Washington University in St. Louis

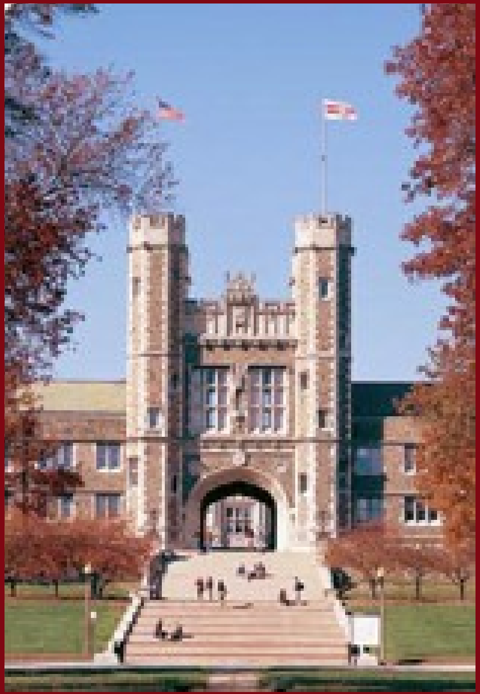
Engineering

High-Speed Networking Mechanisms

Perpetual Challenges & Opportunities

Patrick Crowley
pcrowley@wustl.edu
www.arl.wustl.edu/~pcrowley

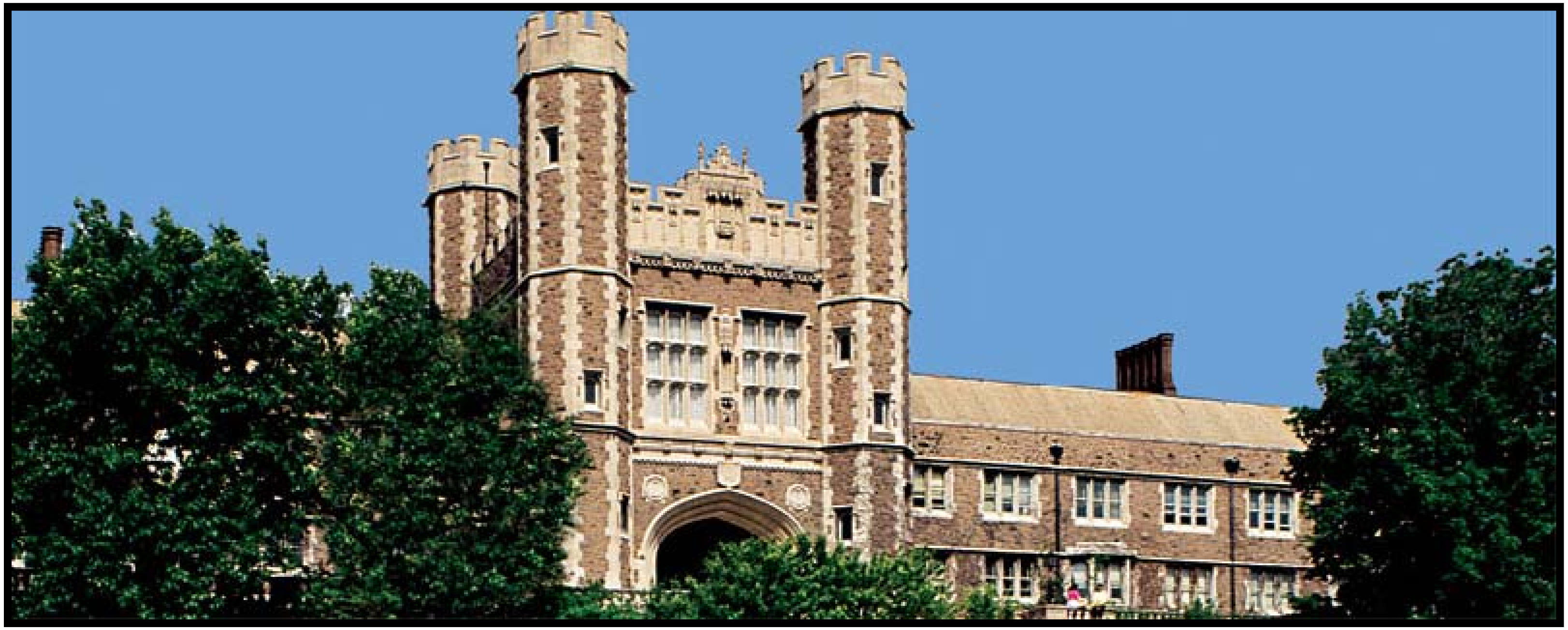
Department of
Computer Science & Engineering



St. Louis

- "Gateway to the West"
- Population: ~3 million
- Diverse economy: IT, engineering, plant & life sciences, manufacturing, aerospace, healthcare

Washington University (www.wustl.edu)



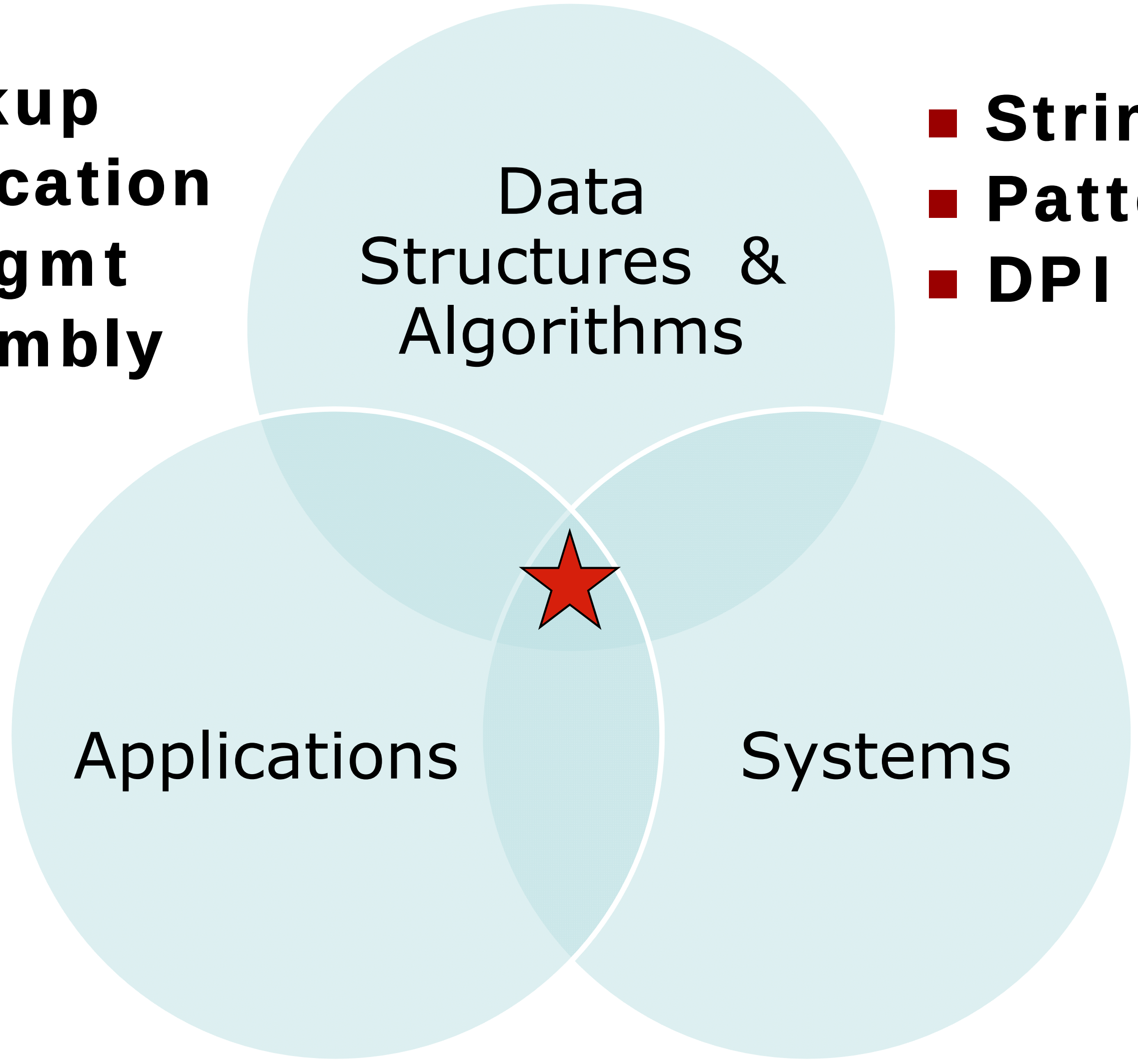
- Private research & teaching university
- 10,000 students (4,000/6,000 undergrads/grads)
- 10,000 faculty and staff
- Research focus: >$1B per year, 23 Nobel prize winners
- Washington U. Med School, among the best in the world
- Very strong research activity in CS & Networking

CSE at Washington University

- Research activities in many areas
  - » Computer architecture and design
  - » Networking and communications, incl. wireless
  - » Software systems
  - » Real-time systems, sensor networks
  - » Robotics, AI, Computer Graphics, Vision, Planning
  - » Human-Computer Interfaces
  - » Optimization
- Entrepreneurial Spirit
  - » Students and faculty frequently create new start-ups
- Strong Partnerships with Industry
  - » Many students/faculty at Google, Intel, Cisco, Microsoft,...

High-Speed Networking Mechanisms

- IP Lookup
- Classification
- Flow Mgmt
- Reassembly



- String match
- Pattern match
- DPI

Ongoing Importance

- Perpetual Motivation #1
  - » Larger networks, faster links
- Perpetual Motivation #2
  - » Networking mechanisms often arise as quick fixes
  - » Redress architectural flaws in prototypes (firewalls)
  - » Meet unanticipated needs (load balancing)

Plan

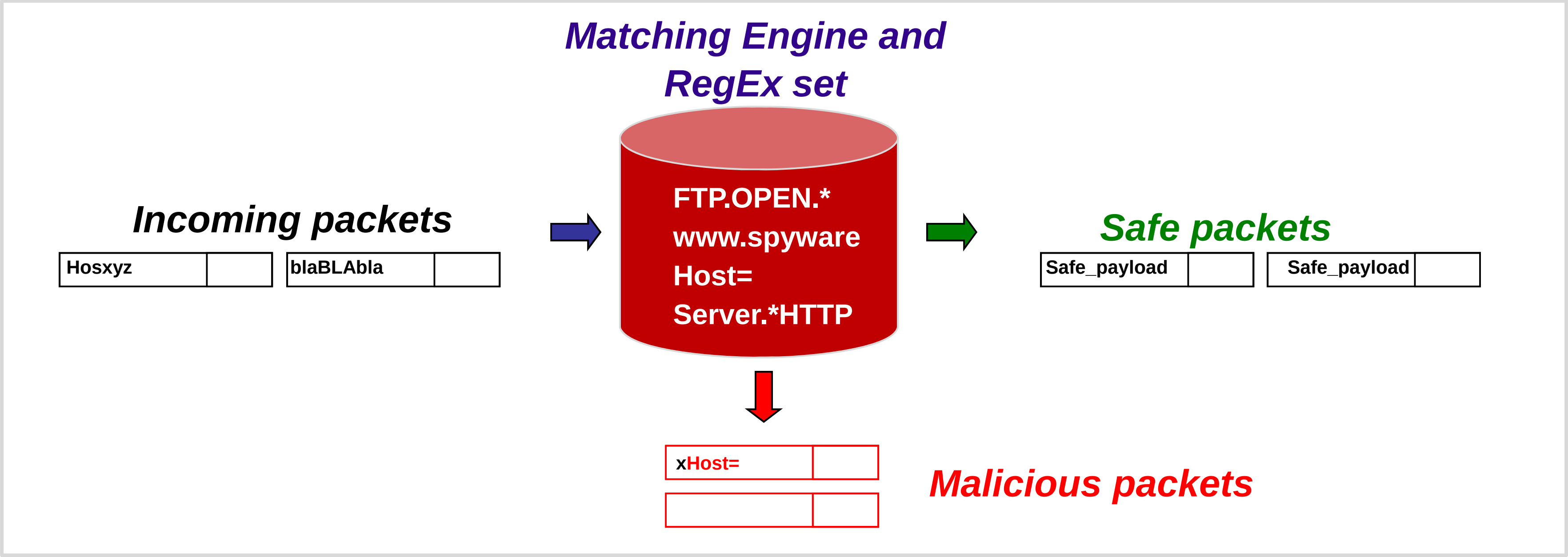
- Introduction
- Sample Mechanism
  - High-Speed Regular Expression Evaluation
- Experimental Environment & Testbed
  - Open Network Laboratory
- Sample Experiments
  - ISP-managed P2P
  - Passive Network Analyzer
- Conclusion

