IP puzzles, probabilistic networking, and other projects at OGI@OHSU

Wu-chang Feng

Louis Bavoil
Damien Berger
Abdelmajid Bezzaz
Francis Chang
Jin Choi
Brian Code
Wu-chi Feng
Ashvin Goel
Ed Kaiser
Kang Li
Antoine Luu
Mike Shea
Deepa Srinivasan
Jonathan Walpole

OGI SCHOOL OF SCIENCE & ENGINEERING

Outline

IP puzzles

- Motivation
- Research challenges
- Design, implementation, and evaluation of a prototype

Other projects at OGI@OHSU

IP Puzzles

otivation

A quick look back on 15 years of not so "Good Times"

SMTP, TCP, ICMP, UDP, FastTrack, SMB, finger, SSL, SQL, etc.

Nachi

Code Red

Fizzer

Fraggle

Klez

Sircam

Iorris worm

Smurf

Slamme

Melissa

SoBig

SYN flood

LoveLetter

hristmas

Delode

Nimda

Michaelangelo

Blaste

988 1993 1998 200

uzzles

- An interesting approach for mitigating DoS activity...
  - Force client to solve a problem before giving service
  - Currently for e-mail, authentication protocols, transport layer
  - Fundamentally changes the Internet's service paradigm
    - Clients no longer have a free lunch
    - Clients have a system performance incentive to behave

contrast in approaches

Leave doors open and unlocked, rely on police/ISPs

- Centralized enforcement (not working)

Give everyone guns to shoot each other with

- Distributed enforcement (may not work either)
- Promising anecdotal evidence with spamming the spammers.
- Harness the infinite energy of the global community to fight problem

osit

Puzzles must be placed in the IP layer to be effective

hy are IP puzzles a good idea?

"Weakest link" corollary to the end2end/waistline argument

Put in the common waistline layer functions whose properties are otherwise destroyed unless implemented universally across a higher and/or lower layer

- DoS prevention and congestion control destroyed if any adjacent or underlying layer does not implement it
  - TCP congestion control thwarted by UDP flooding
  - DoS-resistant authentication protocols thwarted by IP flooding
- Until puzzles are in IP, it will remain one of the weakest links

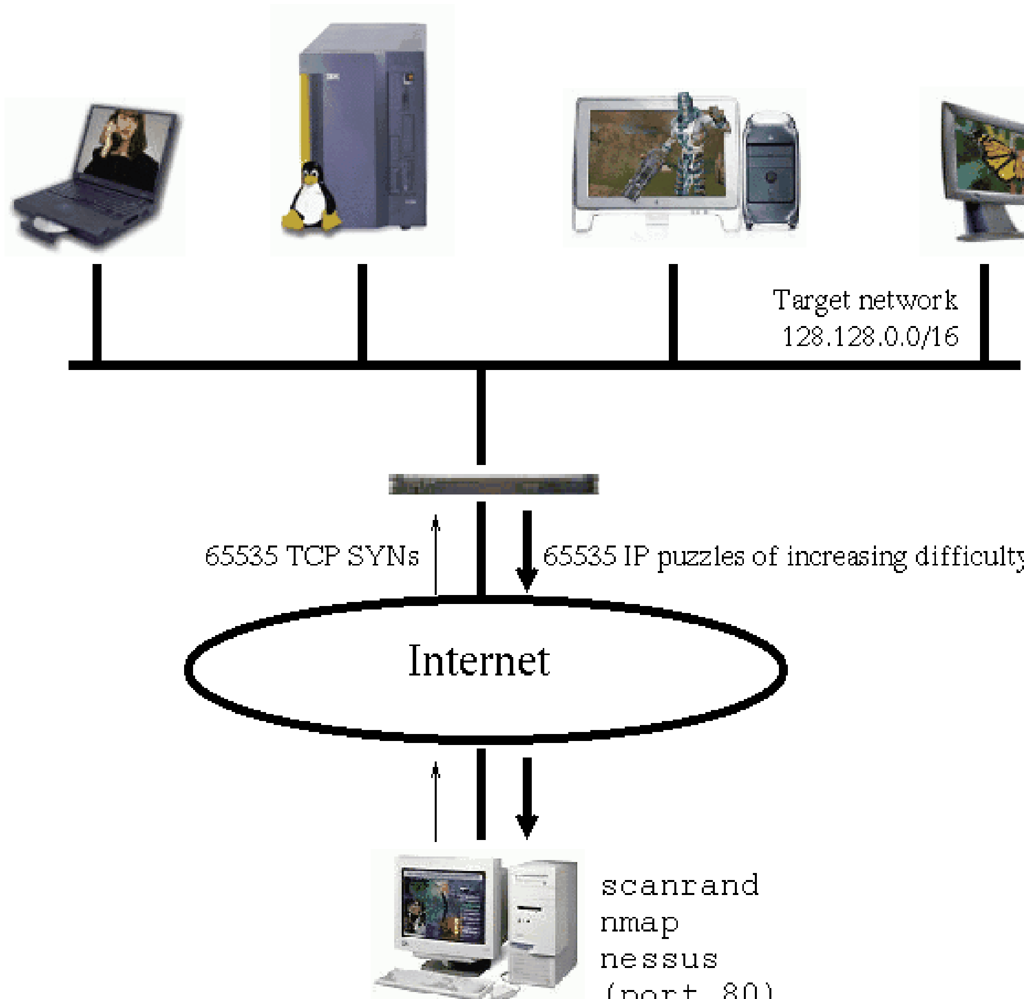
puzzle scenario #1

Port and machine scanning

- Instrumental to hackers and worms for discovering vulnerabl systems
- The nuclear weapon: `scanrand`
  - Inverse SYN cookies and a single socket
  - Statelessly scan large networks in seconds
    - 8300 web servers discovered within a class B in 4 seconds
  - Technique not used in any worm....yet
    - Forget Warhol and the 15 minute worm (SQL Slammer)
    - Need a new metric: "American Pie" worm => done in 15 seconds?
    - Finally, a grand networking challenge!

' puzzle scenario #1

- Mitigation via a "push-back" puzzle firewall



hy are IP puzzles a bad idea?
What are the research challenges?)

- Tamper-resistance
- Performance
- Control
- Fairness

amper-resistance

A tool to both prevent and initiate DoS attacks

- Disable a client by...
  - Spoofing bogus puzzle questions to it
  - Spoofing its traffic to unfairly trigger puzzles against it
- Disable a router or server by...
  - Forcing it to issue loads of puzzles
  - Forcing it to verify loads of bogus puzzle answers
  - Replaying puzzle answers at high-speed
- Probably many more....

erformance

- Must support low-latency, high-throughput operation
  - Must not add latency for applications such as on-line games
  - Must support high-speed transfers
  - Must not add large amounts of packet overhead
- Determines the granularity at which puzzles are applied
  - Per byte? Per packet? Per flow? Per aggregate?
  - Driven by performance and level of protection required

ontrol

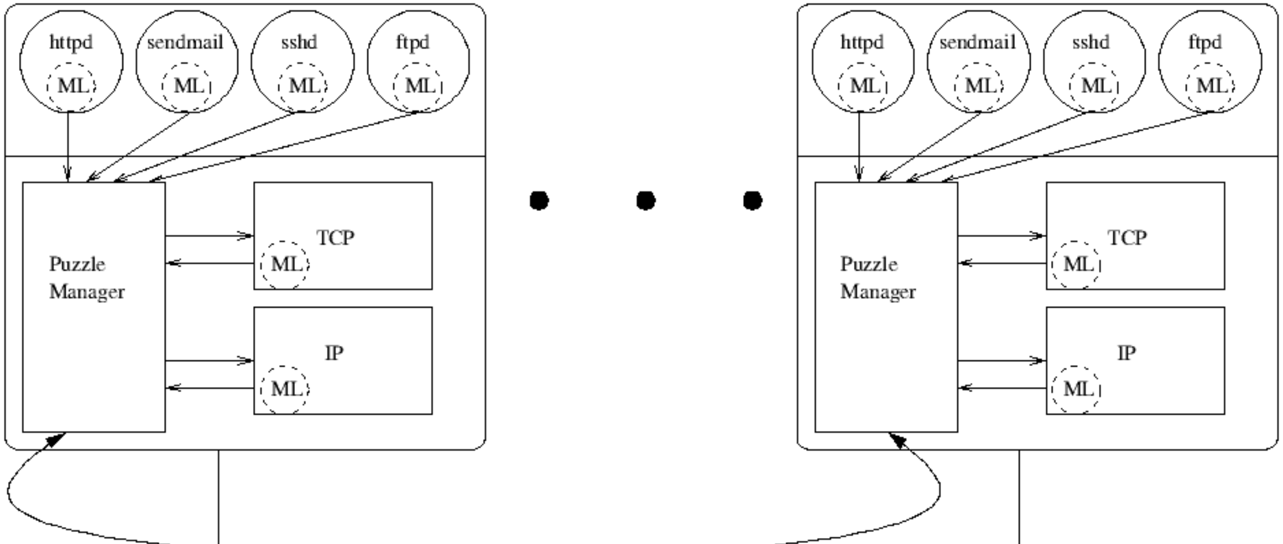
- Control algorithms required to maintain high utilization and low loss
  - Mandatory, multi-resolution ECN signals that can be given at any time granularity
  - Can apply ideas from TCP/AQM control
    - Adapt puzzle difficulty within network based on load
    - Adapt end-host response to maximize throughput while minimizing system resource consumption (natural game theoretic operation)

airness

- Minimize work for “good citizens”, maximize work for bad one
  - Problem: mechanism is in a layer with minimal information
    - Can support bandwidth-based puzzle delivery
    - Can support some differentiation to deter Smurf/Fraggle

202.183.197.116 - - [02/Jun/2003:02:08:29 -0700] "GET /default.ida?XXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u7801%u909
0%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%
u531b%u53ff%u0078%u0000%u00=a HTTP/1.0" 404 306 "-" "-"

- Need a “puzzle manager”
  - Drive IP-layer puzzle policy based on application input



eputation-based networking

- Reputation determines puzzle difficulty
    - *f(OS, Applications, Admins, EndUser)*
- Implications
    - Software vendors
        - Making "trustworthy computing" mandatory (not marketing)
        - Long-term, computational tax for poorly designed software
    - System administrators and IT practices
        - Making responsible system management mandatory
        - Disturbing pervading notion: "cheaper to leave infected than patch"
        - Long-term, computational tax on poorly administered systems
    - End-users
        - Making users choose more secure software and adopt better practices
        - Punish users behaving "badly"
        - Long-term, computational tax on ignorance and maliciousness

"Nothing is certain but death and taxes." - Benjamin Franklin

hy is this good for Intel?

Keeping the Internet healthy via CPU cycles

Drives a whole new market for faster CPUs

- Make the incompetent, the lazy, and the malicious “pay” for use of the Internet
- Computational tax paid directly to Intel

Demand for a whole new class of network devices

- Puzzle proxies and firewalls based on IXP network processor

Is this for real?

Yes

- Protocol design
- Puzzle design
- Prototype implementation
- Evaluation

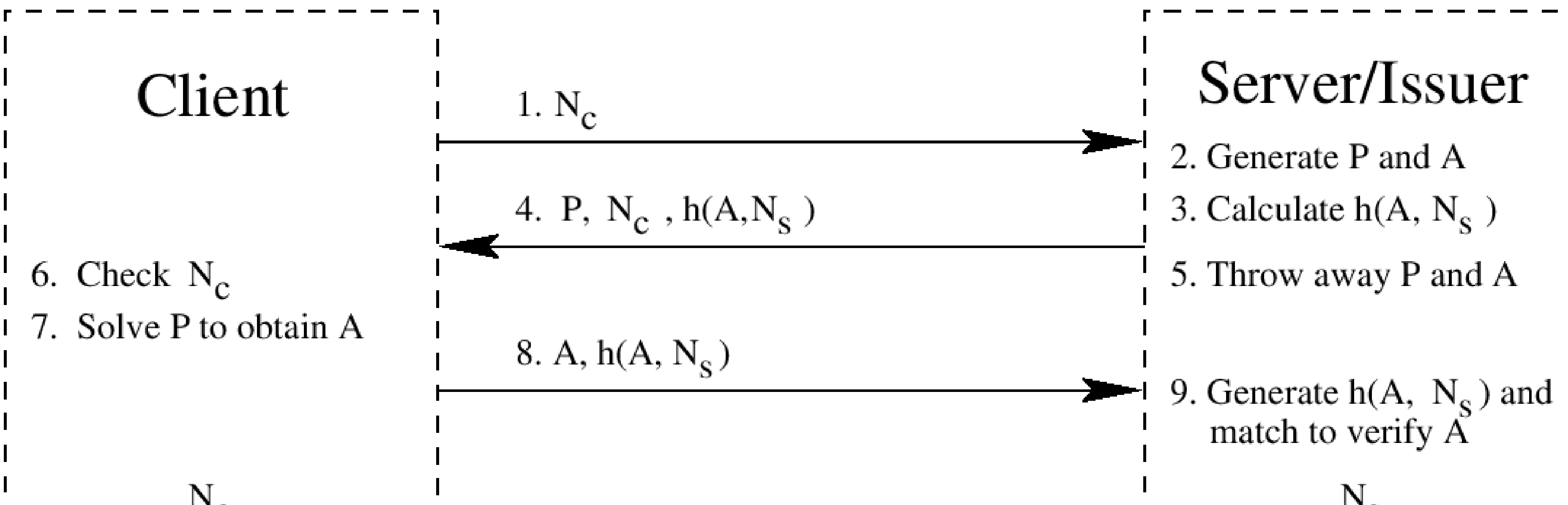
Basic protocol

Based on

- SYN cookies [Bernstein1997]
- Puzzle-protected authentication systems [Aura2001, Leiwo2000]

Features

- Stateless
- Resistant to puzzle spoofing



Understanding the basic protocol

Client nonce

- Client attaches nonce that server must echo in puzzle messag
- Prevents bad guy from spoofing a puzzle to the client

Server nonce and puzzle generation

- Server generates puzzle/answer on the fly
- Uses secret nonce to “sign” a hash of the answer
- Sends puzzle along with above hash
- Throws away the puzzle and answer

Client response

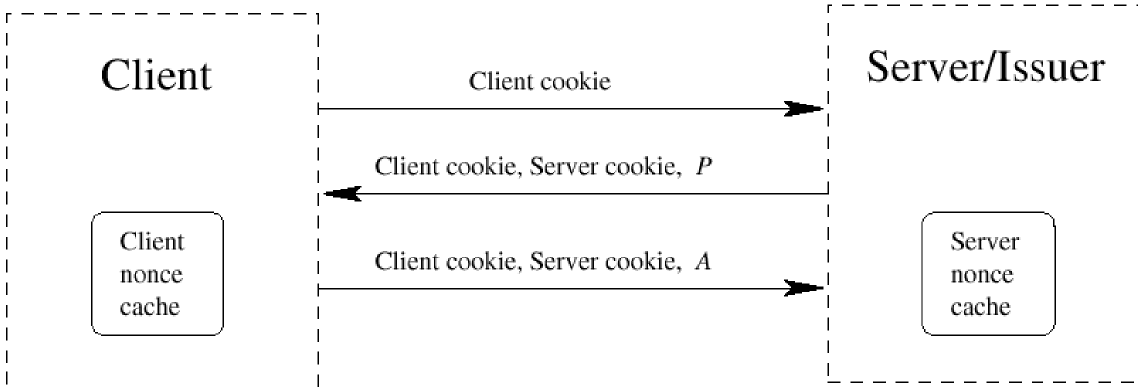
- Attaches answer along with signed hash
- Server verifies valid answer via correctly signed hash

Our modifications

What about….

- Brute-force attacks on N_s
  - Randomly generated circular nonce array continuously updated
- Efficient verification
  - Add logical timestamp to index into circular nonce array (O(1) lookup)
- Infinite replay
  - Add puzzle expiration time
- Streaming applications
  - Issue puzzles ahead of time to client and add puzzle maturity time
- Slow clients
  - Send difficulty estimates to give clients the option to abstain

Final protocol design



Protocol field	Description
Client cookie	TS_c, N_c
Server cookie	$D_p, T_m, T_e, TS_s,$ $h(A, T_m, T_e, TS_s, N_s, TS_c, N_c)$
TS_c	Client timestamp
N_c	Client nonce
P	Puzzle
A	Answer
TS_s	Server timestamp
N_s	Server nonce
D_p	Puzzle difficulty
T_m	Puzzle maturity time
T_e	Puzzle expiry time
$h()$	Fixed hash function

Puzzle algorithms

Have the body of the car (i.e. the protocol)

Need a good engine (i.e. the puzzles)

Can one develop a puzzle algorithm that can support….

- Puzzle generation at line speed
- Puzzle verification at line speed
- Fine-grained control of puzzle difficulty

Puzzle algorithms

- Time-lock puzzles
- Hash reversal
- Multiple hash reversal
- Our approach
  - Hash-based range puzzles

Puzzle algorithms: Time-lock Puzzle

Based on notion of repeated squaring [Rivest,Shamir,Wagner]

- Fine-grained control over difficulty
  - Multiples of squaring time (~1μs)
- Slow to generate (~2ms)
  - 2^t(mod ((p-1)(q-1)))
  - a^e(mod pq)

