Next Stop 100 Gb/s: Scaling Network Security Monitoring to New Operational Challenges

Robin Sommer

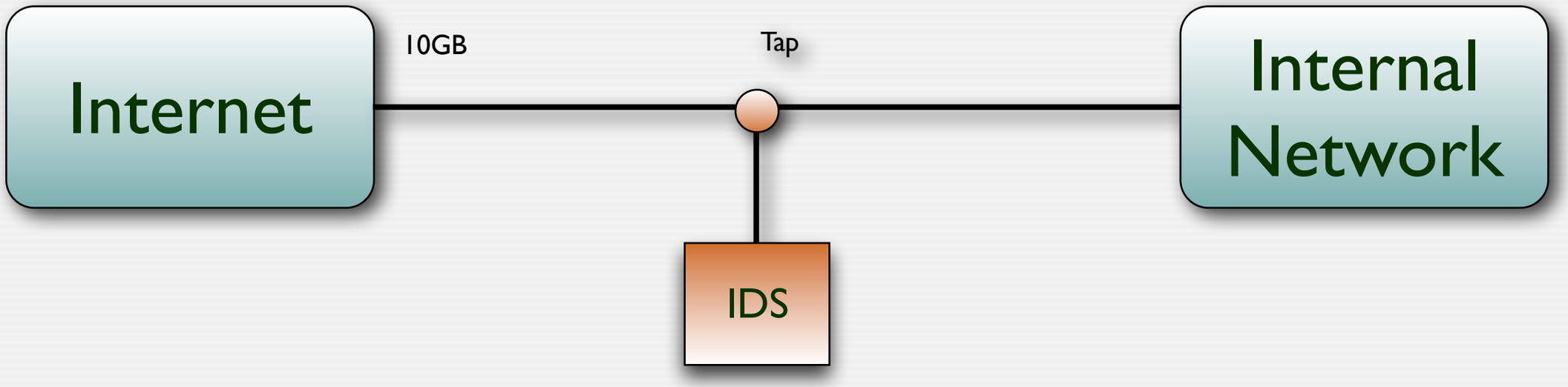
International Computer Science Institute, &
Lawrence Berkeley National Laboratory

robin@icsi.berkeley.edu
http://www.icir.org

ICSI Board Meeting
October 2011

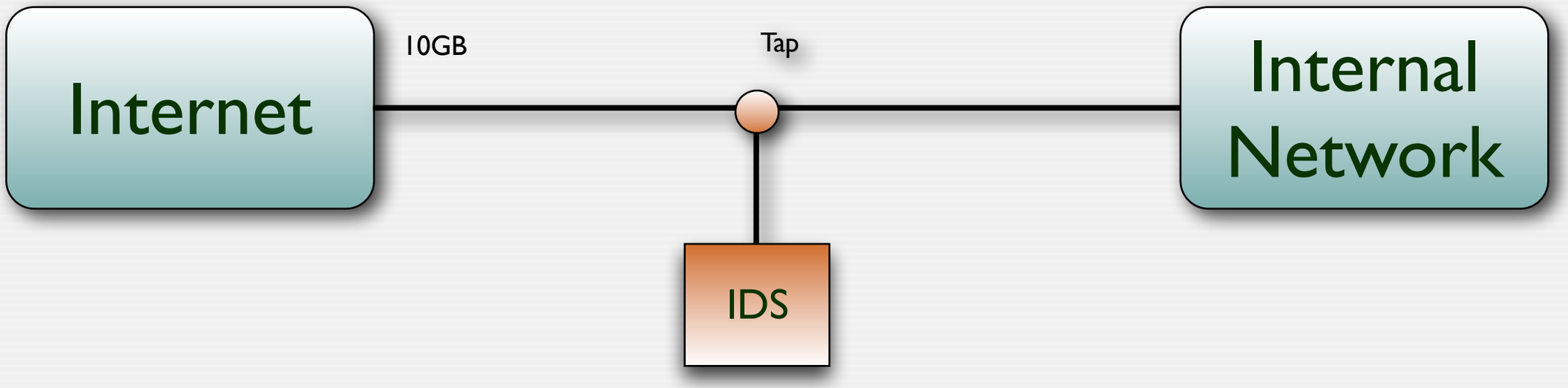
Monitoring Network Security

Traditional process is conceptually quite simple.



Monitoring Network Security

Traditional process is conceptually quite simple.



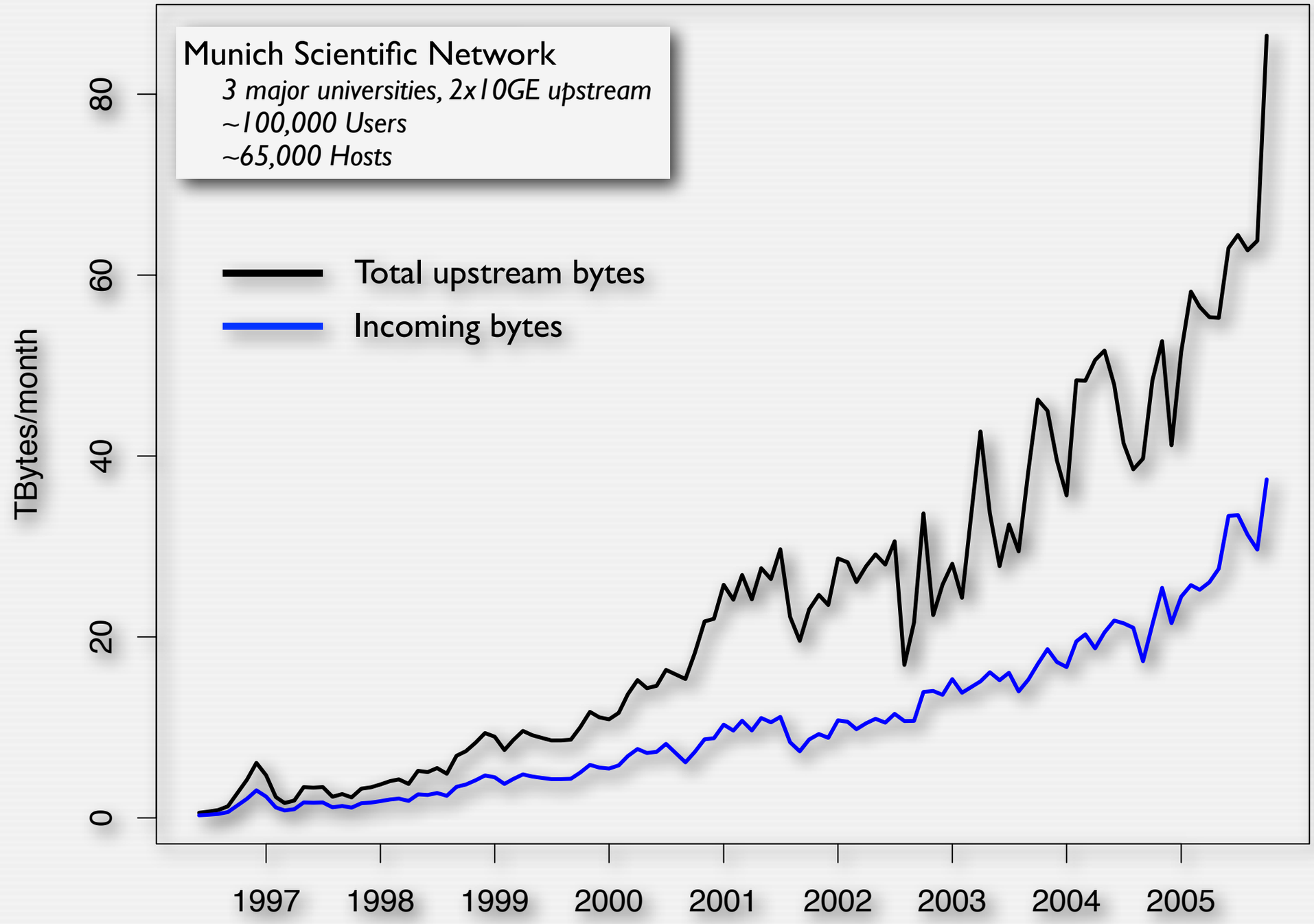
Example: Finding downloads of known malware.

1. Find and parse all Web traffic.
2. Find and extract binaries.
3. Compute hash and compare with database.
4. Report, and potentially kill, if found.

Back in 2005 …

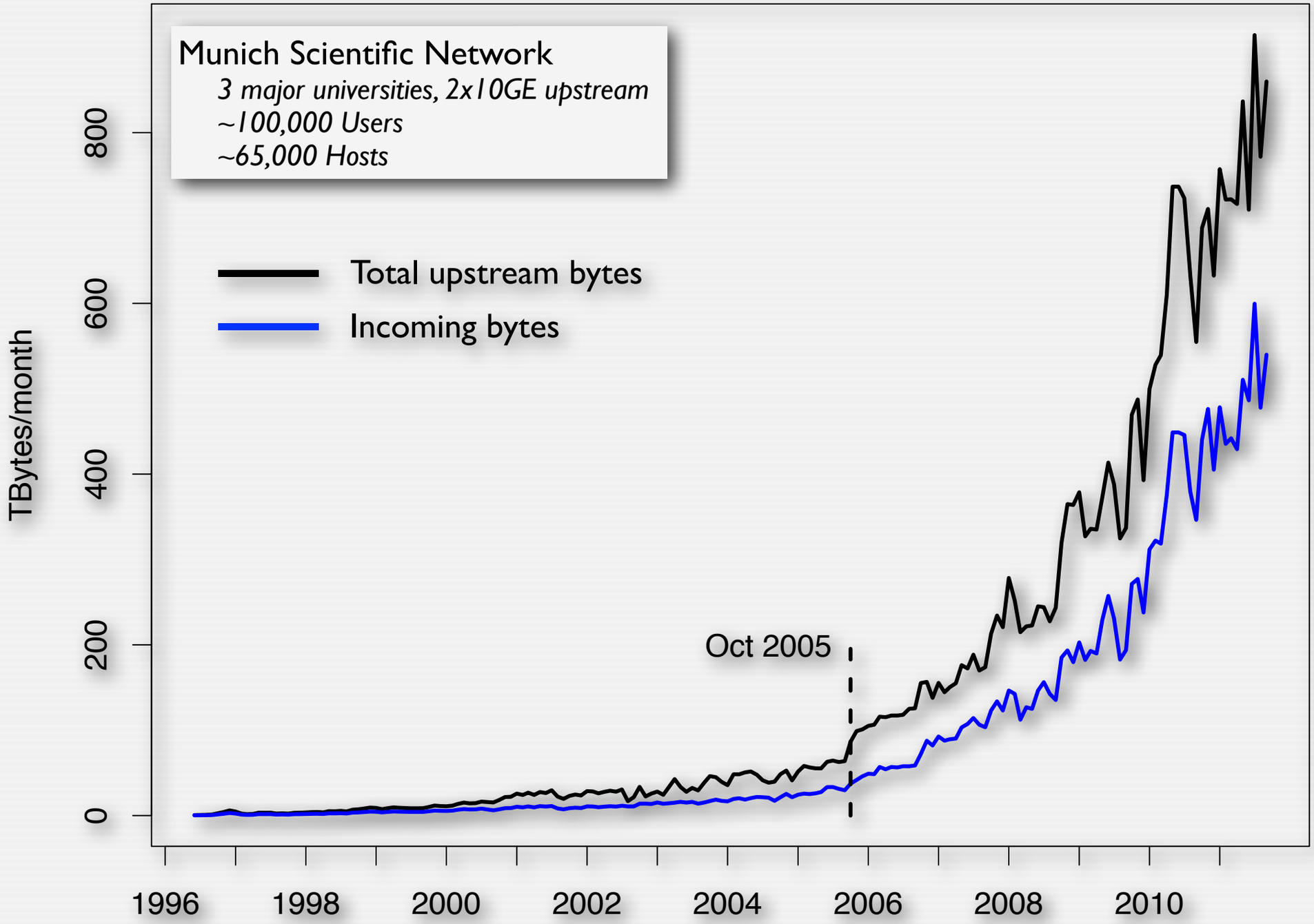
Data: Leibniz-Rechenzentrum, München

Back in 2005 ...



Data: Leibniz-Rechenzentrum, München

Today ...



Data: Leibniz-Rechenzentrum, München

Gap: Research vs Operations

Gap: Research vs Operations

Conceptually simple tasks can be hard in practice.

- Academic research often neglects operational constraints.
- Operations cannot leverage academic results.

Gap: Research vs Operations

Conceptually simple tasks can be hard in practice.

- Academic research often neglects operational constraints.
- Operations cannot leverage academic results.

At ICSI, opportunity to work *with* operations.

- Close collaborations with several large sites (LBNL,UC Berkeley, NCSA).
- Extremely fruitful for both sides.

Gap: Research vs Operations

Conceptually simple tasks can be hard in practice.

- Academic research often neglects operational constraints.
- Operations cannot leverage academic results.

At ICSI, opportunity to work *with* operations.

- Close collaborations with several large sites (LBNL,UC Berkeley, NCSA).
- Extremely fruitful for both sides.

Operational constraints pose research questions.

- Results go back into operations.

Bro: Bridging The Gap for 15 years

Bro: Bridging The Gap for 15 years

Bro IDS one primary research platform.

- Designed and developed by Vern Paxson in 1996.
- Open-source, BSD-license, maintained at ICSI.
- Conceptually very different from other IDS.

http://www.bro-ids.org

Bro: Bridging The Gap for 15 years

Bro IDS one primary research platform.

Designed and developed by Vern Paxson in 1996.
Open-source, BSD-license, maintained at ICSI.
Conceptually very different from other IDS.

http://www.bro-ids.org

In operational use since its beginnings.

Corner-stone of LBNL network security.
Installations in science, government, industry.

Bro: Bridging The Gap for 15 years

Bro IDS one primary research platform.

Designed and developed by Vern Paxson in 1996.
Open-source, BSD-license, maintained at ICSI.
Conceptually very different from other IDS.

http://www.bro-ids.org

In operational use since its beginnings.

Corner-stone of LBNL network security.
Installations in science, government, industry.

Much functionality came out of research.

Bro: Bridging The Gap for 15 years

Bro IDS one primary research platform.

Designed and developed by Vern Paxson in 1996.
Open-source, BSD-license, maintained at ICSI.
Conceptually very different from other IDS.

http://www.bro-ids.org

In operational use since its beginnings.

Corner-stone of LBNL network security.
Installations in science, government, industry.

Much functionality came out of research.

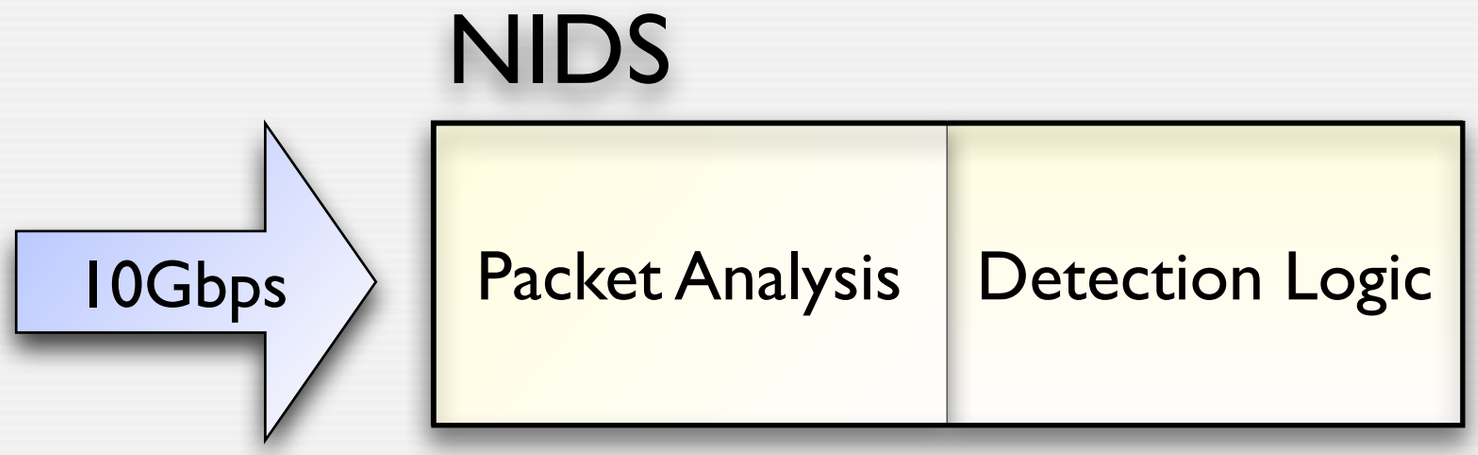
Example: Processing performance.

LBNL operations had trouble keeping up.
Research question: How can Bro scale?