Sample Mechanism

- High-Speed Regular Expression Evaluation
- This is the work of my recently graduated student **Michela Becchi**

Context

- Network intrusion detection and prevention systems



- Intrusion detection and prevention
- Email monitoring
- Content based routing
- Application level prioritizing and filtering
- …

Challenges

- Networking context
  - » *Line rate operation* (several Gbps)
  - » *Parallel search* over data-sets consisting of *hundreds or thousands of patterns*

    - Bound per-character processing
    - Pre-computed large data structures

  - » On memory-centric architectures

Memory bandwidth

Memory size

Challenges (cont'd)

- Snort rule-set, November 2007 snapshot
  - 8536 rules
    - 5549 Perl Compatible Regular Expressions

• 99% with **character ranges**	*mi[ck][ch]ela*
• 16.3 % with **dot-star terms**	*mi.*ela mi[^\n\r]*ela*
• 44 % with **counting constraints**	*mi.{2,3}ela*
• 6% with **back-references**	*mi(ch|k)ela bec\1i*

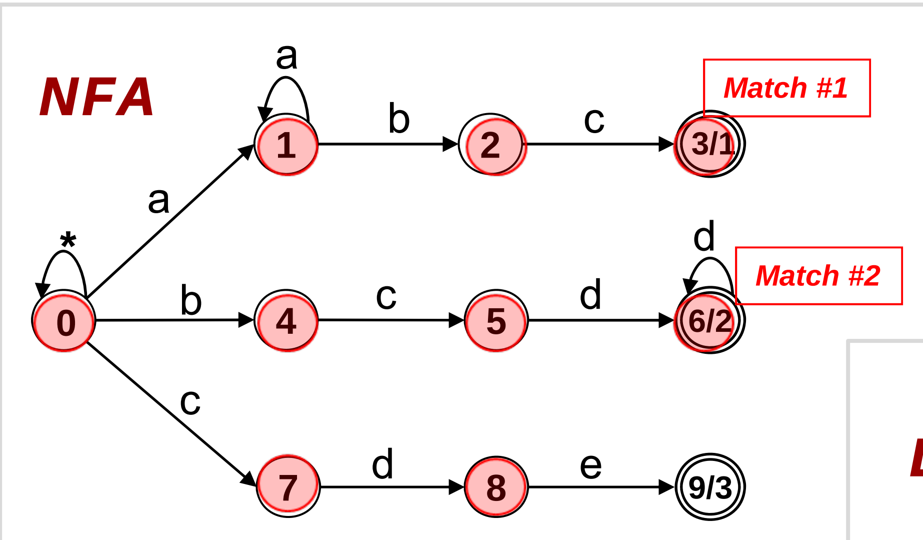
- Note:
  - *Lazy/greedy quantifiers*
  - *Positive/negative lookahead*
  - *Atomic groups*
  - *…*

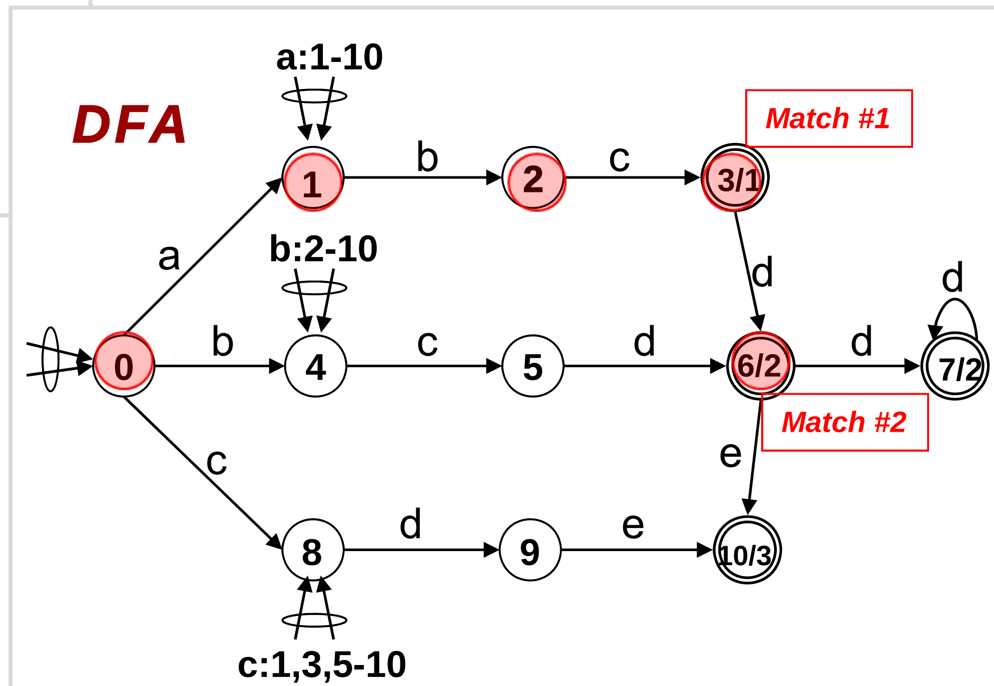
  - *No expressive power added*
  - *Speed up text-based engines*

Deterministic vs. Non-Deterministic FA

RegEx: (1) $a^{+}bc$ (2) bcd^{+} (3) cde

Text: *a b c d*





MEMORY SIZE:
# of states and transitions

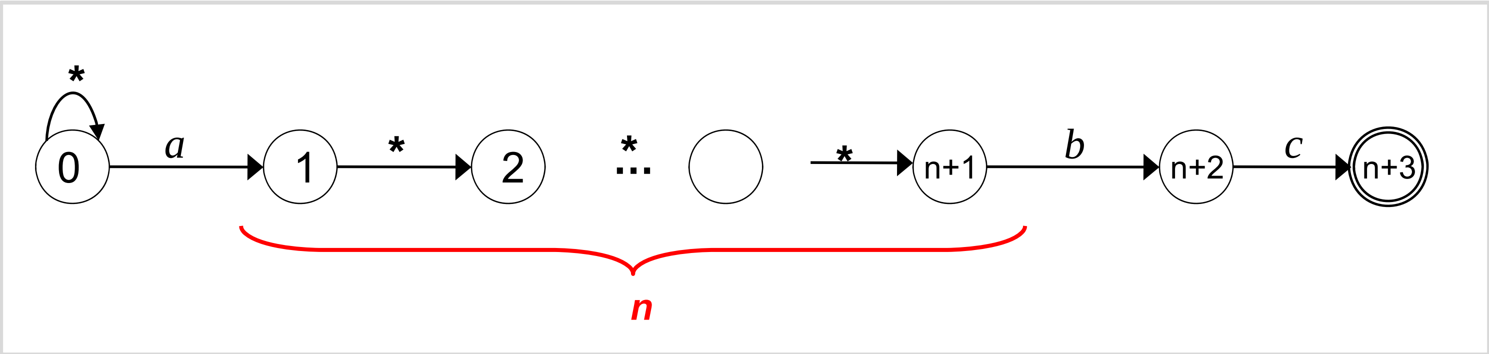
Better for NFAs

MEMORY BANDWIDTH:
# of state traversals per input character

Better for DFAs

Counting constraints – NFA

E.g: a.{n}bc

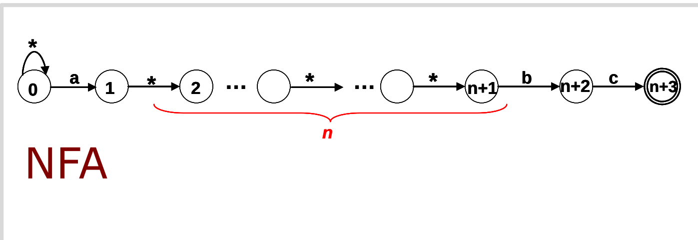


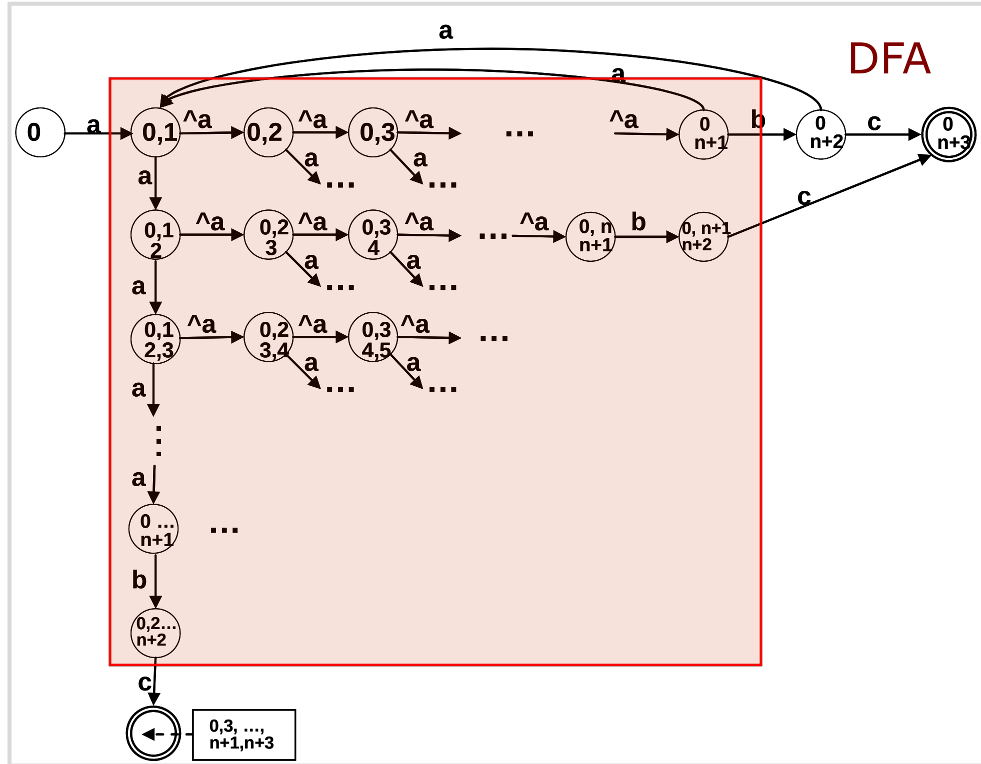
- Memory size
  - » For large *n*, number of states N_{NFA} ***linear*** in *n*

- Memory bandwidth
  - » Input text: *aaaaaaa…aaabc* ⇨ *n* states active in parallel
  - » For large *n* ~ N_{NFA} memory accesses/input character

Counting constraints - DFA

E.g: a.{n}bc

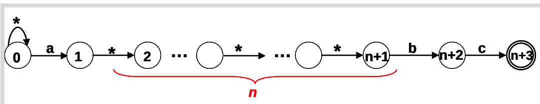


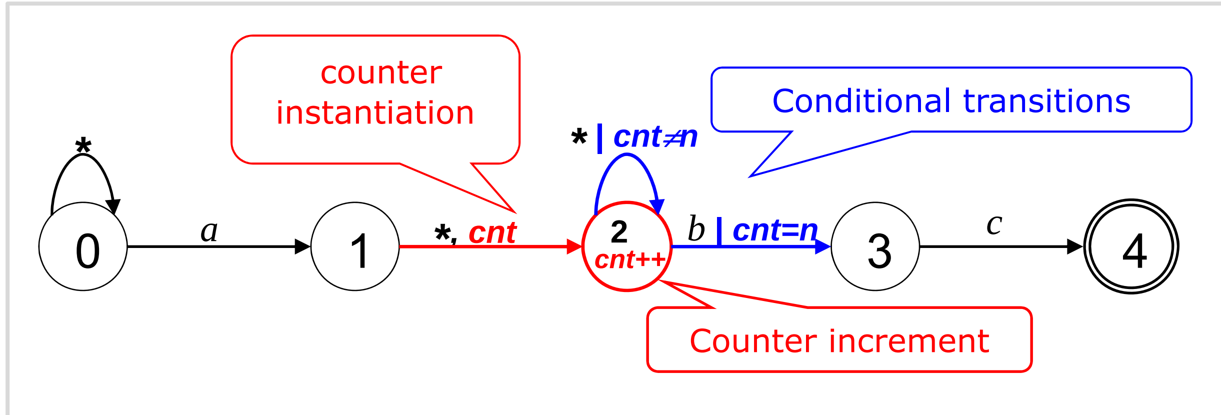


- Memory size
  - » For large *n*, number of states D_{NFA} ***exponential*** in *n*
  - » For large *n DFA practically infeasible*
    - *e.g. n=40→ ~1000 billion states*

Counting-NFAs

E.g: a.{n}bc





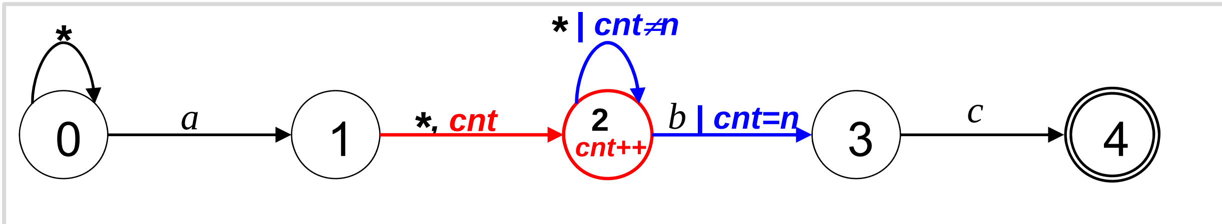
- Advantage: Limited size (*independent of n*)
- Functional equivalence: is one counter enough?
  - » E.g.: *a.{3}bc*:
    - text: *axaybcz* ⇨ match is detected
    - text: *axaybzbc* ⇨ match is missed!
  - ⇨ Multiple (up to *n)* counter instances necessary

$$0 \xrightarrow{a} 0,1 \xrightarrow{x} 0,2(c_1=1) \xrightarrow{a} 0,1,2(c_1=2) \xrightarrow{y} 0,2(c_1=3, c_2=1) \xrightarrow{b} 0,2,3(c_2=2) \xrightarrow{z} 0,2(c_2=3) \xrightarrow{b} (0,3) \xrightarrow{c} (0,\underline{4})$$

- *n* active counter instances ⇨ unmodified memory bandwidth requirement!

