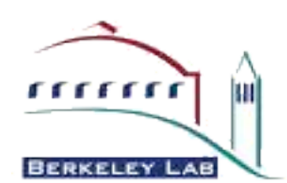
# *Network Security Today:*
# Finding Complex Attacks at 100Gb/s


Robin Sommer

International Computer Science Institute, &
Lawrence Berkeley National Laboratory

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





# Outline

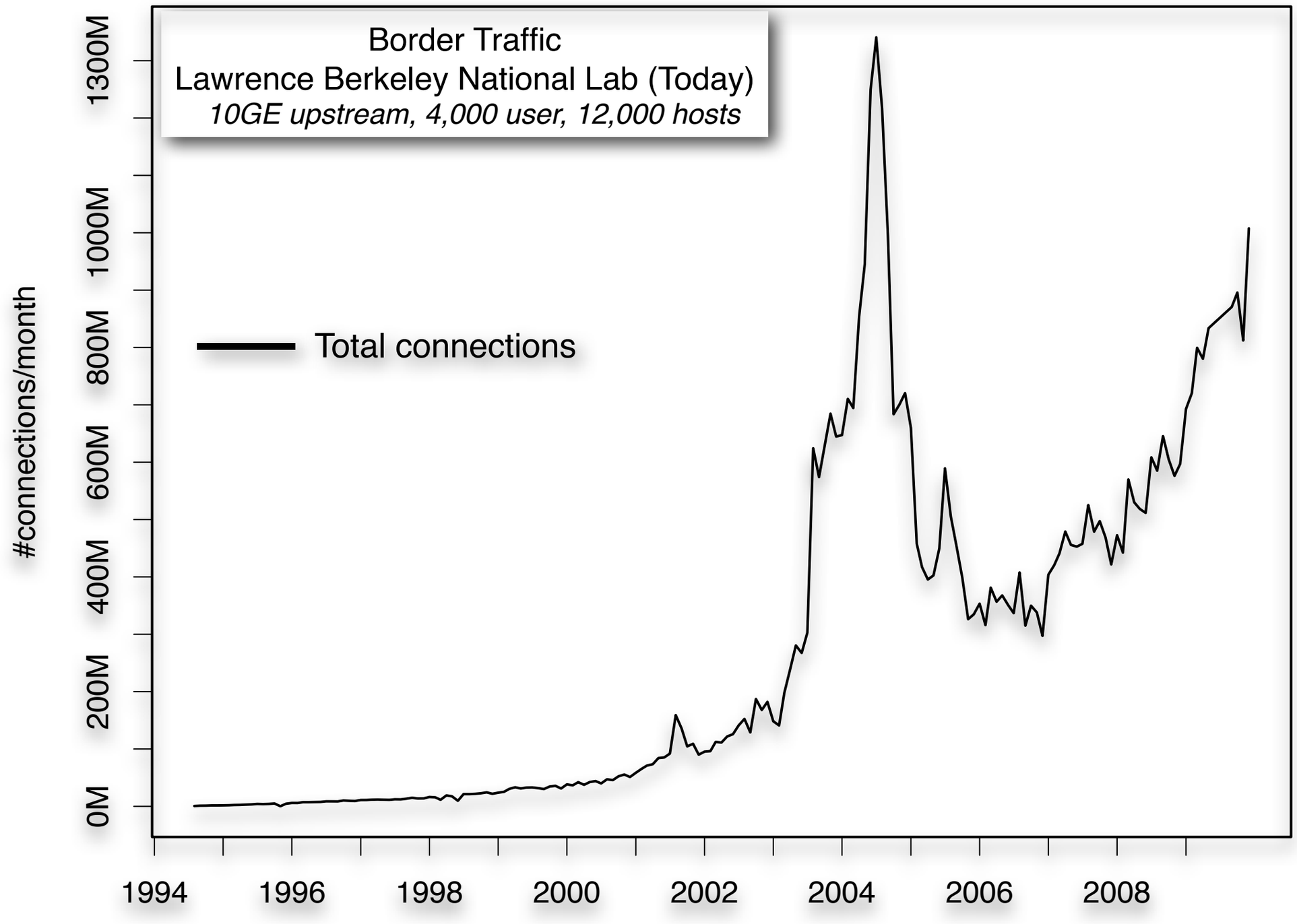
# Outline

Today's Threats.