Load-balancing Architecture

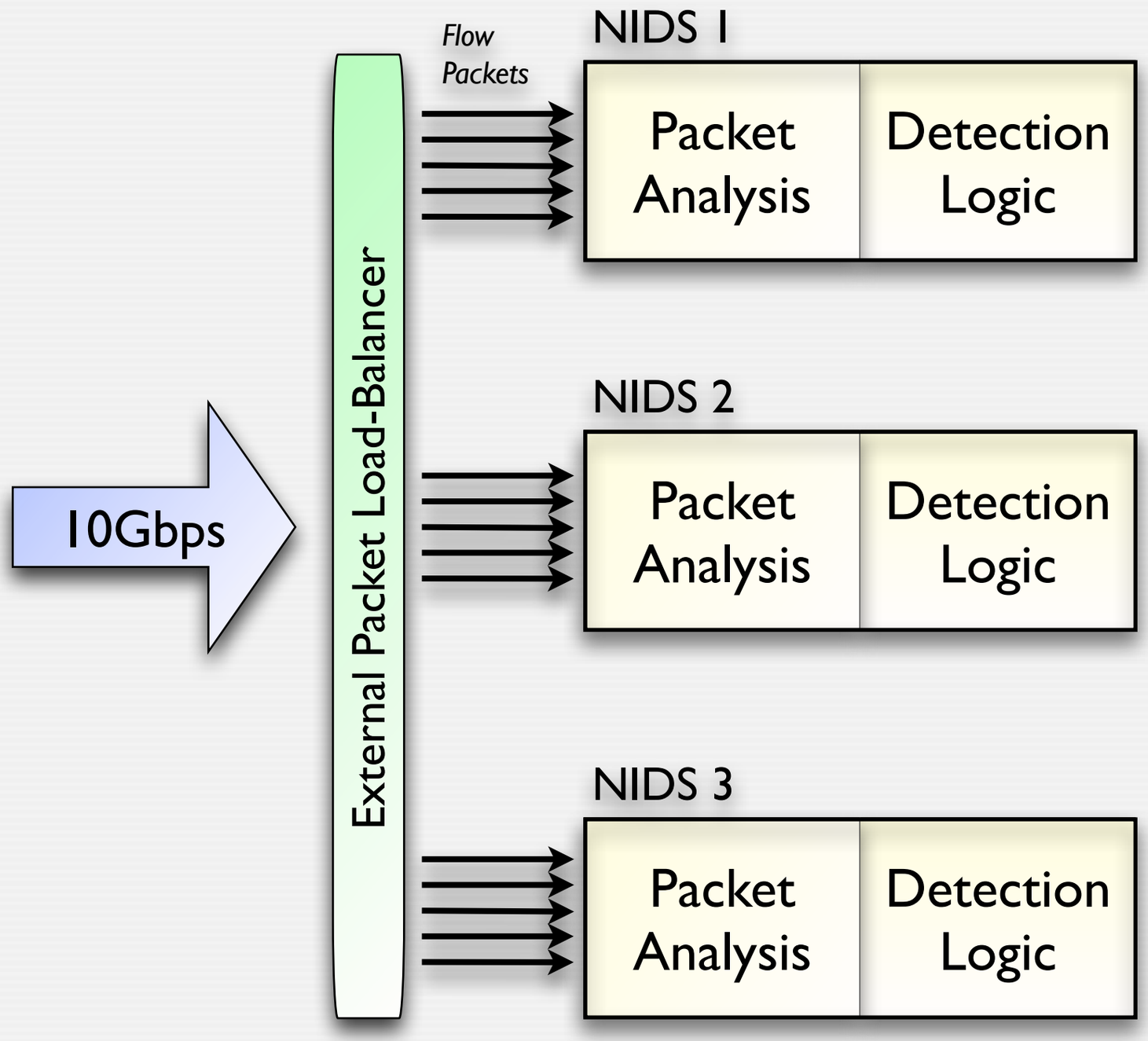
[Vallentin et al. 2007]

Load-balancing Architecture



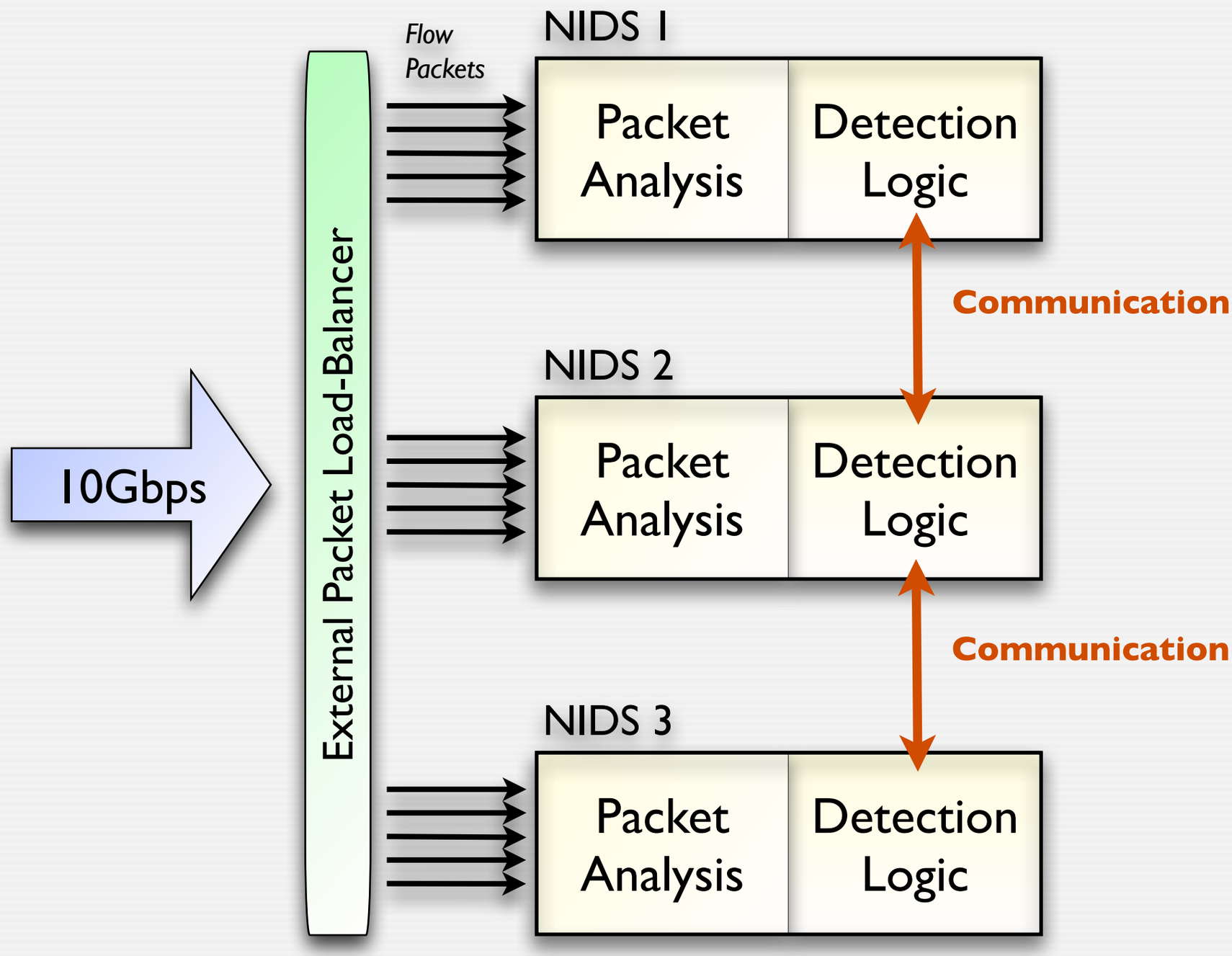
[Vallentin et al. 2007]

Load-balancing Architecture



[Vallentin et al. 2007]

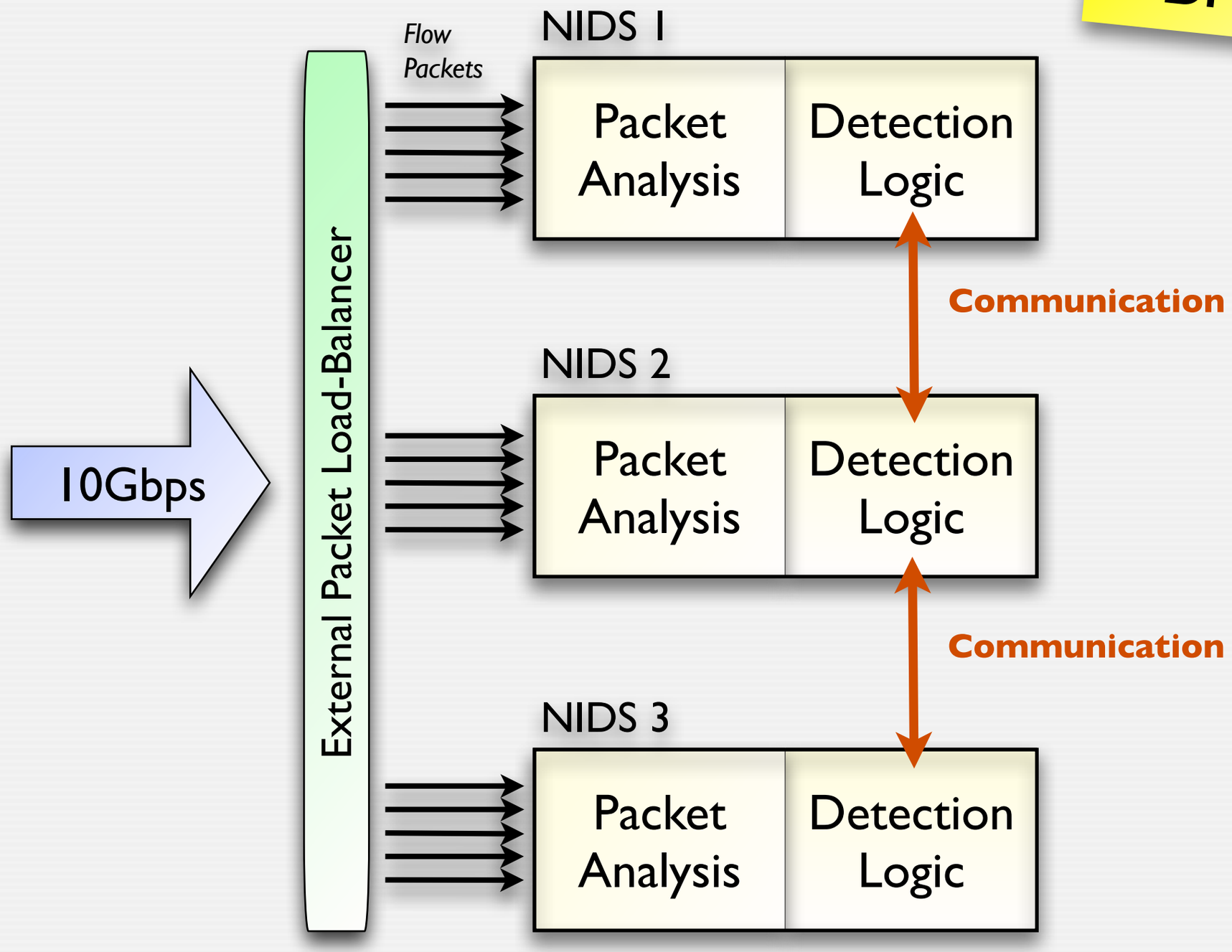
Load-balancing Architecture



[Vallentin et al. 2007]

Load-balancing Architecture

"Bro Cluster"



[Vallentin et al. 2007]

Cluster goes Operation

Cluster goes Operation

Load-balancer operates a line-rate.

1. Receive packet.
2. Calculate hash.
3. Rewrite MAC address.
4. Send packet out.

Cluster goes Operation

Load-balancer operates a line-rate.

1. Receive packet.
2. Calculate hash.
3. Rewrite MAC address.
4. Send packet out.

Research prototype limited to 2 Gb/s.

Linux box using kernel-level *Click*.

Cluster goes Operation

Load-balancer operates a line-rate.

1. Receive packet.
2. Calculate hash.
3. Rewrite MAC address.
4. Send packet out.

Research prototype limited to 2 Gb/s.

Linux box using kernel-level *Click*.

LBNL wanted reliable 10 Gb/s device.

Difficult to find a robust line-rate solution.

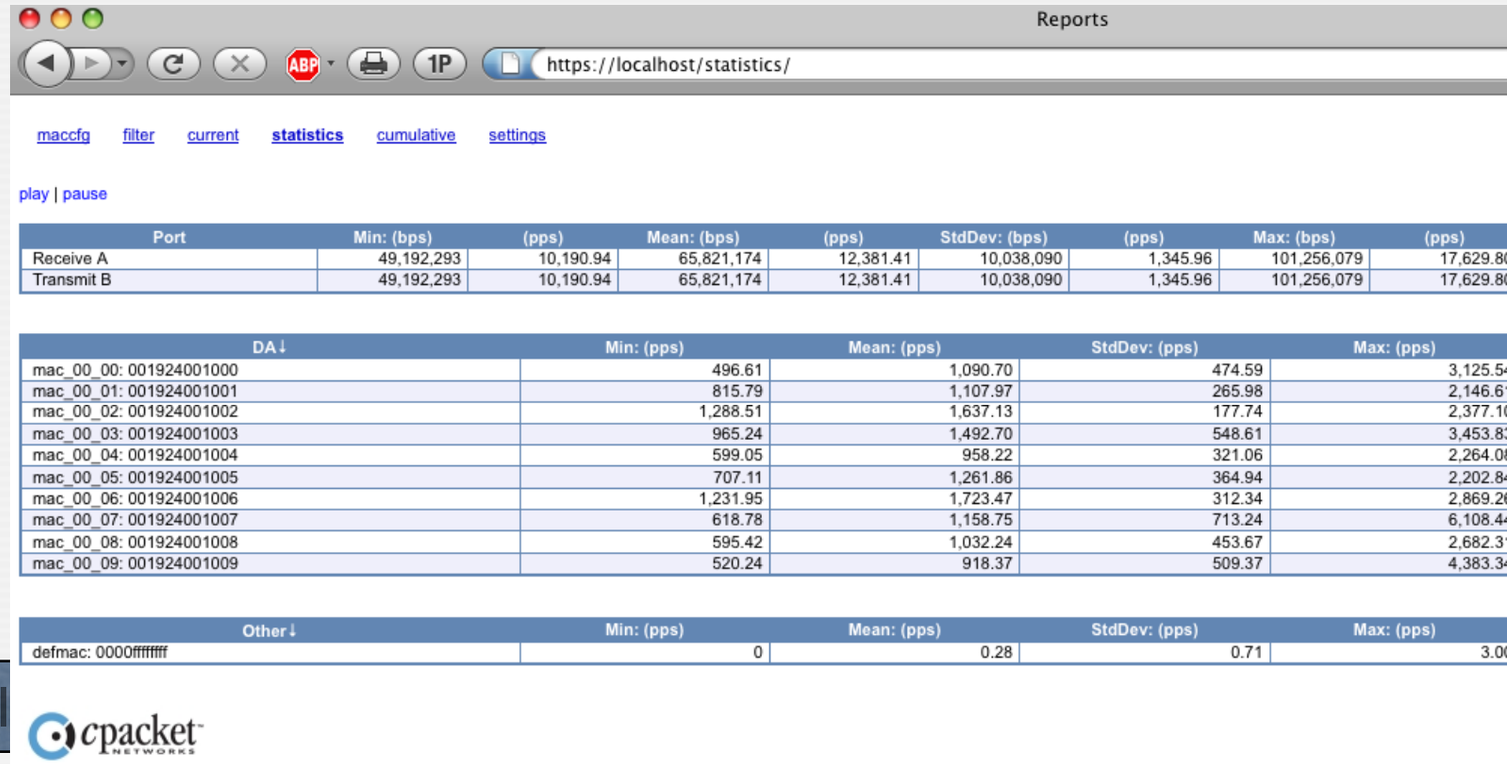
Eventually contracted vendor to build device.