Counting-NFAs: limiting memory bandwidth

E.g: a.{n}bc



- **Observation:**
  - Counter instances updated in parallel
  - Difference between c_i and c_j constant over time
- **Idea:**
  - *Differential representation:* store oldest (and largest) instance c_i' and, for $j>i$, $\Delta c_j = c_j - c_{j-1}$

8	5	3	1

c'	Δc_i		
8	3	2	2
9	3	2	2
7	-	2	2

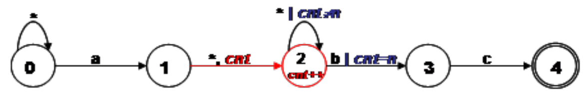
10

$n=10$

  - *Condition evaluation:*
    - $cnt=n$: $c_i'=n$
    - $cnt\neq n$: $c_i' \neq n$ OR another instances c_j exists
- **Advantage:**
  - Even if *n* instances are active, **only 2 must be queried / updated**

Counting-DFAs

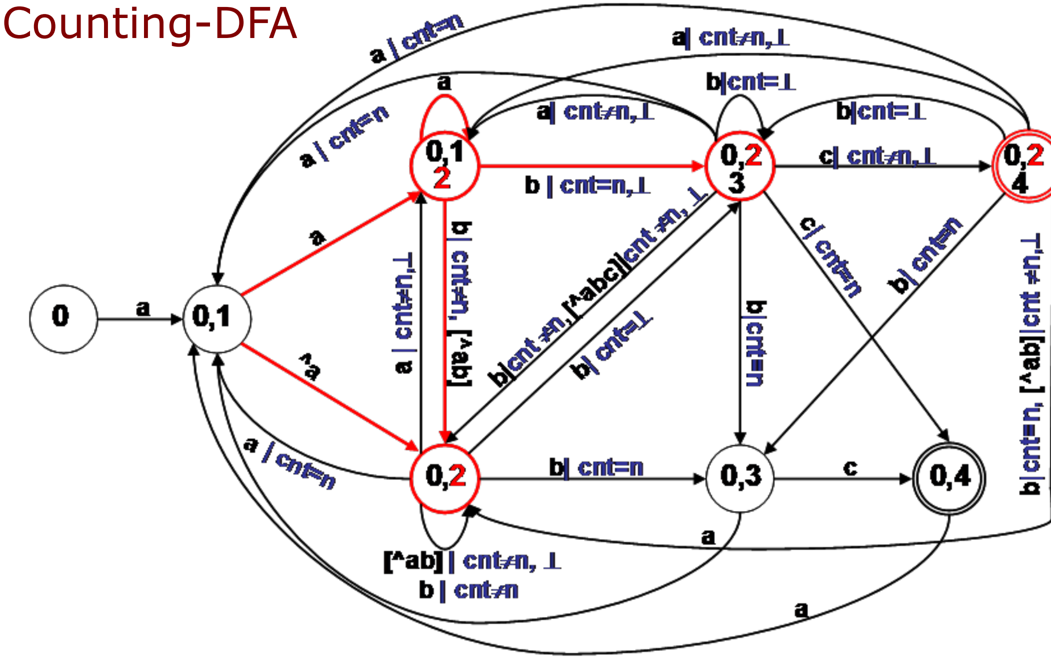
E.g: a.{n}bc



Counting-NFA

- Extended NFA-DFA transformation
  - Counting states
  - Instantiating transitions
  - Conditional transitions

Counting-DFA



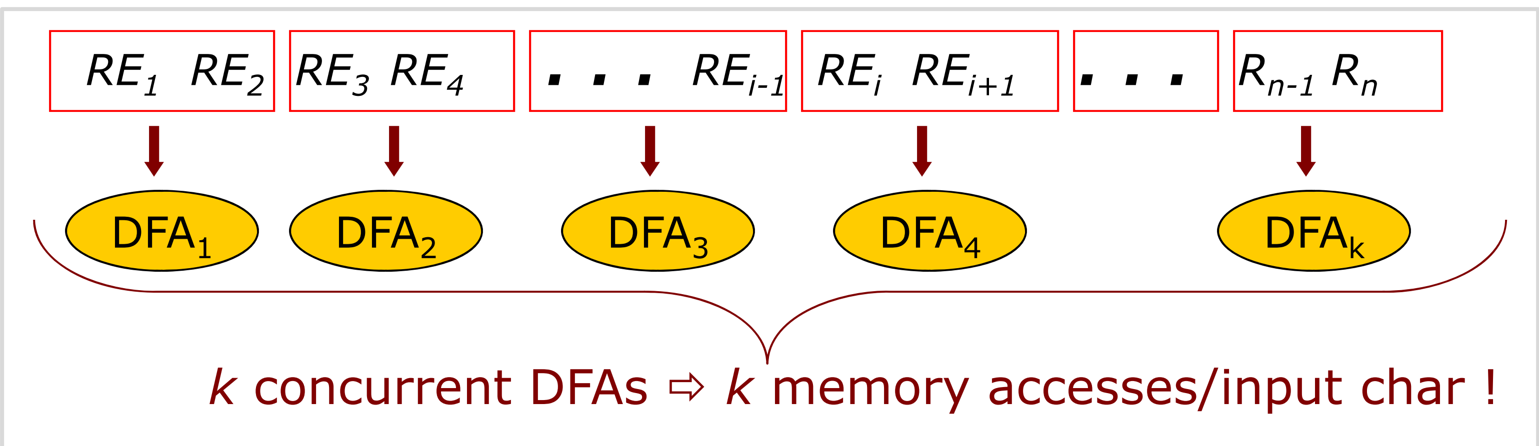
- Possible conditions:
  - cnt=n: $c_i'=n$ *and* c_i' *is single instance*
  - cnt≠n: $c_i' \neq n$
  - cnt=⊥: $c_i'=n$ *and another instance* c_j exists

- Consequences:
  - Limited memory bandwidth (1 state + 2 counter instances)
  - Limited size (*independent of n*)

Combining multiple regex

Patterns = $\{RE_1, RE_2, RE_3, \ldots RE_n\}$

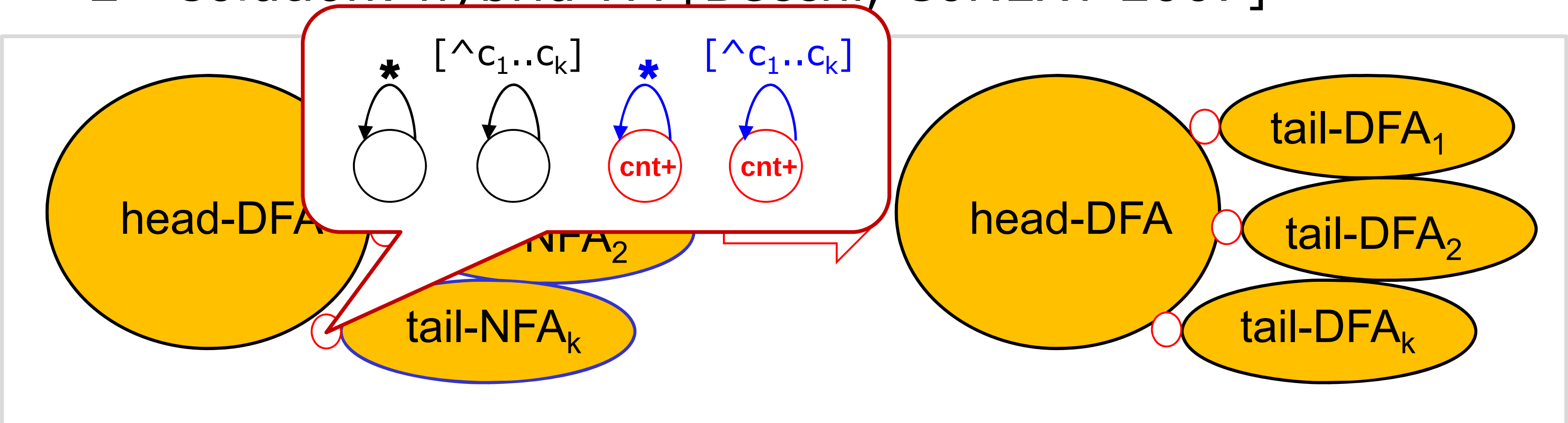
- 1^{st} solution: regex partitioning [Brodie, ISCA'06][Yu, ANCS'06]



➡ High parallelism and memory bandwidth: ASIC, FPGA

Combining multiple regex (cont'd)

- 2nd solution: hybrid-FA [Becchi, CoNEXT 2007]



- Memory Size:
  - » Limited, independent of # of closures states
- Memory Bandwidth:
  - » Average:
    - » only head-DFA active
    - » one state traversal/character
  - » Worst case:
    - – All tail-FAs are active
    - – Bandwidth= # DFAs state traversal + 2 accesses/counters, per char

➡ Low-Medium parallelism and memory bandwidth: GPP, small CMP

Back-references

- Idea: a given sub-expression must be matched multiple times with the same text

- Examples
  - *(abc|bcd).\1y* matches *abcdabcdy*, does not match *abcdaabcy*
  - *a([a-z]$^+$)a\1y* matches *babacabacy*

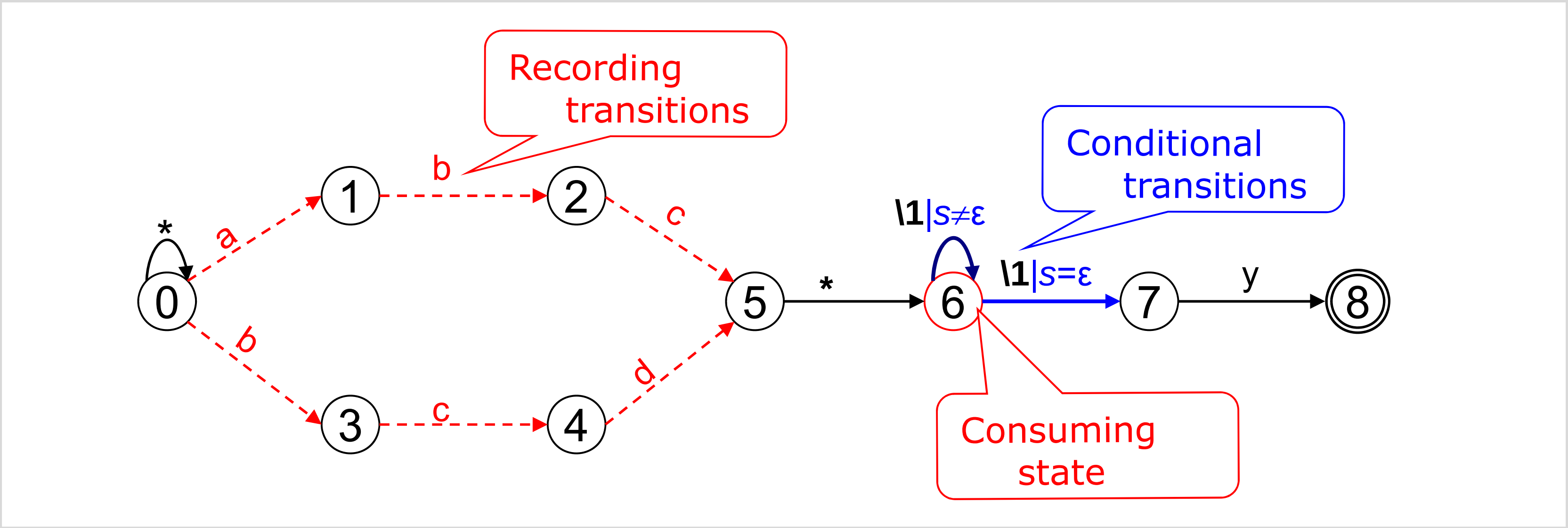
- Observations
  - The alternative in the referenced sub-expression may overlap
  - The captured sub-expression may overlap w/ previous/next char
  - The length of the referenced sub-expression may be variable

