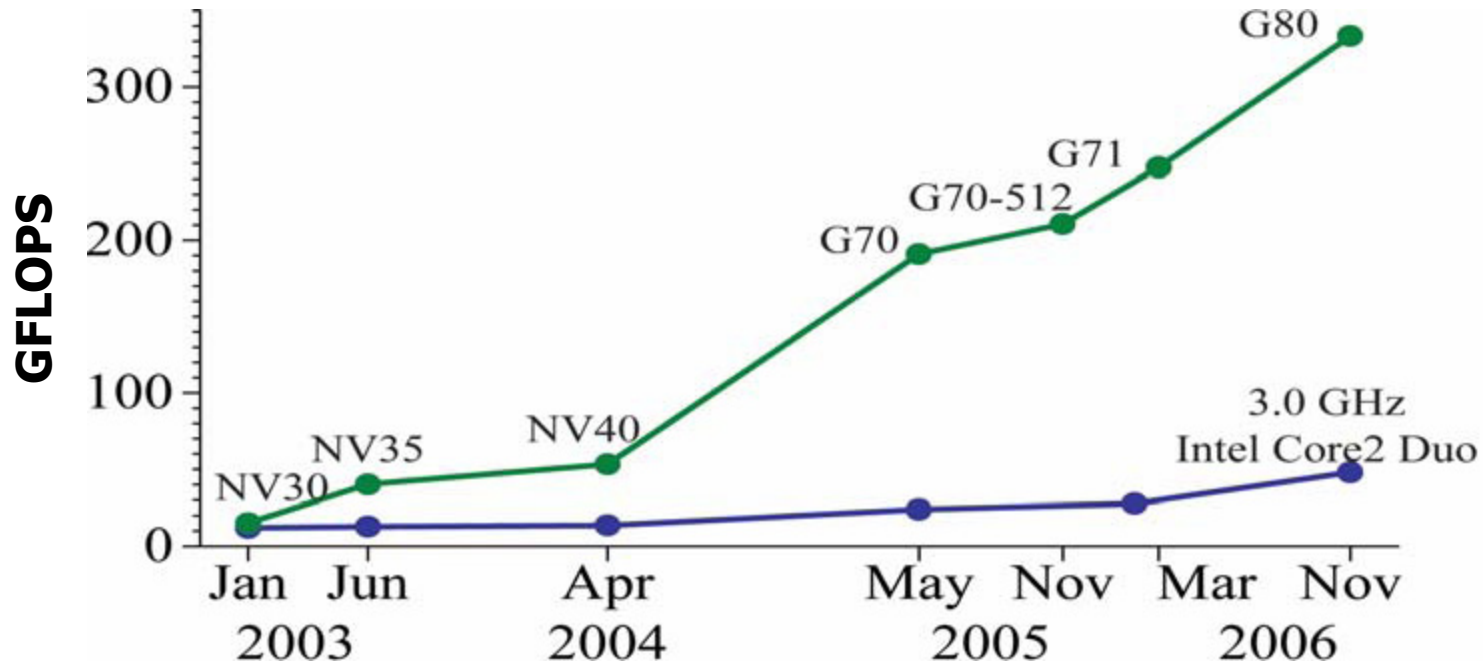
Signatures = {  
     $s_1$ : /(.\*shadow/ ,  
     $s_2$ : /(.\*user(.\*)root/ ,  
     $s_3$ : /(.\*)[Pp][Aa][Ss][Ss][Ww][Dd]/ }



# Processor Architectures



# Why Consider GPUs?



Source: NVIDIA

- “The future of GPUs is programmable processing”
- GPUs specialized for compute-intensive highly parallel computation (not just for graphics)

# GPUs for Signature Matching: Challenges

---

- Recursive data access patterns
- Not purely data parallel (divergence occurs)
- Key Questions:
  - *Do GPU/SIMD architectures have necessary flexibility?*
  - *Does mapping to such architectures negate performance gains?*

# This Work

---

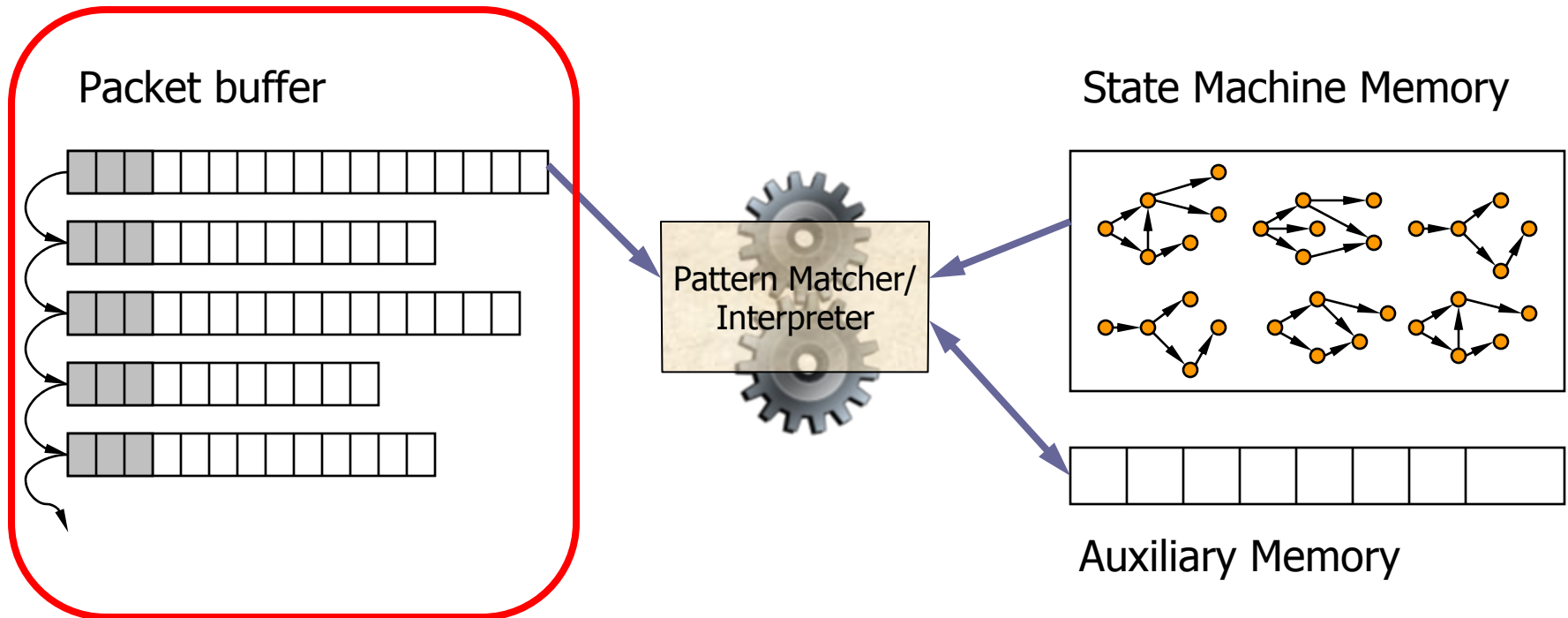
- Overall Goal: Assess suitability of GPUs for signature matching
- Conclusion: GPUs/SIMD provide higher performance than CPUs at similar costs
- In support of this:
  - Characterize signature matching in terms of control flow, memory access patterns, and concurrency;
  - Build and evaluate fully-functional prototype signature matcher on an NVIDIA G80 GPU;