A Production Load-Balancer

cFlow: 10GE line-rate, stand-alone load-balancer

10 Gb/s in/out
Web & CLI
Filtering capabilities

Available from cPacket



Port	Min: (bps)	(pps)	Mean: (bps)	(pps)	StdDev: (bps)	(pps)	Max: (bps)	(pps)
Receive A	49,192,293	10,190.94	65,821,174	12,381.41	10,038,090	1,345.96	101,256,079	17,629.8
Transmit B	49,192,293	10,190.94	65,821,174	12,381.41	10,038,090	1,345.96	101,256,079	17,629.8

DA↓	Min: (pps)	Mean: (pps)	StdDev: (pps)	Max: (pps)
mac_00_00: 001924001000	496.61	1,090.70	474.59	3,125.5
mac_00_01: 001924001001	815.79	1,107.97	265.98	2,146.6
mac_00_02: 001924001002	1,288.51	1,637.13	177.74	2,377.1
mac_00_03: 001924001003	965.24	1,492.70	548.61	3,453.8
mac_00_04: 001924001004	599.05	958.22	321.06	2,264.0
mac_00_05: 001924001005	707.11	1,261.86	364.94	2,202.8
mac_00_06: 001924001006	1,231.95	1,723.47	312.34	2,869.2
mac_00_07: 001924001007	618.78	1,158.75	713.24	6,108.4
mac_00_08: 001924001008	595.42	1,032.24	453.67	2,682.3
mac_00_09: 001924001009	520.24	918.37	509.37	4,383.3

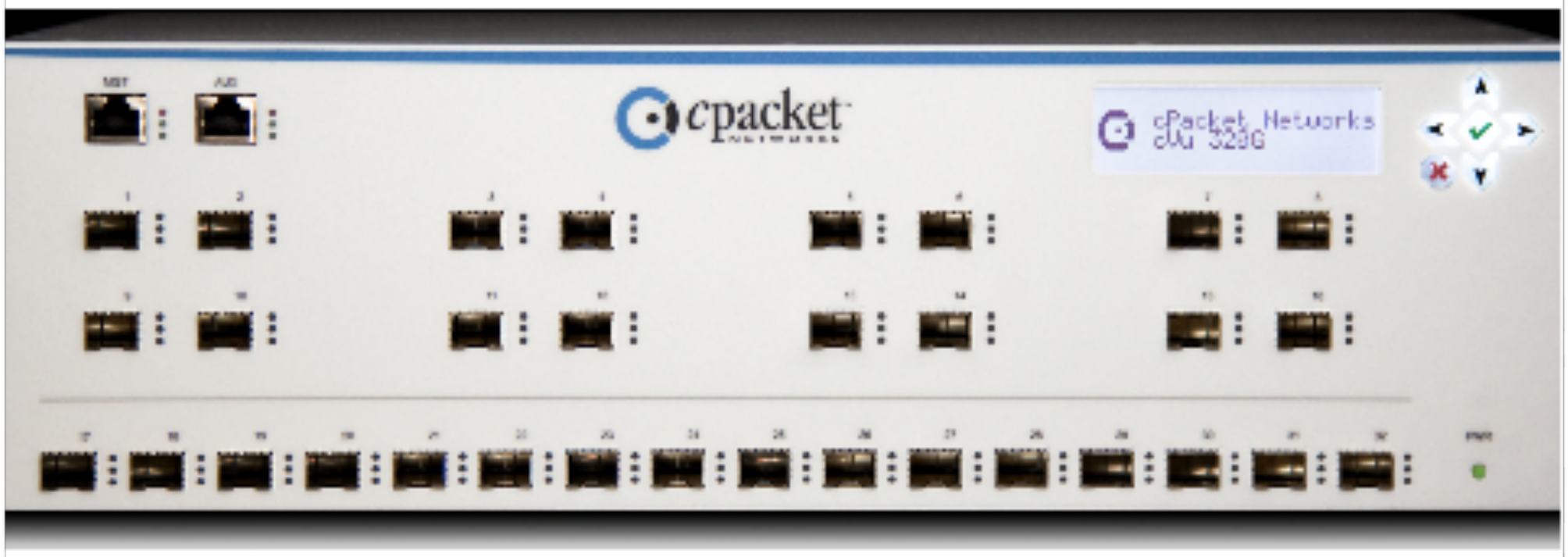
Other↓	Min: (pps)	Mean: (pps)	StdDev: (pps)	Max: (pps)
defmac: 0000ffffffff	0	0.28	0.71	3.0

A Production Load-Balancer

cPacket cVu 320G

cpacket NETWORKS

32 x 10G SFP+ Traffic Monitoring Switch
Aggregation, Complete Packet Inspection Filtering, Automatic Flow Balancing



Available from cPacket

mac_00_02: 001924001002	1,288.51	1,637.13	177.74	2,377.10
mac_00_03: 001924001003	965.24	1,492.70	548.61	3,453.83
mac_00_04: 001924001004	599.05	958.22	321.06	2,264.08
mac_00_05: 001924001005	707.11	1,261.86	364.94	2,202.84
mac_00_06: 001924001006	1,231.95	1,723.47	312.34	2,869.26
mac_00_07: 001924001007	618.78	1,158.75	713.24	6,108.44
mac_00_08: 001924001008	595.42	1,032.24	453.67	2,682.31
mac_00_09: 001924001009	520.24	918.37	509.37	4,383.34

Other↓	Min: (pps)	Mean: (pps)	StdDev: (pps)	Max: (pps)
defmac: 0000ffffffff	0	0.28	0.71	3.00

cpacket NETWORKS

What Does This Mean for Us?

What Does This Mean for Us?

Operations is using our technology.

- LBNL has 3 operational Bro Clusters w/ 15 backends.
- Other sites have, or are building, similar setups.

What Does This Mean for Us?

Operations is using our technology.

LBNL has 3 operational Bro Clusters w/ 15 backends.

Other sites have, or are building, similar setups.

New research *platforms* for us.

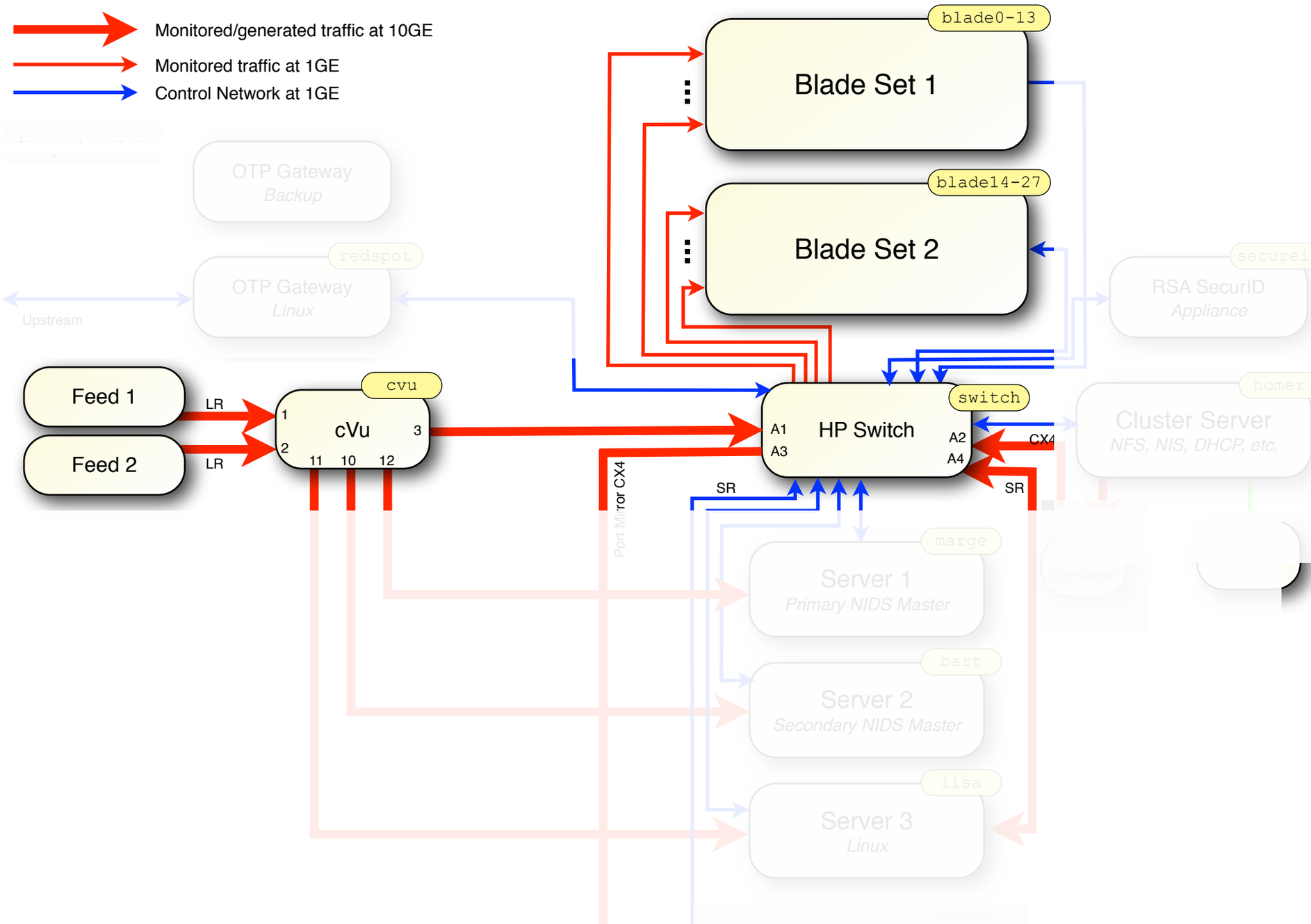
Research cluster at LBNL w/12 backends.

Research cluster at UC Berkeley w/ 28 backends.

UC Berkeley Cluster

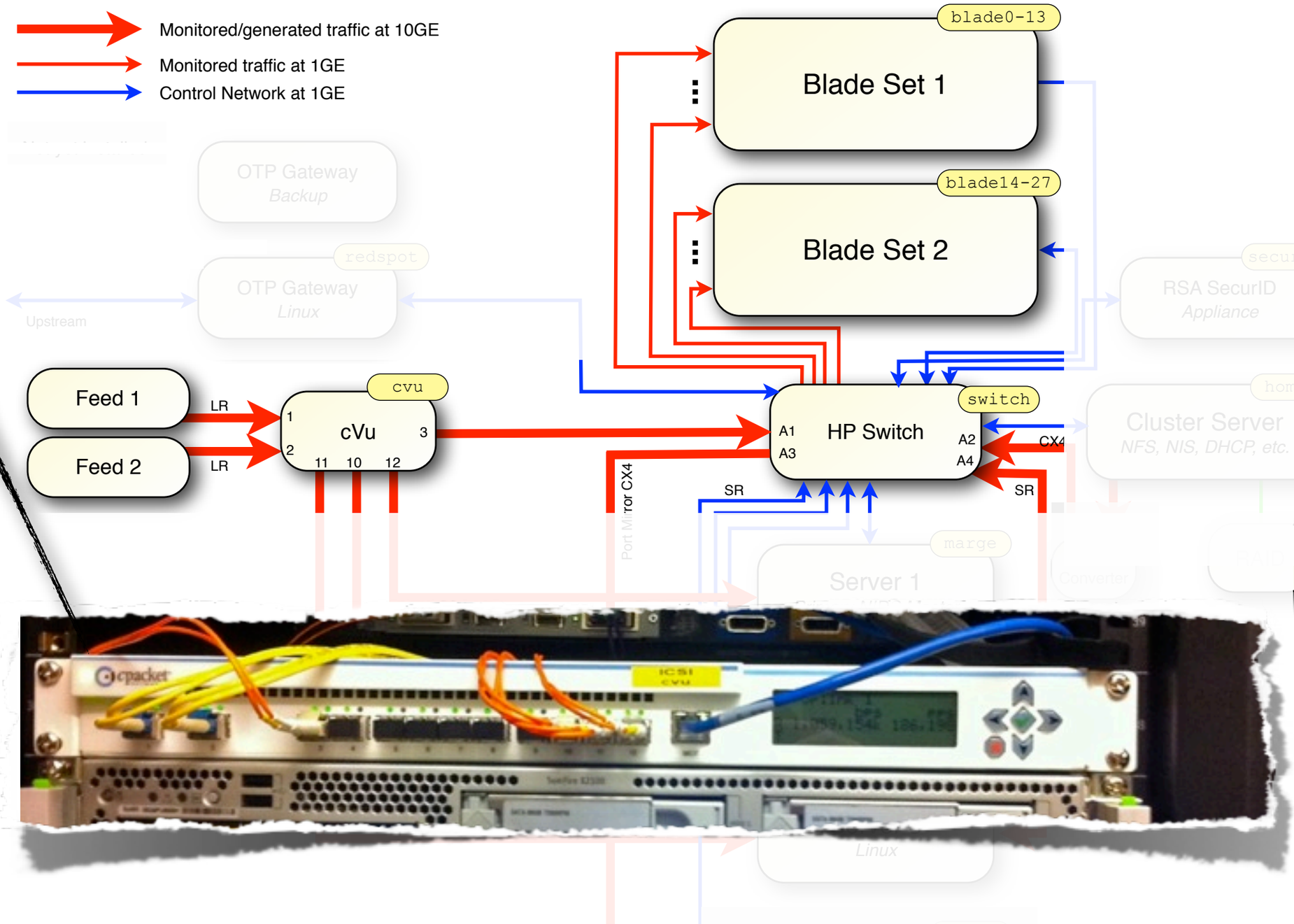
Large-scale network, 100,000 hosts, 2x10Gbps upstream, 2-4Gb/sec.
30 individual machines. Collaboration with Campus SNS.
Funded by NSF's Computing Research Infrastructure program.

UC Berkeley Cluster



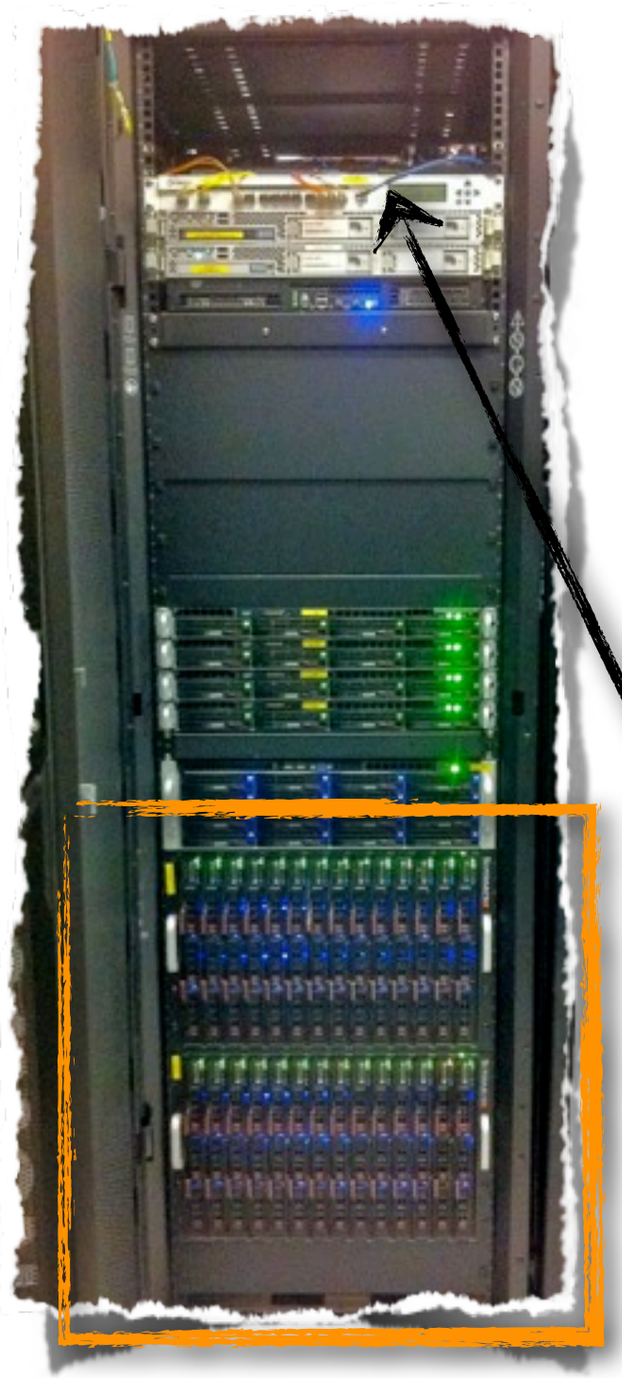
Large-scale network, 100,000 hosts, 2x10Gbps upstream, 2-4Gb/sec.
30 individual machines. Collaboration with Campus SNS.
Funded by NSF's Computing Research Infrastructure program.

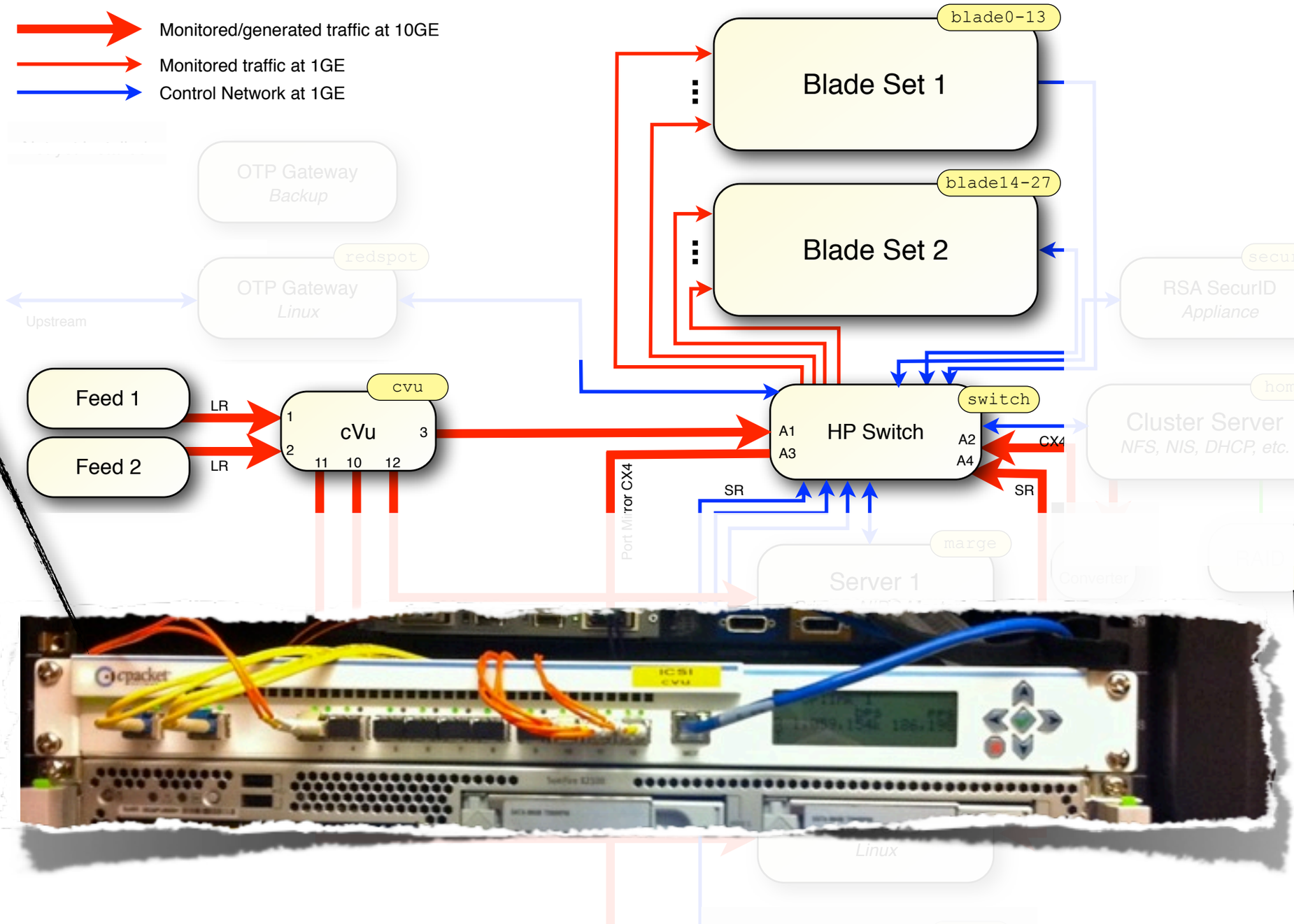
UC Berkeley Cluster



Large-scale network, 100,000 hosts, 2x10Gbps upstream, 2-4Gb/sec.
30 individual machines. Collaboration with Campus SNS.
Funded by NSF's Computing Research Infrastructure program.