Memory needed

GOAL: preserve NFA-like operation:
- Find all matches/stop at the first
- Process each char once
- Allow parallel RegEx processing

Extended-FA

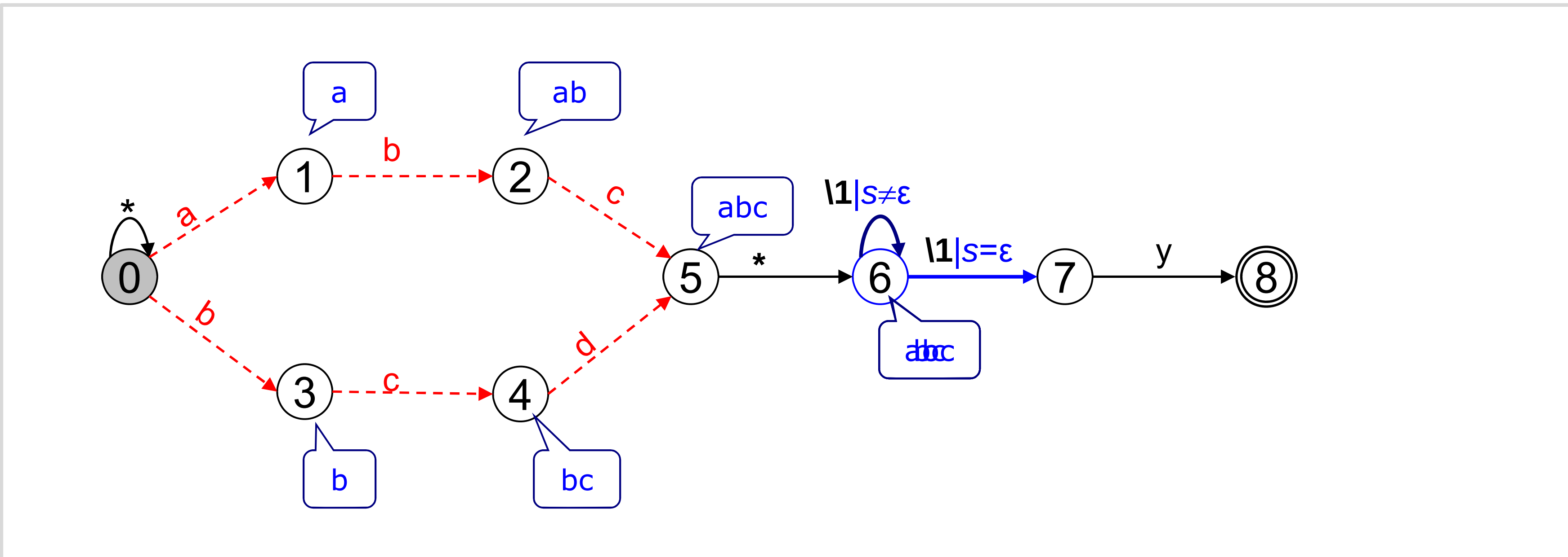
E.g.: (abc|bcd).\1y



- Extensions:
  - » Recording and conditional transitions, consuming states
  - » Each state associated with a set $\{PM_k\}$ of partial match strings