# Outline

---

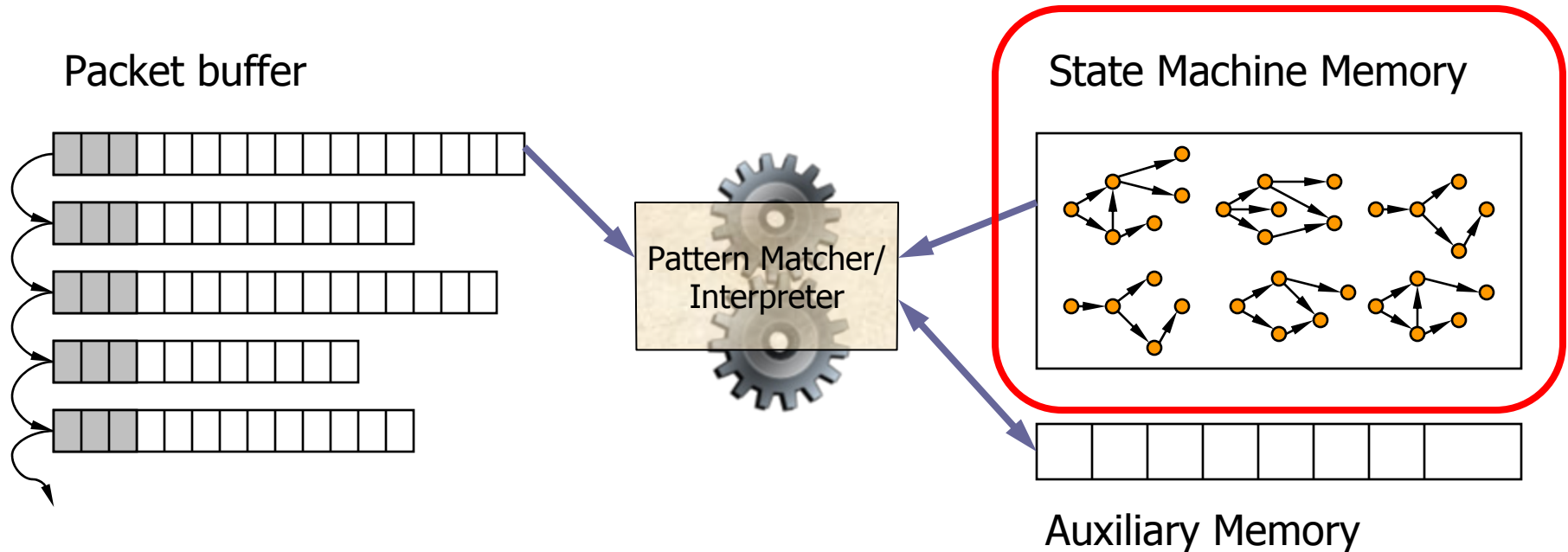
- Signature Matching Characteristics
- NVIDIA G80 Prototype
- Experimental Results
- Discussion and Conclusions

# Signature Matching Components



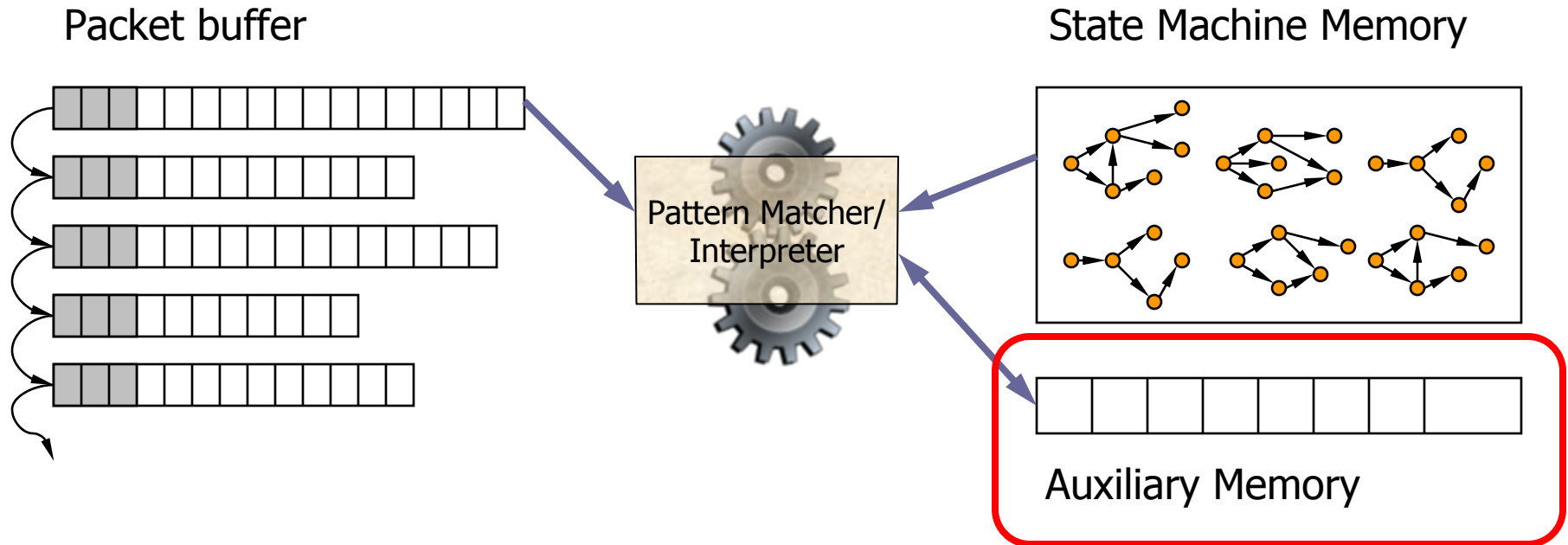
- ❑ Several MB in size
- ❑ DMA, etc. to copy from NIC to memory
- ❑ Regular Memory Access