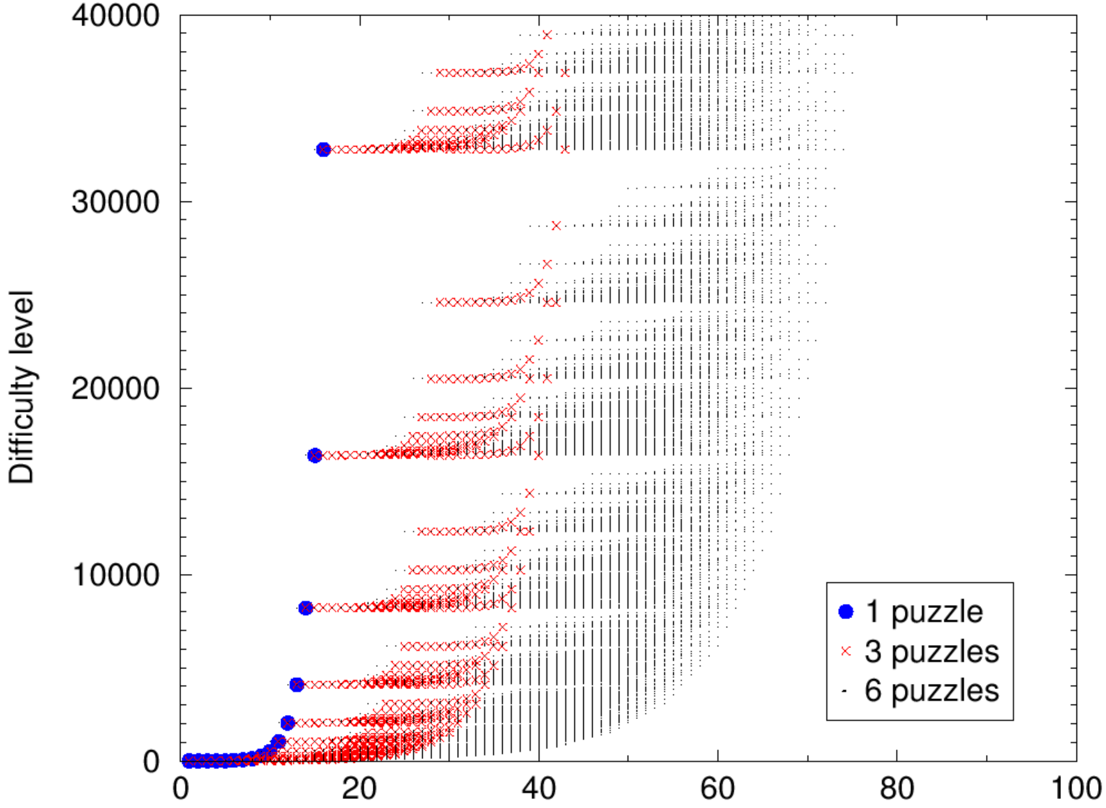
Puzzle algorithms: Hash reversal

Based on reversing a hash

- Brute-force search of input space to find match
- Coarse-grained control over difficulty
  - Difficulty growth as powers of 2
- Fast to generate (~1μs)
  - Hardware support for hashing common
  - IXP 2850

reversal

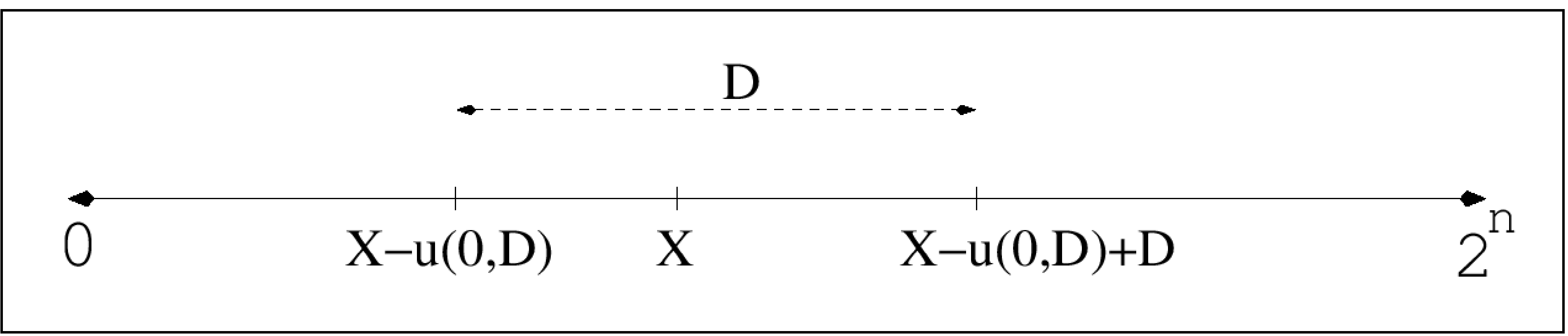
- Reverse multiple hashes
  - Finer control of difficulty
    - Support O(210+211) difficulty?
    - One 11-bit hash = too easy
    - One 12-bit hash = too hard
    - One 10-bit hash and one 11-bit hash = just right
- Fast to generate, but…
  - Linear increase in generation overhead over single hash
  - Linear increase in space/bandwid for puzzle



puzzles

Reverse a single hash given a hint

- Randomly generated range that solution falls within
- Brute-force search within range
- Fine-grain difficulty adjustment
  - Difficulty adjusted via range adjustment
  - Multiples of hash time (~1μs)
- Fast to generate (~1μs)



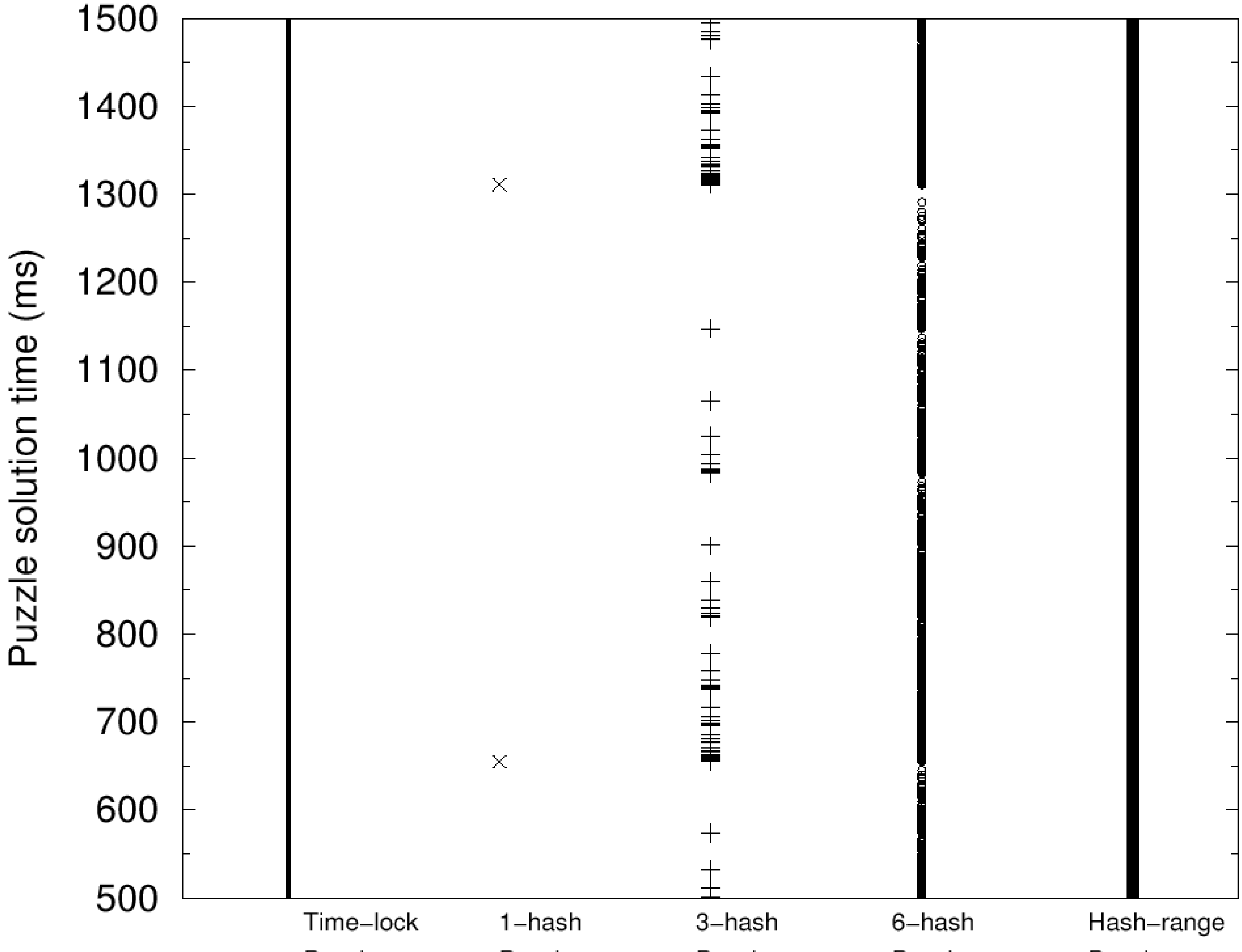
Granularity comparison

Derived analytically…

Puzzle type	Unit work (u)	Range	Mean granularity	Maximum granularity	Exact control	Parallel computation
Time-lock repeated squaring	squaring with 2^t ($0.75\mu s$)	$O(2^n)$	u	u	Yes	No
Single hash reversal	hash ($1.09\mu s$)	$u * 2^n$	$\frac{u*2^n}{n}$	$u * 2^{n-1}$	No	Yes
Multiple hash reversal ($k < n$)	hash ($1.09\mu s$)	$u * k * 2^n$	$\frac{u*k*2^n}{\sum_{i=0}^{k}(n-i)\binom{n}{i}}$	$u * 2^{n-1}$	No	Yes
Multiple hash reversal ($k > n$)	hash ($1.09\mu s$)	$u * k * 2^n$	$\frac{u*k*2^n}{(k-n+1)2^n+\sum_{i=0}^{n-1}(n-i)\binom{n}{i}}$	$u * 2^{n-1}$	No	Yes
Single hash reversal with range	hash ($1.09\mu s$)	$u * 2^n$	u	u	No	Yes

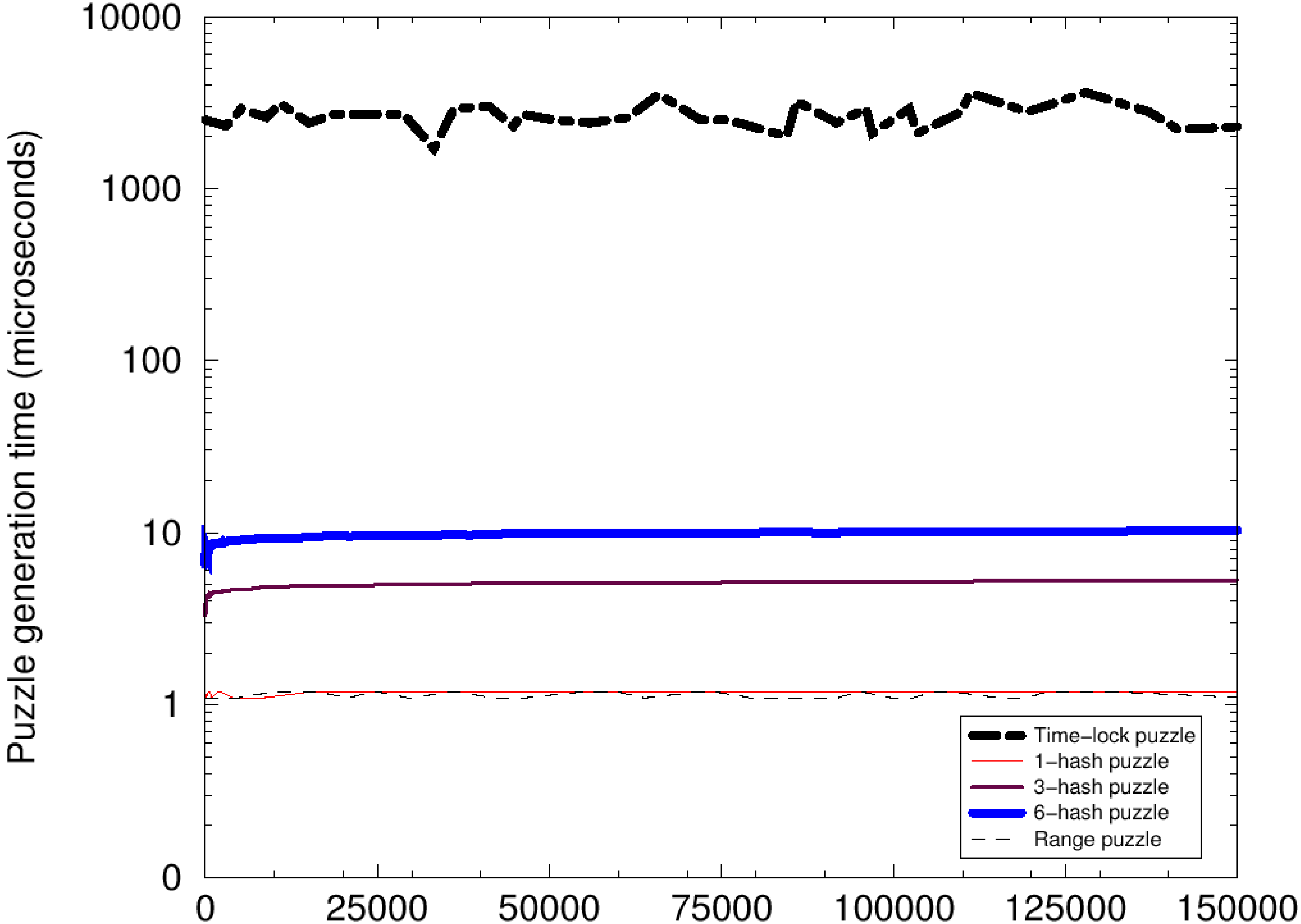
Granularity comparison

Actual difficulty levels on 1.8GHz Pentium 4



Generation comparison

Measured across 10,000 puzzles



Putting it together

First car: Puzzle-protected UDP

- Works great
- Lots of good results
- Not car we wanted

Second car: Puzzle-protected IP

- Work-in-progress…

Puzzle-protected IP protocol

Implemented within IP

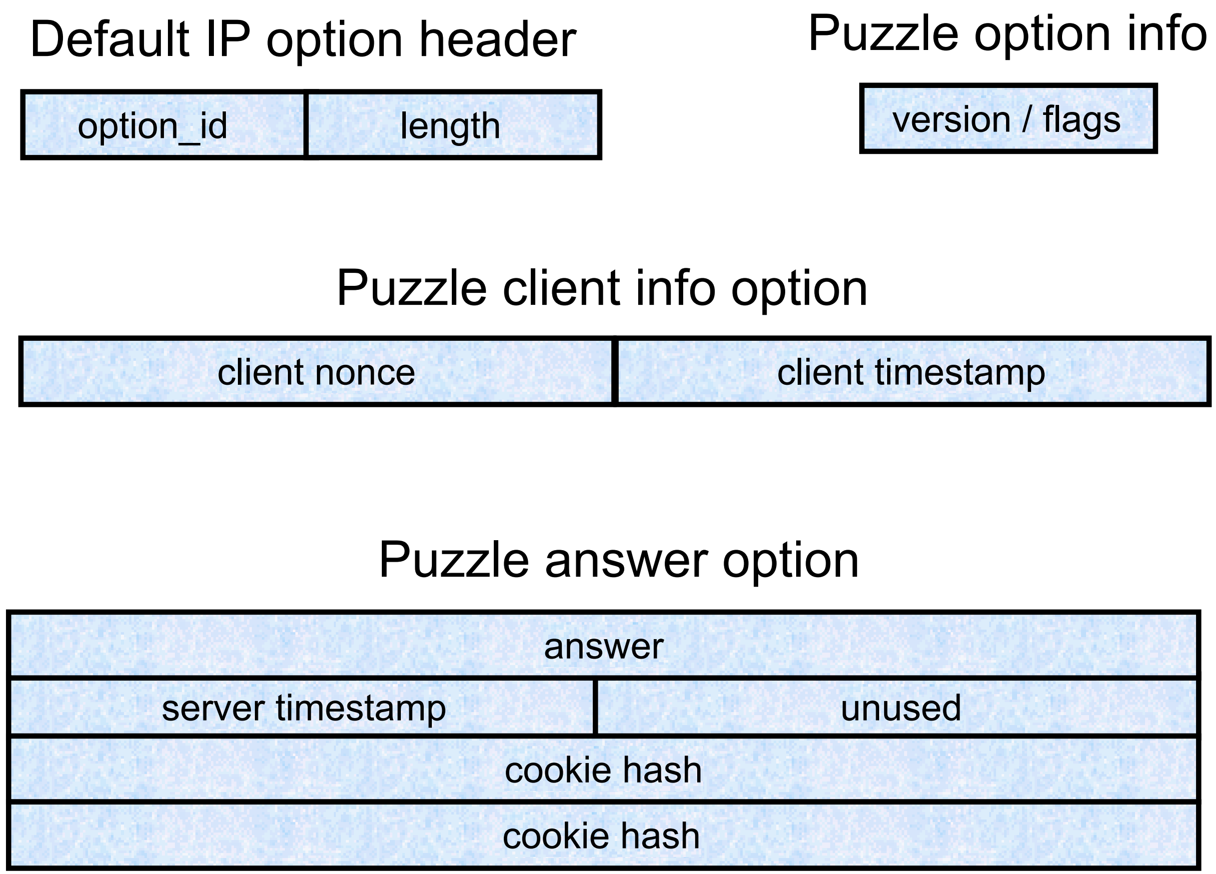
- New IP options
- New ICMP options (to support > 40 bytes)

Allows for transparent deployment

- No modifications to pseudo-header for transport checksums
- Can run between proxies and firewalls
  - No modification to end-hosts required
  - Proxies
    - Can attach nonces on behalf of clients
    - Can answer puzzles and attach answers on behalf of clients
  - Firewalls
    - Can issue and verify puzzles on behalf of servers

Puzzle client IP options

- Client info
- Puzzle answer

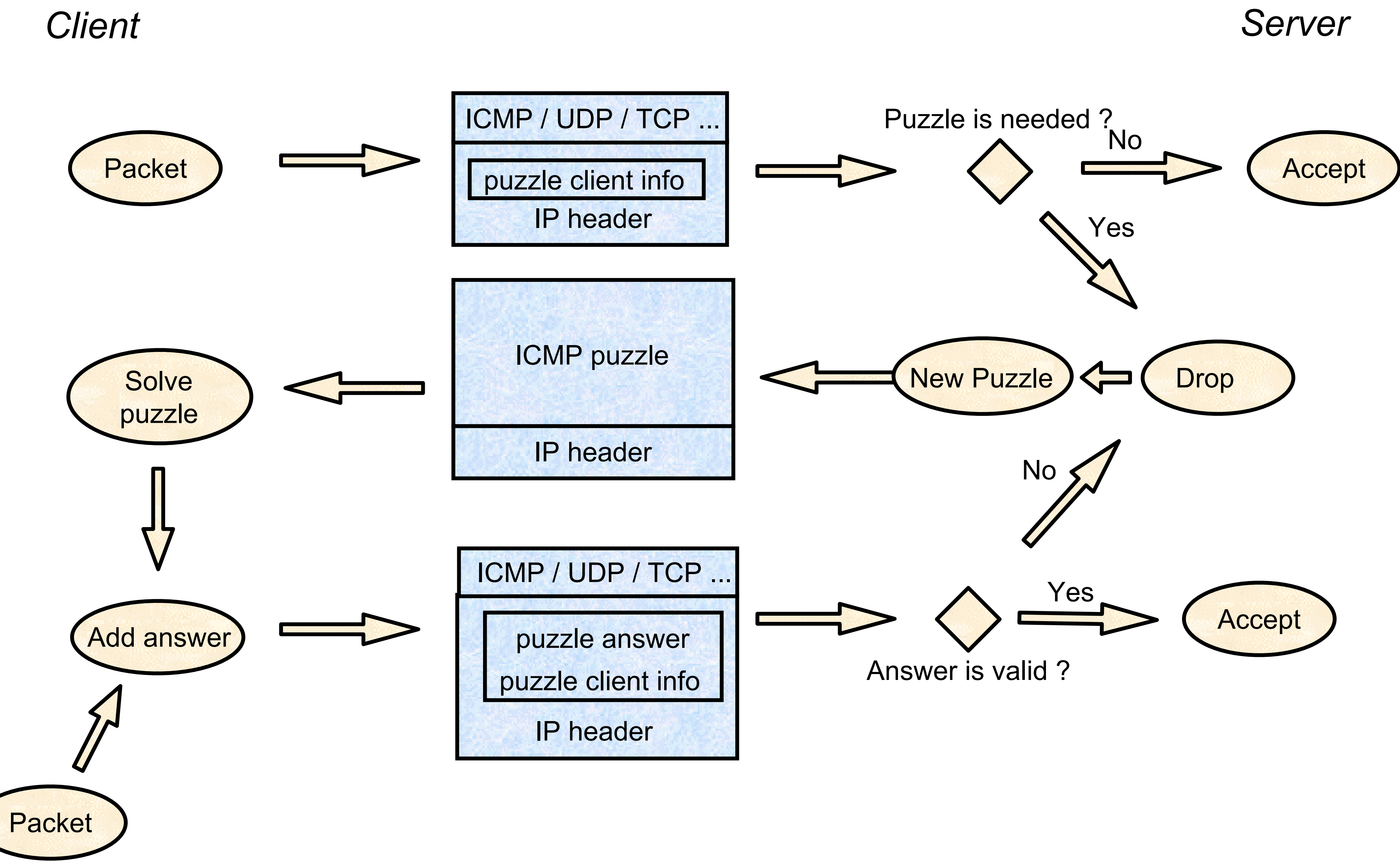


Puzzle server ICMP message

ICMP type 38

<table>
<tr><td>Type 38</td><td>Code (version)</td><td>Checksum</td></tr>
<tr><td colspan="2">Identifier</td><td>Sequence Number</td></tr>
<tr><td>No. of Puzzles</td><td>Protocol</td><td>Server Timestamp</td></tr>
<tr><td colspan="2">Client Nonce</td><td>Client Timestamp</td></tr>
<tr><td colspan="3">Puzzle maturity time</td></tr>
<tr><td colspan="3">Puzzle expiration time</td></tr>
<tr><td colspan="3">Cookie Hash</td></tr>
<tr><td colspan="3">Cookie Hash</td></tr>
<tr><td colspan="3">Min</td></tr>
<tr><td colspan="3">Max</td></tr>
<tr><td colspan="3">Difficulty</td></tr>
<tr><td colspan="3">Puzzle Hash</td></tr>
<tr><td colspan="3">Puzzle Hash</td></tr>
</table>