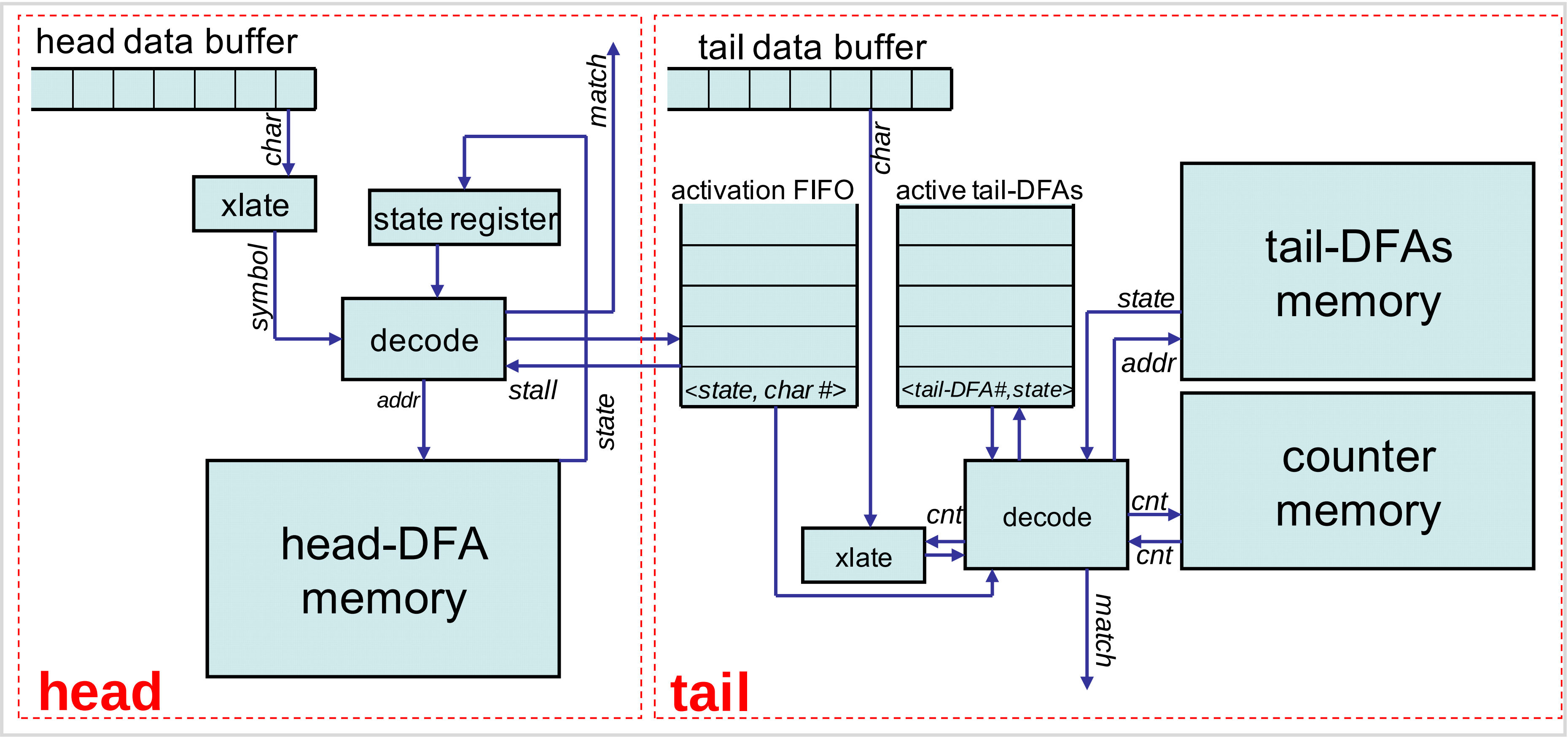
Extended-FA operation

E.g.: (abc|bcd).\1y

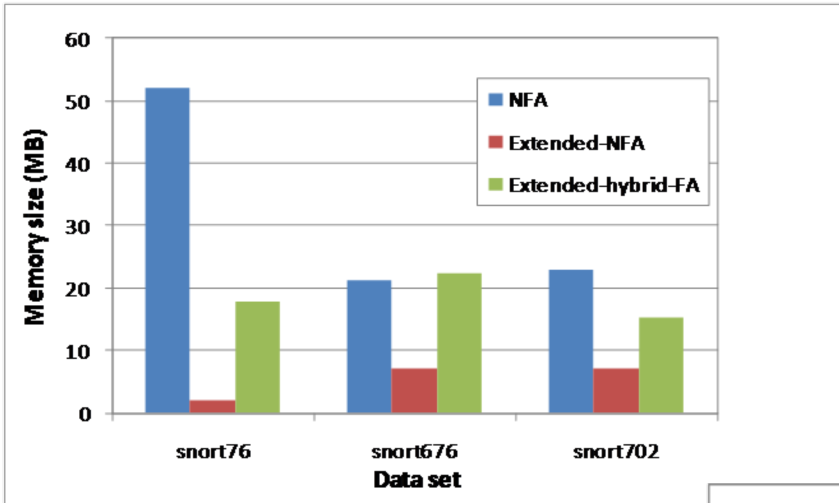


Text: a b c e a b c y

Matching architecture



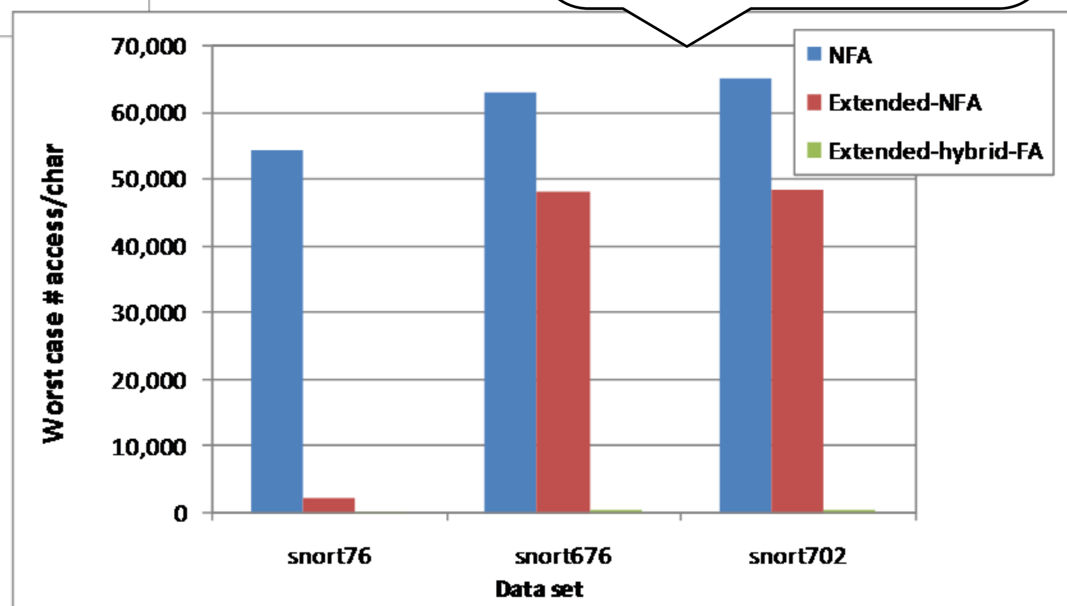
Results



Memory size

	# counters	# backref
snort76	46	0
snort676	[illegible]	[illegible]
snort702	[illegible]	[illegible]

Hybrid-FA has memory bandwidth from 10X to 100X lower



Worst case memory bandwidth

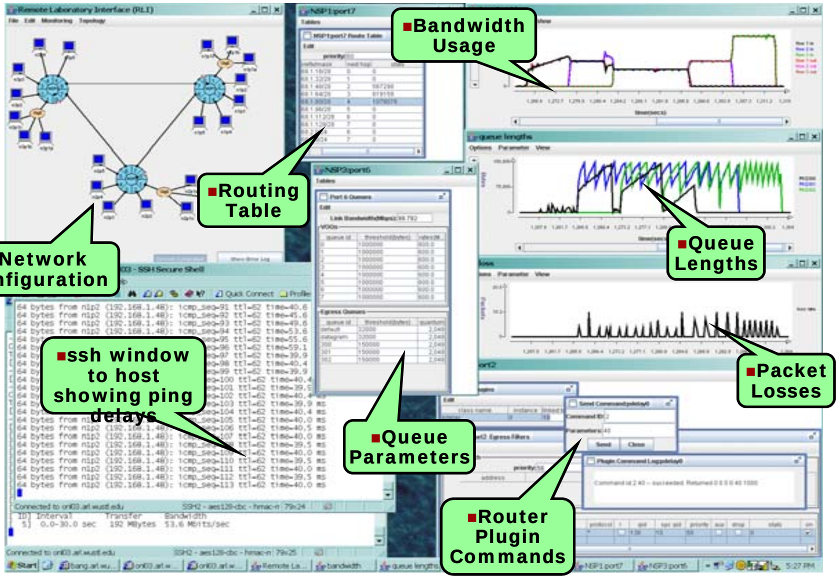
Experimental Environment & Testbed

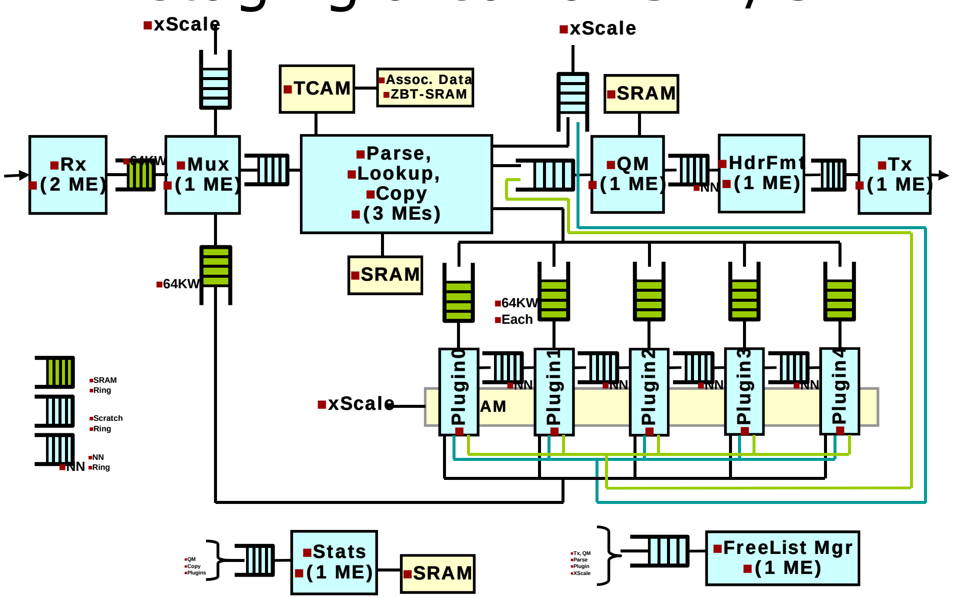
- Open Network Lab

- This is the result of the combined efforts of around 20 faculty, students, and staff led primarily by **Jon Turner**

Open Network Lab

- ONL is an Internet-accessible networking lab (onl.wustl.edu)
- Major recent expansion
  - » 14 new NP-based routers
  - » staging area for SPP/GENI





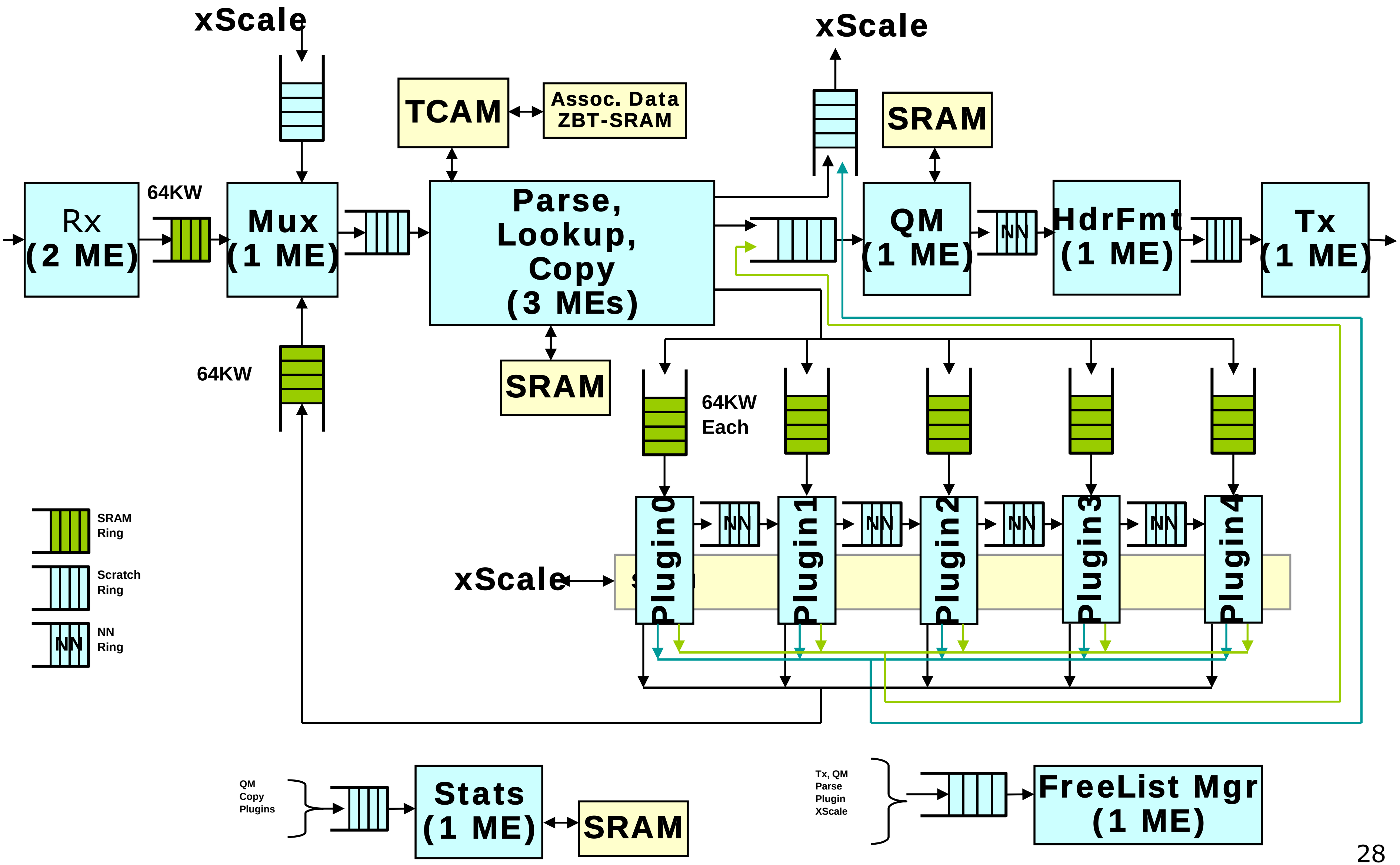
ATCA & NP-based Routers

- Development of ATCA enables use of commercial router components

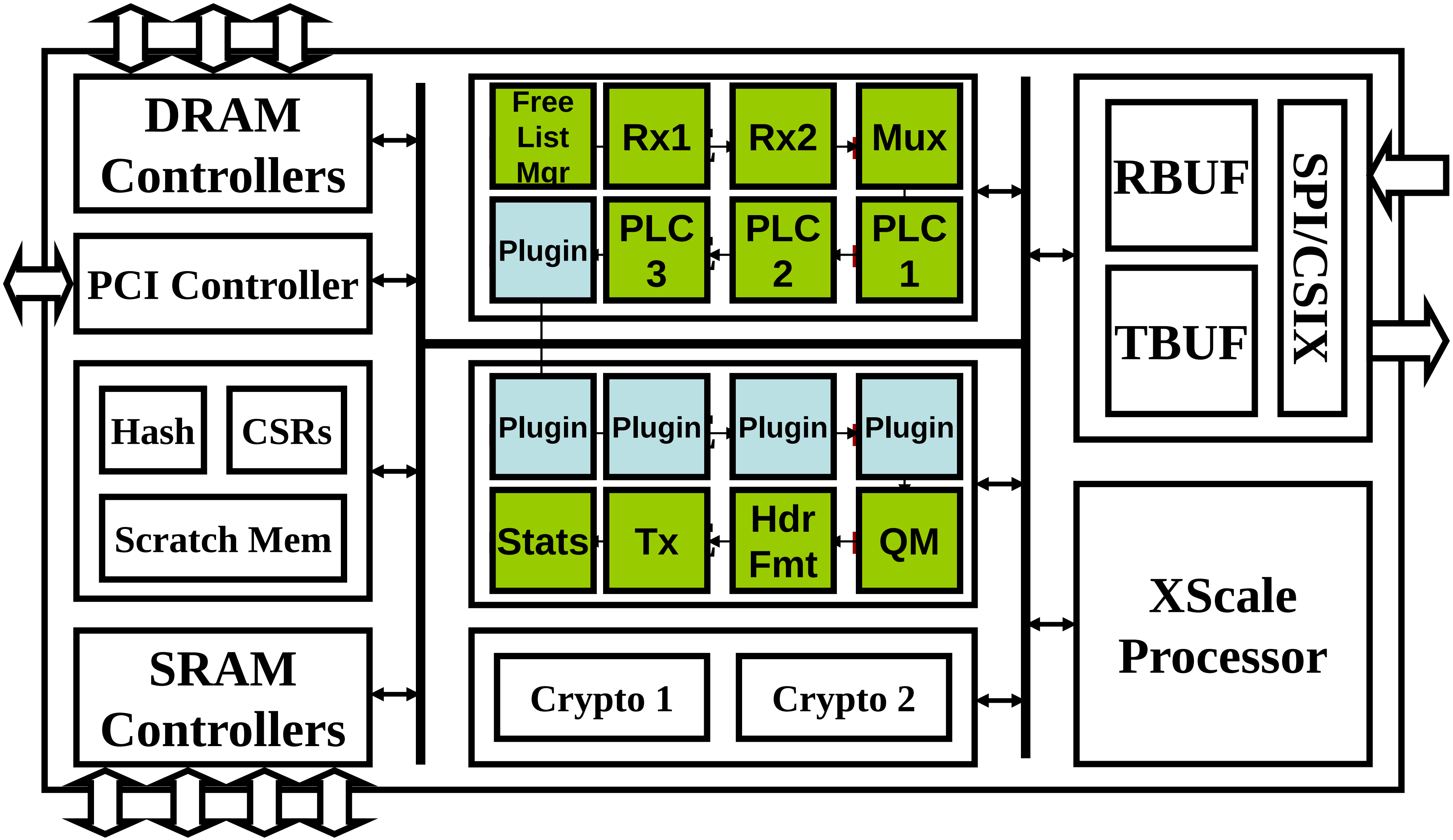
- Packet Processing: Radisys ATCA-7010
  - 10 1 Gbps or 1 10 Gbps links
  - 2 Intel IXP 2855 NPs
    - 17 programmable processor cores each
  - 1.4 GB high-speed RDRAM, 48 MB QDR SRAM, 1 shared 18 Mb TCAM

- *Radisys ATCA-7010*

Router SW Organization



Where can users insert their code?



Sample Applications

- ISP-Managed P2P
- This is the work of my current student **Shakir James**
- The full work will be published later this month at IEEE P2P 2010 in Delft

P2P: Two Points of View

- A user's point of view
  - » Support many applications
  - » Offer "inexpensive" scalability
  - » Recover quickly from failures

Image from talk "P2P: An ISP's Point of View," by Pablo Rodriguez

- An ISP's point of view
  - » Route traffic over costly **transit** links
  - » Increase broadband customers**, but**
  - » Surge in traffic ≠ surge in $$$

The Problem

- Duality of P2P
  - » Cheap for content providers, **but**
  - » Expensive for ISPs
- “Cat and mouse” game
  - » 1. ISPs: Install traffic-shaping devices
  - » 2. P2P : Obfuscate traffic
  - » 3. Repeat until…