# Signature Matching Components



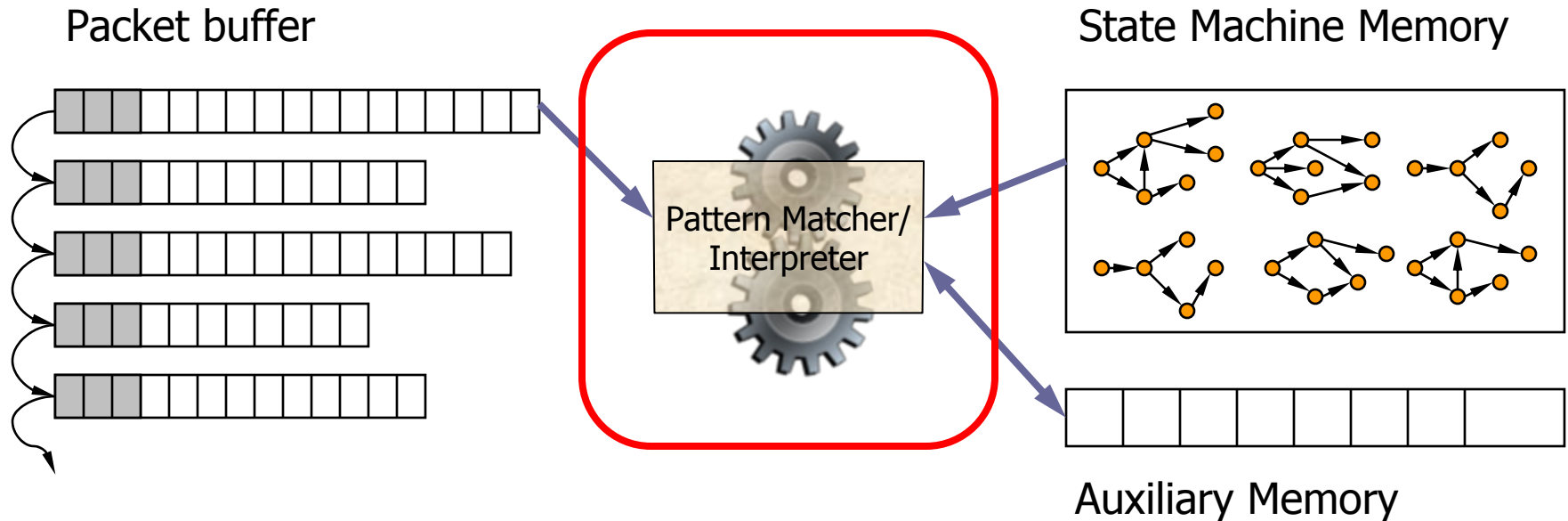
- ❑ 1 KB per state, thousands to millions of states
- ❑ Recursive data structures, irregular accesses
- ❑ Accesses serially dependent, driven by input

# Signature Matching Components



- ❑ Holds temporary data, acceptance indicators
- ❑ Typically less than 10K in size

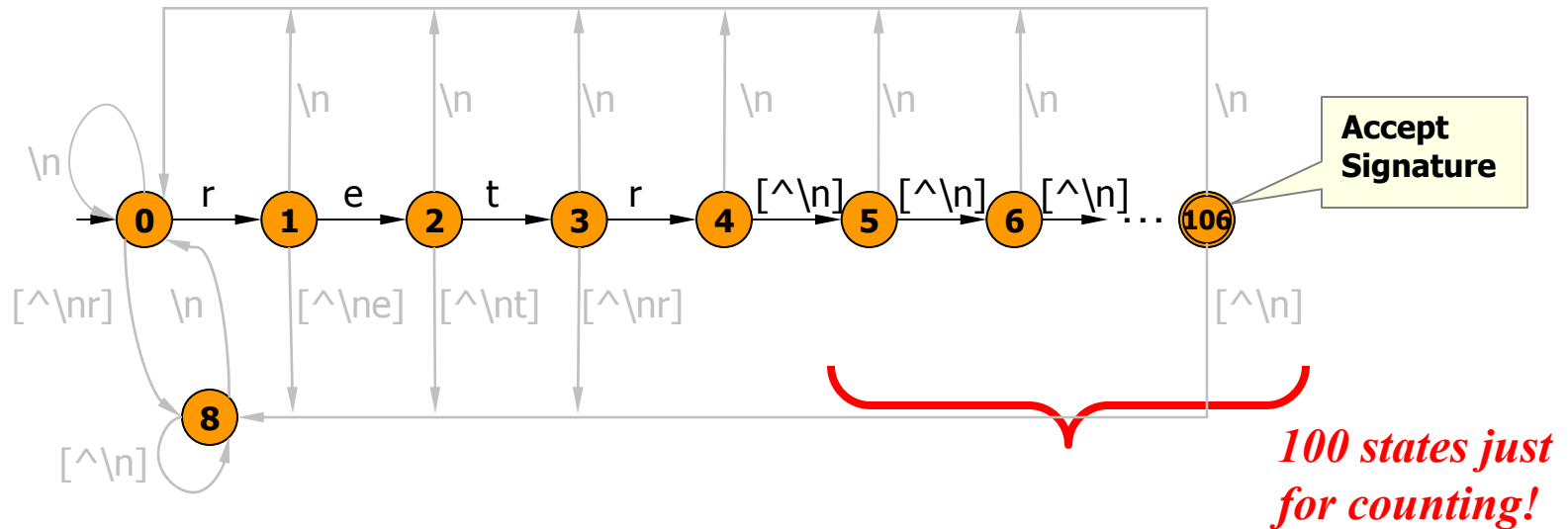
# Signature Matching Components



- ❑ Reads packet input
- ❑ Selects and traverses state machine
- ❑ Updates auxiliary data