In action



Client
Server
Packet
ICMP / UDP / TCP ...
puzzle client info
IP header
Puzzle is needed ?
No
Accept
Yes
ICMP puzzle
IP header
Solve puzzle
New Puzzle
Drop
No
Add answer
ICMP / UDP / TCP ...
puzzle answer
puzzle client info
IP header
Yes
Accept
Answer is valid ?
Packet

Puzzle-protected IP implementation

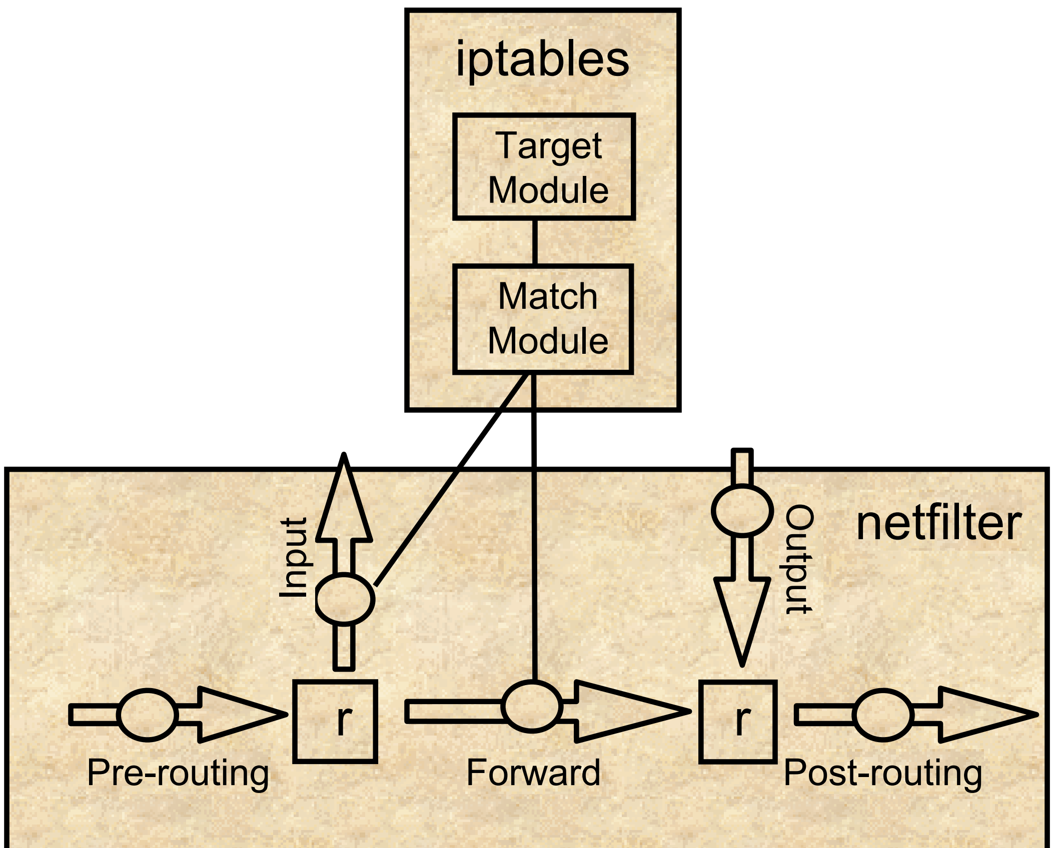
Linux via `iptables/netfilter`

- No kernel modifications
- Minimal modifications to `iptables` to add puzzle module hooks
- Compatibility with pre-existing `iptables` rulesets
- Flexibility in deployment
  - Client, server, proxy, firewall implementations via simple rule configuration
  - Programmable selection of puzzle victims

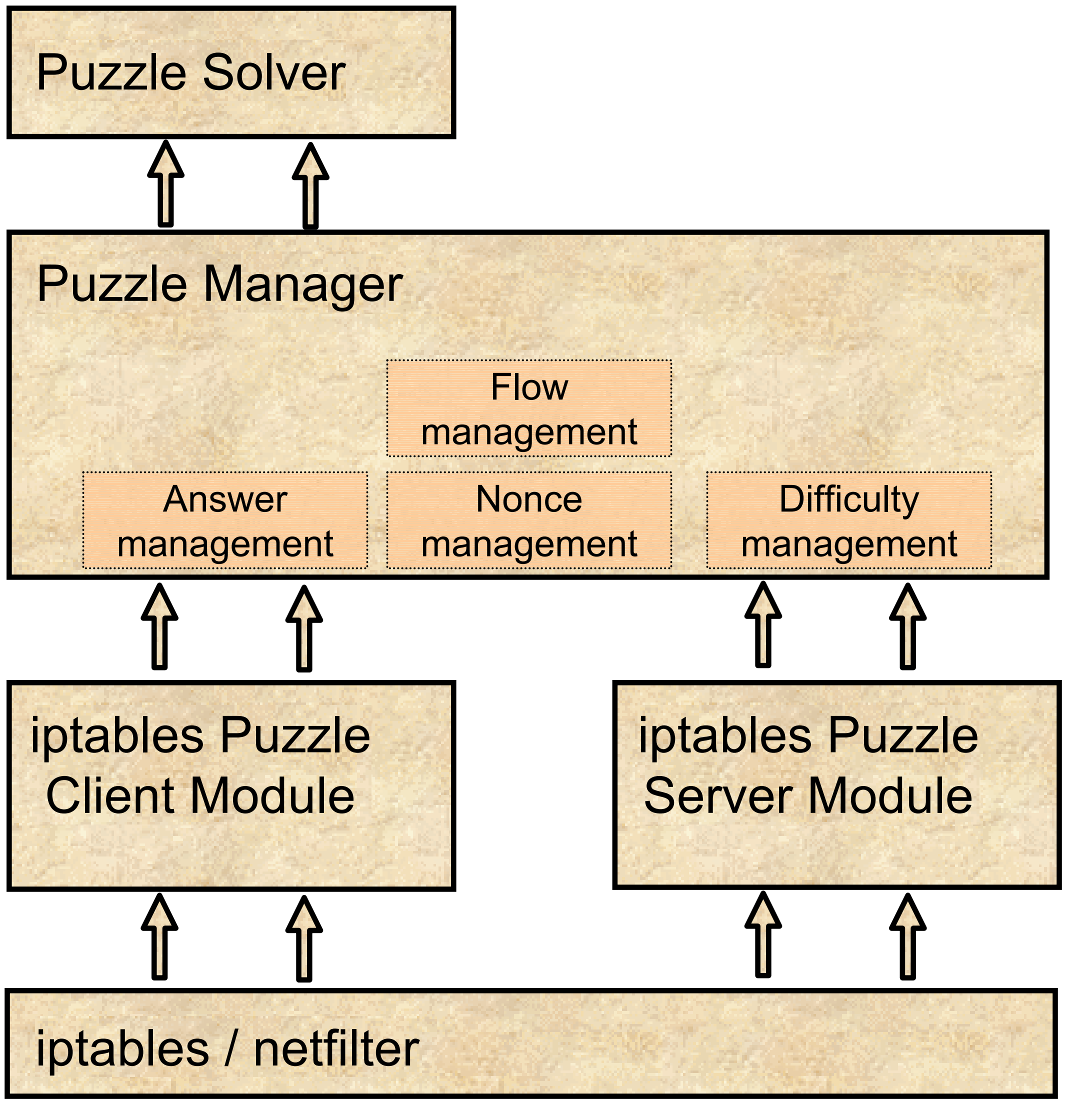
iptables/netfilter

netfilter matching at select packet processing locations

- INPUT, OUTPUT, PREROUTING, FORWARD, POSTROUTING
- Hooks for sending packets to particular iptables modules



ptables puzzle module



Example #1: Simple client and serve

Server issues puzzles on all incoming TCP SYN segments without a valid puzzle answer

- Server

```
mp5% insmod ./puzzlenet_mgr.o
mp5% insmod ./ipt_puzServer.o
mp5% iptables -t mangle -A INPUT -p tcp --syn -j puzServer
```

- Client

```
ak47% insmod ./puzzlenet_mgr.o
ak47% insmod ./ipt_puzClient.o
ak47% iptables -t mangle -A INPUT -p icmp -icmp-type 38 -j puzClient
ak47% iptables -t mangle -A POSTROUTING -j puzClient
ak47%
ak47% telnet mp5
Trying 10.0.0.7…
Connected to 10.0.0.7.
Escape character is ‘^]’.
```

- `tcpdump` trace

```
17:09:28.983779 10.0.0.6.12799 > 10.0.0.7.23: S
17:09:28.983822 10.0.0.7 > 10.0.0.6: icmp: type-#38
17:09:31.980573 10.0.0.6.12799 > 10.0.0.7.23: S
17:09:31.980637 10.0.0.7.23 > 10.0.0.6.12799: S ack
```

Example #2: Proxy and firewall

Firewall issues puzzles on all packets without valid answer

Proxy attaches nonces and answers puzzles on behalf of all clients

- Firewall

```
firewall% insmod ./puzzlenet_mgr.o
firewall% insmod ./ipt_puzServer.o
firewall% iptables -t mangle -A INPUT -j puzServer
firewall% iptables -t mangle -A FORWARD -j puzServer
```

- Proxy

```
proxy% insmod ./puzzlenet_mgr.o
proxy% insmod ./ipt_puzClient.o
proxy% iptables -t mangle -A INPUT -p icmp -icmp-type 38 -j puzClient
proxy% iptables -t mangle -A FORWARD -p icmp -icmp-type 38 -j puzClient
proxy% iptables -t mangle -A POSTROUTING -j puzClient
```

Example #2: Proxy and firewall

Client (ak47)

- Connection to closed port on server (mp5)
- Connection to non-existent machine

```
ak47% telnet mp5 2601
Trying 10.0.2.6…
telnet: Unable to connect to remote host: Connection refused
ak47% telnet 10.0.2.123
Trying 10.0.2.123…
```

- `tcpdump` trace

```
17:12:53.632512 10.0.0.6.14698 > 10.0.2.6.2601: S
17:12:53.632566 10.0.1.2 > 10.0.0.6: icmp: type-#38
17:12:56.630212 10.0.0.6.14698 > 10.0.2.6.2601: S
17:12:56.630287 10.0.2.6.2601 > 10.0.0.6.14698: R
17:13:05.456542 10.0.0.6.14699 > 10.0.2.123: S
17:13:05.455725 10.0.1.2 > 10.0.0.6: icmp: type-#38
17:13:08.454862 10.0.0.6.14699 > 10.0.2.123: S
17:13:14.453935 10.0.0.6.14699 > 10.0.2.123: S
```

Status

Fully functional `iptables/netfilter` implementation

- Tamper-resistance
  - Tamper-proof operation (must be along path to deny service)
- Performance
  - 100,000 puzzles/sec on commodity hardware
    - 1Gbs+ for per-packet puzzles with MTU packets
    - Puzzle generation ~1μs
    - Puzzle verification ~1μs, constant amount of state
  - Small packet overhead
    - Puzzle question ~40 bytes
    - Puzzle answer ~20 bytes
  - Low latency
    - Can play puzzle-protected Counter-strike transparently
- Control
  - Fine-grained puzzle difficulty adjustment
  - Simple controller
- Fairness

Questions?

PuzzleNet and Reputation-based Networking

http://www.cse.ogi.edu/sysl/projects/puzzles

Wu-chang Feng, "The Case for TCP/IP Puzzles", *in Proceedings of ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA-03)*

Wu-chang Feng, Antoine Luu, Wu-chi Feng, "Scalable Fine-Grained Control of Network Puzzles", *in submission*

Other projects at OGI@OHSU

Packet classification

- Approximate caches
- Exact cache architectures
- Mapping algorithms onto the IXP
- TCPivo: A high-performance packet replay engine

Multimedia systems

- Panoptes: A flexible platform for video sensors

Questions?

Approximate Caches for Packet Classification

Francis Chang
Wu-chang Feng
Kang Li

OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY

in Proceedings of ACM SIGCOMM (Poster session) August 2003.

Motivation

Increasing complexity in packet classification function

- Number of flows
- Number of rules
- Number of fields to classify
  - Firewalls, NATs, Diffserv/QoS, etc.
- Header size
  - IPv6
- Require large, fast memory to support line speeds

Problem

- Storing large headers in fast memory prohibitively expensive
  - Large memory slow
  - Fast memory expensive
- Classic space-time trade-off

Probabilistic Networking

Throw a wrench into space-time trade-off

Reduce memory requirements by relaxing the *accuracy* of packet classification function

Specific application to packet classification caches

What quantifiable benefits does sacrificing accuracy have on the size and performance of packet classification caches?

Summary slide

But the network is *always* right

Not really….

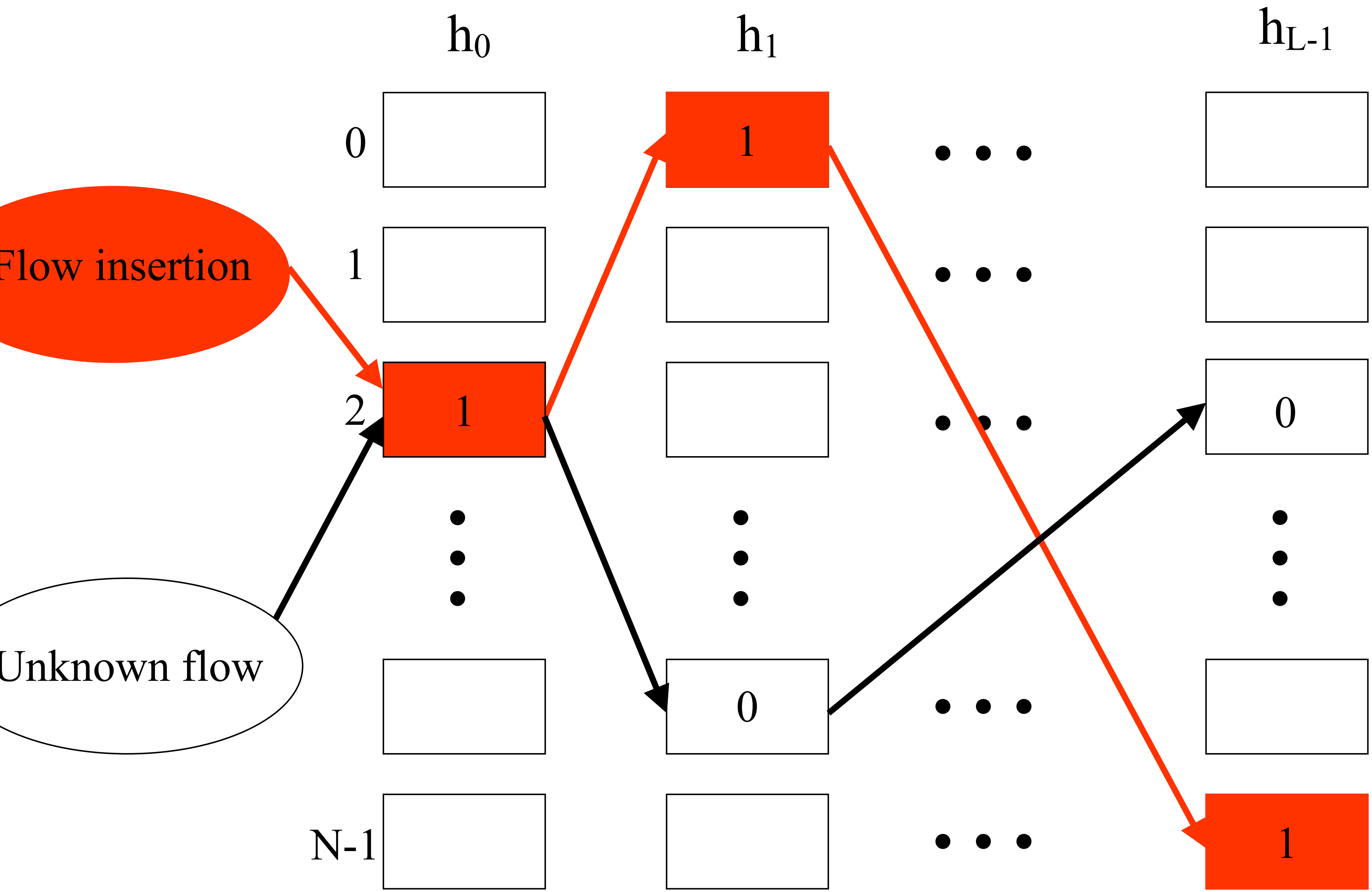
- Bad packets
  - Stone/Partridge SIGCOMM 2000
  - Lots of packets are bad, some are undetectably bad
    - 1 in 1100 to 32000 TCP packets fail checksum
    - 1 in 16 million to 10 billion TCP packets are UNDECTABLY bad
    - UDP packets are not required to have cksum
    - Even if the cksum is bad, OS will give the packet to the application (Linux)
- Routing problems
  - Transient loops
  - Outages

Our approach

Bloom filter

- An approximate data structure to store flows matching a bina predicate
  - L x N array of memory
  - L independent hash functions
  - Each function addresses N buckets
- Use for packet classification caches
  - Store known flows into filter
  - Lookup packets in filter for fast forwarding

Bloom filter



N^L virtual bins out of L*N actual bins

Bloom filter

Things to note

- Collisions cause inaccurate classifications
- Storage capacity invariant to header size and number of fields
  - Size of filter determined only by
    - Number of flows
    - Desired accuracy
  - Exact caches grow with increasing header size and fields
    - IPv4-based connection identifier = 13 bytes
    - IPv6-based connection identifier = 37 bytes

Characterizing Bloom filters

Misclassification rates a function of…

- N = number of bins per level
- L = number of levels
- k = number of flows stored

$$p_{misclassification} = \left(1 - \left(1 - \frac{1}{N} \right)^{k} \right)^{L}$$

Characterizing Bloom filters

How many flows can a Bloom filter support?

- After an approximation and some more derivation….

$$\kappa = -\frac{M}{L}\ln(1 - p^{1/L})$$

- For fixed misclassification rate (p), number of elements is linear to size of memory

What setting of L minimizes p?