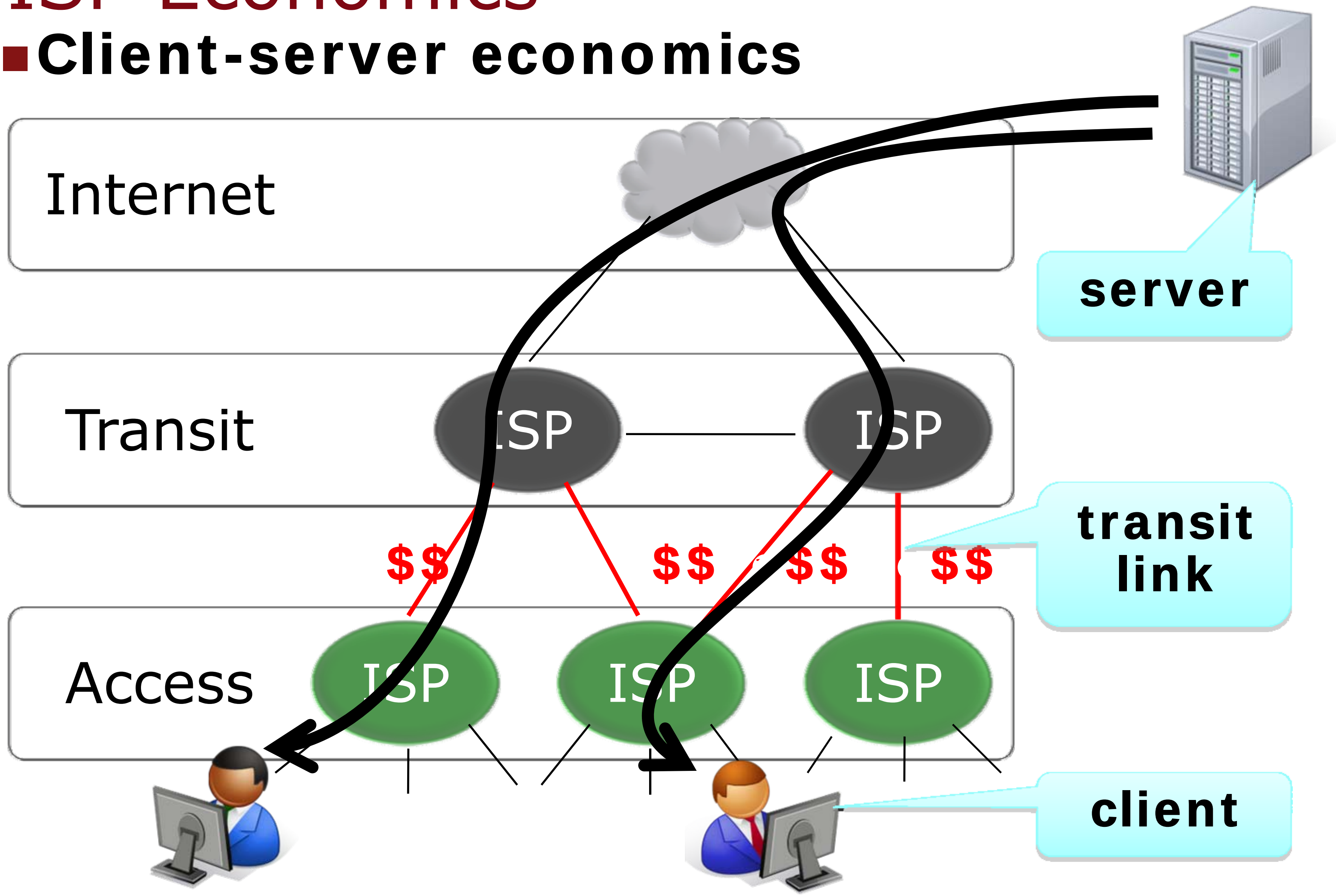
- No end in sight!
  - » FCC forced ISPs to (temporarily) capitulate
  - » Damaged relationship in long-term

Our Goals

- Build a network device that
  - » Controls costs for the ISP, **and**
  - » Maintains good performance for end-users
- Show that ISPs can take unilateral action to
  - » Foster a sustainable co-existence with P2P
  - » Take the first step in fixing relationship
- Two issues to resolve
  - » Illegal content? DMCA "Safe Harbor" a la YouTube
  - » **How does P2P increase costs for ISPs?**