Deep Packet Inspection at High Speed

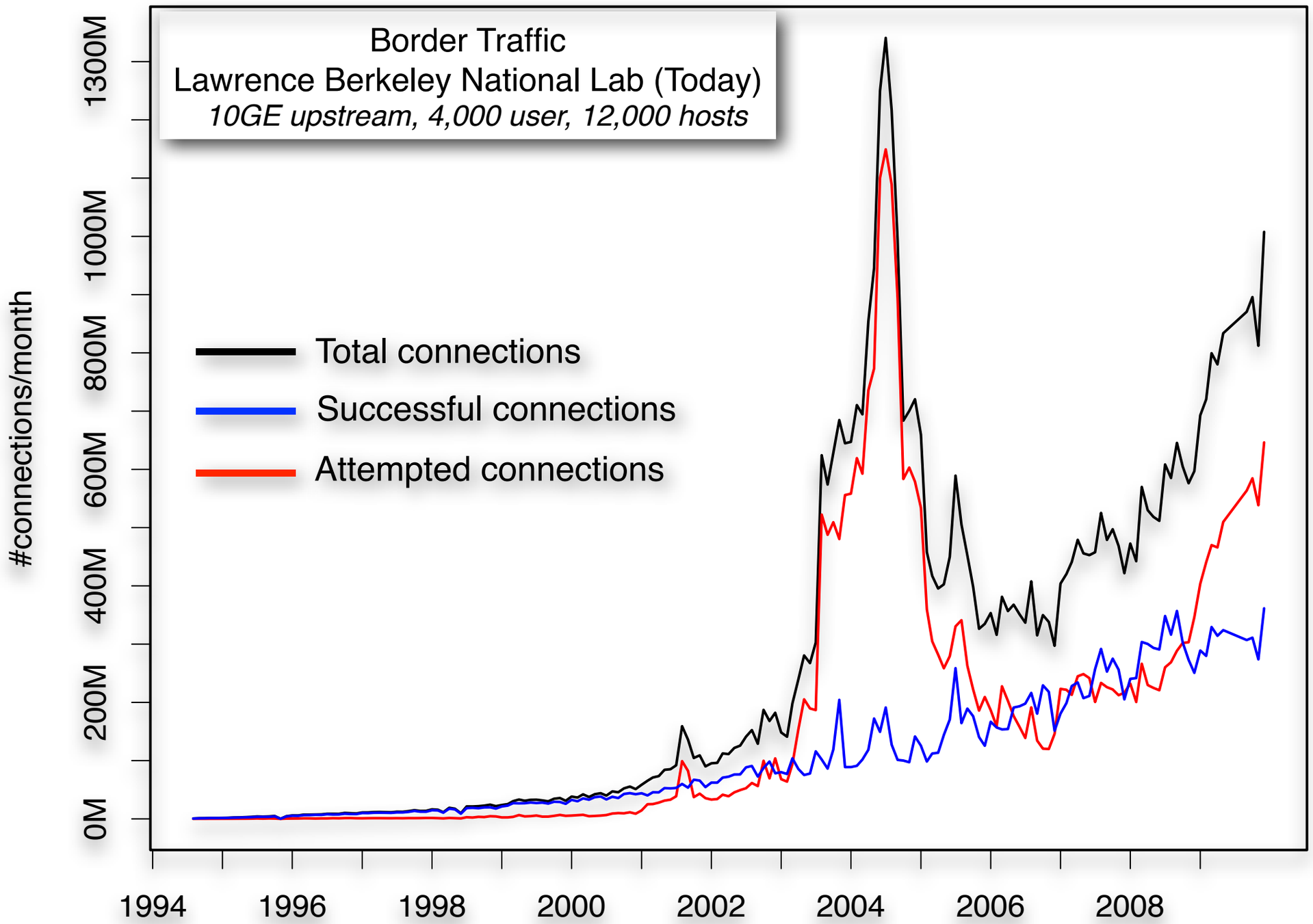
## Semantic Analysis at Global Scale
### The ICSI SSL Notary