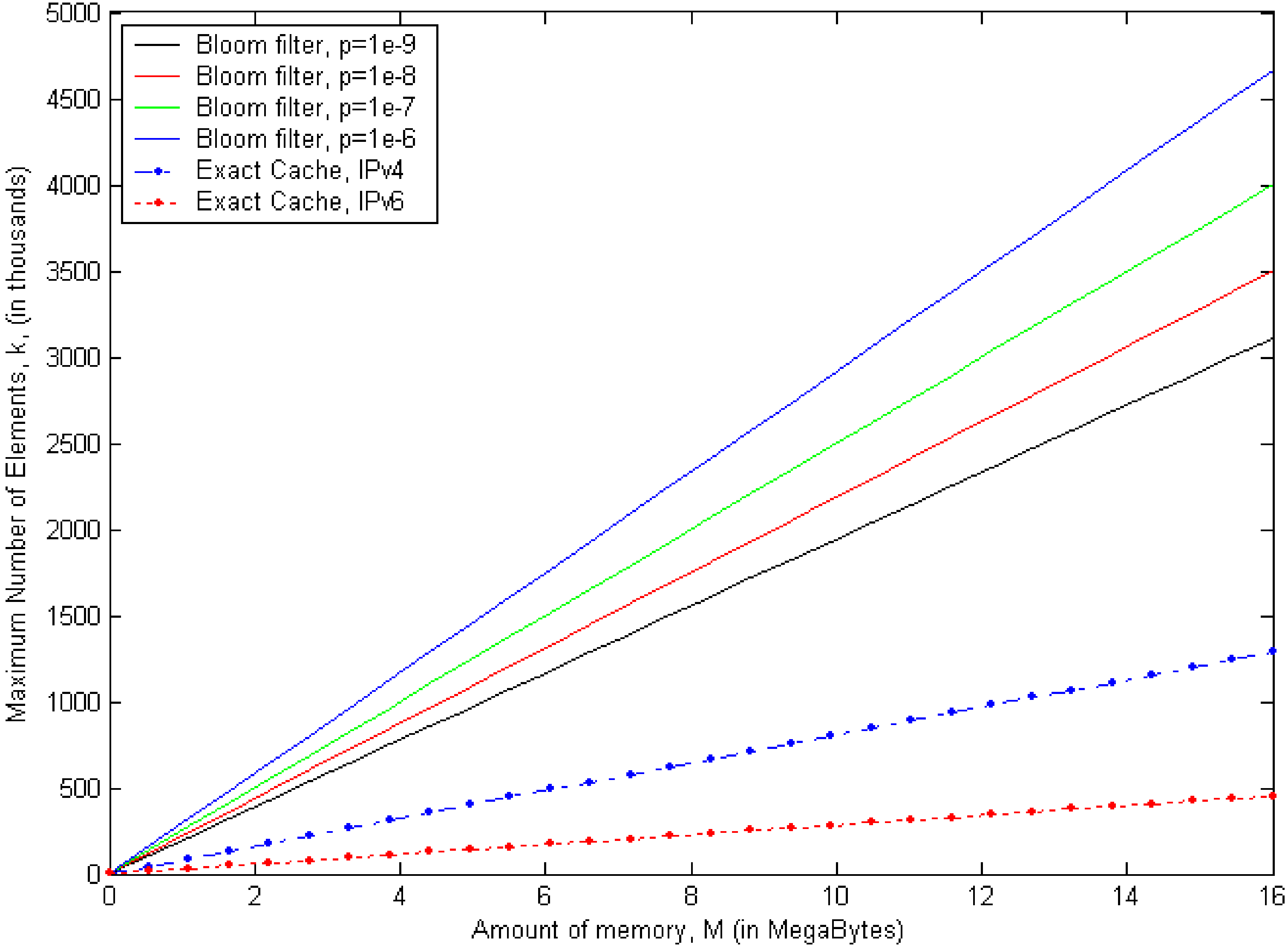
- After some more derivation

$$L = -\log_2 p$$

- L depends only on p
  - Smaller P = Larger L

Comparison to exact approaches

For fixed misclassification rates and optimal L

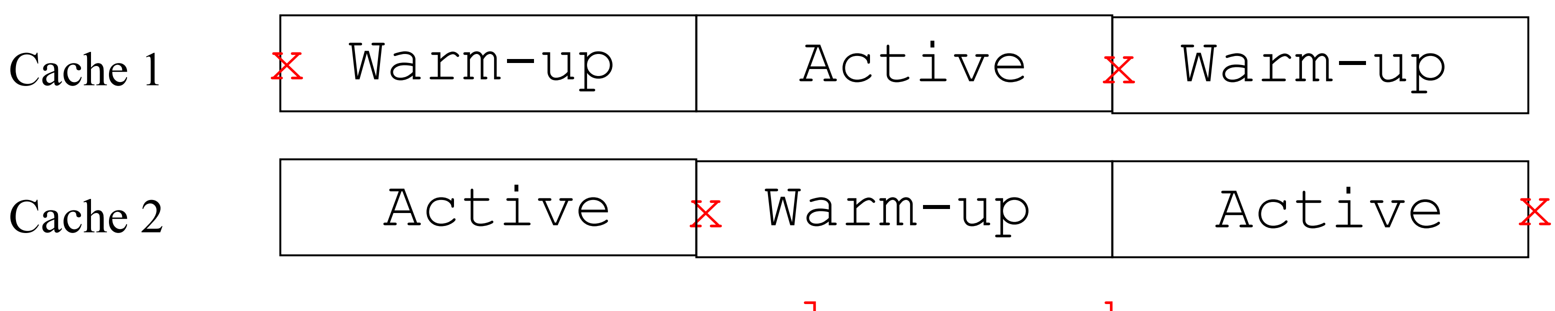


Some modifications

Supporting multiple predicates (see paper)

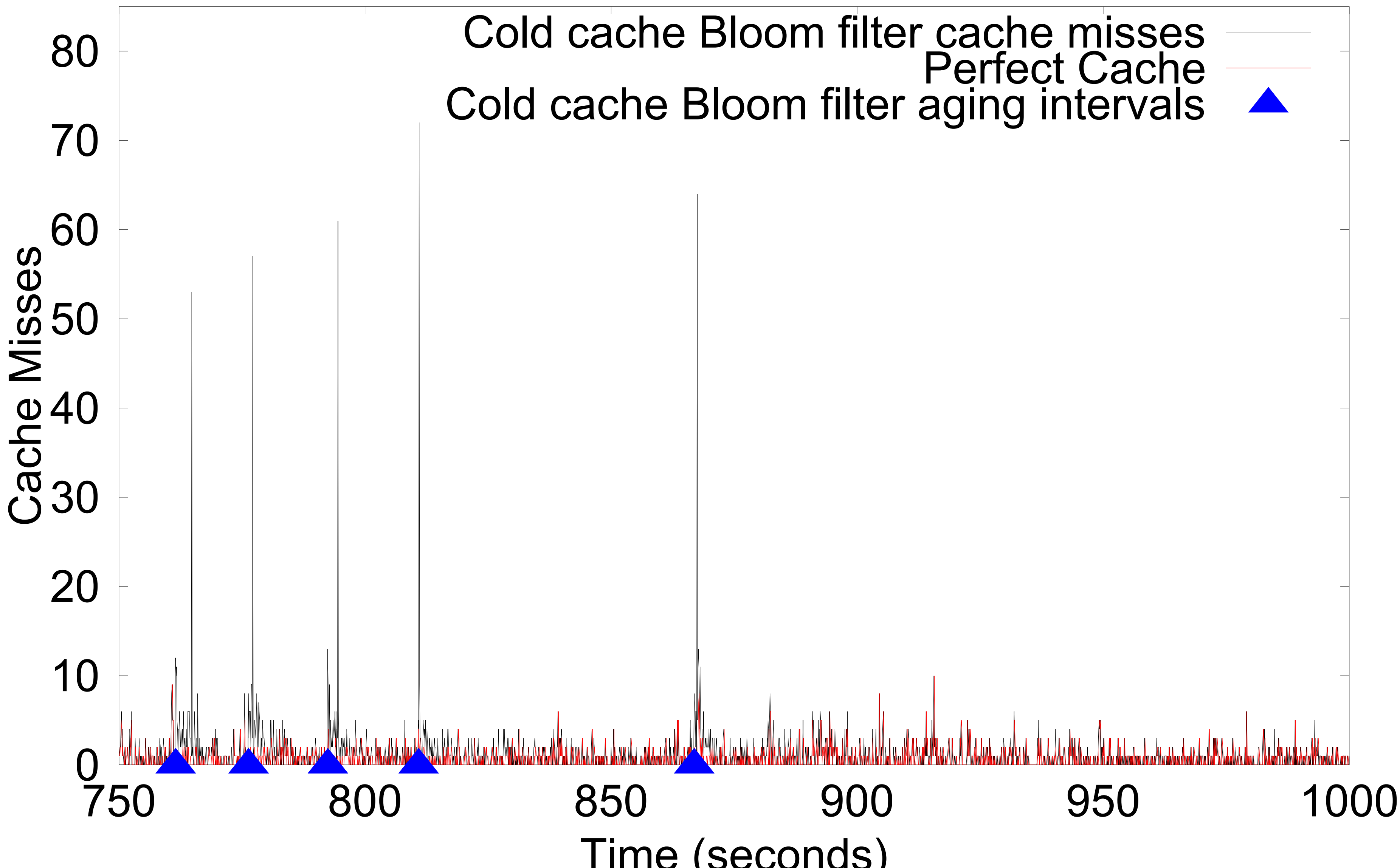
Aging the filter to bound misclassification

- Cold caching
  - Count the number of flows inserted
  - Reset entire cache when misclassification limit reached
  - Problem: large miss rates upon cache clearing
- Double-buffered caching
  - Split into 2 caches: active and warm-up
  - Insert into both caches, check only in active cache
  - Stagger insertion and periodic clearing of cache (every *k* insertions)

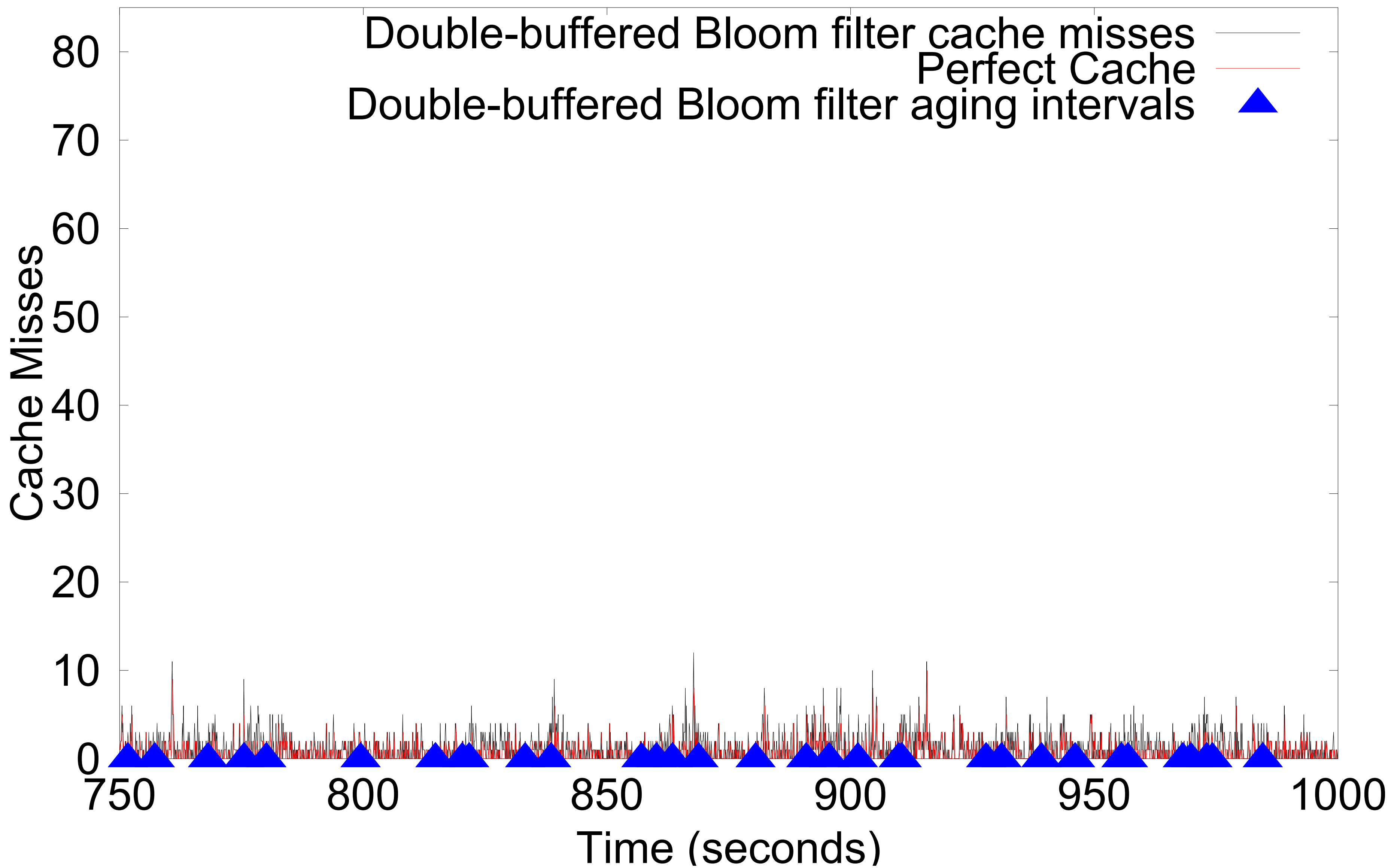


Cold caching

OGI OC-3c trace

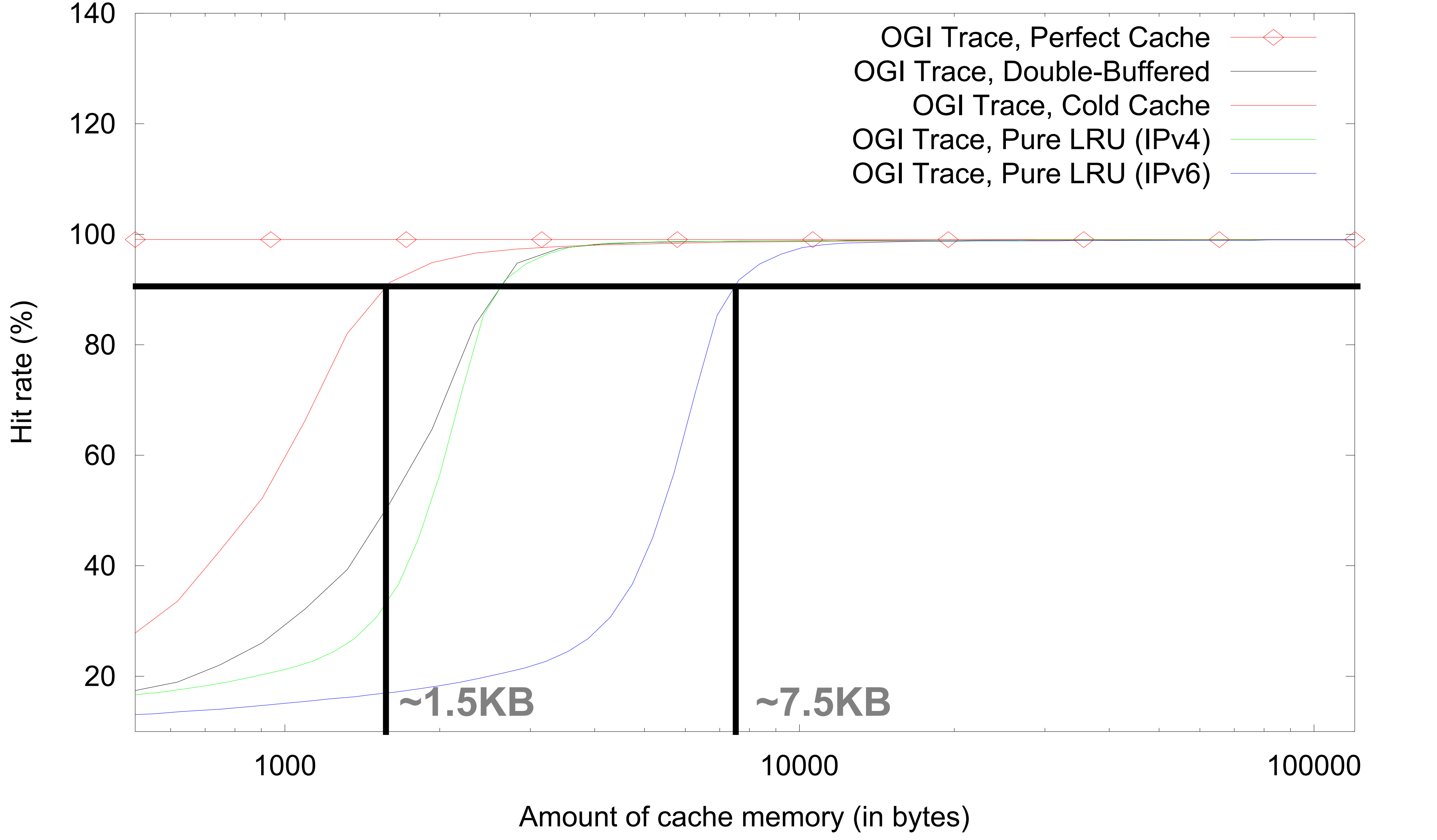


Double-buffered caching



OGI cache hit-rates

Note: all exact caches assumed fully-associative



Dealing with misclassification

Firewall

- Fully classify all TCP SYN

Routers

- Longer routes possible
- TTL prevents loops
- Periodically change hash functions to avoid persistent misclassifying

End-systems

- Manual retry with new flowID

Implementation

IXP1200

- Not the optimal hardware showcase
- Could use support for Bloom filters
  - Parallel hashing
  - Parallel memory access
  - Bit-addressable memory access
- Details in paper

Questions?

Approximate packet classification

- http://www.cse.ogi.edu/sysl/projects/ixp

Francis Chang, Kang Li, Wu-chang Feng, "Approximate Caches for Packet Classification", *in ACM SIGCOMM 2003 Poster Session*, Aug. 2003. **Poster**

Francis Chang, Kang Li, Wu-chang Feng, "Approximate Caches for Packet Classification", *in submission*. **Paper**

Back

Architectures for Packet Classificati
Caches

Kang Li
Damien Berger
Francis Chang
Wu-chang Feng

OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY

in Proceedings of IEEE International Conference on Networks (ICON 2003) Sept. 2003.

otivation

Caching essential for good performance

Impacted by traffic and address mix

Recent work on analyzing..