# Matching with DFAs

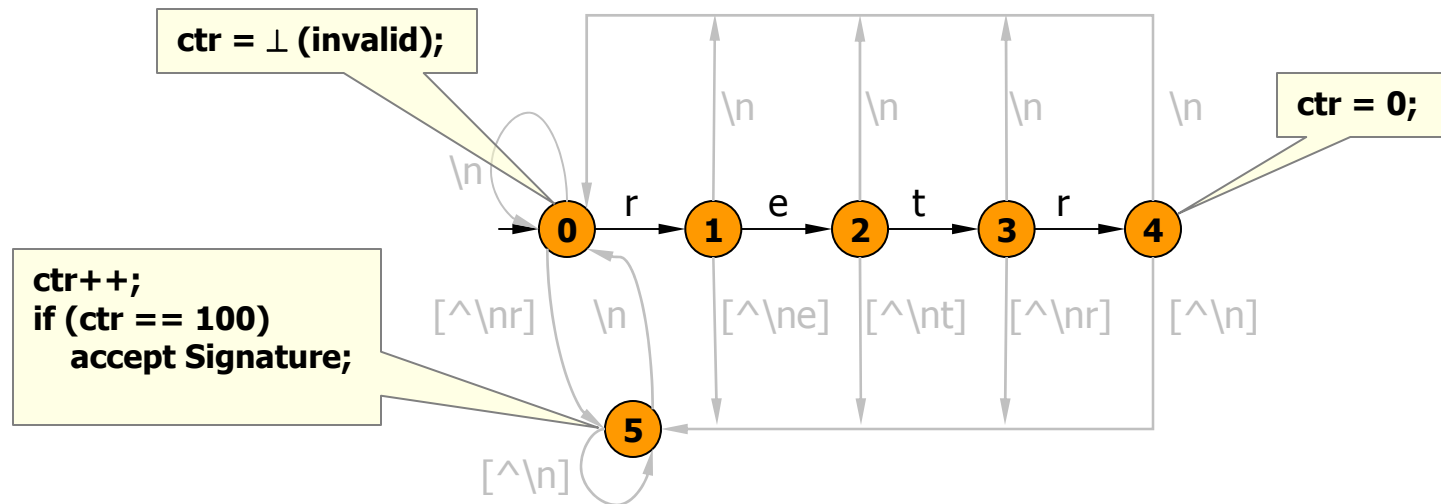
□ Regular Expression: `/^retr[^\n]{100}/`



- Simple computation model, easy to combine
- One table look-up per input byte
- Subject to state-space explosion when combined

# Matching with XFAs

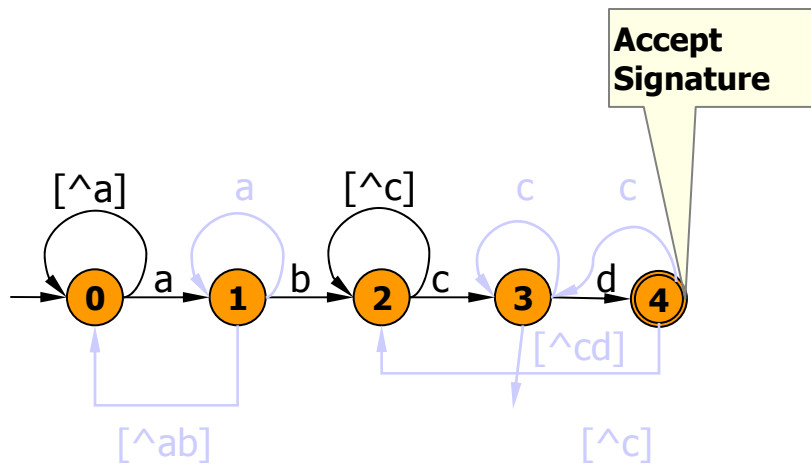
- Regular Expression: `/^retr[^\n]{100}/`



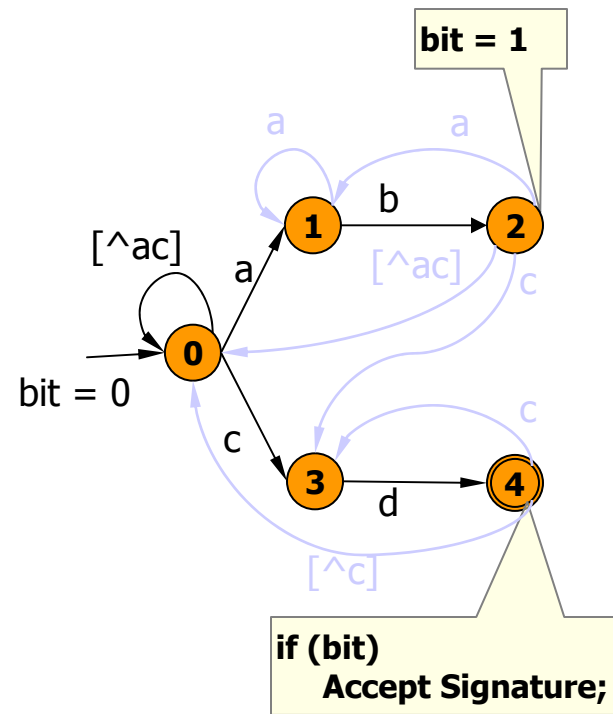
- Use variables to reduce DFA state explosion
- Manipulate variables with instructions attached to states
- Same semantics, complicated execution model