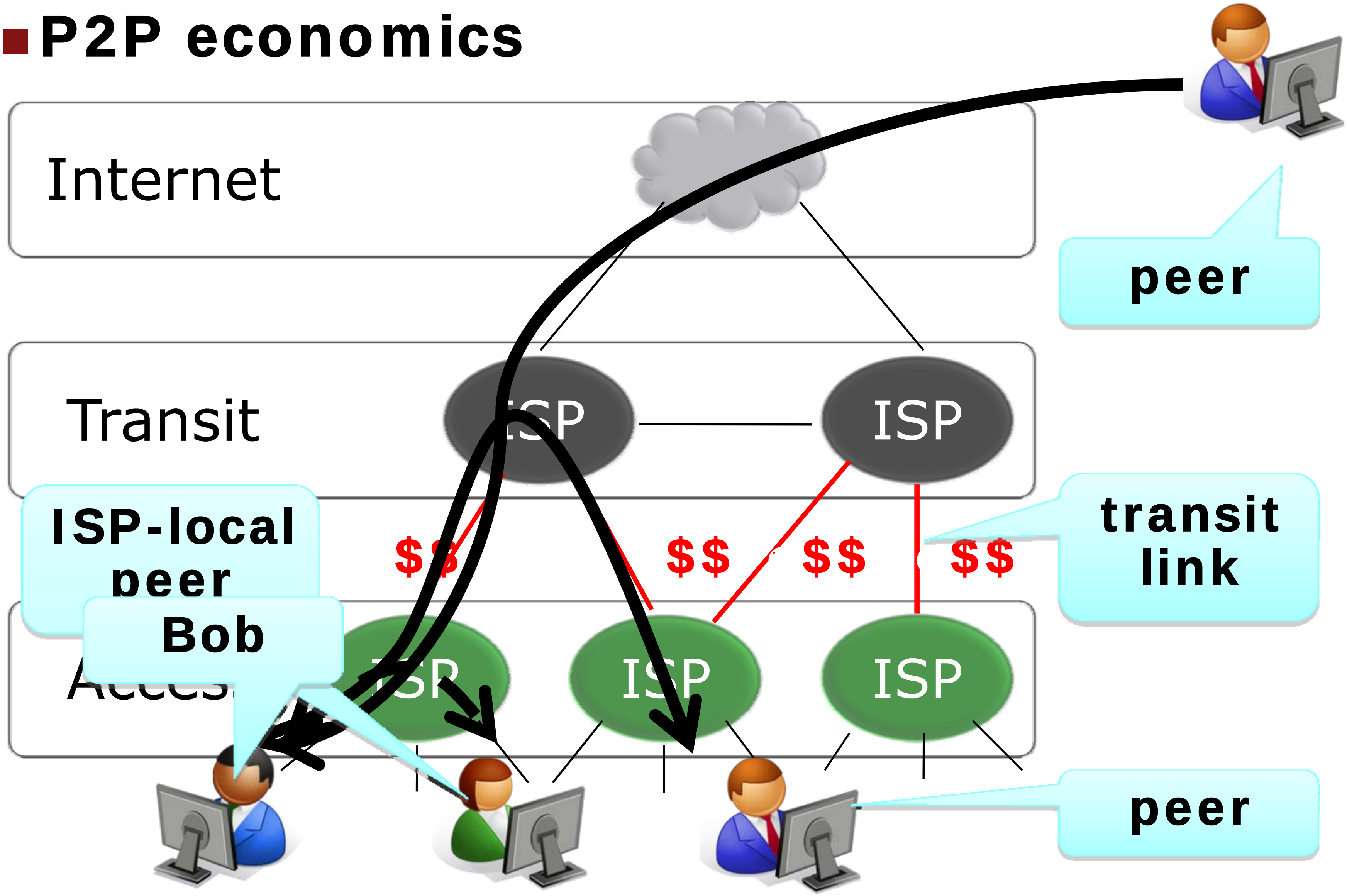
ISP Economics

- Client-server economics

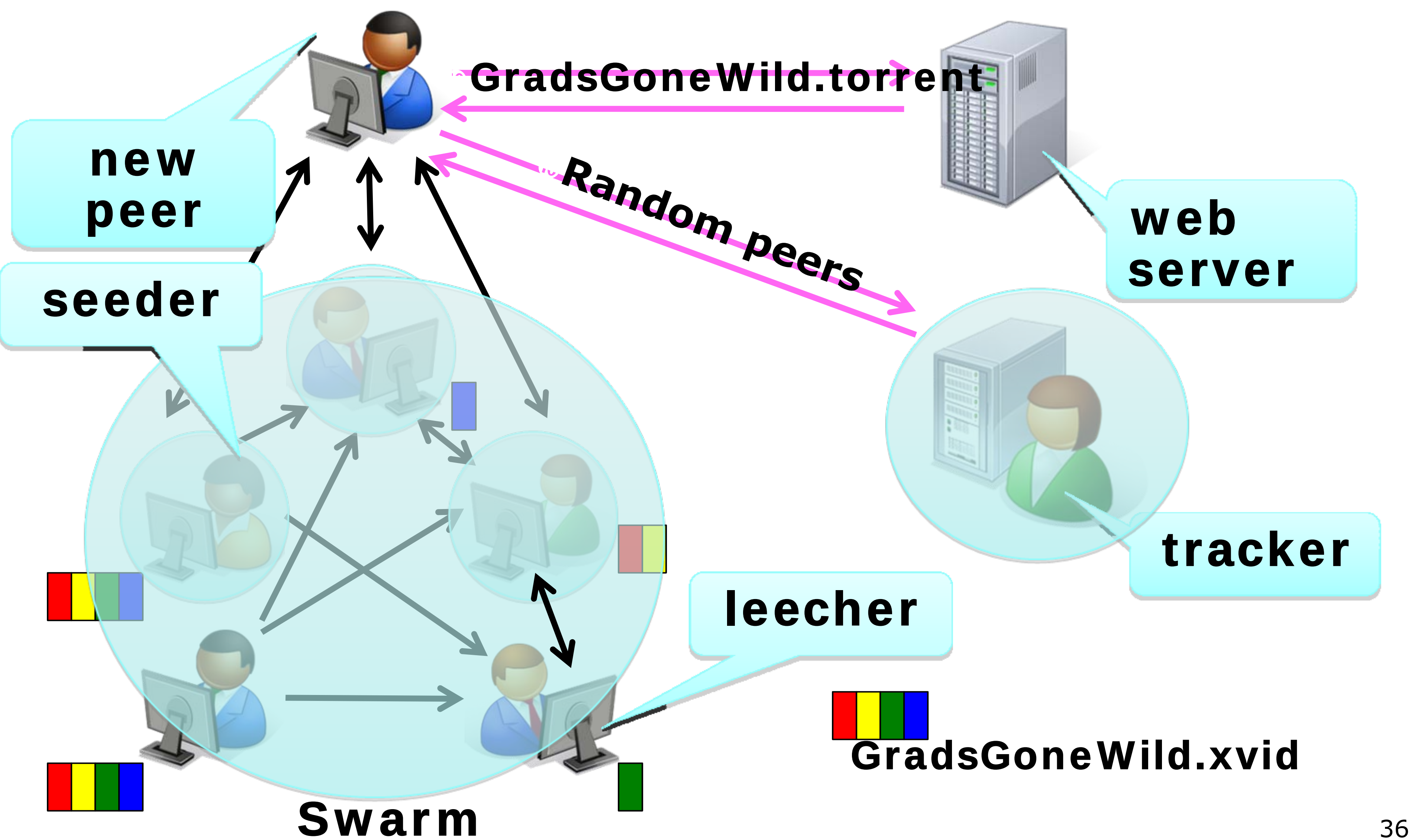


ISP Economics

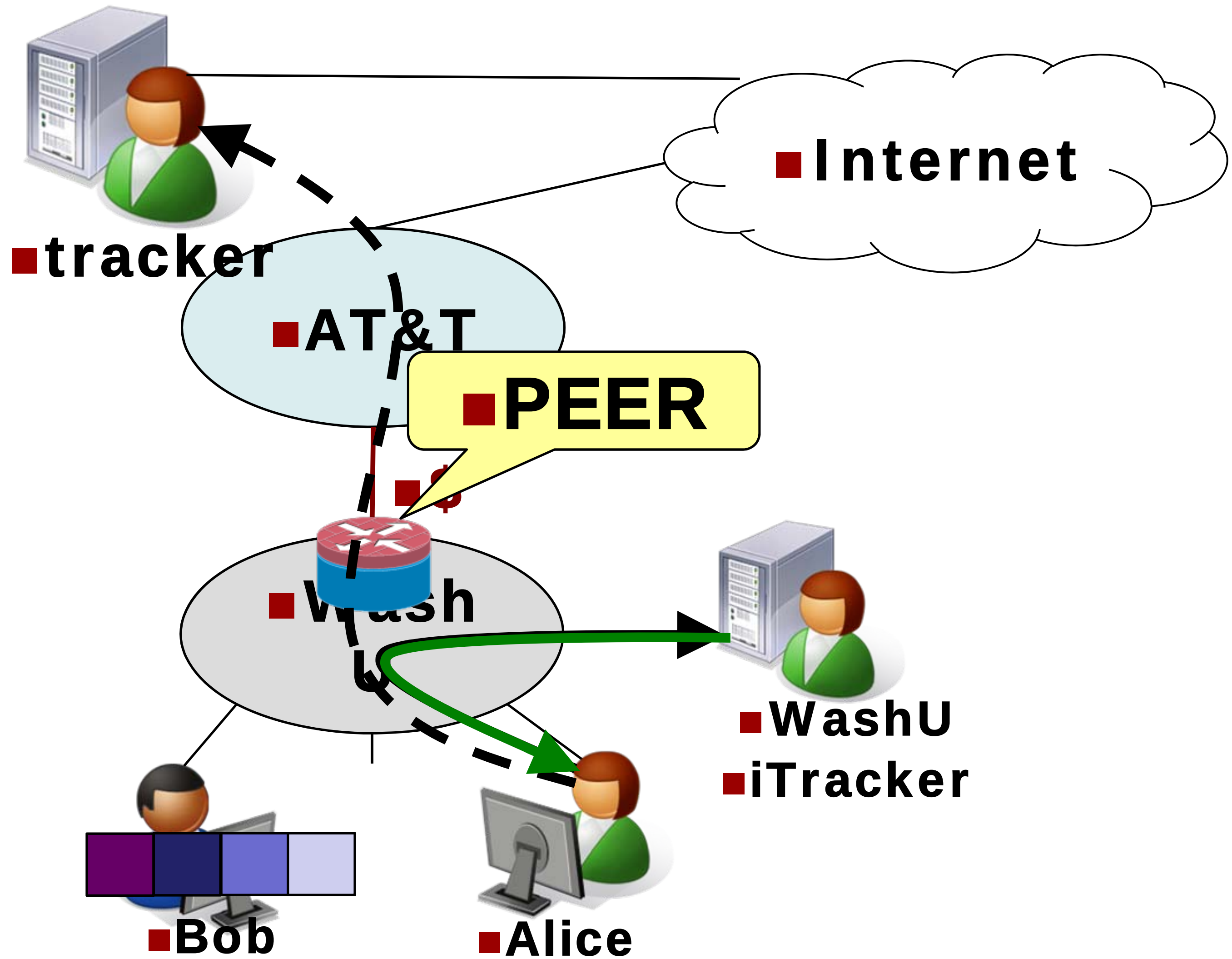
- **P2P economics**



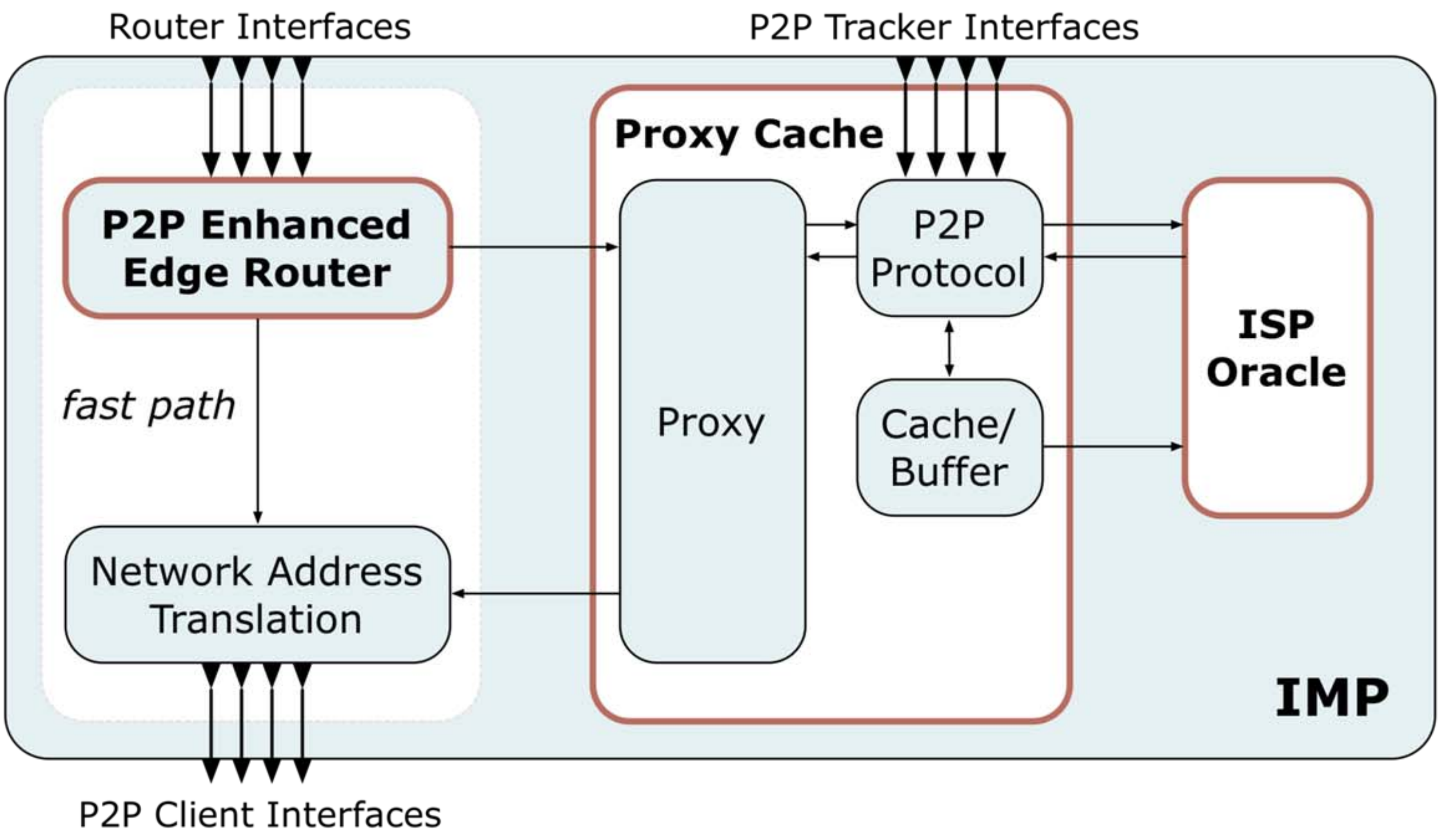
BitTorrent Operation



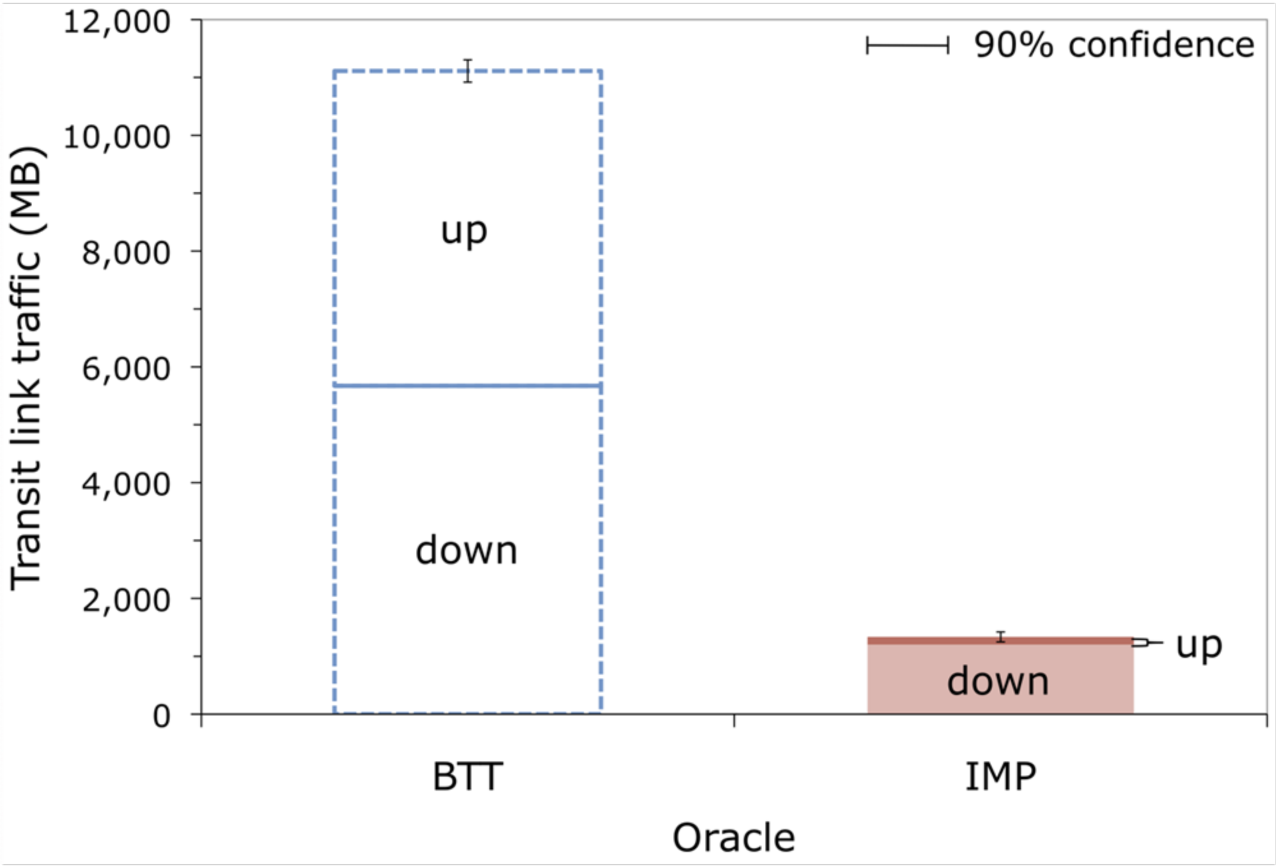
IMP: ISP-Managed P2P



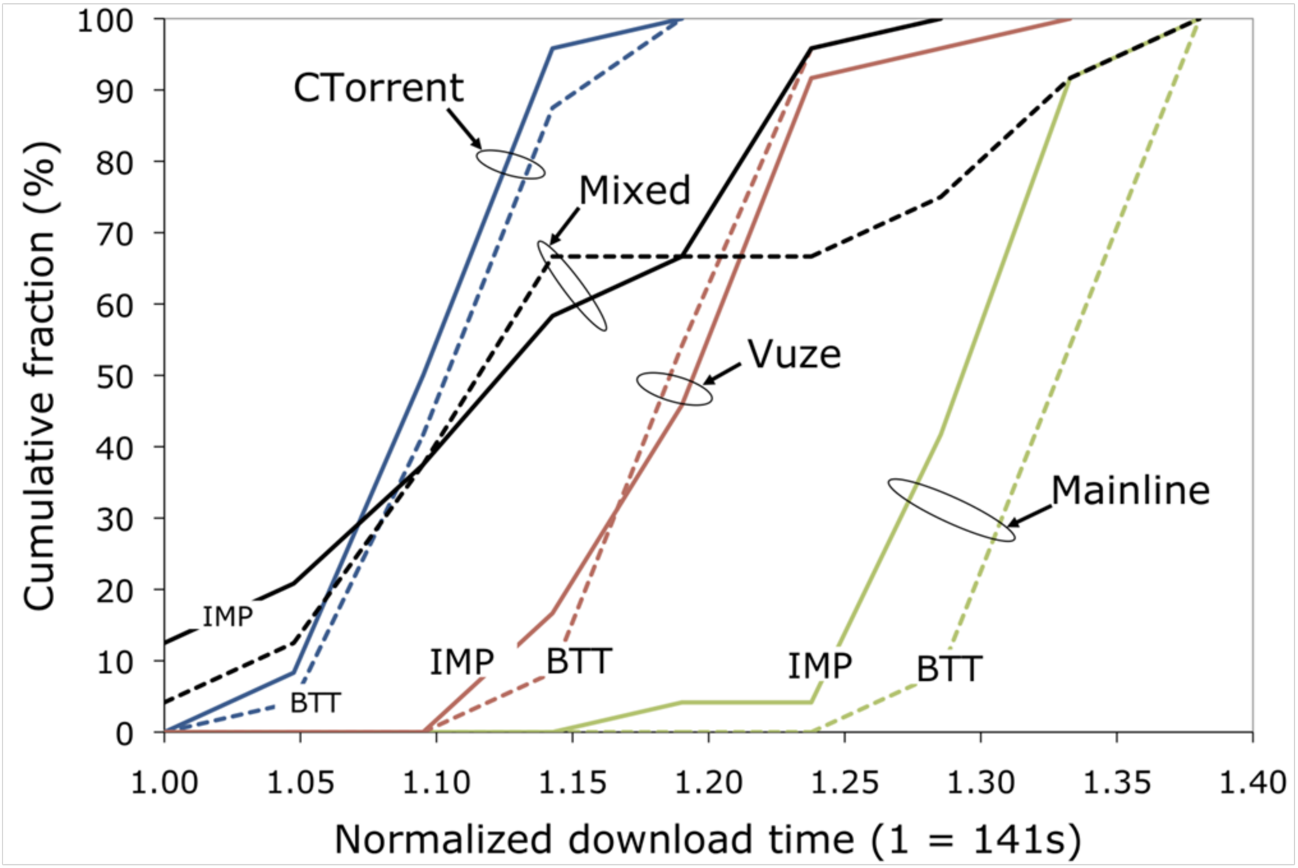
IMP Architecture



IMP: Cross-ISP traffic

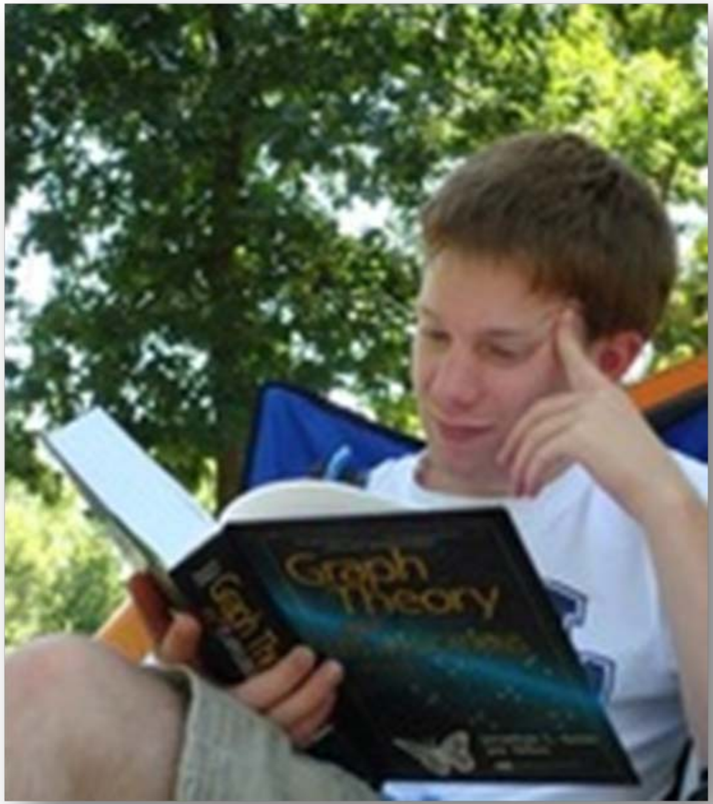


IMP: Download Time/Multiple clients



Sample Applications

- Passive Network Analyzer
- This is primarily the work of my current student **Michael Schultz**