- Internet address allocation
- Traffic characteristics of emerging applications such as game and multimedia

Our study

- How does recent work impact design of caches?
  - Hash function employed in cache (IXP hash unit vs. XOR)
  - Replacement policies (LFU vs. LRU)

Summary slide

Caching

Used currently in IP destination-based routing

- One-dimensional classifier
- Avoid route lookups by caching previous decisions
- Instrumental in building gigabit IP routers
- Good caches make ATM, MPLS less important

Previous caching work

Cache of 12,000 entries gives 95% hit rate [Jain86, Feldmeier88, Heimlich90, Jain90, Newman97, Partridge98]

“A 50 Gb/s IP Router” [Partridge98]

- Alpha 21164-based forwarding cards (separate from line carc
  - First level on-chip cache stores instructions
    - Icache=8KB (2048 instructions), Dcache=8KB
  - Secondary on-chip cahe=96KB
    - Fits 12000 entry route cache in memory
    - 64 bytes per entry due to cache line size
  - Tertiary cache=16MB
    - Full double-buffered route table

Packet classification caching

Multi-field identification of network traffic

- Typically done on the 5-tuple
- <SourceIP, DestinationIP, SourcePort, DestinationPort, Protocol>
- Inherently harder than Destination IP route lookup
- Extremely resource intensive

Many network services require packet classification

- Differentiated services (QoS), VPNs, NATs, firewalls

Packet classification caching

Overhead of full, multi-dimensional packet classificatio
makes caching even more important

- Full classification algorithms much harder to do versus route lookups
- Per-flow versus per-destination caching results in much lowe hit rates
- Rule and traffic dependent

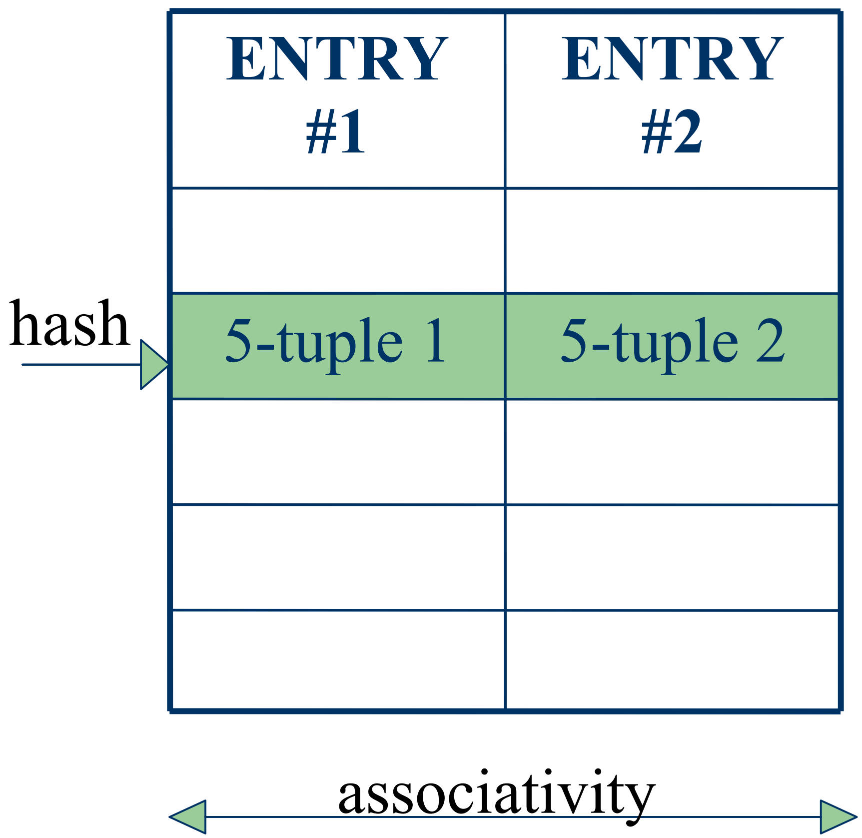
Goal of study

Attack the packet classification caching problem in the context of emerging traffic patterns

Resource requirements and data structures for high performance packet classification caches

- What cache size should be used?
- How much associativity should the cache have?
- What replacement policy should the cache employ?
- What hash function should the cache use

eneral cache architecture



Current approaches

Direct-mapped hashing with LRU replacement

- Typical for IP route caches [Partridge98]

Parallel hashing and searching with set-associative hardware [Xu00]

- ASIC solution with parallel processing and a fixed, LRU replacement scheme

Approach

Collect real traces

- http://pma.nlanr.net
- OGI/OHSU OC-3 trace

Simulation

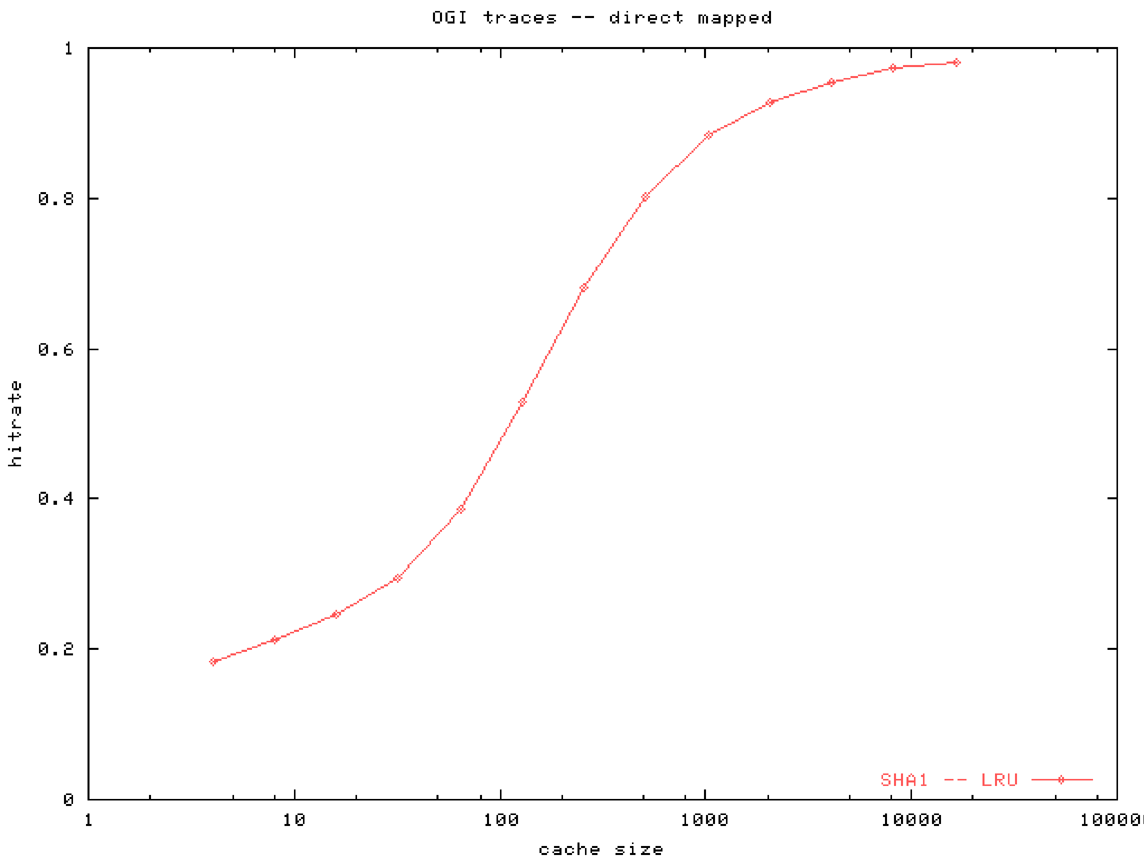
- PCCS

Real Hardware tests

- IXP1200

How large should the cache be?

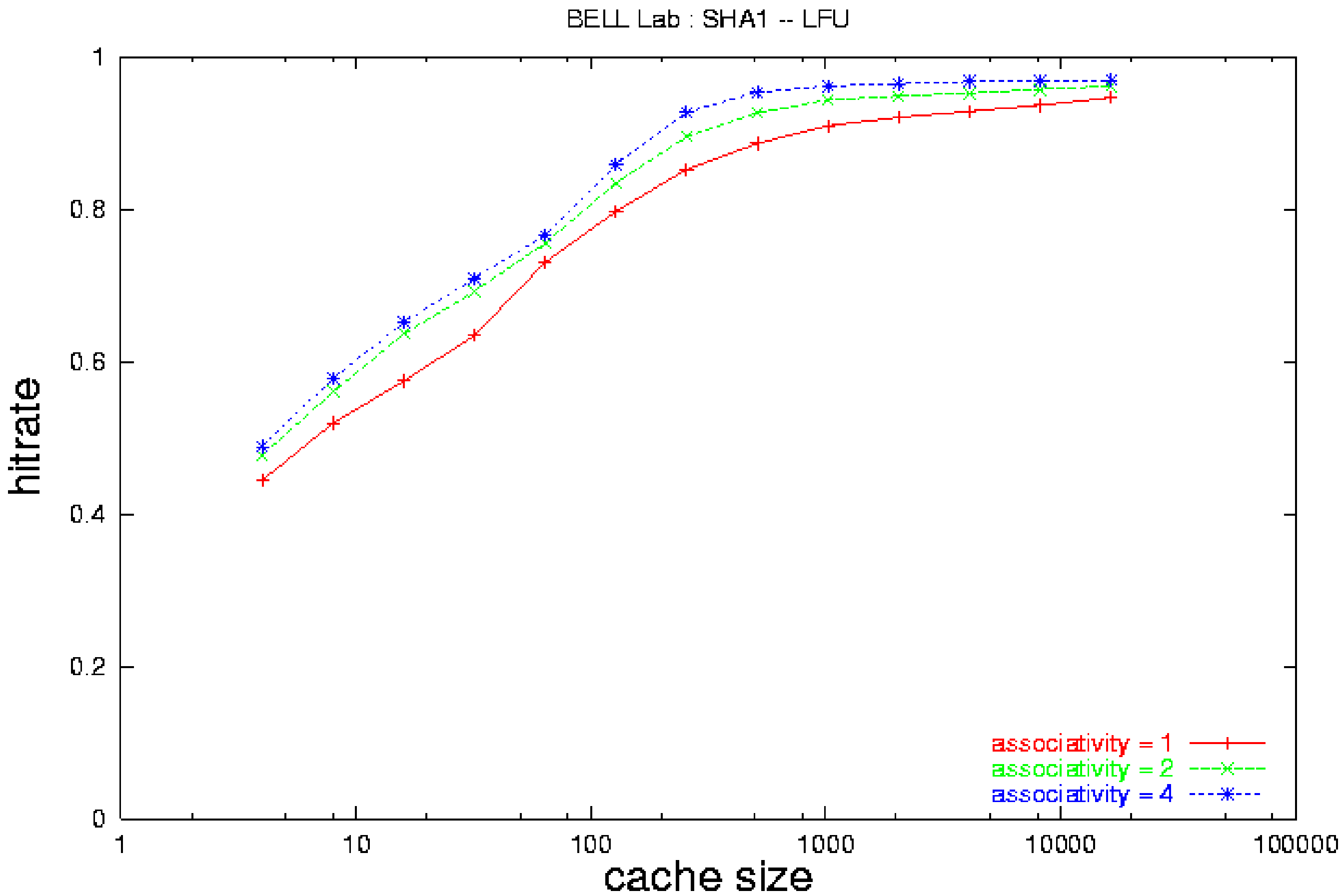
Depends on number of simultaneously active flows present (assuming each new flow has a new 5-tuple)



needed?

Associativity increases hit rates

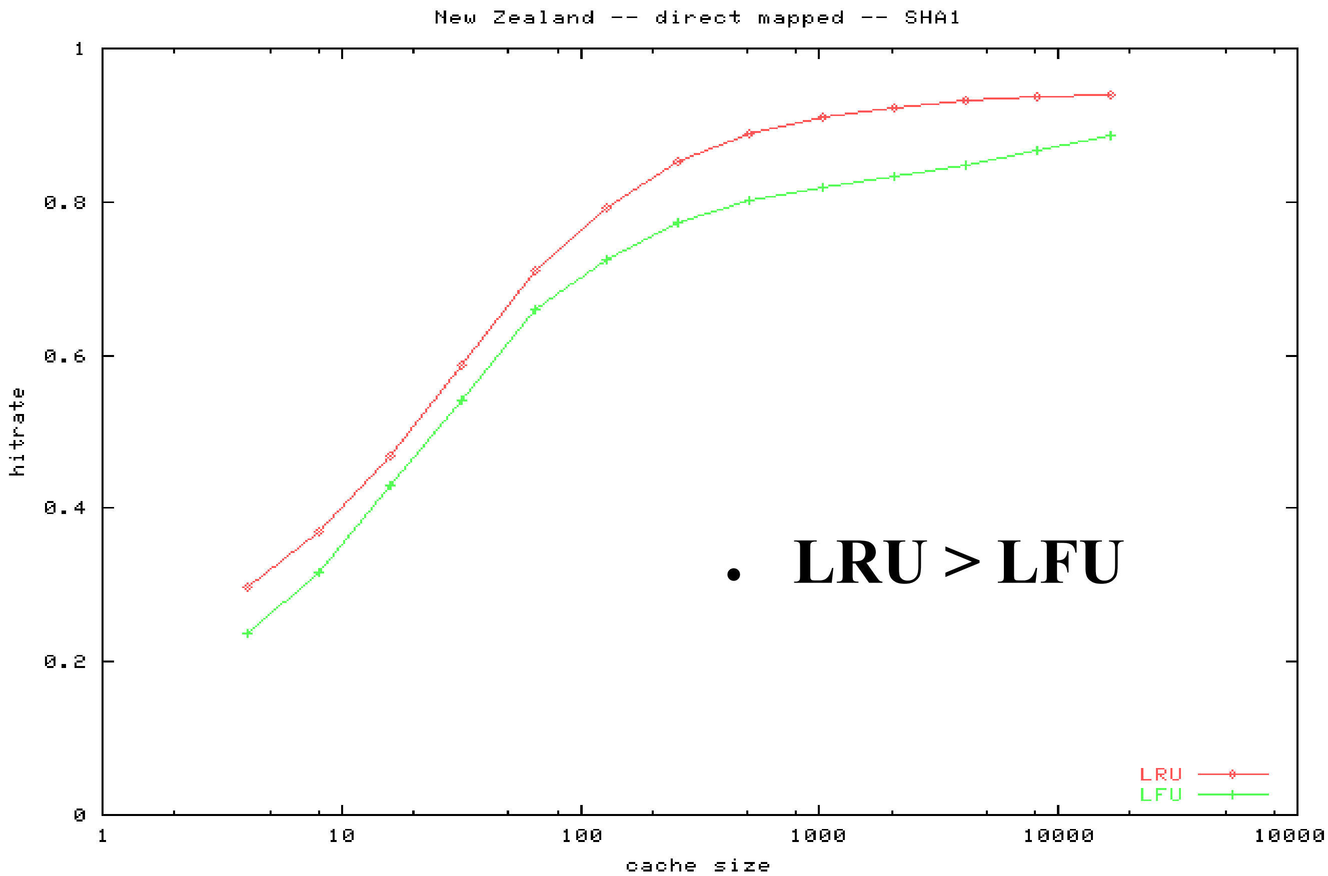
Benefits diminish with increasing associativity and large
cache sizes



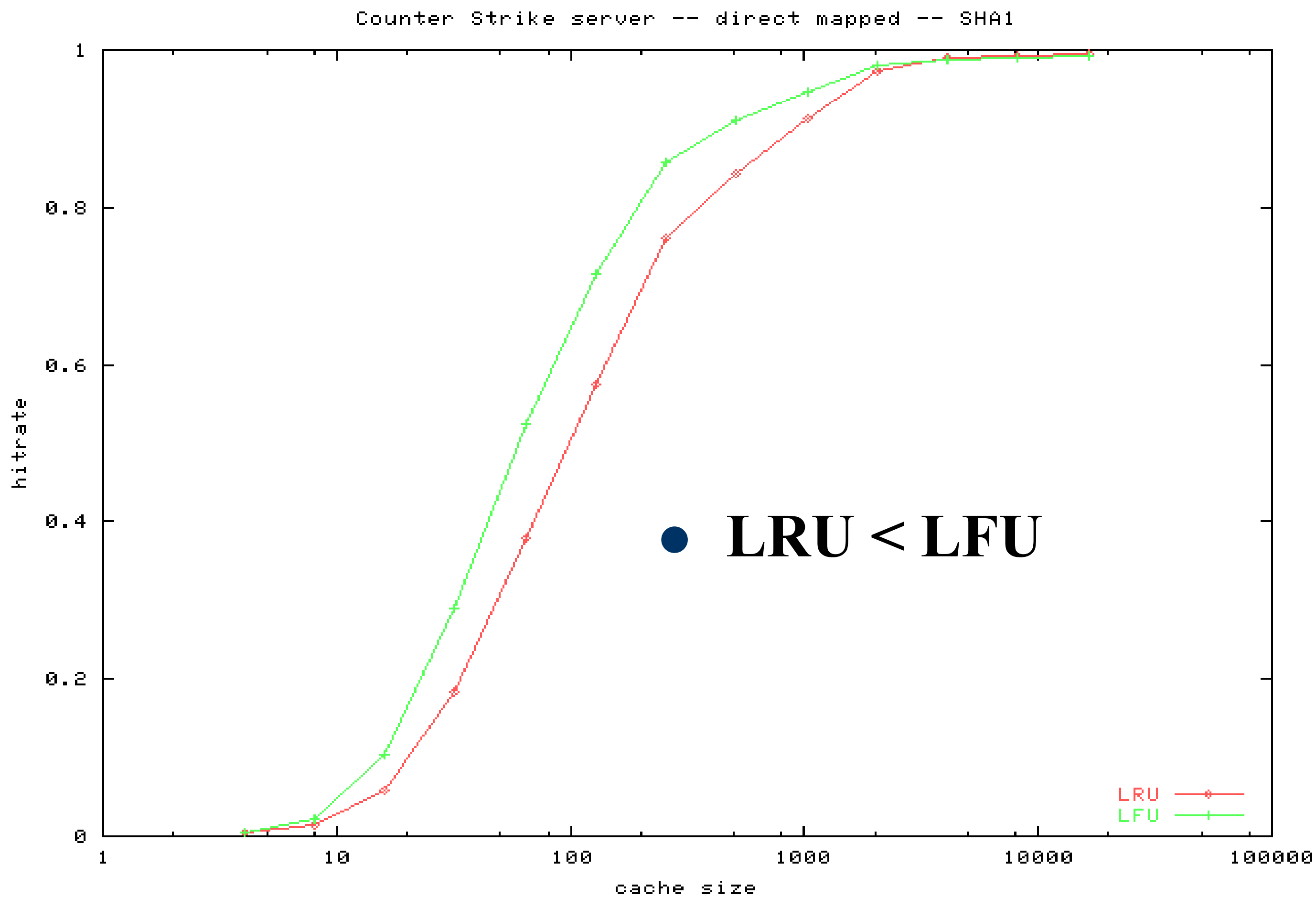
What replacement policy is needed?

LRU: Least-recently used

LFU: Least-frequently used



hat replacement policy is needed?



- **LRU < LFU**

Observations

Game traffic

- Large number of periodic packets
- Extremely small packet sizes
- Persistent flows
- Without caching, a packet classification disaster

Web traffic

- Bursty, heavy-tailed packet arrival
- Transient flows

Consider a mixture of game and web traffic

- LFU prevents pathologic thrashing

What hash function is needed?