# The Old Days ...



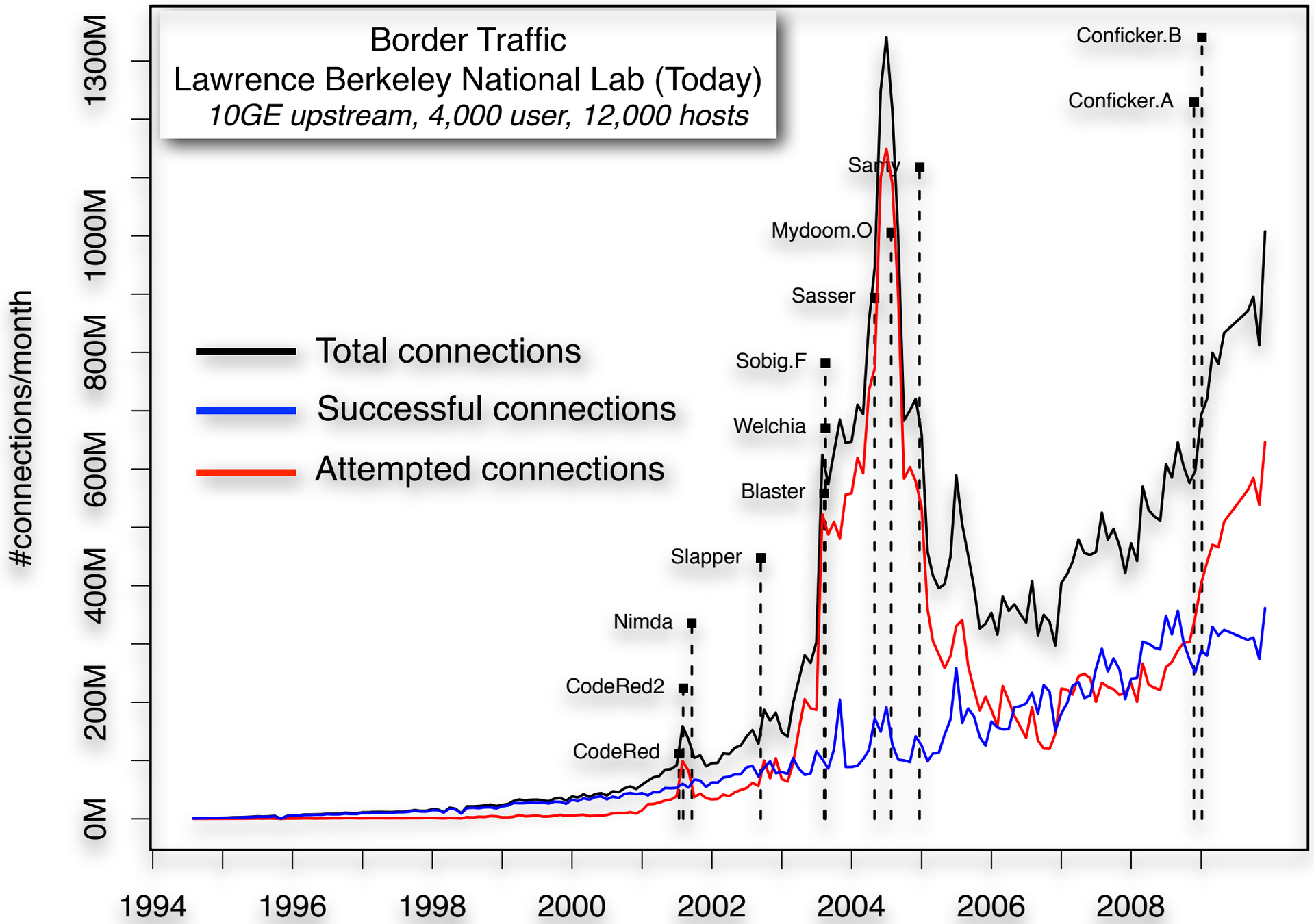

Data: Lawrence Berkeley National Lab

# The Old Days ...




Data: Lawrence Berkeley National Lab

# The Old Days ...




Data: Lawrence Berkeley National Lab

# Today's Threats

# Today's Threats

## Trend 1: Commercialization of attacks

Thriving underground economy ("Crime-as-a-Service").

Bear Race: Attack is good enough if it pays.

# Today's Threats

Trend 1: Commercialization of attacks

Thriving underground economy ("Crime-as-a-Service").
Bear Race: Attack is good enough if it pays.

Trend 2: High-skill / high-resource attacks.

Activist Hacking.
Advanced Persistent Threats.

> Advanced Persistent Threat (APT). MANDIANT defines the APT as a group of sophisticated, determined and coordinated attackers that have been systematically compromising U.S. government and commercial computer networks for years. The vast majority of

Source: MANDIANT

# Today's Threats

## Trend 1: Commercialization of attacks

Thriving underground economy ("Crime-as-a-Service").
Bear Race: Attack is good enough if it pays.