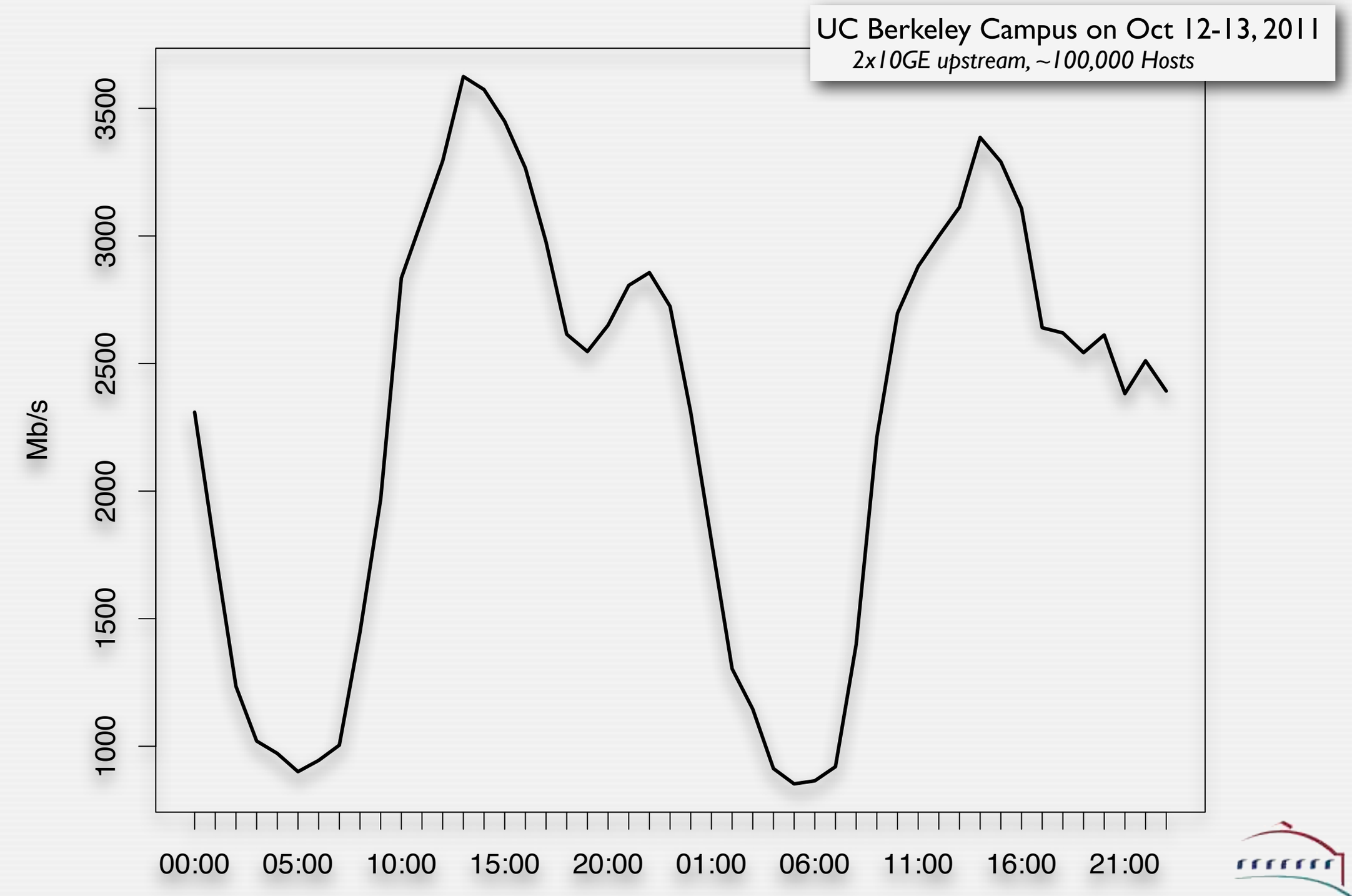
UC Berkeley Cluster





Large-scale network, 100,000 hosts, 2x10Gbps upstream, 2-4Gb/sec.
30 individual machines. Collaboration with Campus SNS.
Funded by NSF's Computing Research Infrastructure program.

Campus Traffic



Campus Bro Cluster

shire

Name	Type	Node	Pid	Proc	VSize	Rss	Cpu	Cmd
manager	manager	marge	56355	parent	131M	113M	35%	bro
proxy-1	proxy	marge	56417	parent	38M	20M	0%	bro
worker-00	worker	blade0	68859	parent	320M	294M	25%	bro
worker-01	worker	blade1	5069	parent	330M	314M	27%	bro
worker-02	worker	blade2	40406	parent	330M	308M	25%	bro
worker-03	worker	blade3	39416	parent	322M	306M	32%	bro
worker-04	worker	blade4	74771	parent	322M	294M	38%	bro
worker-05	worker	blade5	9257	parent	324M	294M	21%	bro
worker-06	worker	blade6	1915	parent	322M	293M	26%	bro
worker-07	worker	blade7	36112	parent	324M	304M	28%	bro
worker-08	worker	blade8	13415	parent	332M	299M	29%	bro
worker-09	worker	blade9	1391	parent	326M	299M	22%	bro
worker-10	worker	blade10	86072	parent	324M	308M	35%	bro
worker-11	worker	blade11	886	parent	326M	309M	32%	bro
worker-12	worker	blade12	89545	parent	328M	310M	29%	bro
worker-13	worker	blade13	5854	parent	342M	319M	27%	bro
worker-14	worker	blade14	24363	parent	324M	302M	26%	bro
worker-15	worker	blade15	49775	parent	324M	308M	23%	bro
worker-16	worker	blade16	433	parent	328M	300M	35%	bro
worker-17	worker	blade17	87068	parent	330M	308M	42%	bro
worker-18	worker	blade18	434	parent	328M	296M	21%	bro
worker-19	worker	blade19	89307	parent	330M	314M	22%	bro
worker-21	worker	blade21	64289	parent	344M	315M	29%	bro
worker-22	worker	blade22	21382	parent	324M	309M	21%	bro
worker-23	worker	blade23	1878	parent	324M	300M	22%	bro
worker-25	worker	blade25	59300	parent	336M	313M	19%	bro
worker-27	worker	blade27	1563	parent	352M	332M	25%	bro

What Does This Mean for Us?

What Does This Mean for Us?

Operations is using our technology.

LBNL has 3 operational Bro Clusters w/ 15 backends.

Other sites have, or are building, similar setups.

New research *platforms* for us.

Research cluster at LBNL w/12 backends.

Research cluster at UC Berkeley w/ 28 backends.

New research directions.

1. Going beyond 10GE ...

Next Stop: 100 Gb/s

DOE/ESNet
100G Advanced Networking Initiative

NEWS CENTER

Contact Us | Biology for Energy and Health | Climate + Environment | Computing | Energy | Physics +

Moving Data at the Speed of Science: Berkeley Lab Lays Foundation for 100 Gbps Prototype Network

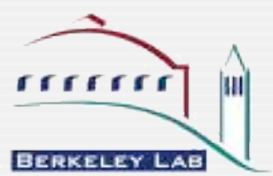
JULY 13, 2011

Source: ESNet

Source: ESNet

100 Gb/s Load-balancer

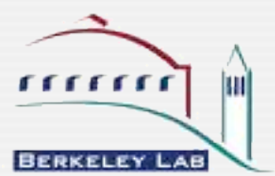
100 Gb/s Load-balancer



100 Gb/s Load-balancer

SBIR Phase 2 project.

Just awarded to cPacket & ICSI.

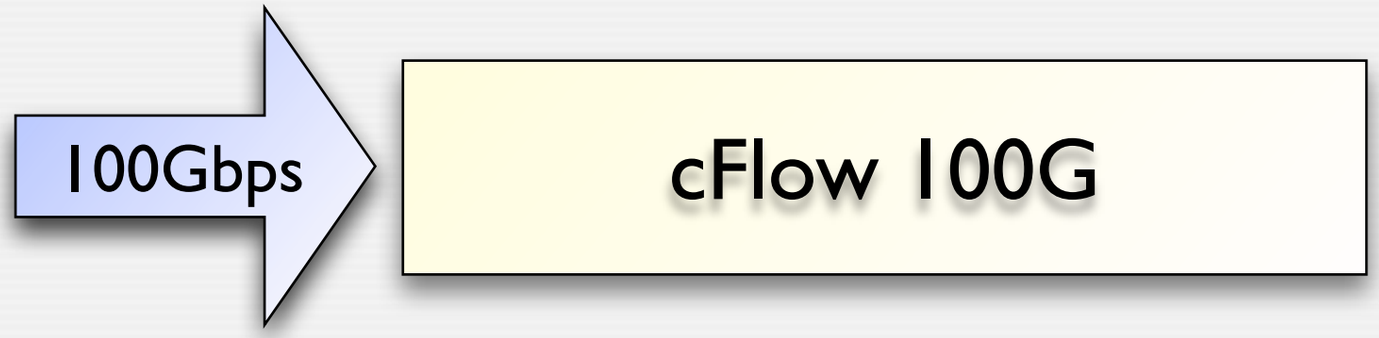


100 Gb/s Load-balancer

U.S. DEPARTMENT OF ENERGY | Office of Science

SBIR Phase 2 project.

Just awarded to cPacket & ICSI.



ICSI International Computer Science Institute

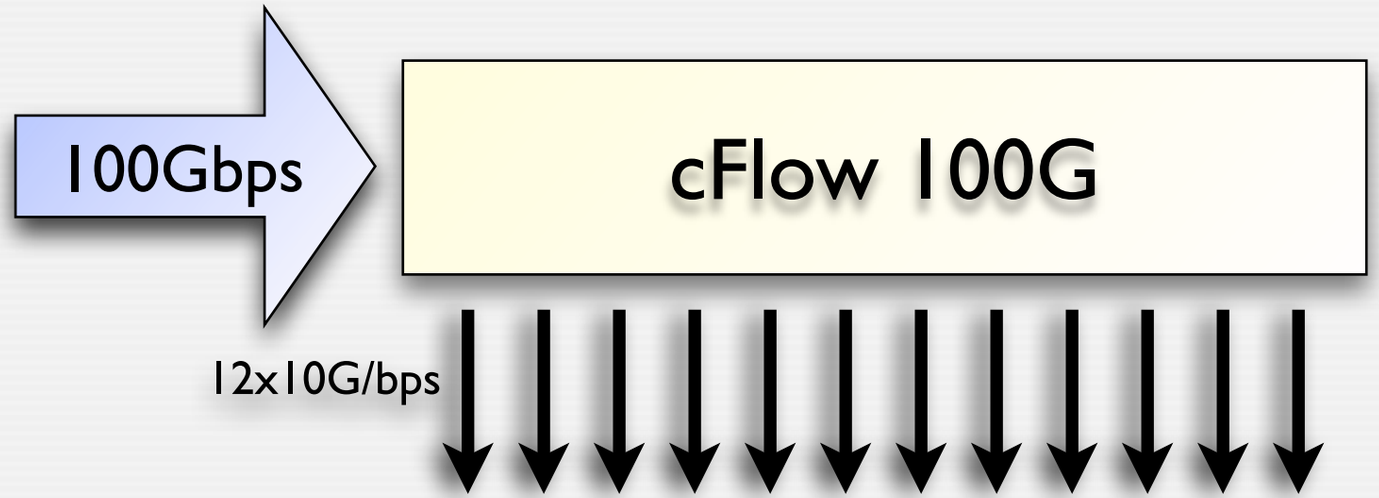
Berkeley Lab

100 Gb/s Load-balancer

U.S. DEPARTMENT OF ENERGY | Office of Science

SBIR Phase 2 project.

Just awarded to cPacket & ICSI.



ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

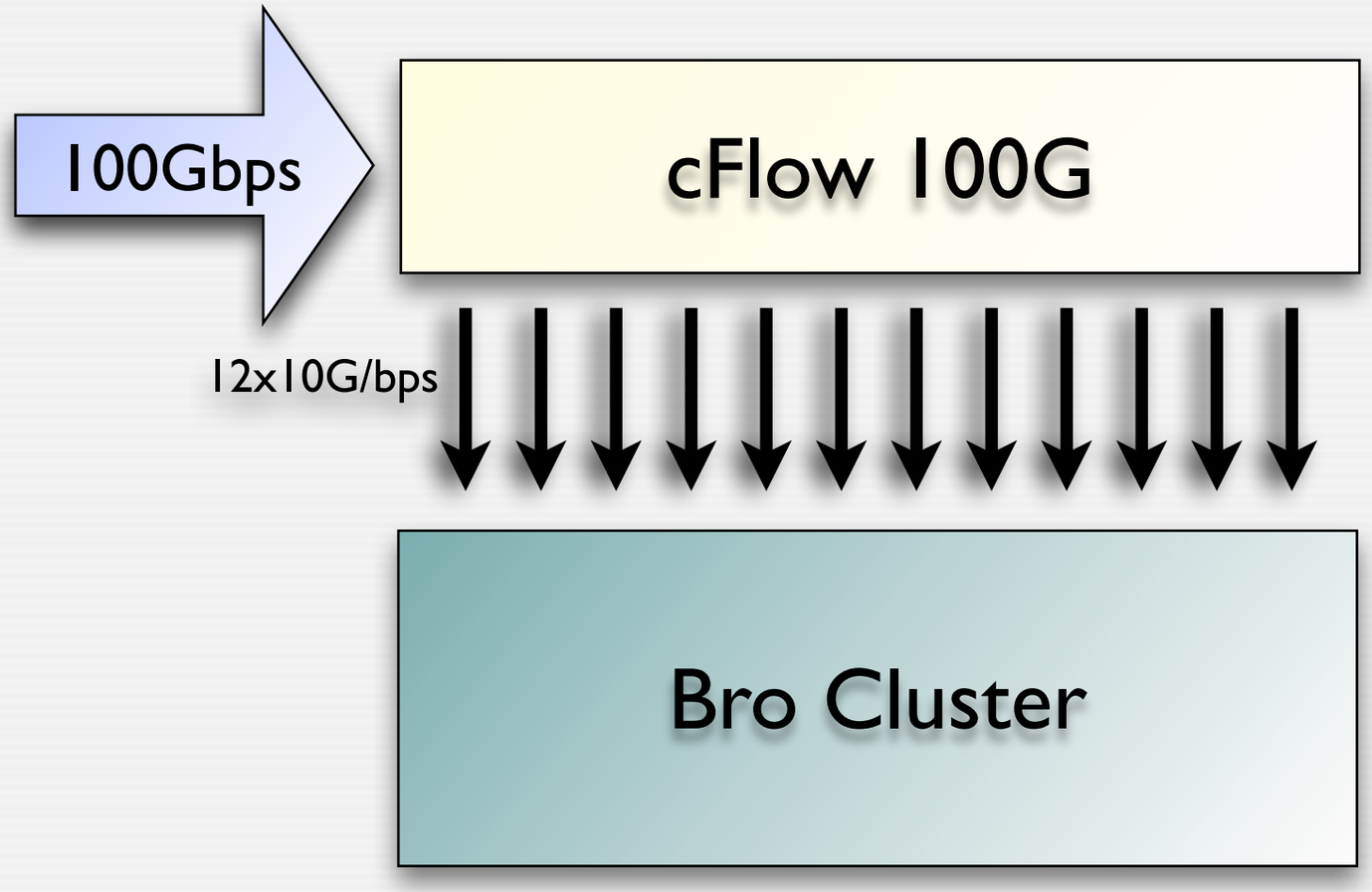
BERKELEY LAB

100 Gb/s Load-balancer

U.S. DEPARTMENT OF ENERGY | Office of Science

SBIR Phase 2 project.

Just awarded to cPacket & ICSI.



ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

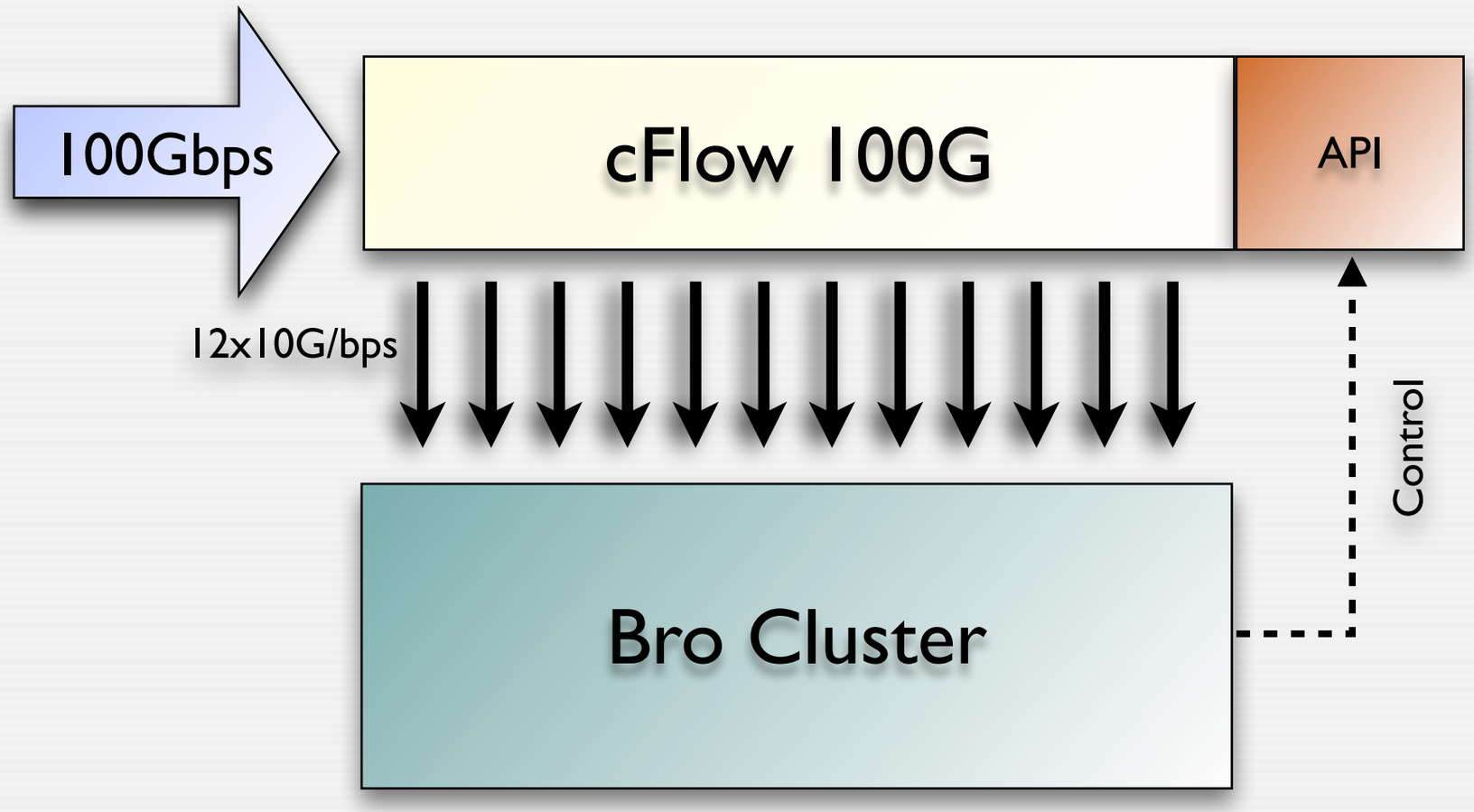
BERKELEY LAB

100 Gb/s Load-balancer

U.S. DEPARTMENT OF ENERGY | Office of Science

SBIR Phase 2 project.

Just awarded to cPacket & ICSI.



ICSI

BERKELEY LAB

What Does This Mean for Us?

Operations is using our technology.

LBNL has 3 operational Bro Clusters w/ 15 backends.

Other sites have, or are building, similar setups.

New research *platforms* for us.

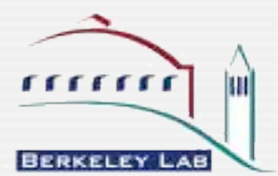
Research cluster at LBNL w/12 backends.

Research cluster at UC Berkeley w/ 28 backends.

New research directions.

1. Going beyond 10GE ...
2. **From multi-system to multi-core ...**

Bro is Single-Threaded ...



Bro is Single-Threaded ...

Cluster clearly has short-comings.

- Backends have multiple cores, which are mostly idling.

Bro is Single-Threaded ...

Cluster clearly has short-comings.

Backends have multiple cores, which are mostly idling.

We *really* want multi-threading.

Must scale well with increasing numbers of cores.

Must be transparent to the operator.

Bro is Single-Threaded ...

Cluster clearly has short-comings.

Backends have multiple cores, which are mostly idling.

We *really* want multi-threading.

Must scale well with increasing numbers of cores.

Must be transparent to the operator.

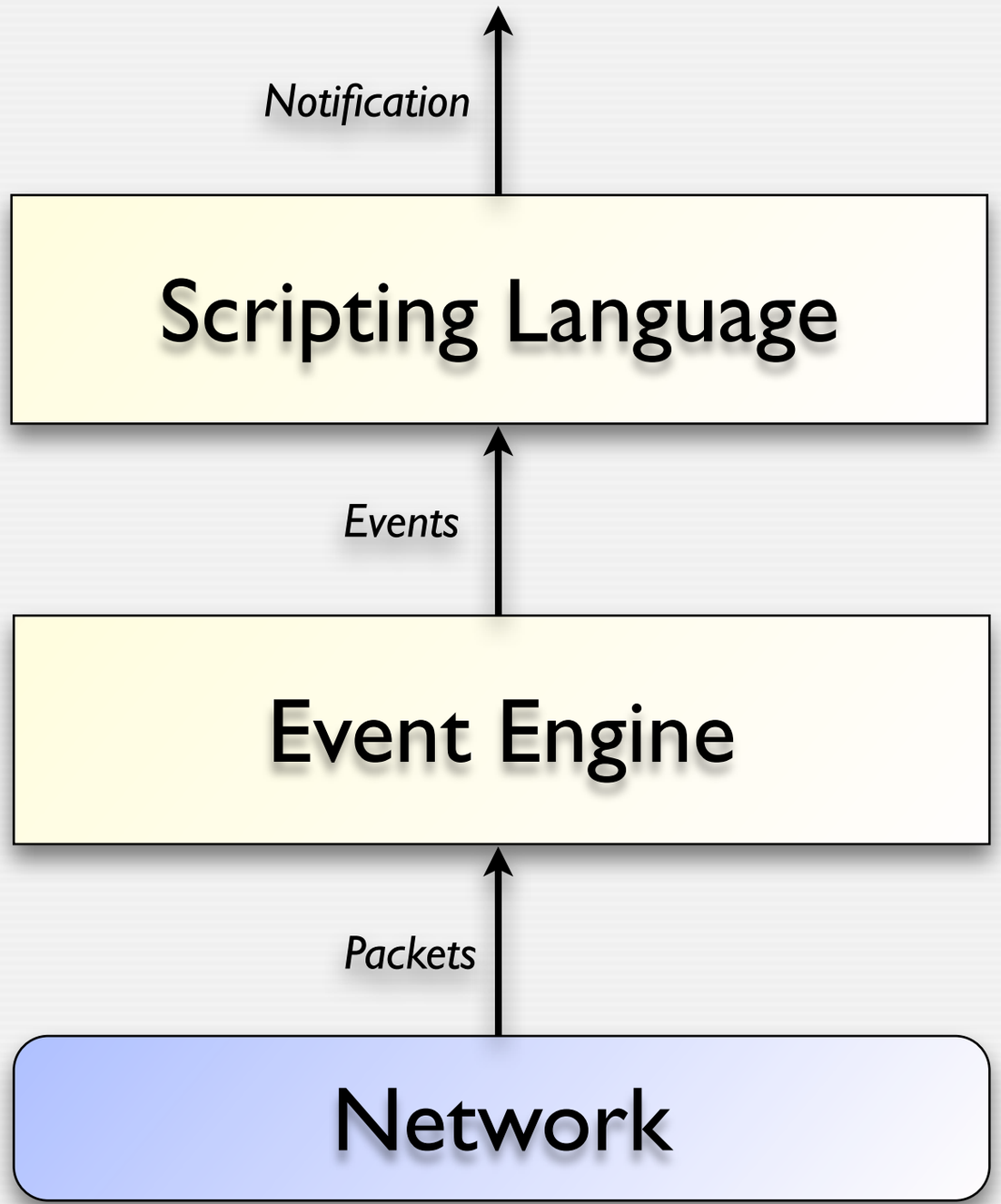
For some IDS, that's not so hard.

For others, it is ...