# Matching with XFAs

□ Regular Expression:  $/.*ab.*cd/$

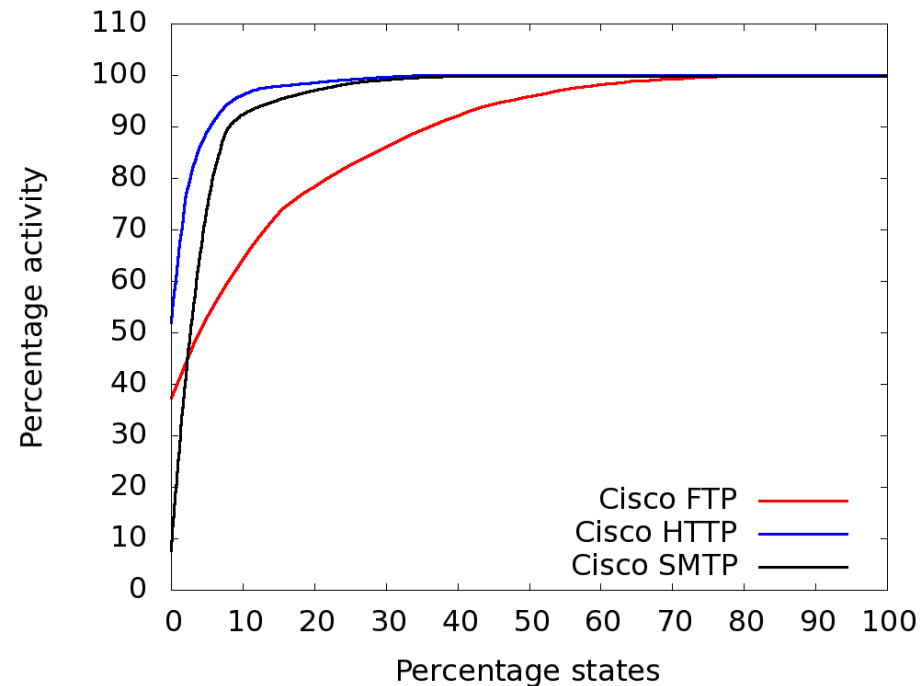
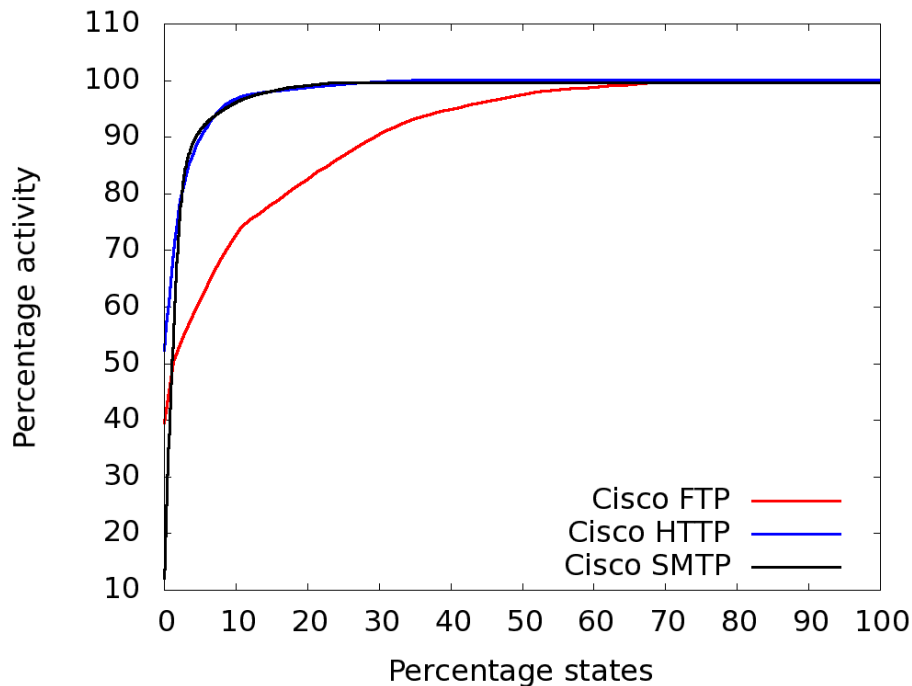


DFA



XFA

# State Frequency



- <math><10\%</math> states contribute to <math>>90\%</math> accesses
- Conclusion: Irregular access but caching, multithreading can hide access latencies

# Control Flow

---

```
StateMachine *SM = read_signatures();
```

```
for each packet in trace:
```

```
    apply(SM, packet.bytes, packet.len);
```

```
apply(StateMachine *SM, char *buf, int len):
```

```
    state *CS = SM.start_state;
```

```
    execute_instrs(CS->instrs);
```

```
    for i=0 to len
```

```
        CS = CS->nextState(buf[i]);
```

```
        execute_instrs(CS->instrs);
```

# Divergence

---

- Occurs when distinct conditional branches are taken in processing elements
  - Measured as the number of distinct conditional branches in SIMD instruction
  
- Sources:
  - Variations in packet size
  - Acceptance of *some* regular expressions
  - Execution of distinct XFA instructions
  
- What are common divergence levels?

# Control Flow and Divergence

---

Protocol	Prediction Accuracy		Divergence	
	DFA	XFA	%	Avg
FTP	97.5	97.6	2.3	2
HTTP	99.2	99.3	0.3	2
SMTP	98.3	98.2	61	2.21

# Signature Matching Requirements

---

## □ Memory Requirements

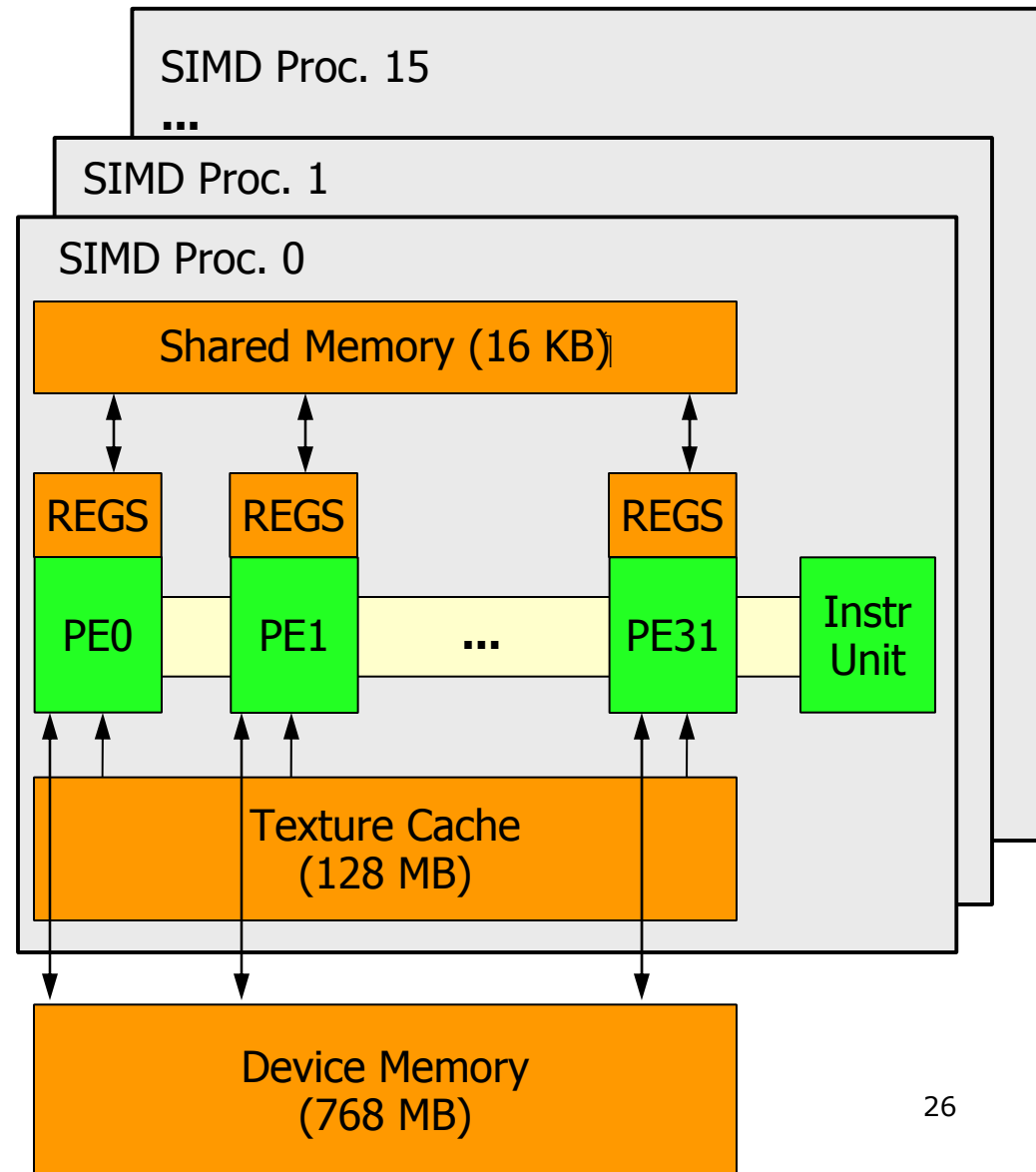
- Efficient regular access for moderate sized-memory
- Fast access for small, local memory
- Hide latency of irregular accesses