IP address and address mixes highly structured

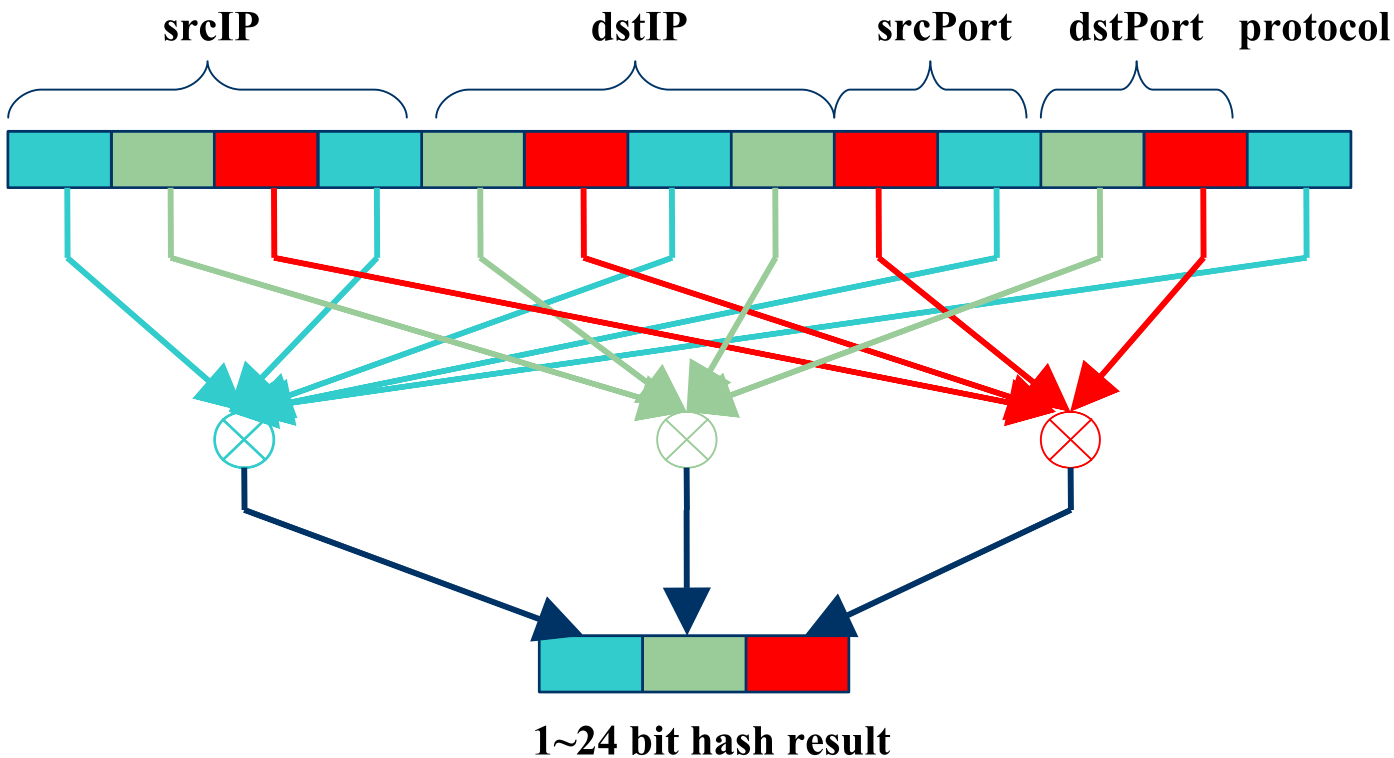
- Strong hash functions prevent collisions
- Weak hashing leads to increased thrashing and misses

Observation: Internet address usage highly structured [Kohler02]

- Structural features around /8, /16, /24
  - Sparseness
  - Sequential allocation from *.*.*.0
- Allows for intelligent construction of weak hash function tha achieves high performance

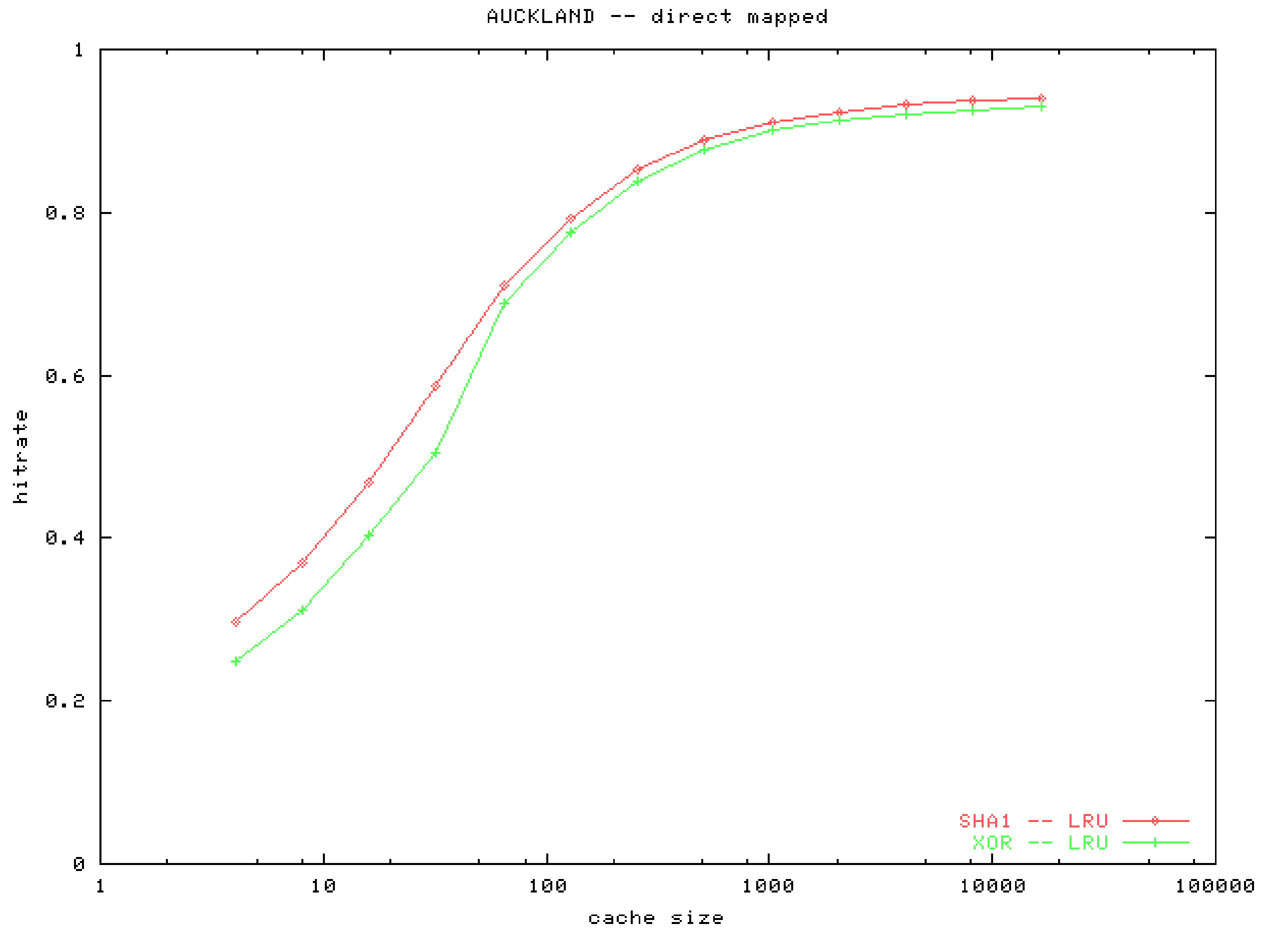
What hash function is needed?

A simple, but effective, “dummy” hash function



What hash function is needed?

Hardware hash units not needed for caching



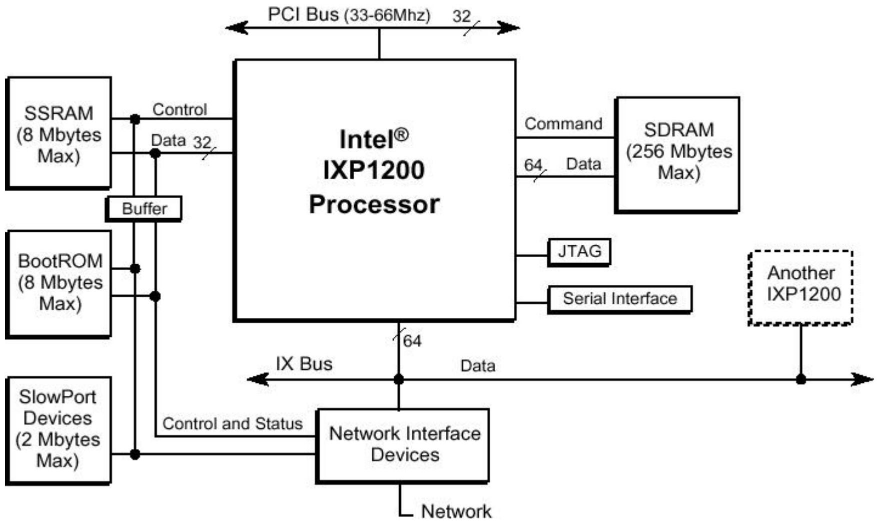
Experimental validation

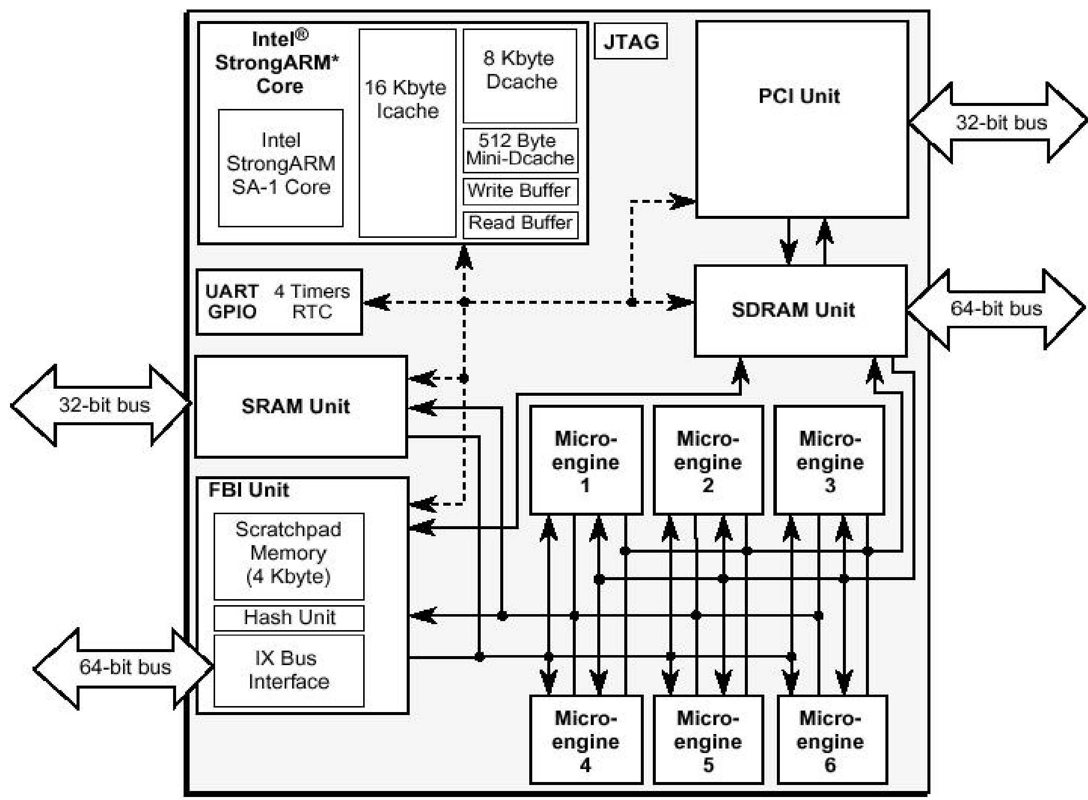
Intel IXP1200

- Programmable network processor platform
- Can be used to explore sizing, associativity, and hashing issu
- Provides a single 64-bit hardware hash unit
  - Fixed multiplicand polynomial
  - Programmable divisor polynomial

Question: Should the IXP's hash unit be used to implement a packet classification cache?

XP1200





Intel IXP1200 Network Processor

IXP performance tests

Hash unit performance test implemented in microC

- Latency ~ 25-30 cycles
- Throughput ~ 1 result every 9 cycles

Dummy hash function

- Latency ~ 5 cycles
- Throughput ~ 1 result every 5 cycles per micro-engine

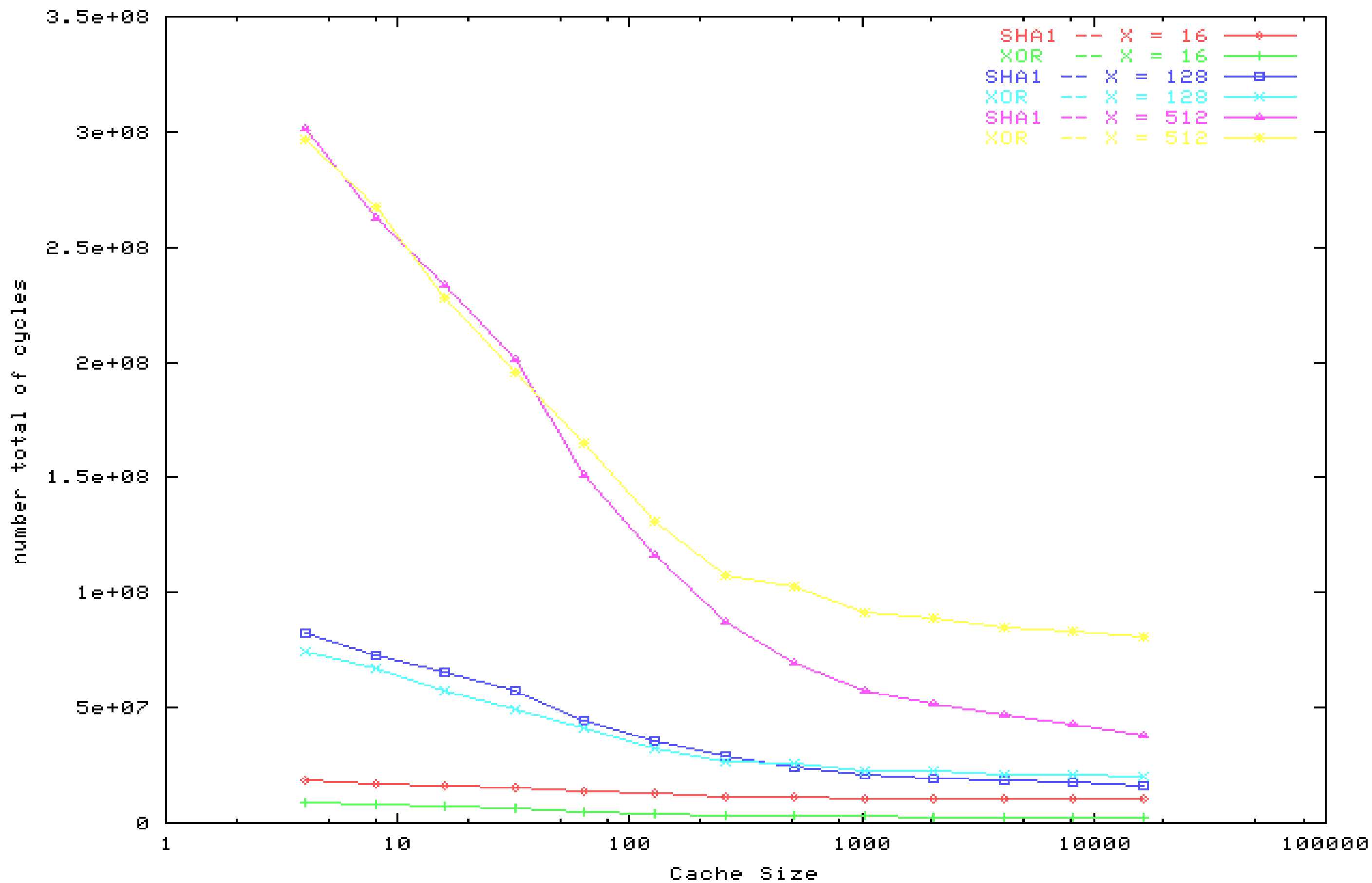
Assume a cache miss incurs a penalty of tX cycles (full classification time)

Find the total number of cycles for each hash function o the same workload

Results

h=hit rate t_h=hash latency t_X=classification latency

Total cycles = $h * t_h + (1-h)*t_X$



Summary

Network hardware designs such as caches must adapt to changing traffic structure

- Cache sizes, associativity, replacement policies, hash functio
- Address allocation policies allow μ-engine based XOR-hashe to outperform stronger hashes (i.e. centralized IXP hash unit)
- LFU provides only marginal improvement over LRU with multimedia traffic

uestions?

- Packet classification
  - http://www.cse.ogi.edu/sysl/projects/ixp

ng Li, Francis Chang, Damien Berger, Wu-chang Feng, "Architectures or Packet Classification Caching", *in Proceedings of International onference on Networks*, Sept. 2003.

Back

TCPivo
A High-Performance Packet Replay Engine

Wu-chang Feng
Ashvin Goel
Abdelmajid Bezzaz
Wu-chi Feng
Jonathan Walpole

OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY

in Proceedings of ACM SIGCOMM Workshop on Models, Methods, and Tool
for Reproducible Network Research (MoMeTools) August 2003.

otivation

- Many methods for evaluating network devices
  - Simulation
    - Device simulated, traffic simulated
    - `ns-2`, IXP network processor simulator
  - Model-based emulation
    - Actual device, traffic synthetically generated from models
    - IXIA traffic generator
  - Trace-driven emulation
    - Actual device, actual traffic trace
    - Particularly good for evaluating functions that rely on actual addres
      mixes and packet interarrival/size distributions

oal of work

Packet replay tool for trace-driven evaluation

- Accurate
- High-performance
- Low-cost
  - Commodity hardware
  - Open-source software

Summary slide

Solution

Solution: `TCPivo`

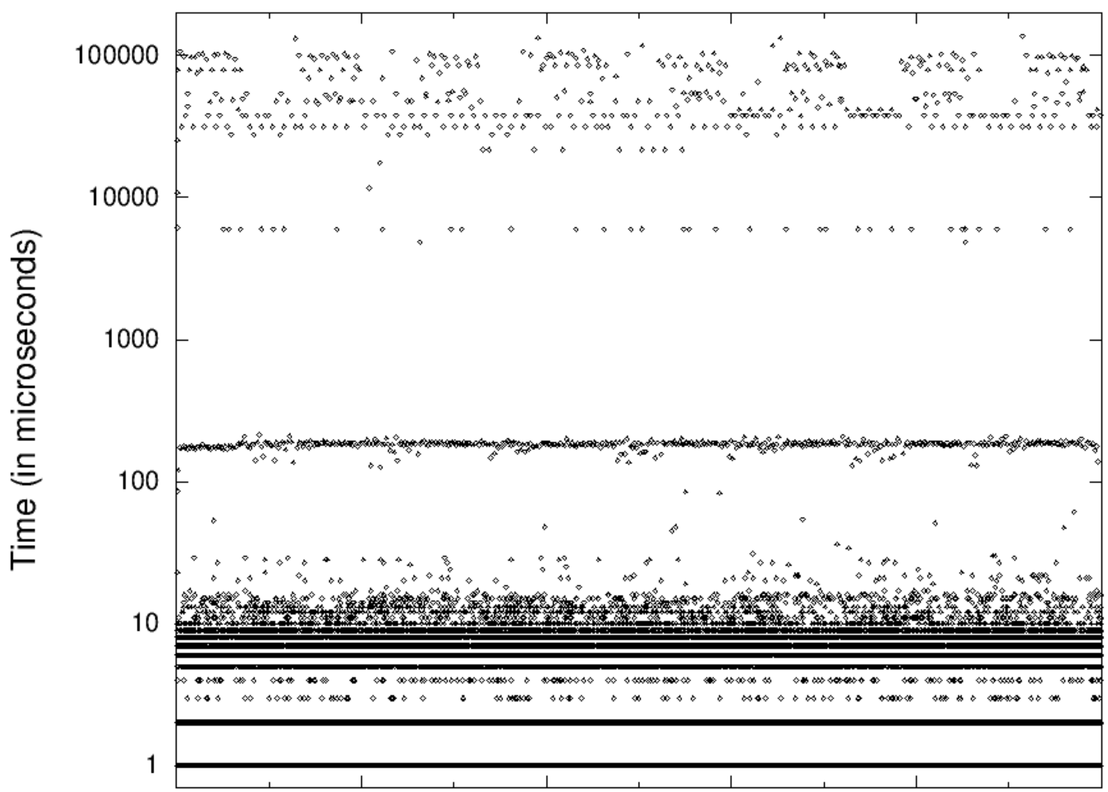
- Accurate replay above OC-3 rates
  - Pentium 4 Xeon 1.8GHz
  - Custom Linux 2.4.20 kernel with ext3
  - Intel 82544 1000Mbs
  - ~$2,000

hallenges

- Trace management
  - Getting packets from disk
- Timer mangement
  - Time-triggering packet transmission
- Scheduling and pre-emption
  - Getting control of the OS
- Efficient sending loop
  - Sending the packet