Bro's Architecture

Detection Logic

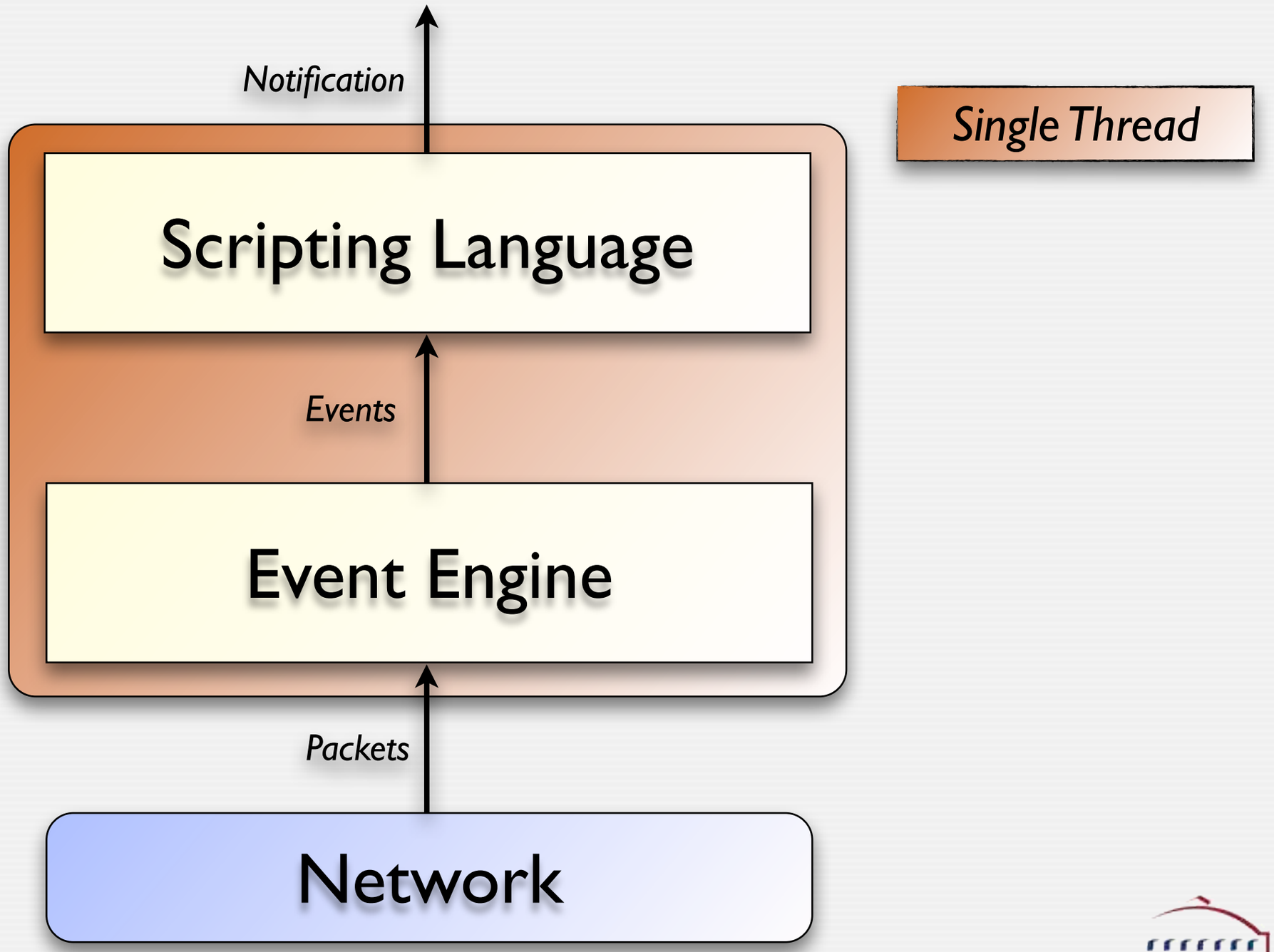
Packet Analysis



Bro's Architecture

Detection Logic

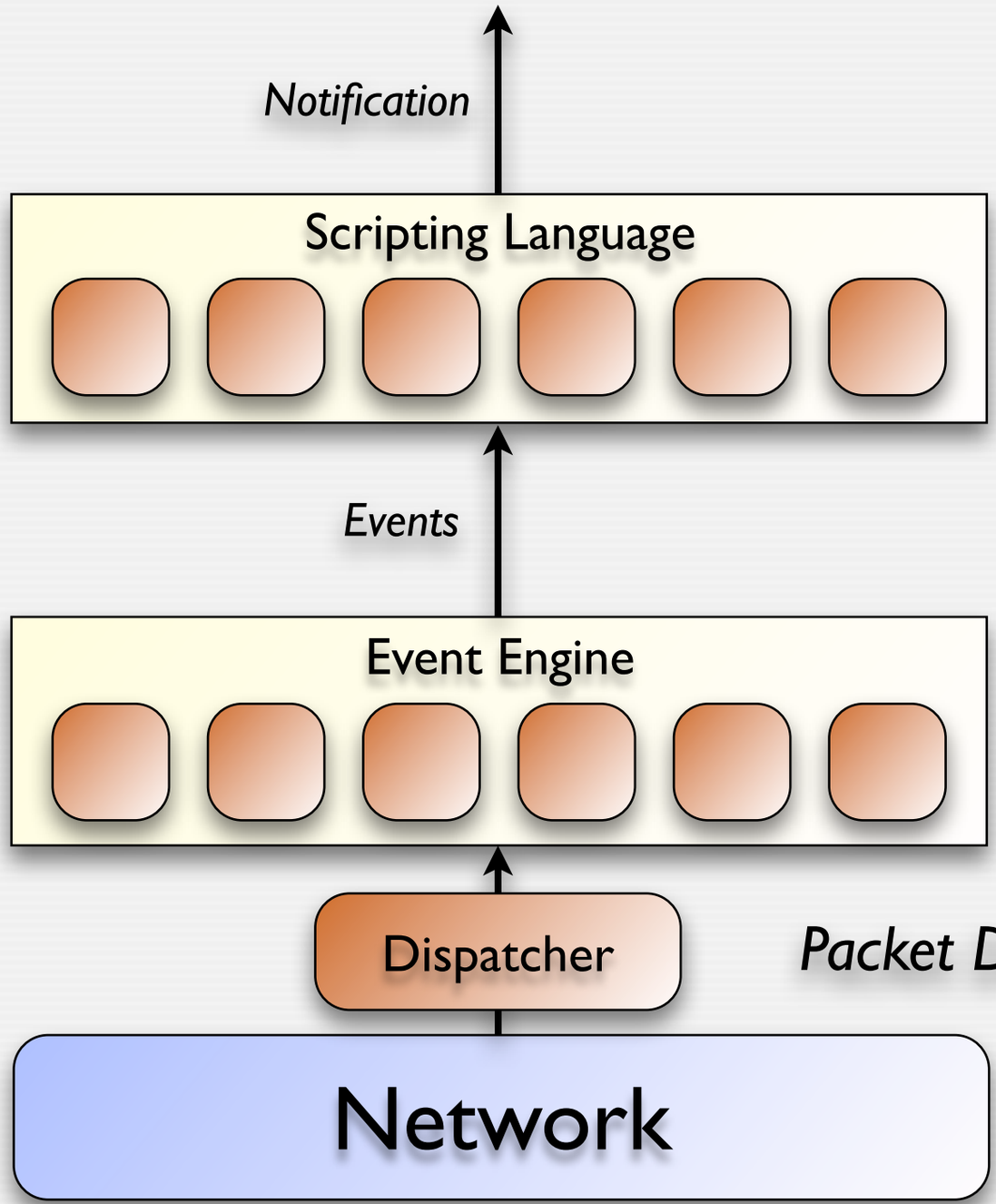
Packet Analysis



Single Thread

Bro's Architecture

Detection Logic



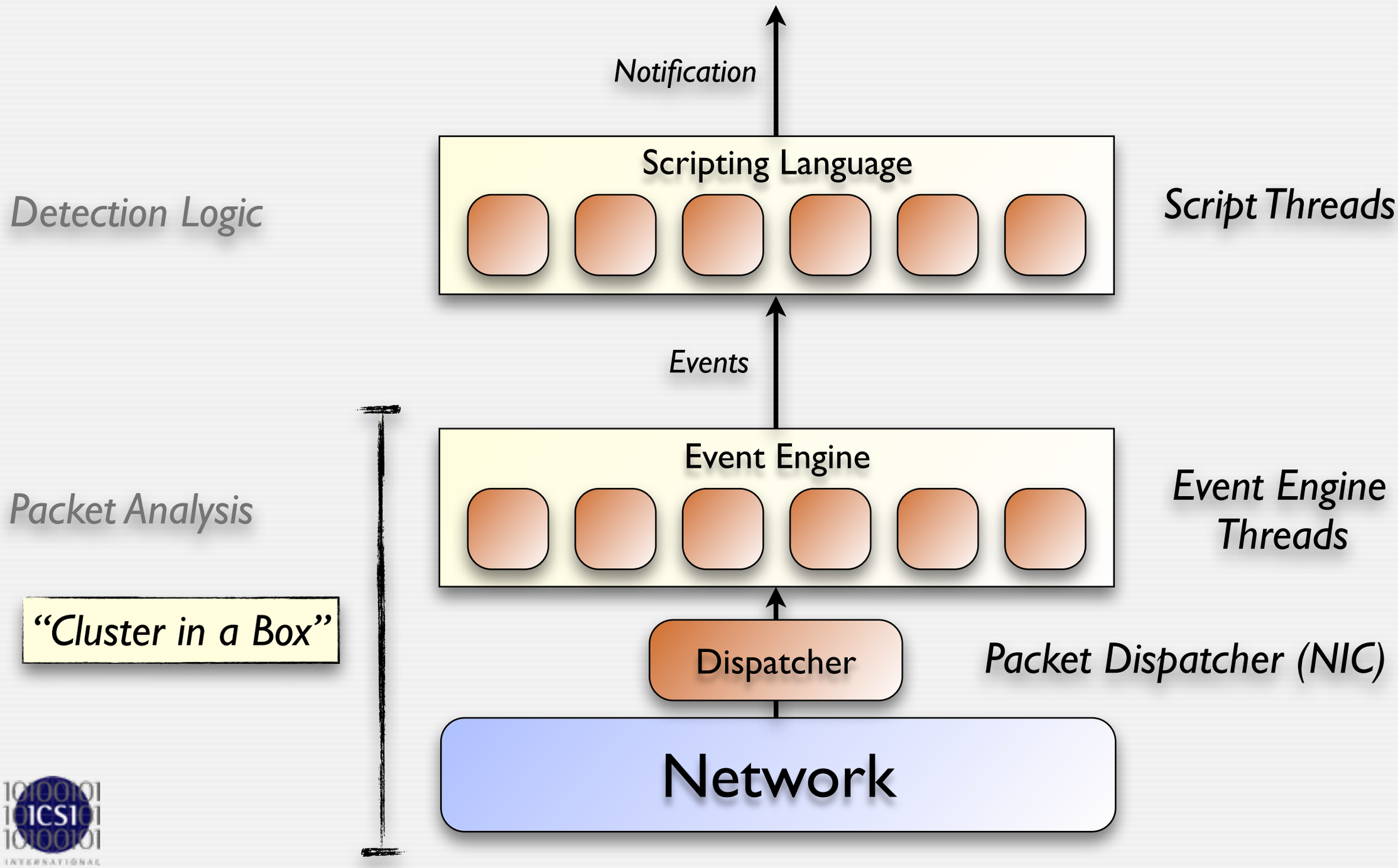
Script Threads

Event Engine Threads

Packet Dispatcher (NIC)

Packet Analysis

Bro's Architecture



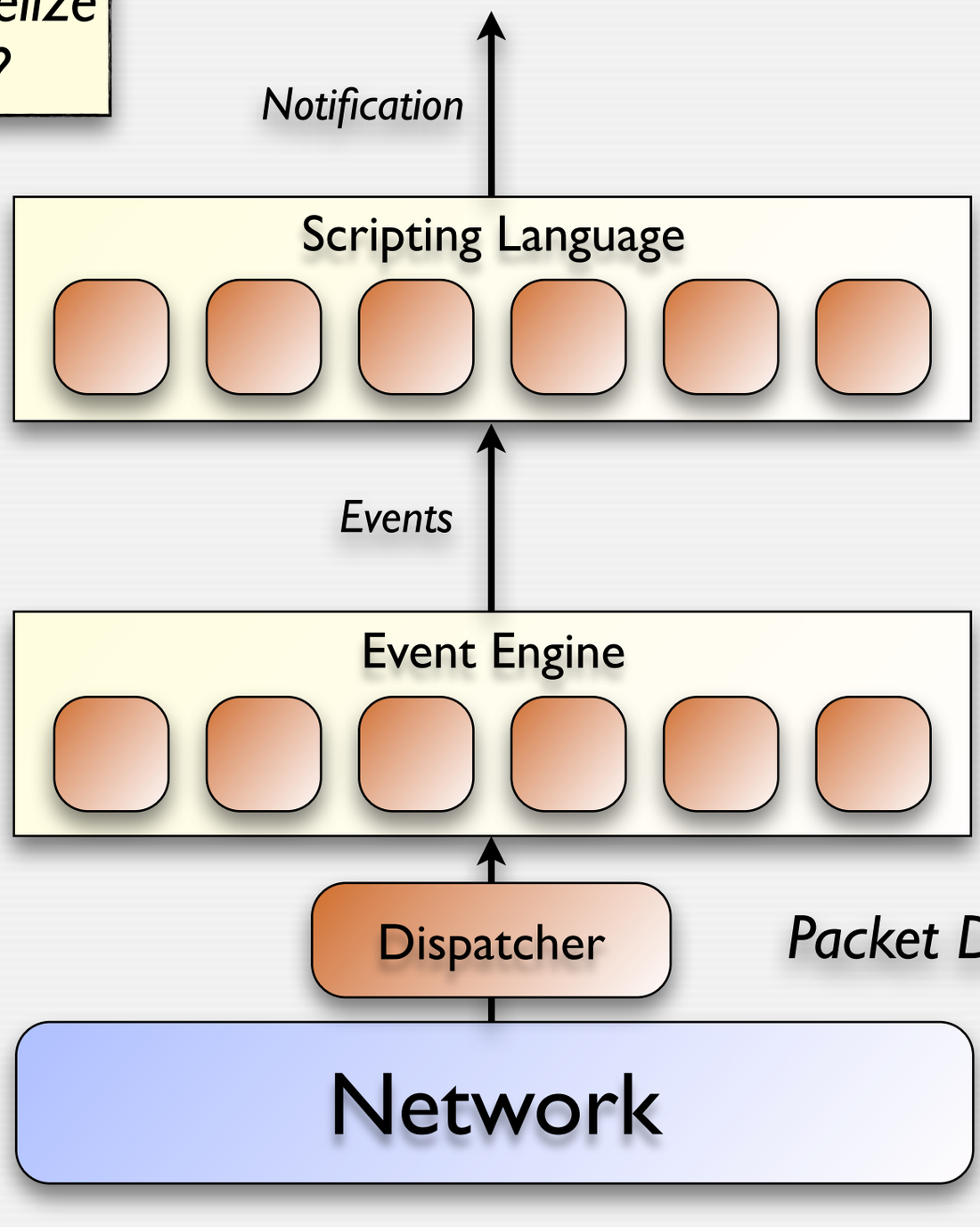
Bro's Architecture

Research: How to parallelize a scripting language?

Detection Logic

Packet Analysis

"Cluster in a Box"

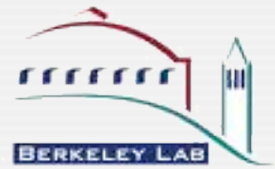


Script Threads

Event Engine Threads

Packet Dispatcher (NIC)

Towards Multi-Threaded Bro



Towards Multi-Threaded Bro

Analysis structured around *units*.

Processing and state expressed in such terms.

Towards Multi-Threaded Bro

Analysis structured around *units*.

Processing and state expressed in such terms.

Can leverage units for scheduling.

Run-time tracks current unit.

Scheduler steers events to thread in charge.

Towards Multi-Threaded Bro

Analysis structured around *units*.

Processing and state expressed in such terms.

Can leverage units for scheduling.

Run-time tracks current unit.

Scheduler steers events to thread in charge.

Simulations predict excellent scalability.

There are plenty independent unit instances in traffic.

Towards Multi-Threaded Bro

Analysis structured around *units*.

Processing and state expressed in such terms.

Can leverage units for scheduling.

Run-time tracks current unit.

Scheduler steers events to thread in charge.

Simulations predict excellent scalability.

There are plenty independent unit instances in traffic.

Implemented a prototype inside Bro

Works somewhat, but painful.

HILTI Abstract Machine

A **H**igh-Level **I**ntermediary **L**anguage for **T**raffic **I**nspection

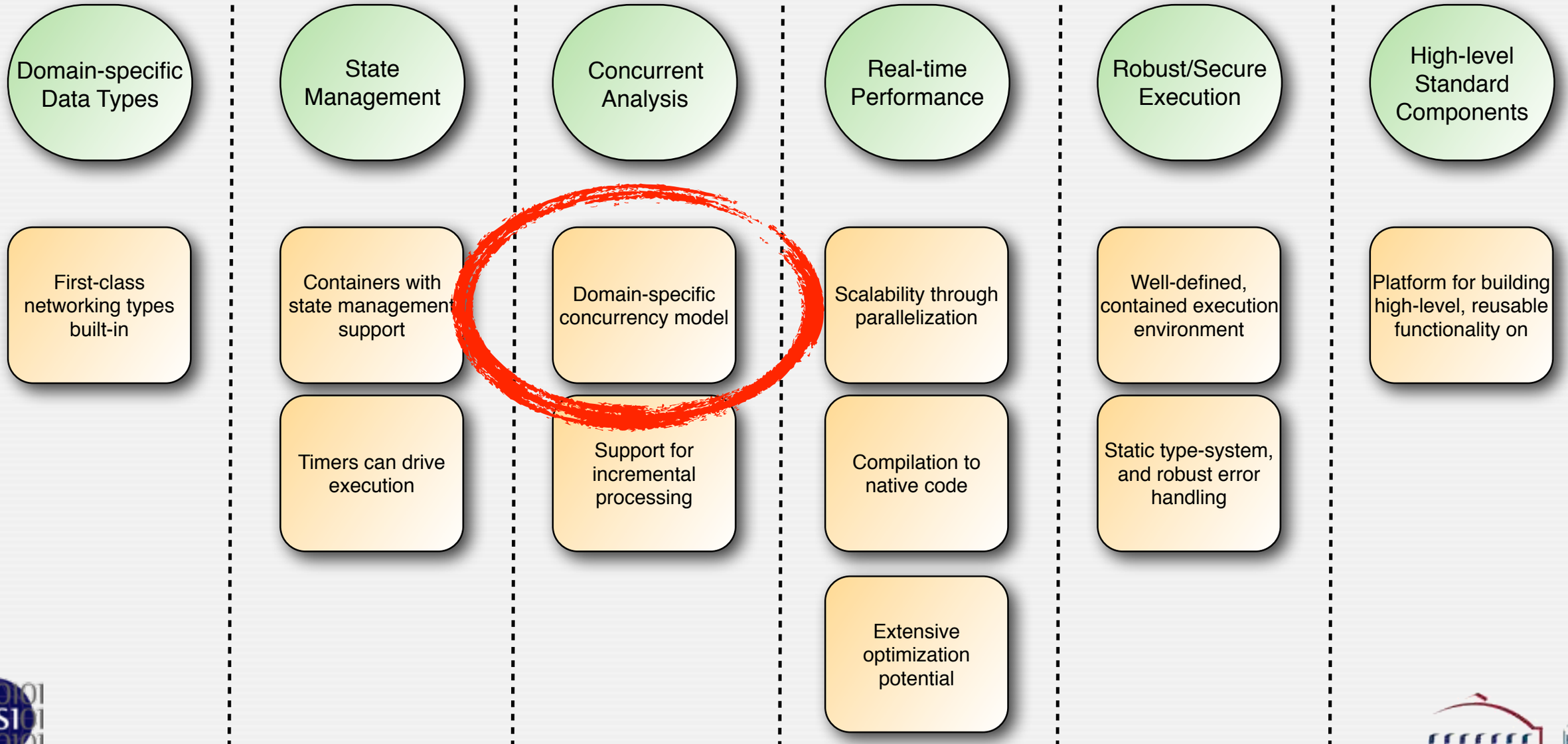
HILTI Abstract Machine

A **H**igh-Level **I**ntermediary **L**anguage for **T**raffic **I**nspection

Domain-specific Data Types	State Management	Concurrent Analysis	Real-time Performance	Robust/Secure Execution	High-level Standard Components
First-class networking types built-in	Containers with state management support	Domain-specific concurrency model	Scalability through parallelization	Well-defined, contained execution environment	Platform for building high-level, reusable functionality on
	Timers can drive execution	Support for incremental processing	Compilation to native code	Static type-system, and robust error handling	
			Extensive optimization potential		

HILTI Abstract Machine

A **H**igh-Level **I**ntermediary **L**anguage for **T**raffic **I**nspection



HILTI Scheduling Example

HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context   (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule  http_request ("/index.html")
schedule  http_request ("/robots.txt")
schedule  connection_attempt (10.1.1.1)
```

HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```

HILTI Scheduling Example

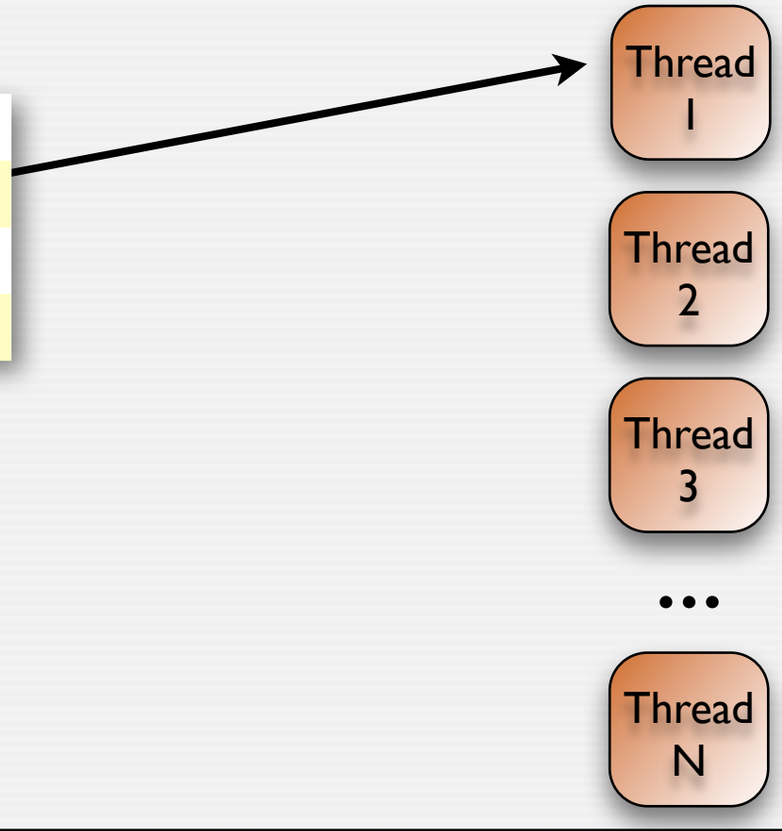
```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```



HILTI Scheduling Example

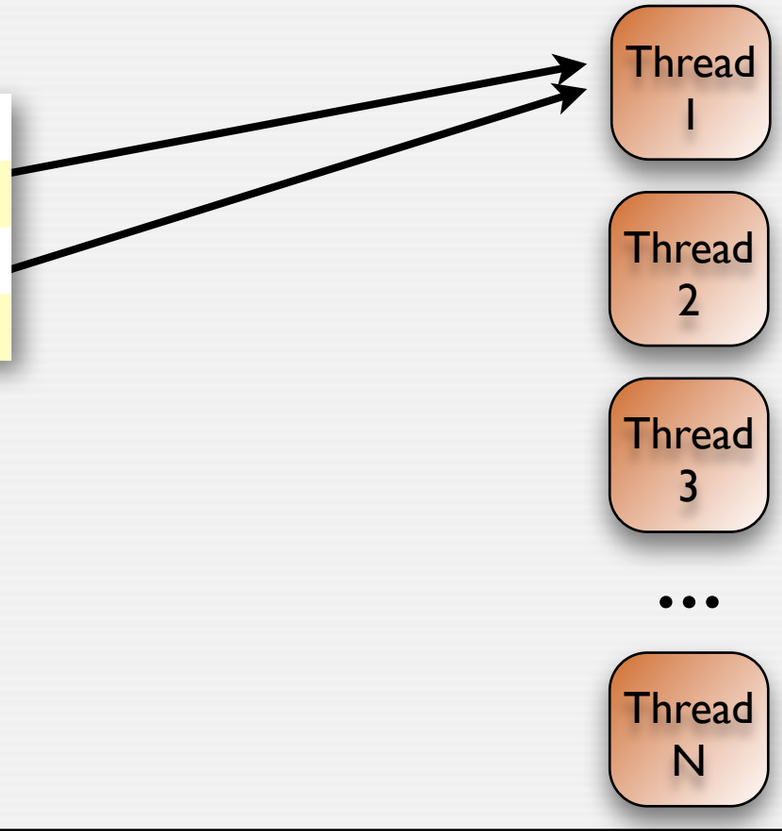
```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```



HILTI Scheduling Example

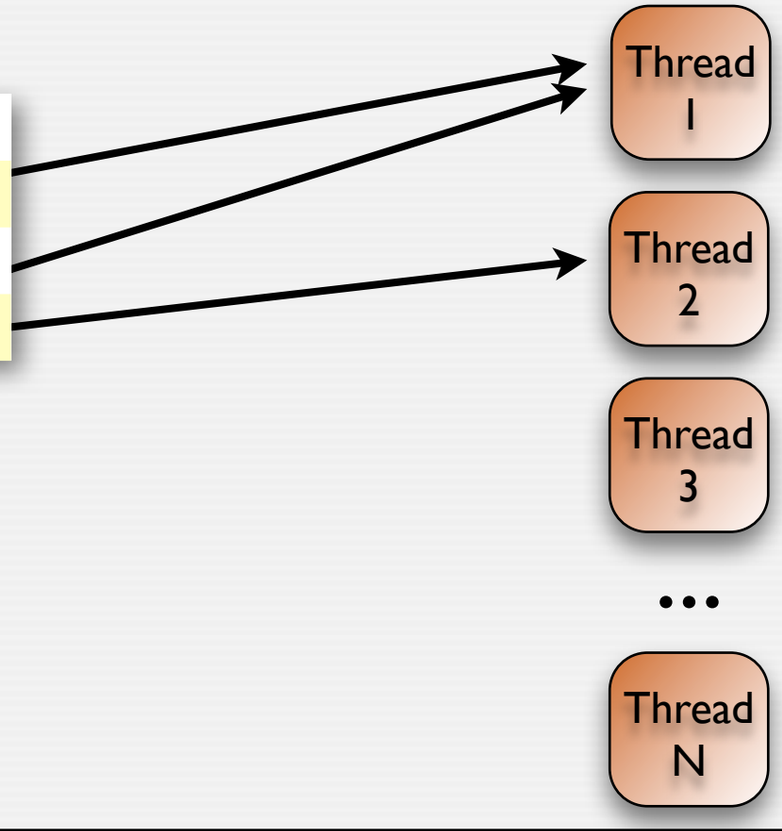
```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```



HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

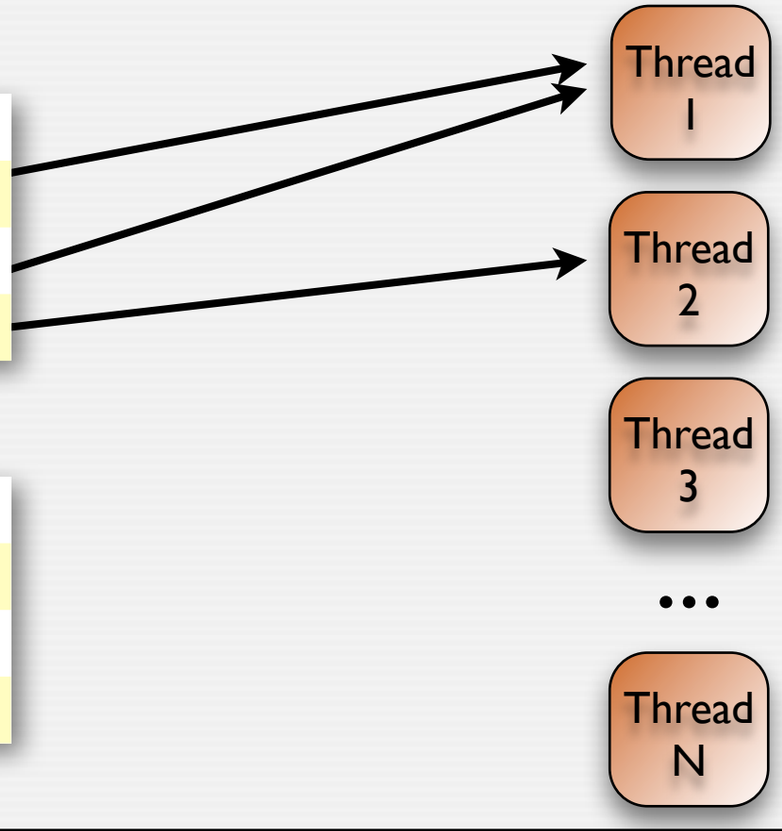
```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```

Packet 2

```
context    (10.1.1.1, 1235/tcp, 10.9.9.9, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```



HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

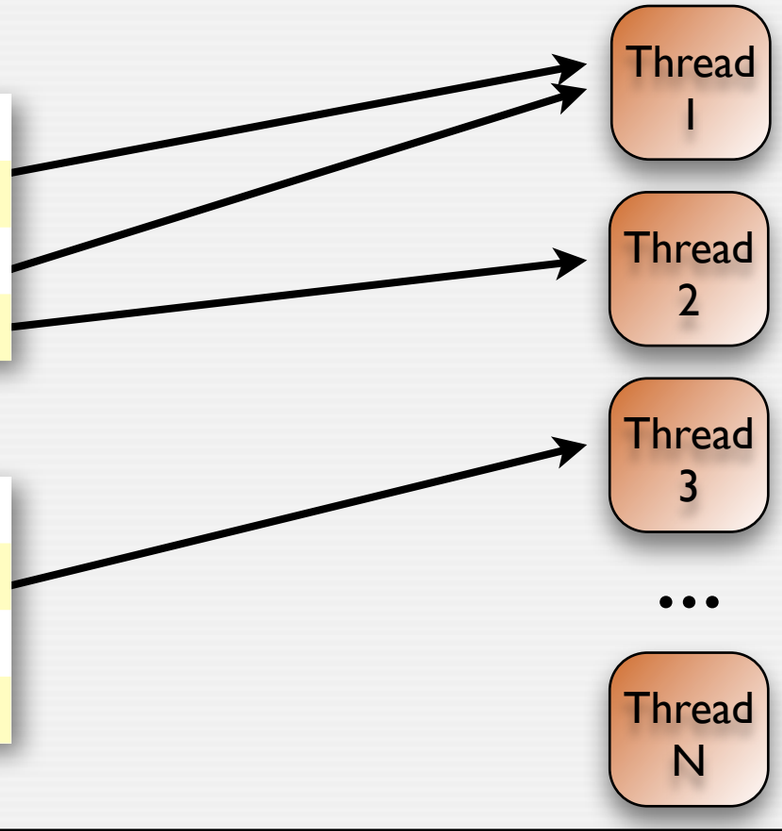
```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```

Packet 2

```
context    (10.1.1.1, 1235/tcp, 10.9.9.9, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```



HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

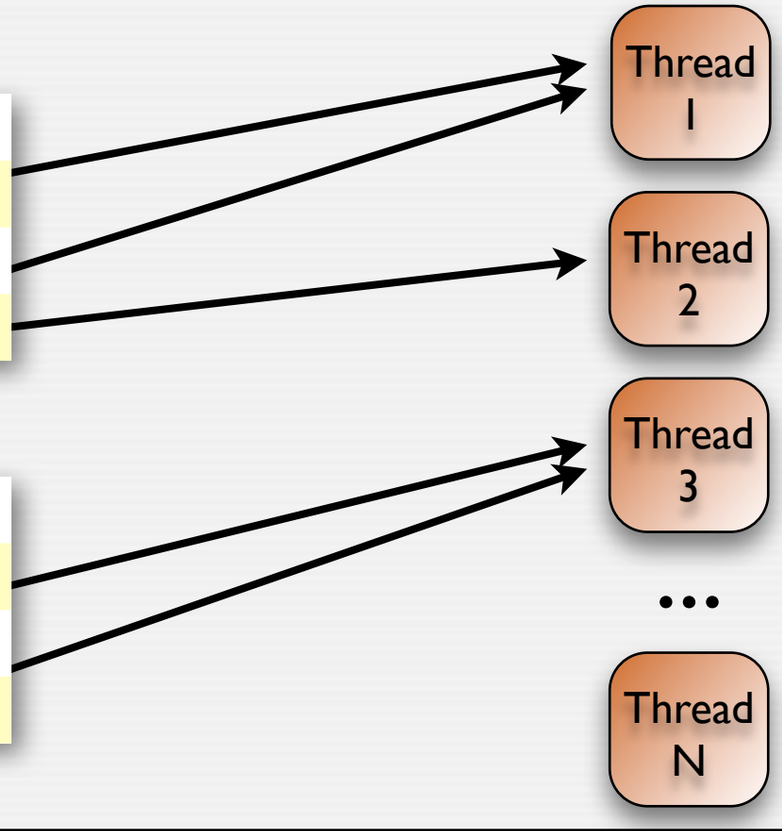
```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```

Packet 2

```
context    (10.1.1.1, 1235/tcp, 10.9.9.9, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```



HILTI Scheduling Example

```
# A function scheduled on a per-flow basis.
scope Flow = { saddr, sport, daddr, dport }

void http_request(bytes url) &scope=Flow
{
     bool isbad = regexp.match /foo=[aA]ttack/ url
     [.. alarm if match found ..]
}
```

(Syntax simplified.)

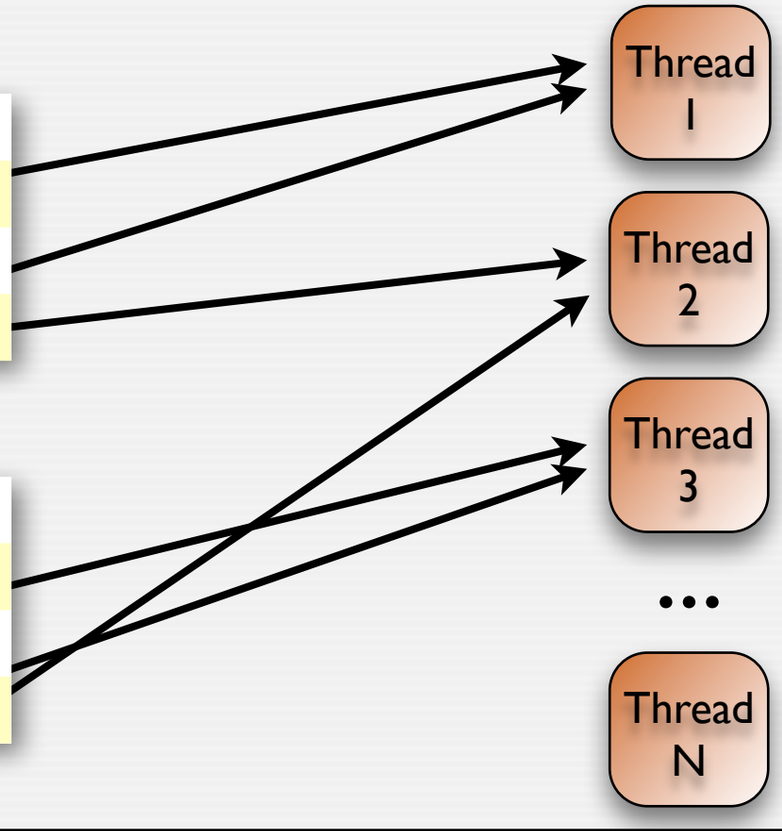
```
# A function scheduled on a per-source basis.
scope Source = { saddr }

map<addr, int> conn_attempts   # Thread-local map.

void connection_attempt(addr host) &scope=Source
{
     int cnt = map.get_default conn_attempts host 0
     cnt = incr cnt
     map.set conn_attempts host cnt
     [ ... alarm if threshold reached ... ]
}
```

Packet 1

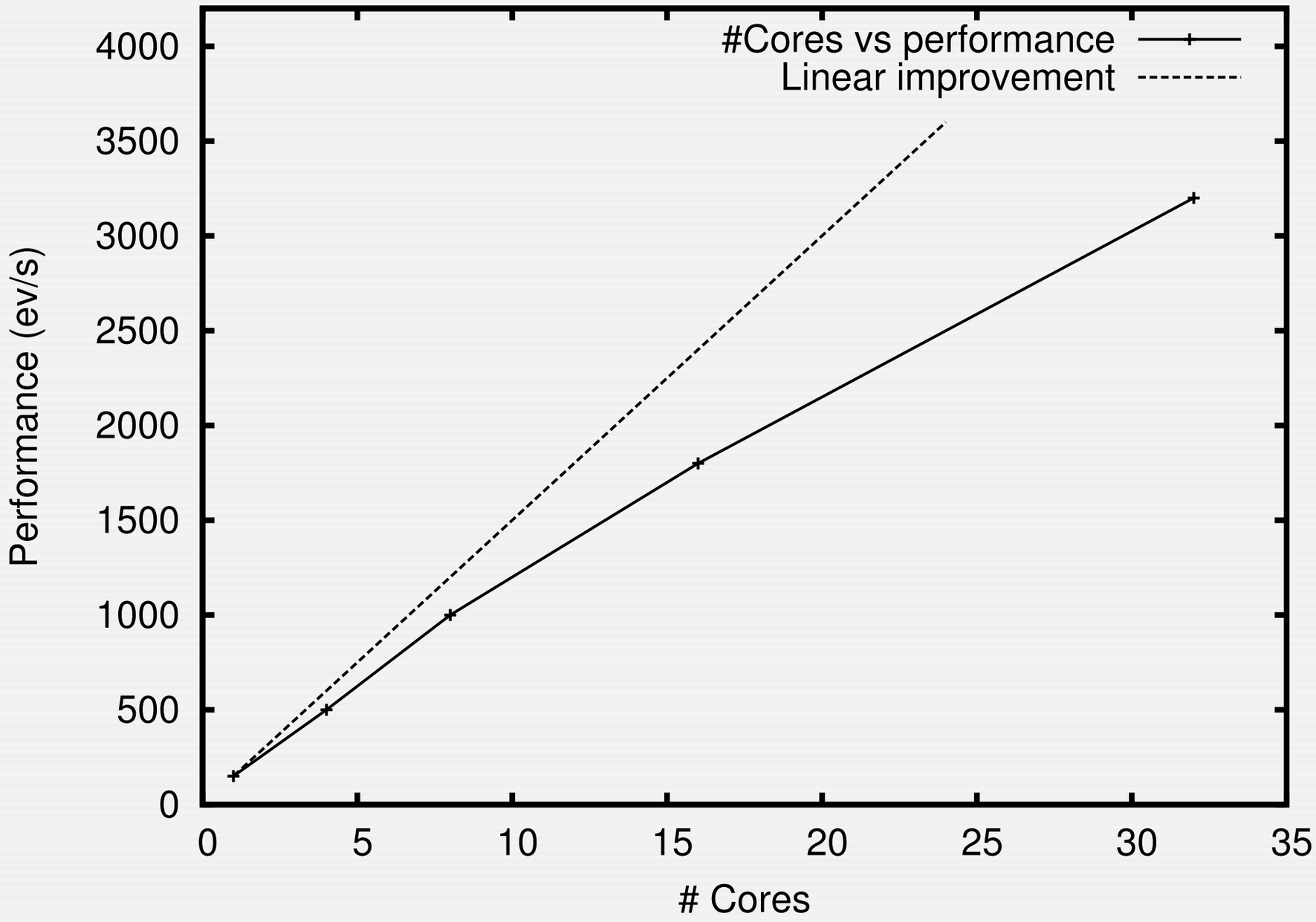
```
context    (10.1.1.1, 1234/tcp, 10.2.2.2, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```

Packet 2