## □ Control Flow

- Divergence occurs, but not prevalent
- Support for data-dependent branching

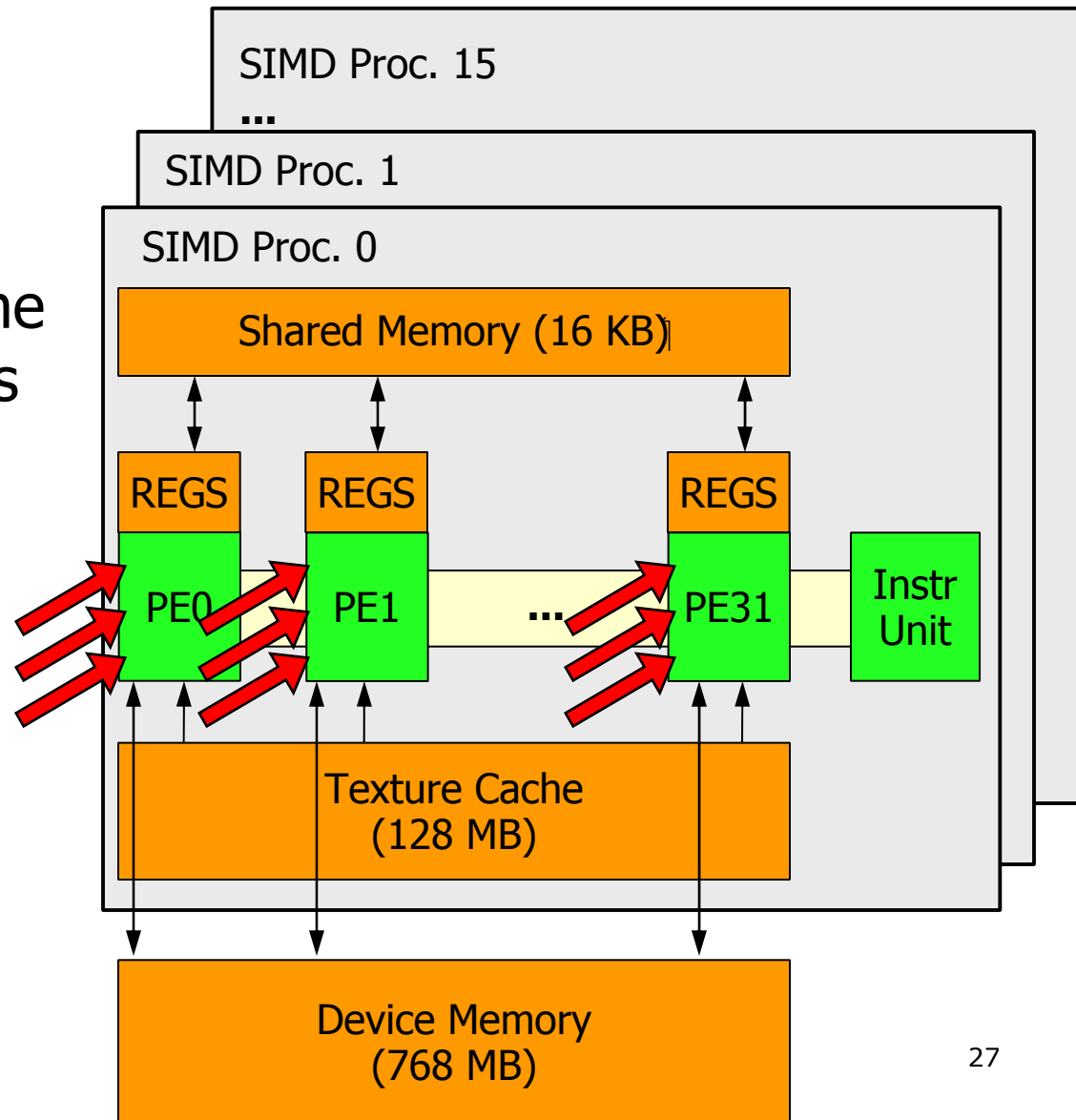
# G80 Architecture

- 16 core, 32 way SIMD
- Fast on-chip shared memory, texture cache
- Off-chip memory large but slow (400-600 cycle access times)
- Texture Cache read-only by GPU