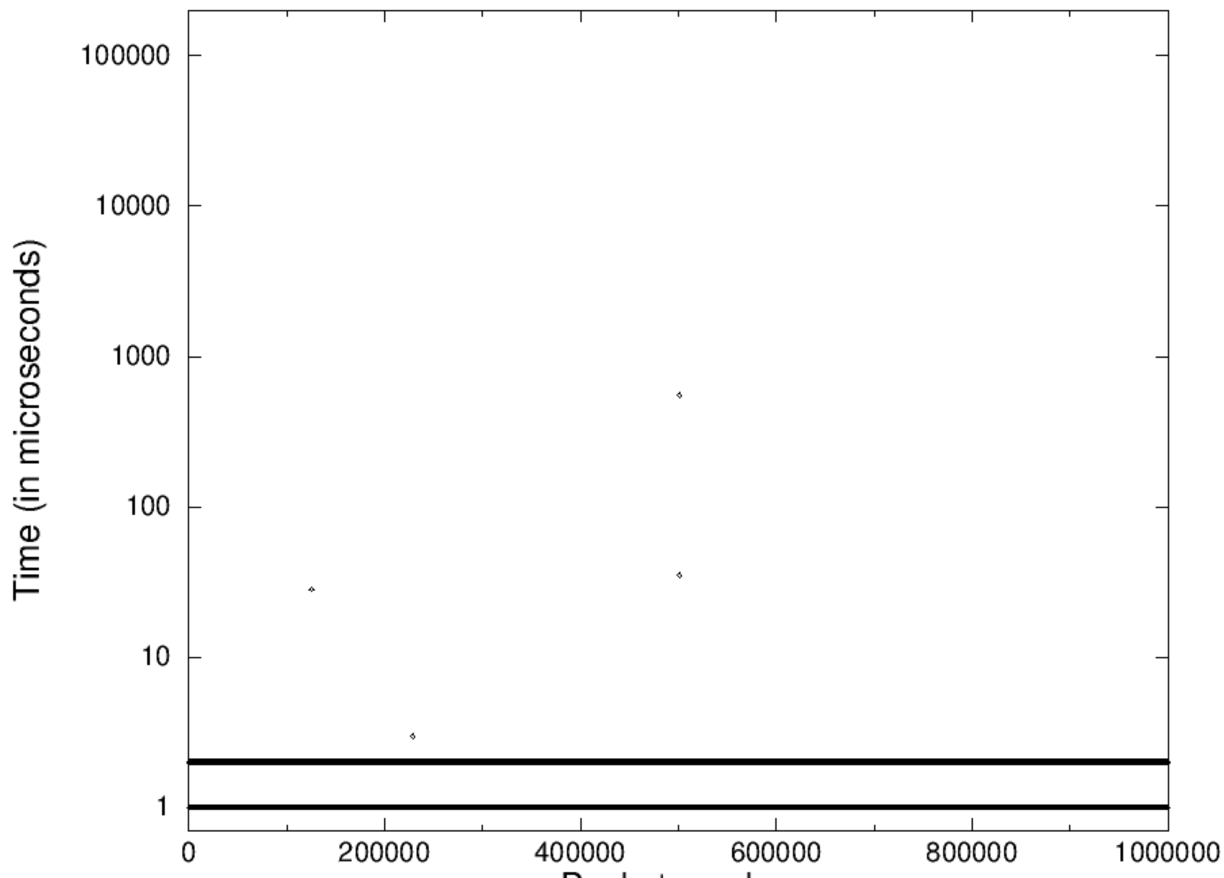
race management problem

- Getting packets from disk
  - Requires intelligent pre-fetching
  - Most OSes support transparent pre-fetch via `fread()`
- Default Linux `fread()` latency reading trace



race management in TCPivo

- Double-buffered pre-fetching
- `mmap()/madvise()` with sequential access hint

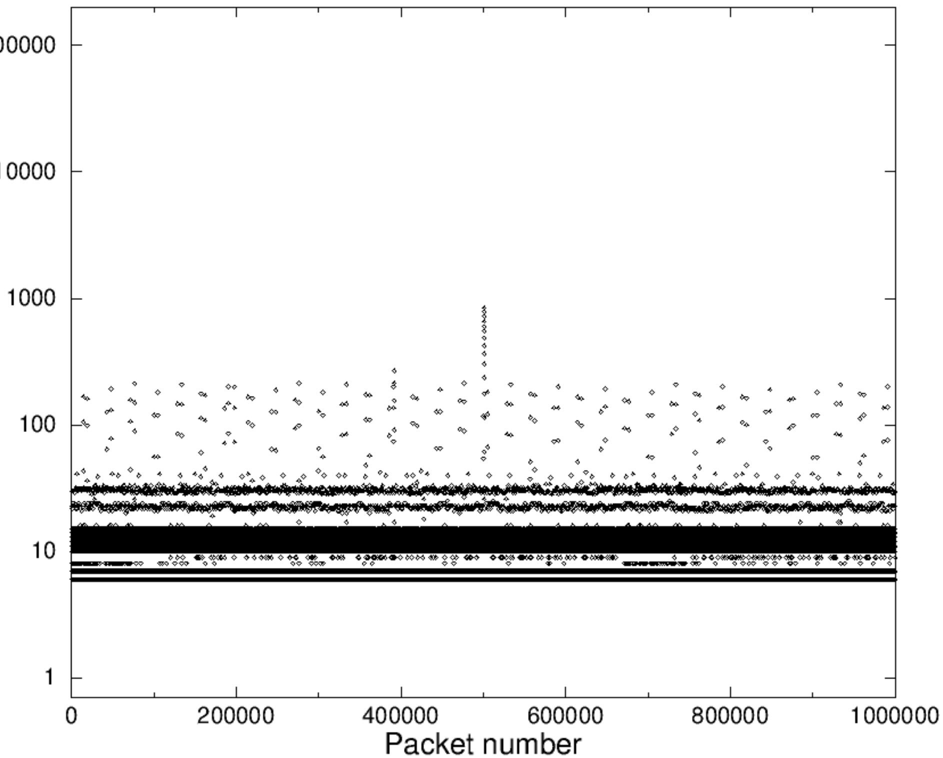


mer management problem

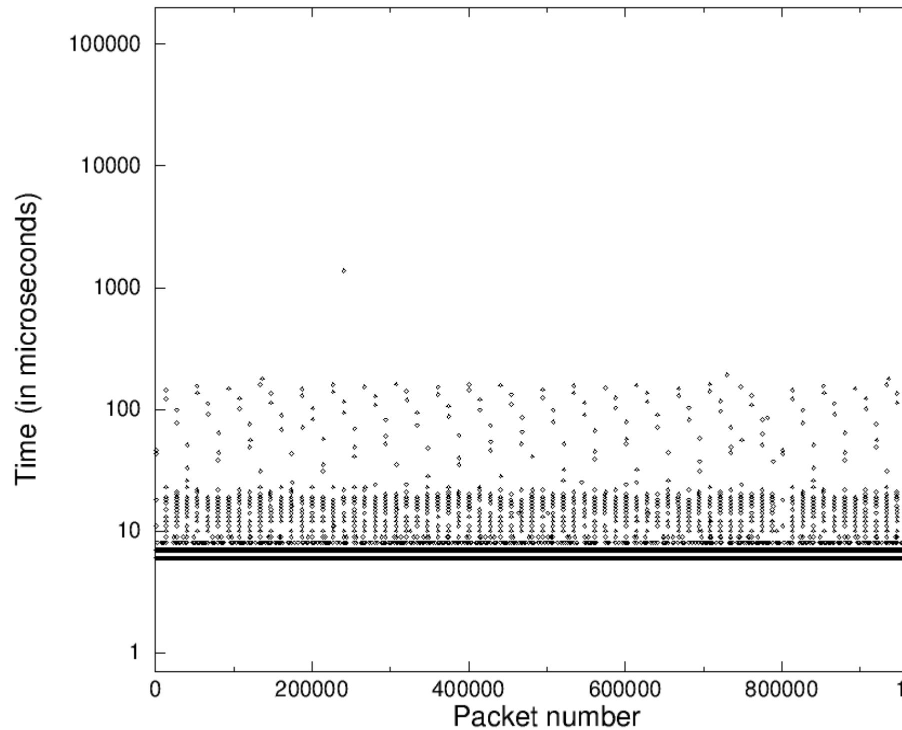
- Must accurately interrupt OS to send packets
- Approaches
  - Polling loop
    - Spin calling `gettimeofday()` until time to send
    - High overhead, accurate
  - `usleep()`
    - Register timer interrupt
    - Low overhead, potentially inaccurate
- Examine each approach using fixed workloads
  - 1 million packet trace
  - Constant-interarrival times δ=70 µsec, δ=2500 µsec

mer management problem

- Polling loop



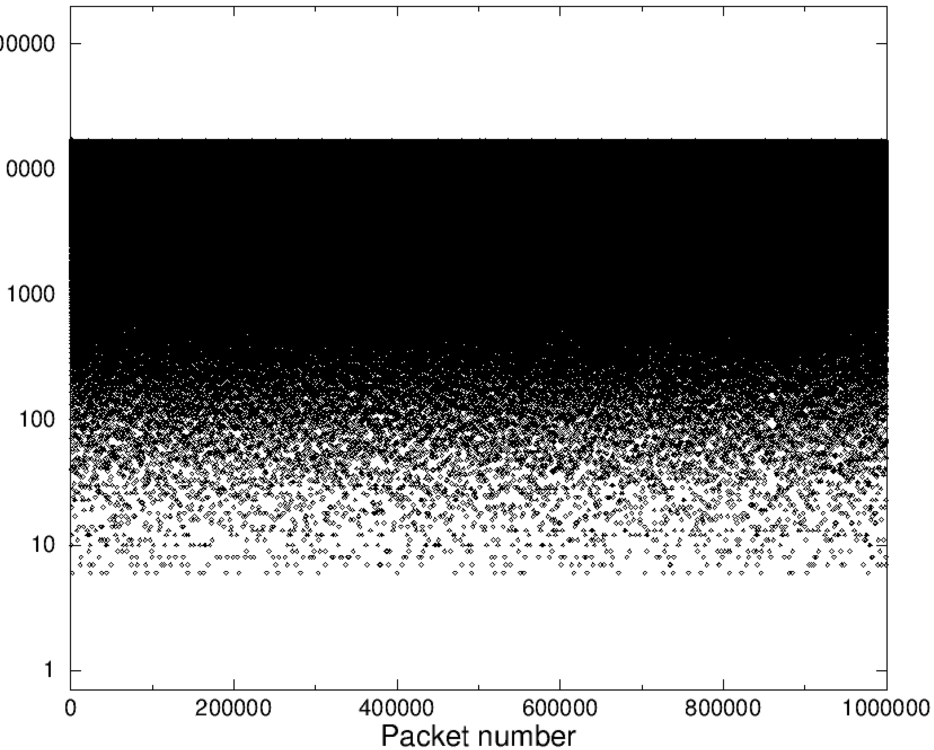
δ=70 μsec
78% User-space CPU utilization



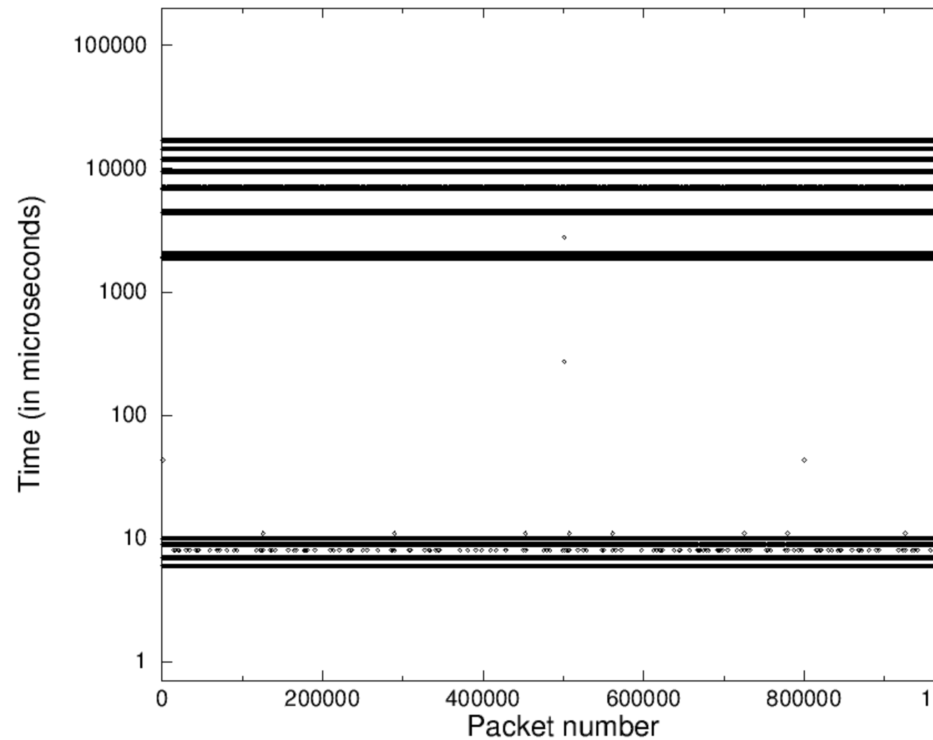
δ=2500 μsec
99% User-space CPU utilizatio

mer management problem

usleep()



δ=70 μsec
40% User-space CPU utilization



δ=2500 μsec
4% User-space CPU utilizatior

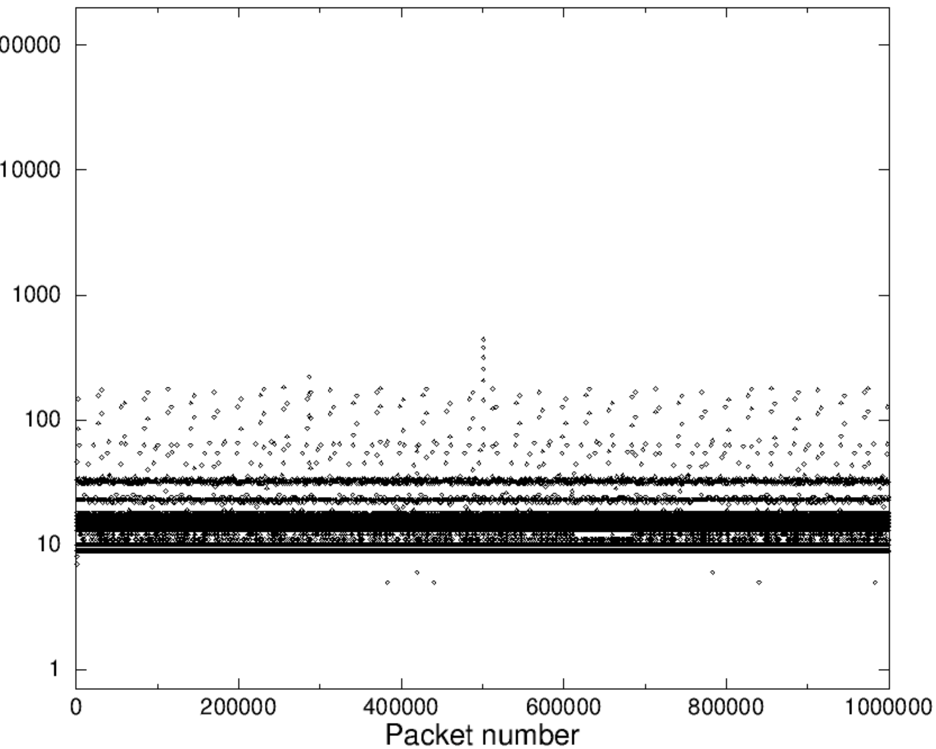
mer management in TCPivo

"Firm timers"

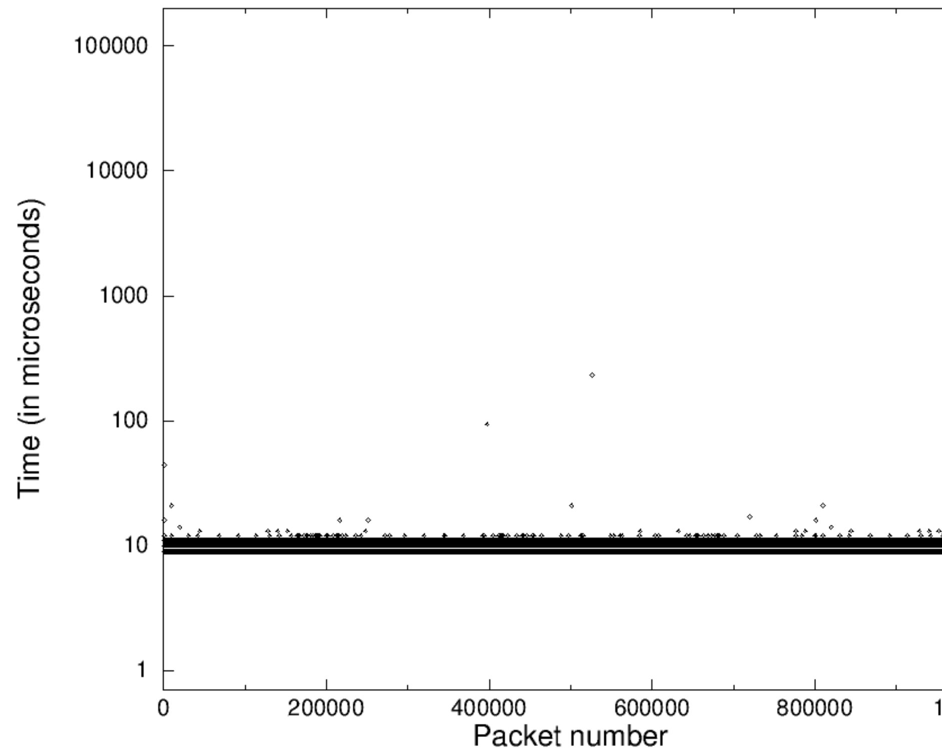
- Combination of periodic and one-shot timers in x86
  - PIT (programmable interval timer)
  - APIC (advanced programmable interrupt controller)
  - Use PIT to get close, use APIC to get the rest of the way
- Timer reprogramming and interrupt overhead reduced via sof timers approach
- Transparently used via changes to `usleep()`

imer management in TCPivo

- Firm timers



δ=70 μsec
19% User-space CPU utilization



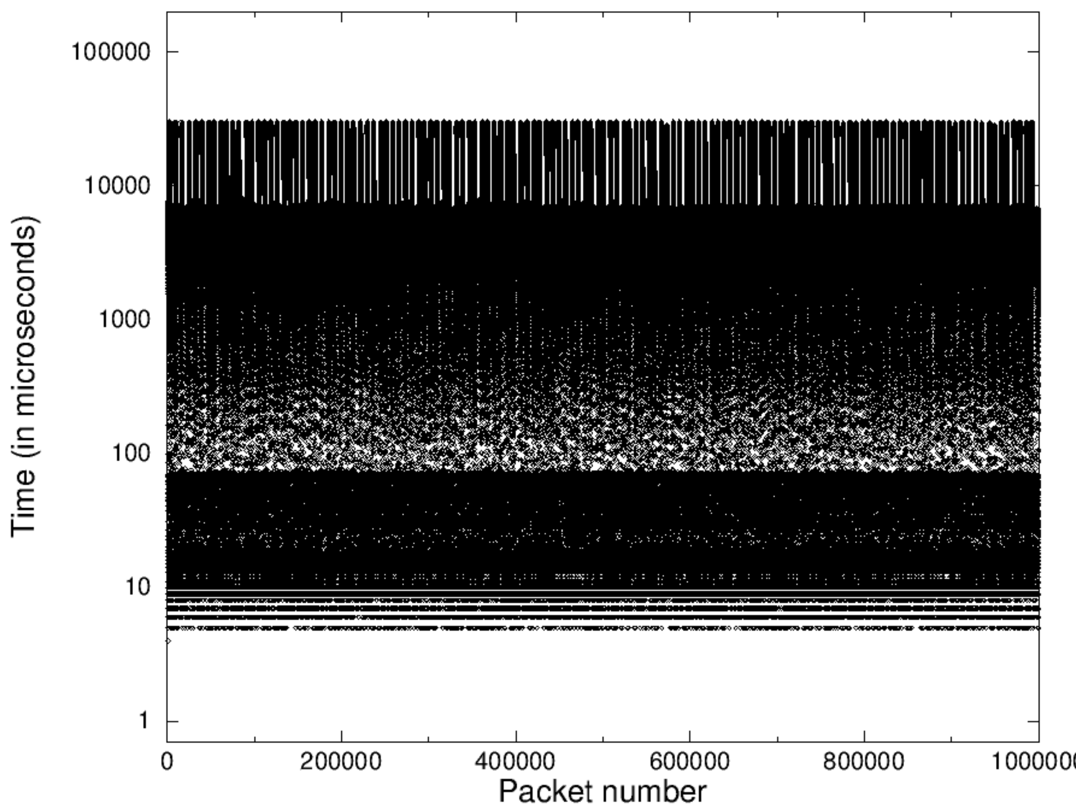
δ=2500 μsec
1% User-space CPU utilizatior

cheduling and pre-emption problem

- Getting control of the OS when necessary
- Low-latency, pre-emptive kernel patches
  - Reduce length of critical sections
- Examine performance under stress
  - I/O workload
    - File system stress test
    - Continuously open/read/write/close an 8MB file
  - Memory workload (see paper)