```
context    (10.1.1.1, 1235/tcp, 10.9.9.9, 80/tcp)
schedule   http_request ("/index.html")
schedule   http_request ("/robots.txt")
schedule   connection_attempt (10.1.1.1)
```

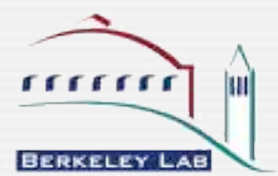


Simulation on Tilera CPU



15s of UC Berkeley campus TCP traffic, 1M event groups.
TileExpress-Pro20G, 64 cores (32 for simulation), 4GB RAM.

HILTI Prototype



HILTI Prototype

Built a prototype HILTI environment.

- Implements all of HILTI's functionality.
- Python-based, with LLVM as backend.

HILTI Prototype

Built a prototype HILTI environment.

- Implements all of HILTI's functionality.
- Python-based, with LLVM as backend.

Ported a number of host applications.

- Bro script compiler.
- BinPAC parser generator.

HILTI Prototype

Built a prototype HILTI environment.

- Implements all of HILTI's functionality.
- Python-based, with LLVM as backend.

Ported a number of host applications.

- Bro script compiler.
- BinPAC parser generator.

Aiming for production-state compiler

- Performance currently limited.
- Needs engineering effort to address bottlenecks.

Bro: The Curse of Success

Bro: The Curse of Success

Bro: The Curse of Success

Success can be problematic with this kind of research.

- Bro is now used operationally by many sites.
- Demands of operational user community hard to meet for small team.

Bro: The Curse of Success

Success can be problematic with this kind of research.

- Bro is now used operationally by many sites.
- Demands of operational user community hard to meet for small team.

Aiming to establish sustainable development model.

Modernize the system to make usage and contributions easier.
Develop a community around the project.

Bro: The Curse of Success

Success can be problematic with this kind of research.

- Bro is now used operationally by many sites.
- Demands of operational user community hard to meet for small team.

Aiming to establish sustainable development model.

- Modernize the system to make usage and contributions easier.
- Develop a community around the project.

NSF supports work through a 3-year *engineering* grant.

- Bro changed a lot over the last year.
- Collaboration with National Center for Supercomputing Applications.

Target: Blue Waters @ NCSA

Target: Blue Waters @ NCSA

10 PF/s peak performance

>1 PF/s sustained on applications
>300,000 cores
>1 Petabyte memory
>10 Petabyte disk storage
>0.5 Exabyte archival storage

Hosted in 88,000-square-foot facility

Target: NCSA

Blue Waters Update—NCSA/IBM Joint Statement

Effective August 6, 2011, **IBM terminated its contract with the University of Illinois** to provide the supercomputer for the National Center for Supercomputing Applications' Blue Waters project.

NCSA is confident that its goal of building a sustained-petascale supercomputer remains achievable in a timely manner. **NCSA is coordinating with the National Science Foundation to ensure project continuity** and that the goals of the project are achieved.

Target: NCSA

Blue Waters Update—NCSA/IBM Joint Statement

Effective August 6, 2011, **IBM terminated its contract with the University of Illinois** to provide the supercomputer for the National Center for Supercomputing Applications' Blue Waters project.

NCSA is confident that its goal of building a sustained-petascale supercomputer remains achievable in a timely manner. **NCSA is coordinating with the National Science Foundation to ensure project continuity** and that the goals of the project are achieved.

NCSA is already planning for 100 Gb/s as well.

Conclusion

Conclusion

Very beneficial to work with operations.

- Puts results into practice and provides feedback.
- Yields new research capabilities.
- Opens new directions for future research.

Conclusion

Very beneficial to work with operations.

- Puts results into practice and provides feedback.
- Yields new research capabilities.
- Opens new directions for future research.

Case-study: Bro

- Primarily a research project.
- Used extensively in operations.

Conclusion

Very beneficial to work with operations.

- Puts results into practice and provides feedback.
- Yields new research capabilities.
- Opens new directions for future research.

Case-study: Bro

- Primarily a research project.
- Used extensively in operations.

Continuous exchange.

- Optimization; Cluster; New Hardware; Multi-core; HILTI.

Conclusion

Very beneficial to work with operations.

- Puts results into practice and provides feedback.
- Yields new research capabilities.
- Opens new directions for future research.

Case-study: Bro

- Primarily a research project.
- Used extensively in operations.

Continuous exchange.

- Optimization; Cluster; New Hardware; Multi-core; HILTI.

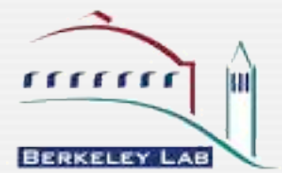
Making results usable costs additional effort.

- Lots of work that does not go into any research paper.
- Separate funding for such efforts can be extremely helpful.

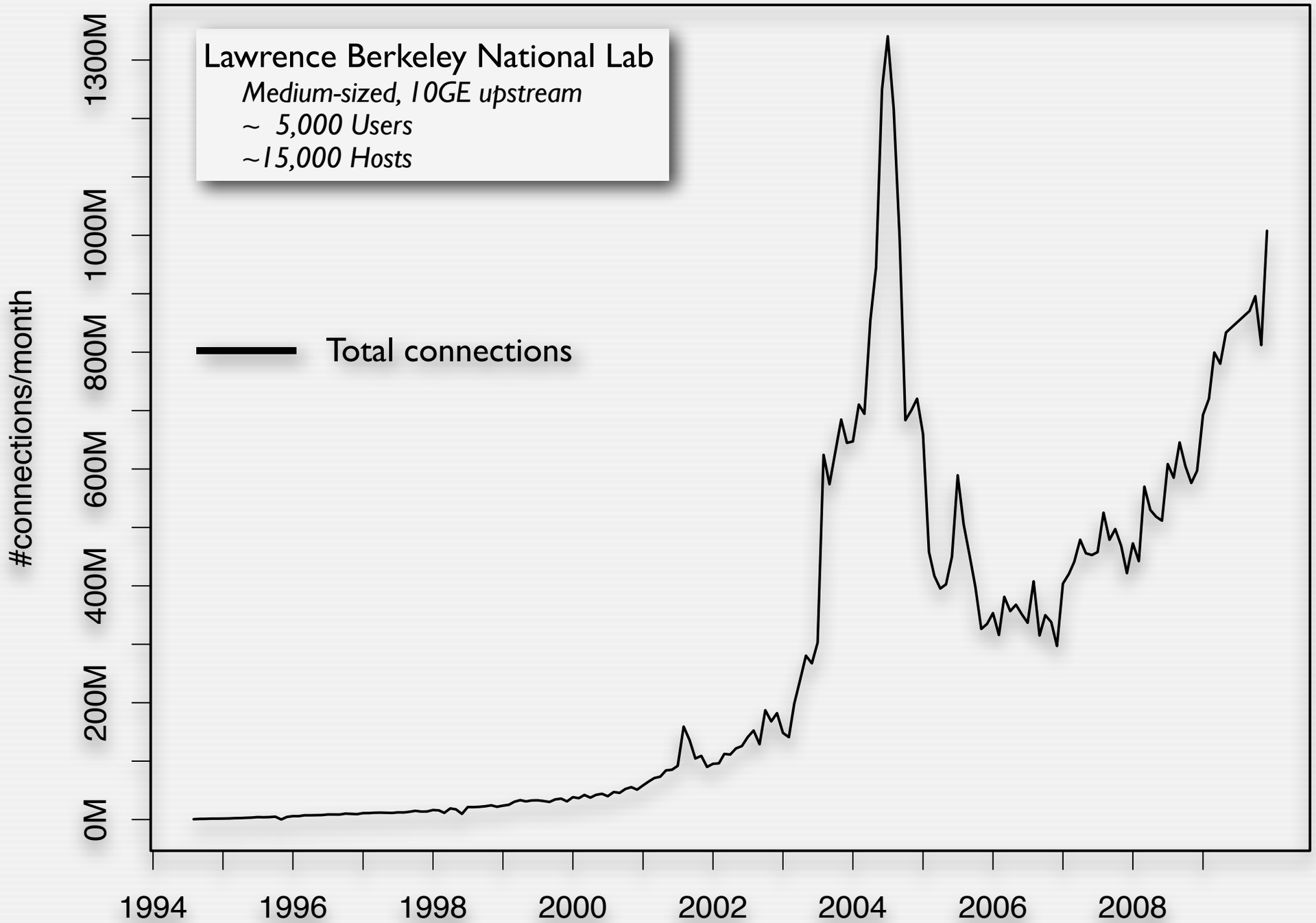
Thanks for your attention.

Robin Sommer
International Computer Science Institute, &
Lawrence Berkeley National Laboratory

robin@icsi.berkeley.edu
http://www.icir.org

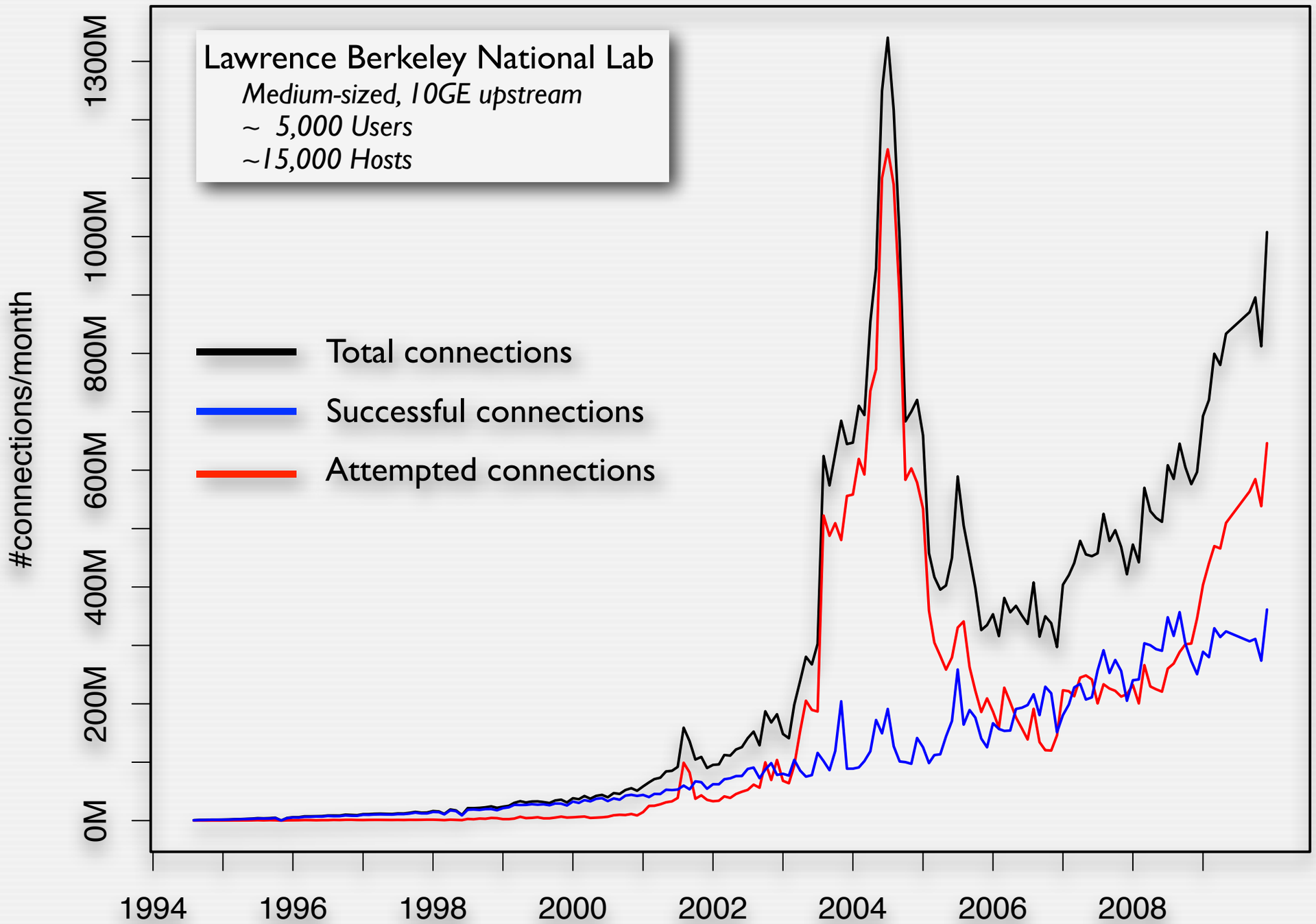


Internet Traffic: Connections



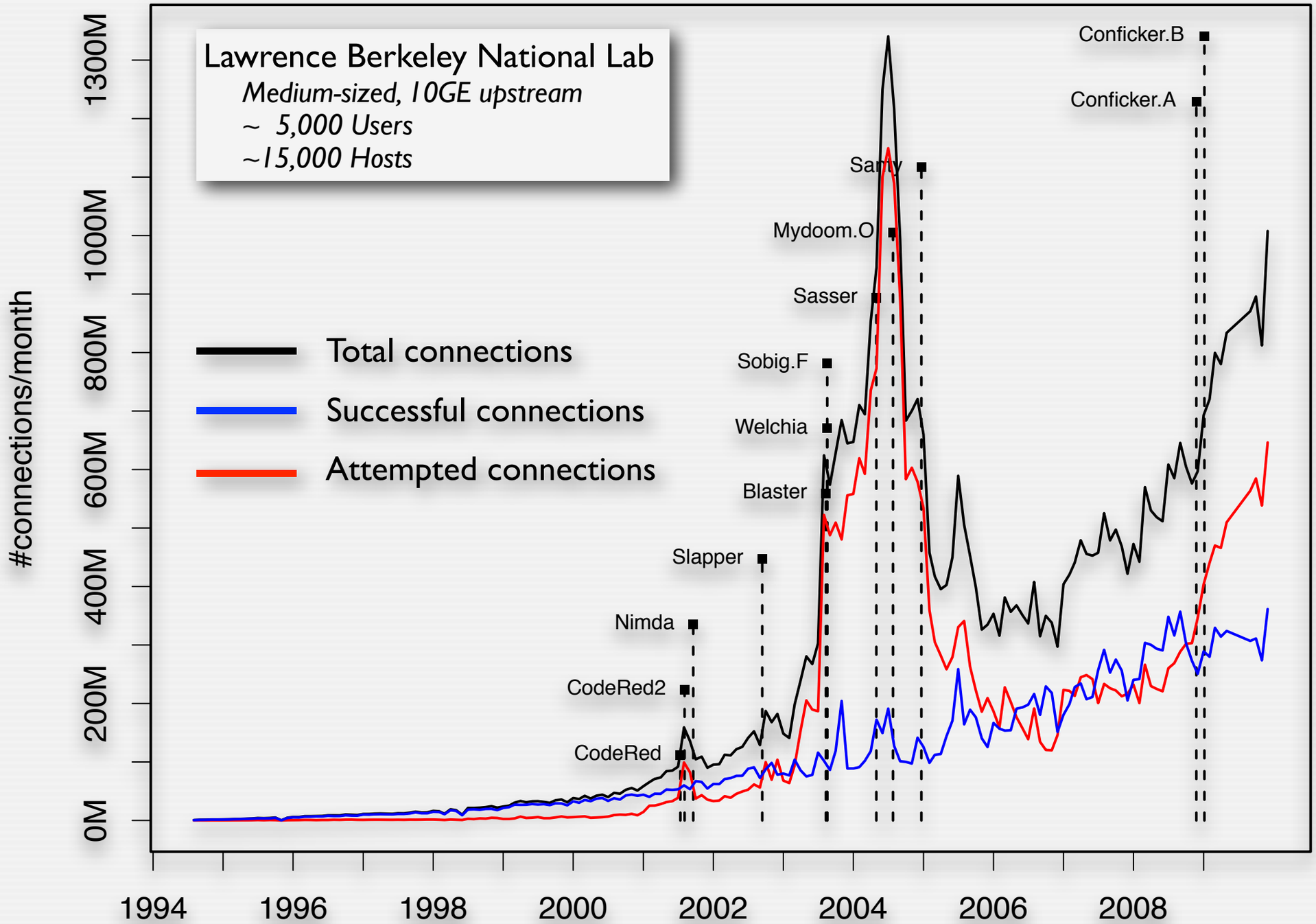
Data: Lawrence Berkeley National Lab

Internet Traffic: Connections



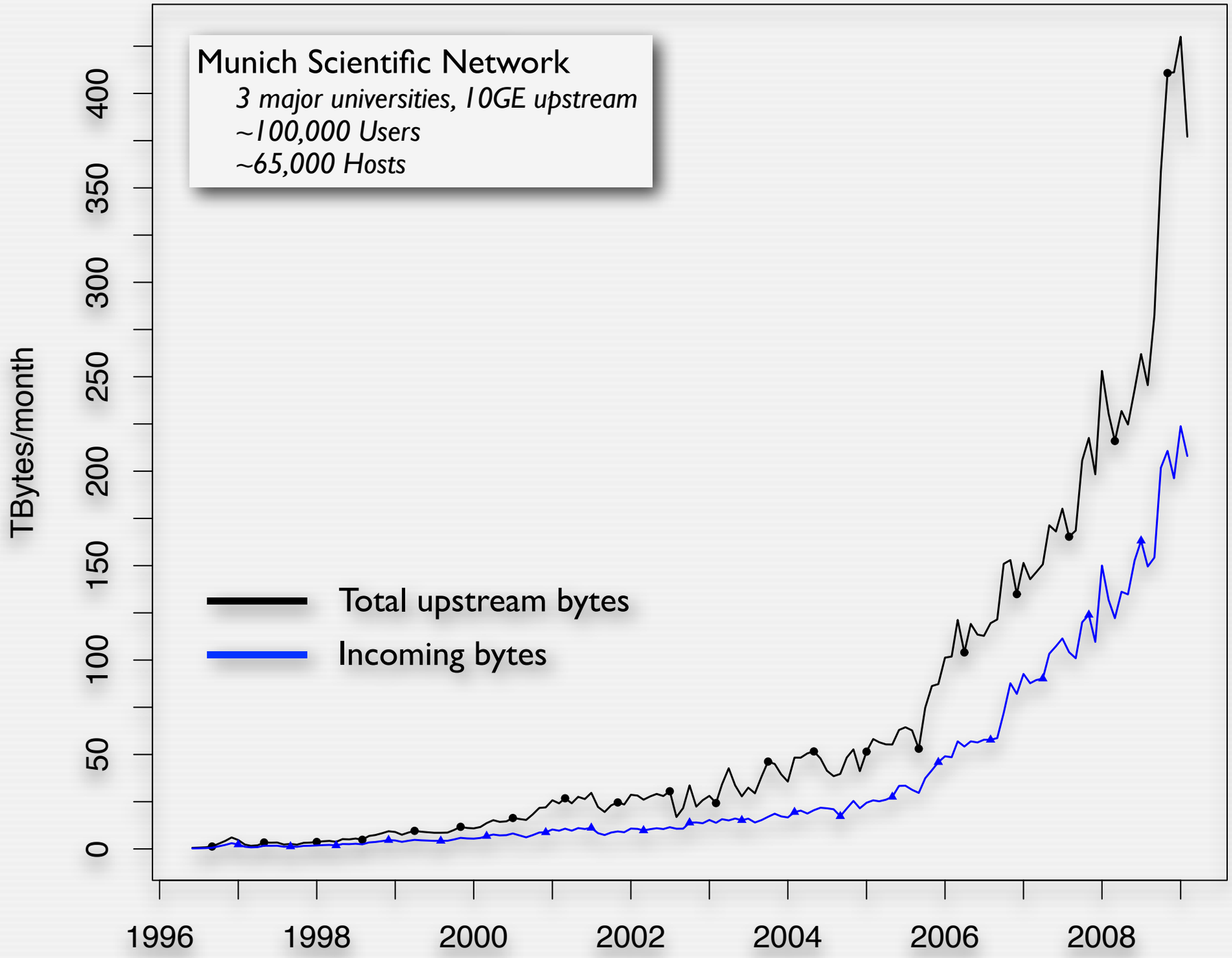
Data: Lawrence Berkeley National Lab

Internet Traffic: Connections



Data: Lawrence Berkeley National Lab

Development of Internet Traffic



Data: Leibniz-Rechenzentrum, München