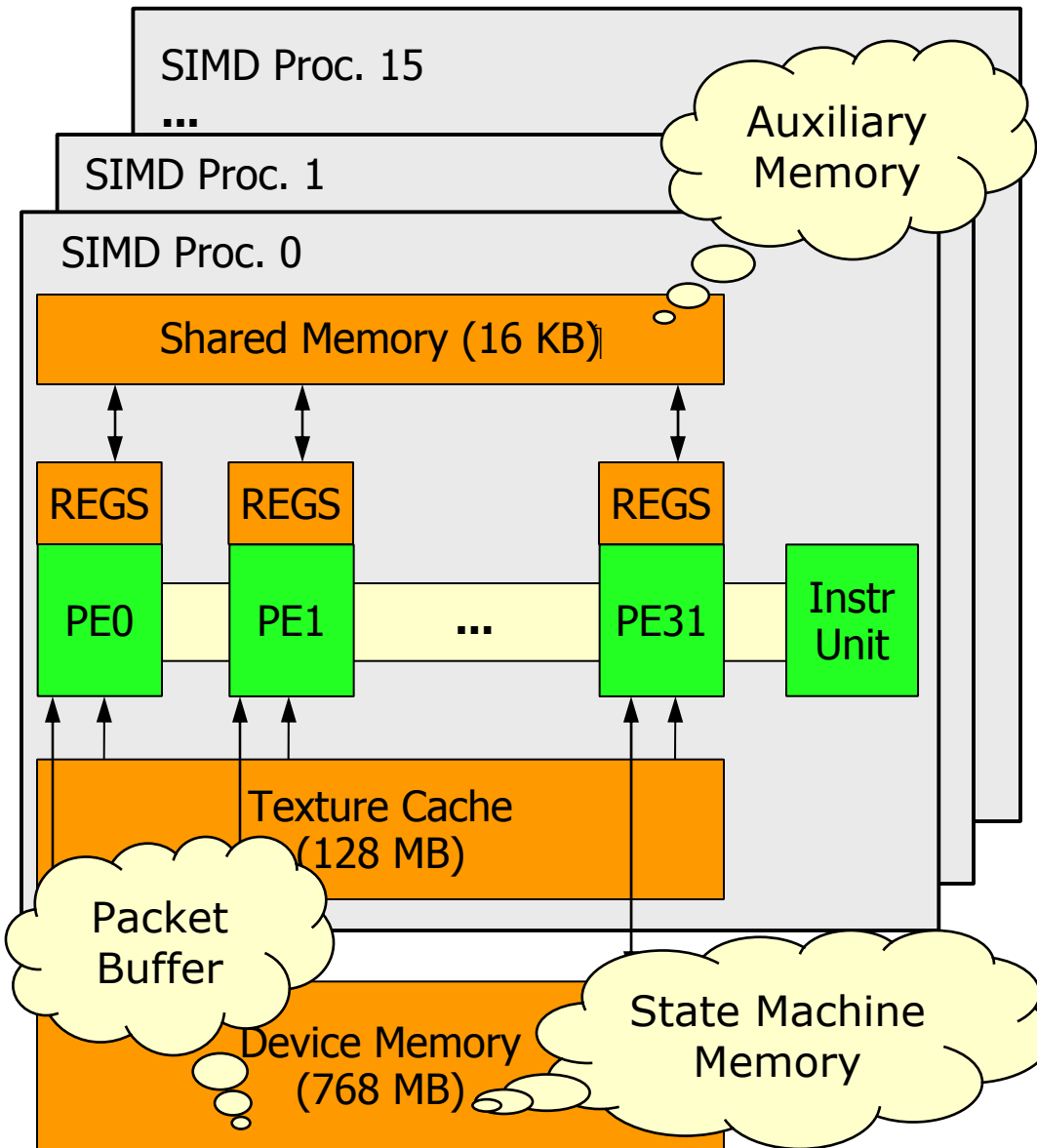


# G80 Architecture

- *CUDA* provides API, drivers to interface
- *Kernels* execute same instruction on all PEs in core
- Time-sliced multi-threading per core
- 31-cycle penalty for data-dependent branching



# G80 Prototype



- Two kernels:
  - *fa\_build* – build state machine on GPU
  - *trace\_apply* – perform DFA and XFA matching
- Sort packets into “groups of 32”
- Batch execution model:
  - Copy packets to GPU
  - Perform matching
  - Retrieve result vector

# Outline

---

- Signature Matching Characteristics
- NVIDIA G80 Prototype
- Experimental Results
- Discussion and Limitations

# Environment

---

## □ Test sets and data

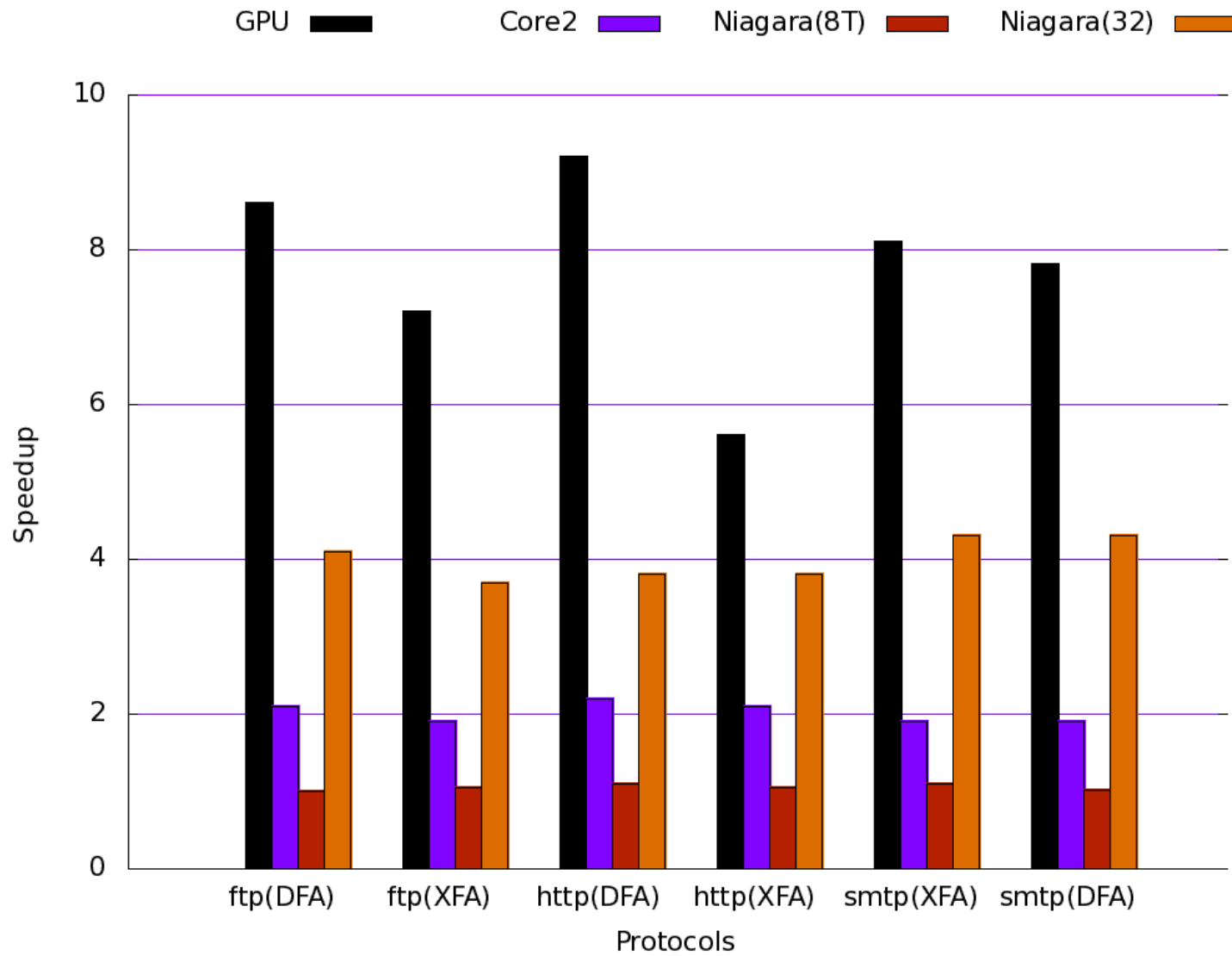
- Extracted FTP, HTTP, and SMTP regular expressions from Cisco IPS signatures
- Collected 10 GB trace from edge of academic network