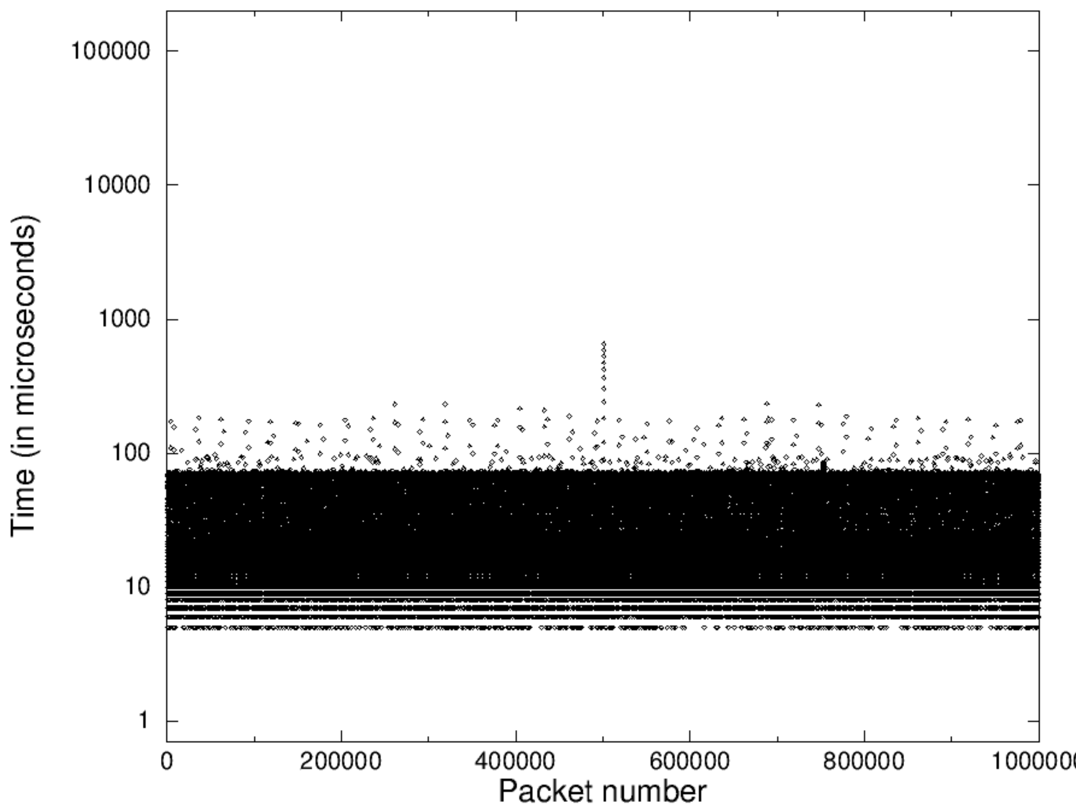
cheduling and pre-emption problem

- Firm timer kernel without low-latency and pre-emptive patches
- I/O Workload, δ=70μsec



cheduling and pre-emption in TCPivo

- Firm timer kernel with low-latency and pre-emptive patches
- I/O Workload, δ=70μsec



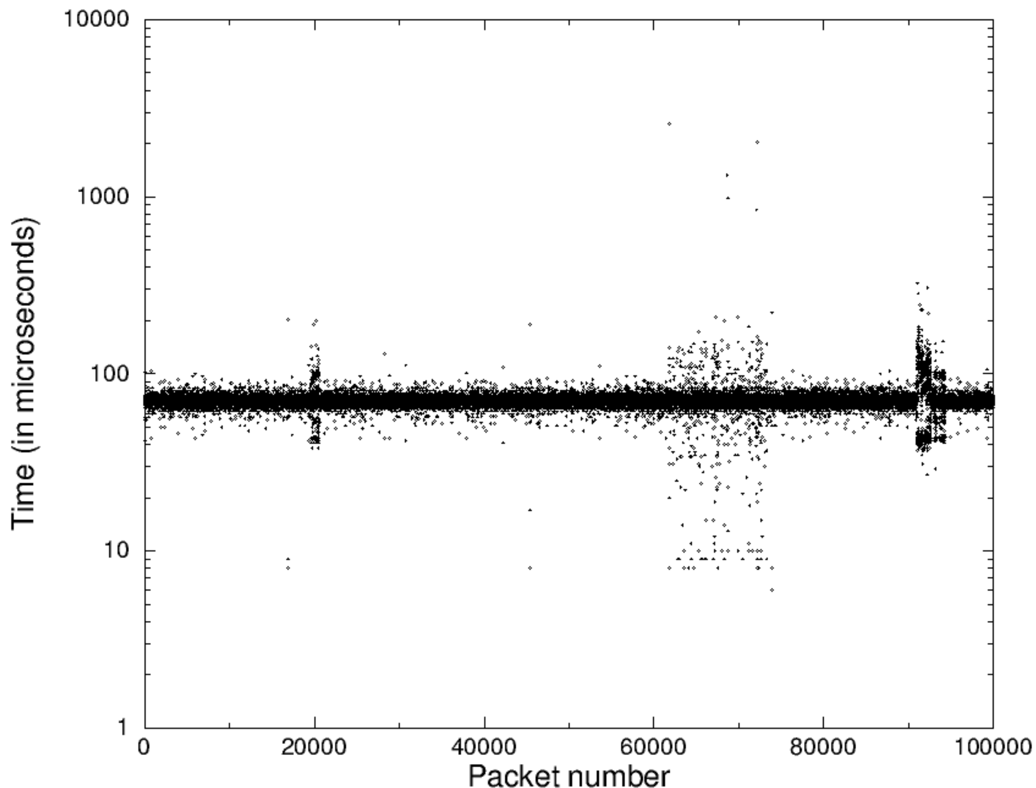
fficient sending loop in TCPivo

- Zeroed payload
- Optional pre-calculation of packet checksums

Task	Average time spent
Trace read	1.30 μsec
Data padding	1.45 μsec
Checksum calculation	1.27 μsec
sendto()	5.16 μsec
Main loop	**9.38 μsec**

utting it all together

- On the wire accuracy
  - δ=70μsec workload at the sender
  - Point-to-point Gigabit Ethernet link
  - Measured inter-arrival times of packets at receiver



oftware availability

- TCPivo
  - http://www.cse.ogi.edu/sysl/projects/tcpivo
  - Formerly known as NetVCR before an existing product of th same name forced a change to a less catchier name.
- Linux 2.4
  - Firm timers
    - http://www.cse.ogi.edu/sysl/projects/TSL
  - Andrew Morton's low-latency patch
    - http://www.zip.com.au/~akpm/linux/schedlat.html
  - Robert Love's pre-emptive patch
    - http://kpreempt.sourceforge.net
- Linux 2.5
  - Low-latency, pre-emptive patches included
  - High-resolution timers via 1ms PIT (No firm timer support)

pen issues

- Multi-gigabit replay
  - Zero-copy
  - TOE
  - SMP
- Accurate, but not realistic for evaluating everything
  - Open-loop (not good for AQM)
  - Netbed/PlanetLab?
    - Requires on-the-fly address rewriting

uestions?

- TCPivo
  - http://www.cse.ogi.edu/sysl/projects/tcpivo

u-chang Feng, Ashvin Goel, Abdelmajid Bezzaz, Wu-chi Feng, Jonathan
lpole, "TCPivo: A High-Performance Packet Replay Engine", *in
roceedings of ACM SIGCOMM Workshop on Models, Methods, and Tools for
eproducible Network Research (MoMeTools)* August 2003.

Back

Performance Analysis of Packet Classification Algorithms on Netwo
Processors

Deepa Srinivasan Wu-chang Feng

OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY

acket classification algorithm mapping

Motivation

- Packet classification is an inherent function of network devic
  - Many algorithms for single-threaded software execution
  - Many hardware-specific algorithms
  - Not a lot for programmable multi-processors

Our study

- Examine algorithmic mapping of a hardware algorithm (BitVector) onto the IXP
  - Pipelined (4 dimensions on 3 μ-engines, 1 combo, 1 ingress, 1 egres
  - Parallel (complete lookup on 4 μ-engines, 1 ingress, 1 egress)

acket classification algorithm mapping

- Initial results
  - Hard to generalize
    - Depends on workload, rulesets, implementation
  - Trie lookups bad for μ-engine health
    - Frequently forced into aborted state due to branching
      - Linear search: ~10-11%,
      - Pipelined Bit-Vector: ~17%
      - Parallel Bit-Vector: ~22%
    - Impacts device predictability and algorithm/compiler design
      - Avoid branches, utilize range-matching?
  - Memory bottleneck favors parallel over pipelined in IXP120
    - Pipelined slightly worse than parallel due to multiple header parsing
    - Will change with IXP2xxx next-neighbor registers

Questions?

Packet classification

- http://www.cse.ogi.edu/sysl/projects/ixp

Deepa Srinivasan, “Performance Analysis of Packet Classification Algorithms on Network Processors”, OGI MS Thesis, May 2003 (submission planned)

Back

Panoptes: A Flexible Platform for Video Sensor Applications

Wu-chi Feng
Brian Code
Ed Kaiser
Mike Shea
Wu-chang Feng
Louis Bavoil

OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY

in Proceedings of ACM Multimedia 2003, November 2003.

Motivation

Emerging video sensor applications with varying requirements

- Environmental observation
- Home health-care monitoring
- Security and surveillance
- Augmented reality
- Robotics
- UAV applications

Goal

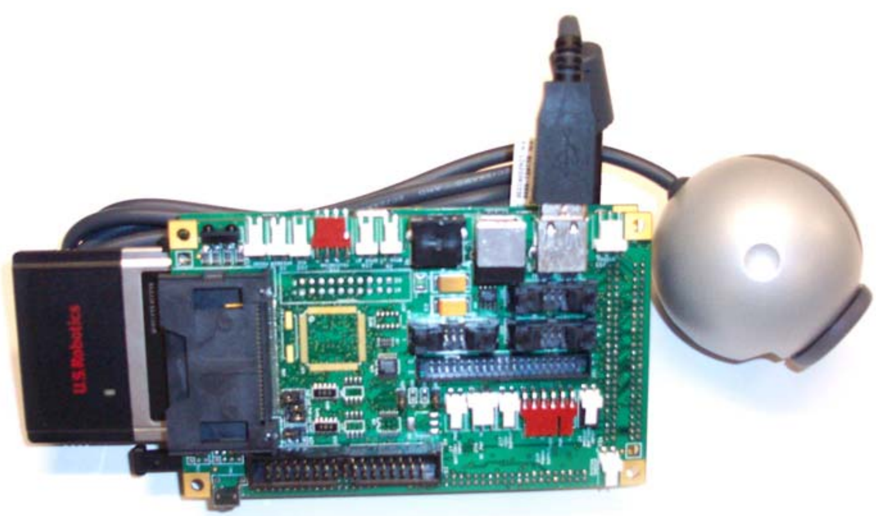
Design, implement, and demonstrate a small, low-power programmable video platform

- Push as much functionality out to the sensors
- Allow easy reconfiguration of functionality to support multiple applications

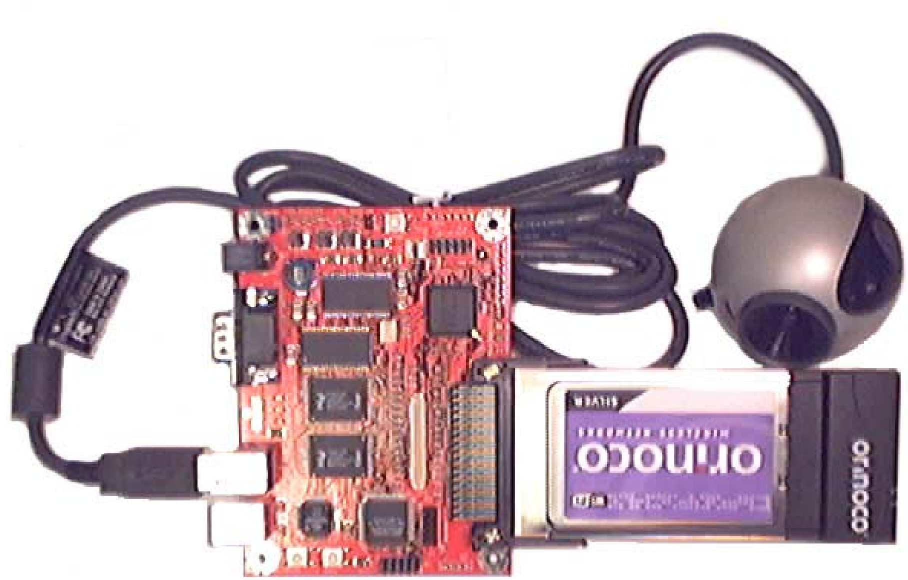
Panoptes

320 x 240 pixel video @ 24 fps
802.11 wireless, USB-based video, Linux

206 MHz Intel StrongARM
~5.5 Watts (fully loaded)



400 MHz Intel Xscale
~4 Watts (fully loaded)



Panoptes

Software architecture

- Functions implemented and compiled in C
  - Buffering
  - Blending
  - Motion detection
  - Dithering
  - Compression
  - Adaptation
- Python scripts to compose functionality
  - Similar to the `ns` simulator and `Tcl`
  - Supports dynamic reconfiguration of video sensors to application-specific needs without recompilation

Demo

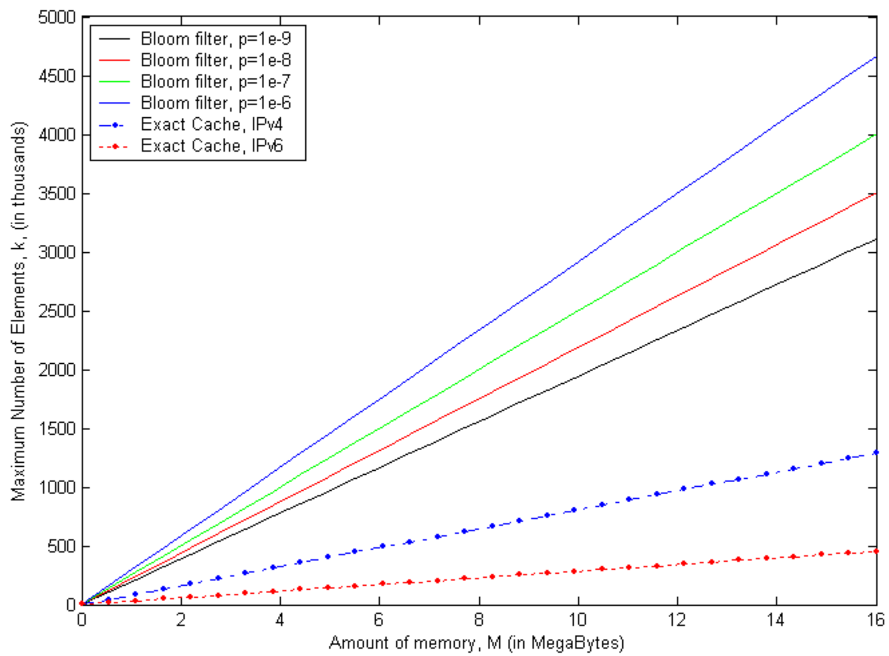
- Little Sister Sensor Networking Application
- Visit OGI for a full demo!

Back

caching

Results

- Order of magnitude space savings for an error rate of one in a billion
- Analytical model for controlling misclassification rate



Francis Chang, Kang Li, Wu-chang Feng, "Approximate Caches for Packet Classification", *in ACM SIGCOMM 2003 Poster Session*, Aug. 2003. **Poster**

Francis Chang, Kang Li, Wu-chang Feng, "Approximate Caches for Packet Classification", *in submission*. **Paper**

xact Packet Classification Caching

- Initial results
  - Address allocation policies allow μ-engine based XOR-hashe to outperform stronger hashes (i.e. centralized IXP hash unit)
  - LFU provides only marginal improvement over LRU with multimedia traffic

Kang Li, Francis Chang, Damien Berger, Wu-chang Feng, "Architectures for Packet Classification Caching", *in Proceedings of International Conference on Networks*, Sept. 2003.

Back

CPivo: High-Performance Packet Replay

- Linux x86-based tool for accurate replay above OC-3
  - Trace management with `mmap()/madvise()`
  - Timer management with firm timers
  - Low transmission overhead
  - Proper scheduling and pre-emption via low-latency and pre-emptive patches
- Software available
  - http://www.cse.ogi.edu/sysl/projects/tcpivo

u-chang Feng, Ashvin Goel, Abdelmajid Bezzaz, Wu-chi Feng, Jonathan alpole, "TCPivo: A High-Performance Packet Replay Engine", *in roceedings of ACM SIGCOMM Workshop on Models, Methods, and Tools or Reproducible Network Research (MoMeTools)* August 2003.

Back

xtra slides

here's the IXP implementation?

- Big issue: IXP1200 is not built for security
  - Pseudo-random number generator can be predicted
  - Internal hash unit cyptographically weak
- Have a very short wish-list of functions
  - IXP 2850 has most of them

Future work

Application interface to puzzle manager

- Integration with IDS
- Integration with applications

Puzzle expiry and pre-issuing system

Better adaptation control

airness

- Inserting a “trust” estimator into the knowledge plane
  - Answer the “WHO” question?
    - Who is a likely source of a future DoS attack?
  - No keys, no signatures, no centralized source
  - Based on time-varying distributed view of client behavior
  - Similar to GeoNetMap's “confidence” measure

puzzle scenario #2

Coordinated DDoS: simultaneous attacks against multip sites from the same set of zombie machines

- Mafiaboy (2000)
- Have zombies initiate low bandwidth attacks on a diverse set of victims to evade localized detection techniques (such as `mod_dosevasive`)

' puzzle scenario #2

- Mitigation using IP puzzles