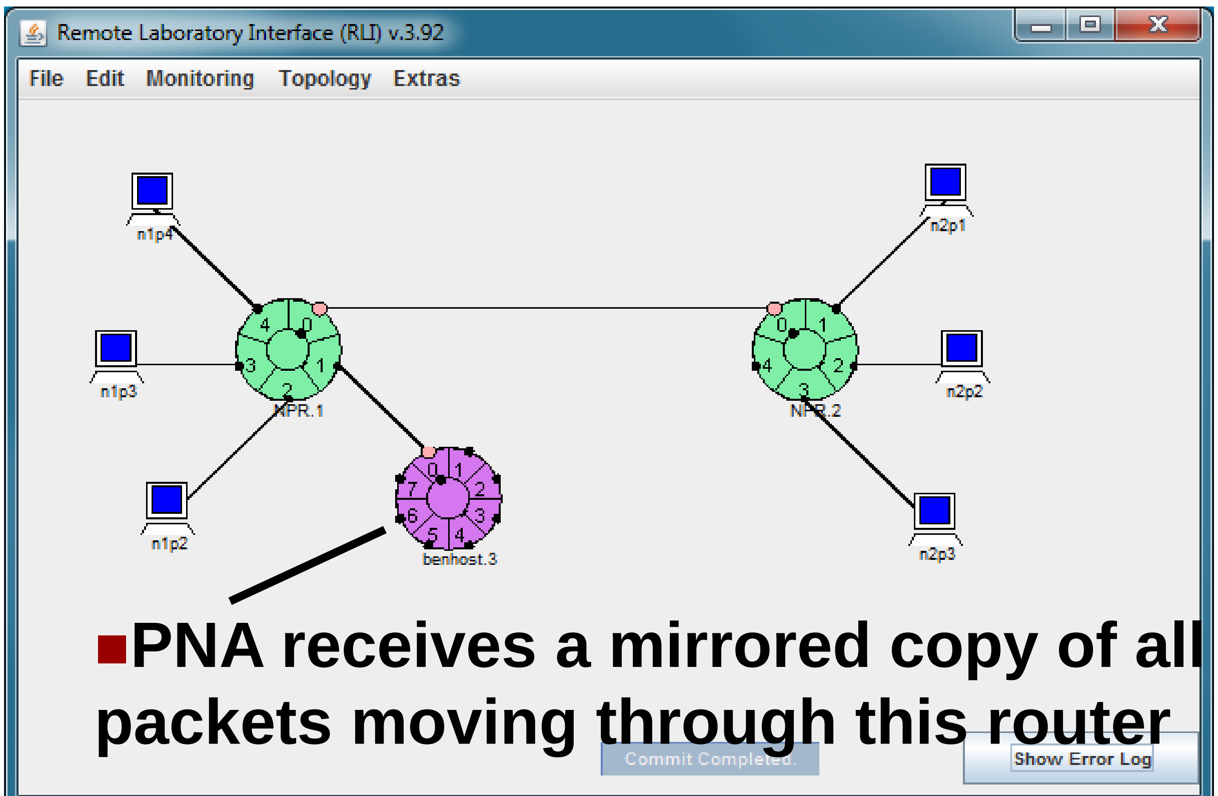
Passive Network Analyzer: A Platform for Real-Time Network Monitoring and Analysis

- Enables the development of customized, real-time "network monitors" to measure net activity
  - Input: mirrored traffic from switches
  - Output: real-time data/models updated instantaneously at packet arrival rates
  - Example monitor: track real-time flow state for all active sessions in an enterprise, log every 10 seconds

- Pertinent technical details
  - Linux SW stack, modified OS kernel
  - Built primarily via "netfilter" API
  - Data currently logged to Amazon S3
  - Leverage due to:
    - Multilevel hash table design to track flow state
    - Efficient kernel modifications

Demonstration Scenario

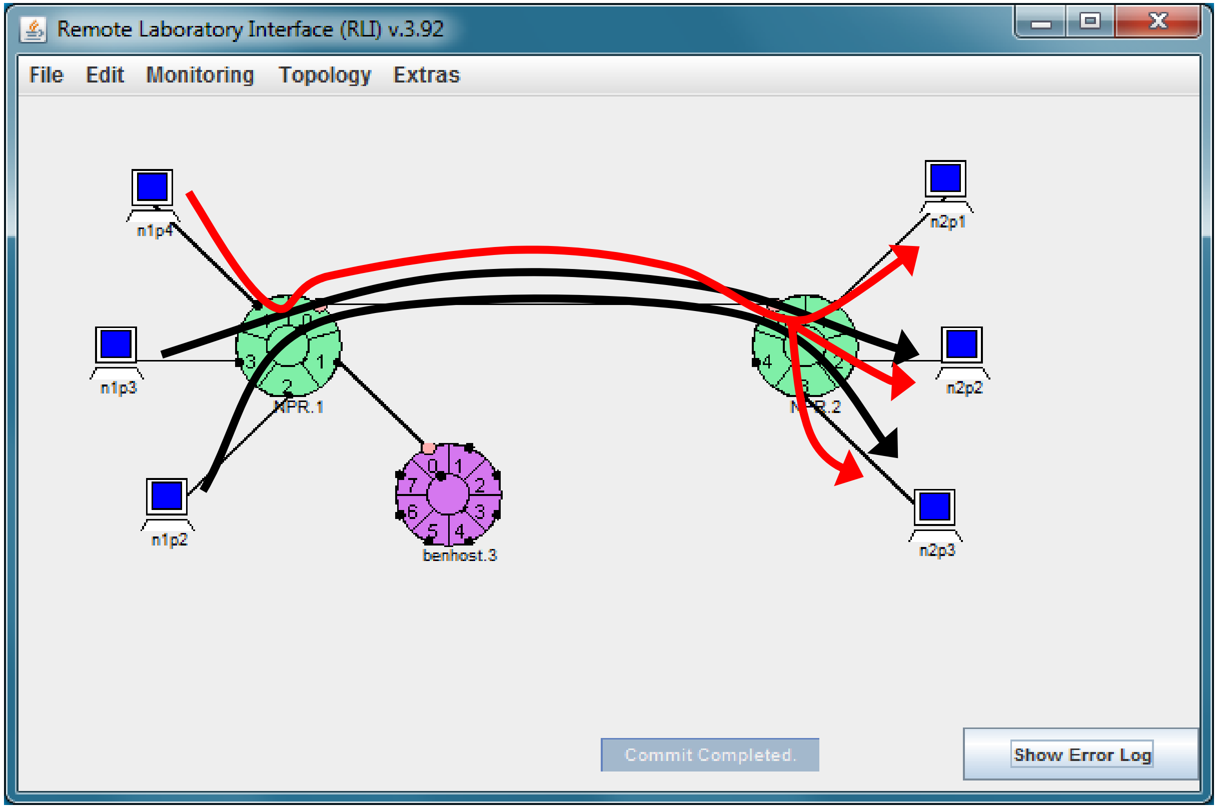
- The PNA system will, in real time,
  - » Detect a troublesome end-system (e.g., one that opens more than 100 sessions within 10 seconds), and
  - » Install a filter to drop its subsequent packets.



- **PNA receives a mirrored copy of all packets moving through this router**

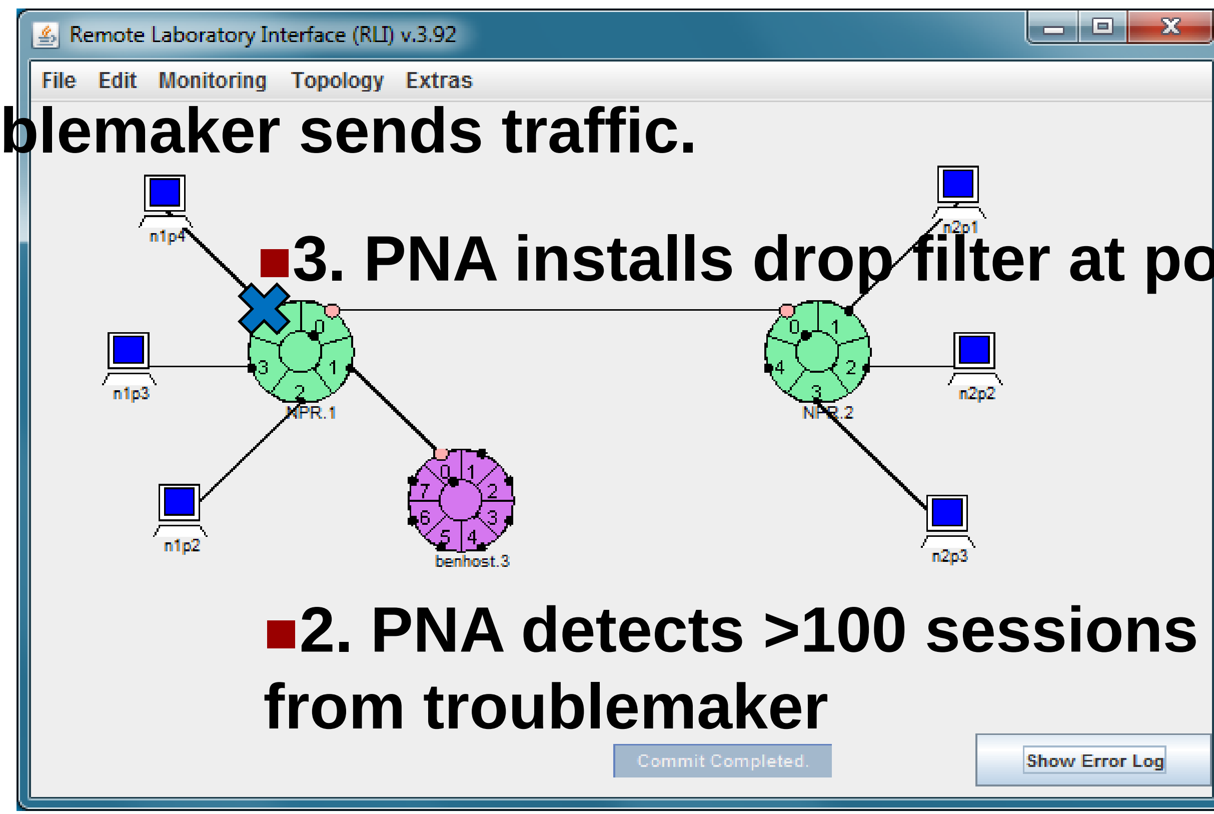
Demonstration Scenario

- 2 sender-receive pairs will send “normal” traffic
- 1 sender will open many sessions on each of the destination machines



Demonstration Scenario

- Demo proceeds in 3 steps



- 1. Troublemaker sends traffic.
- 3. PNA installs drop filter at port
- 2. PNA detects >100 sessions from troublemaker

Conclusion

- High-speed networking mechanisms
  - » Appear to have perpetual importance
  - » Require more smart people to work in the area
  - » Provide a rare combination of satisfying intellectual contributions with near- and long-term industry impact

- The next big mechanism (in my opinion): Names!
  - » URL processing at frightening scales
  - » Forwarding in Van Jacobson's Named Data Networking

- For more information
  - » www.arl.wustl.edu/~pcrowley
  - » The ANCS conference: www.ancsconf.org