## □ Platforms

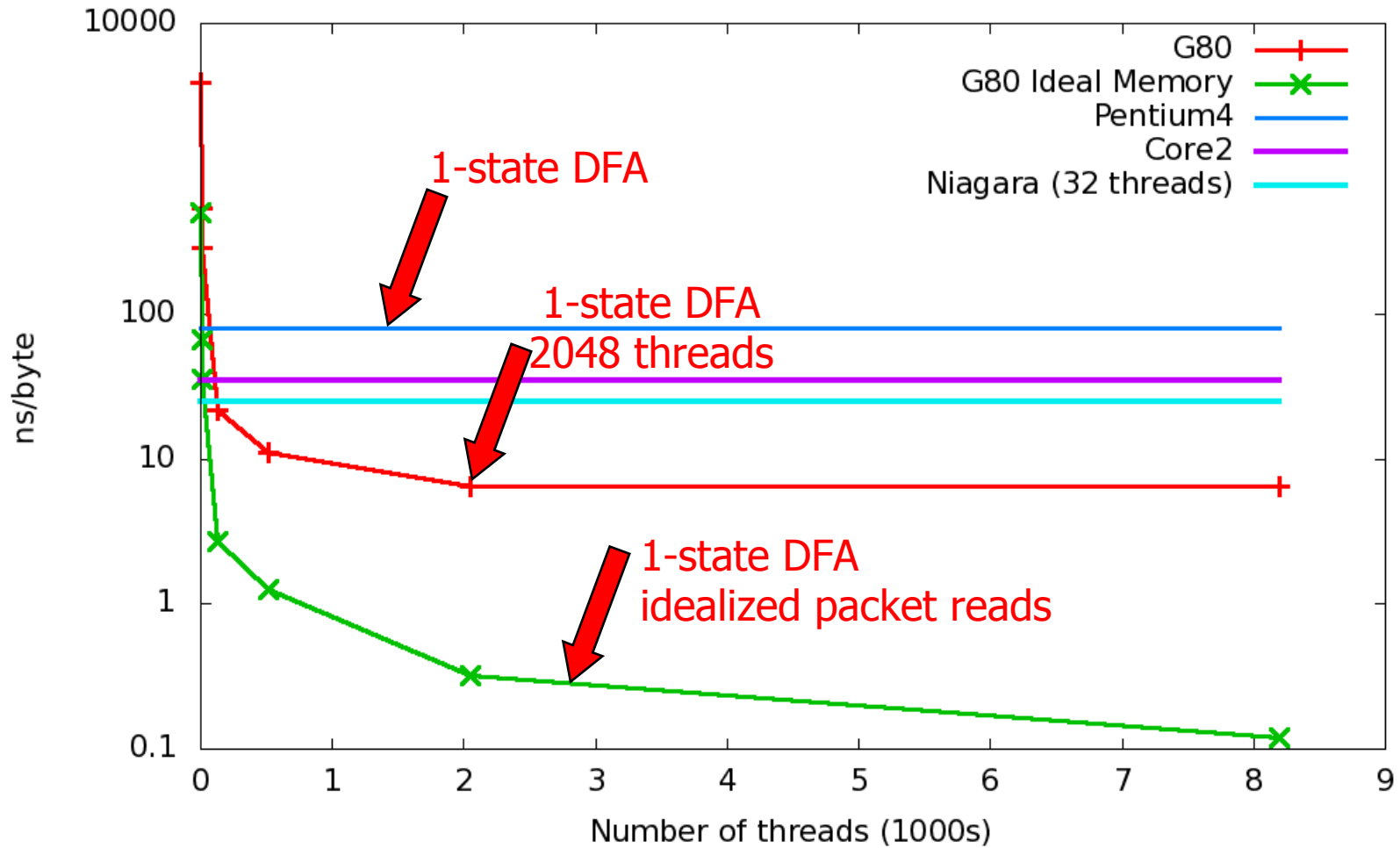
- Prototype: NVIDIA G880 GTX, Pentium 4 host
- Baseline: Software implementation on P4 at 3.0GHz
- Also: Intel Core2, Niagara

## □ All code implemented in “flat” C++

# Observed Speedup



# Ideal Speedup



# Results Summary

---

- G80 achieves 8.6x (DFAs) and 6.7x (XFAs) speedup over Pentium 4 baseline.
  - Key: many threads hide memory access latency
- Peak performance estimate is 36x speedup
- Better memory utilization (texture memory) may move closer to ideal

# Discussion and Limitations

---

- Batch processing and (no) texture cache usage
  - Artifact of prototype environment
  - Can resolve with a shared address space
- Sorted packets
  - Solution 1: consider approximate sorting or binning
  - Solution 2: break serial dependency of state traversal for DFAs
- Other functionality
  - Must maintain per-flow state, perform normalization

# Evaluating GPUs for Network Packet Signature Matching



Thank you