## Trend 2: High-skill / high-resource attacks.

Activist Hacking.
Advanced Persistent Threats.

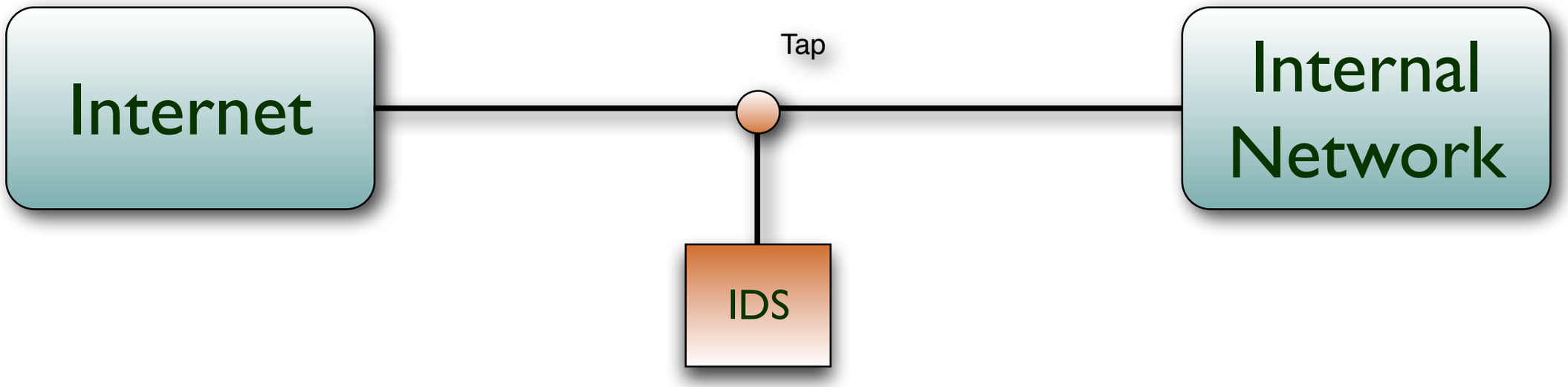
## Trend 3: Insider Attacks

Sabotage
Exfiltration

# Deep Packet Inspection at High Speed

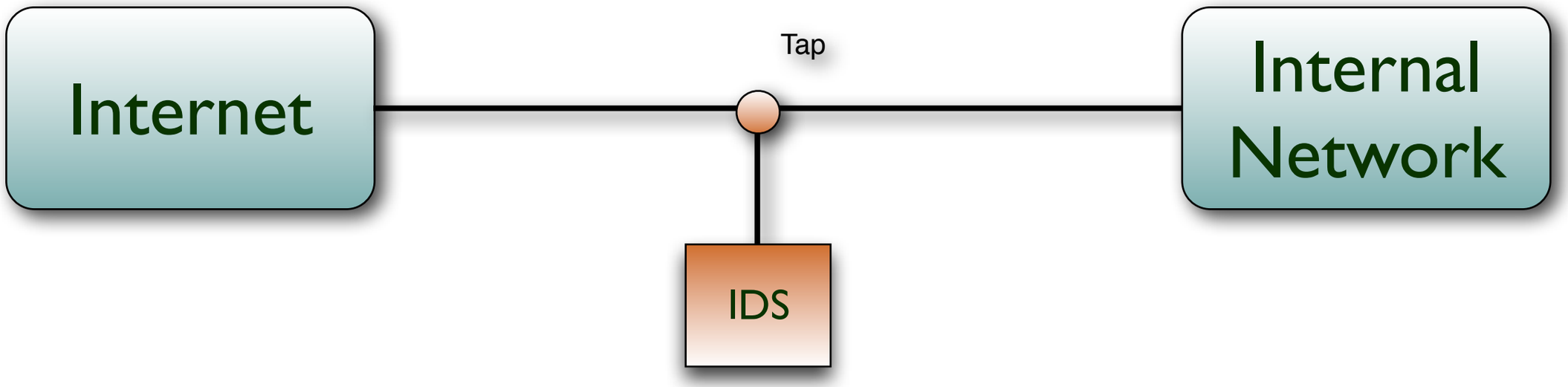

# Analyzing Semantics

# Analyzing Semantics




Example: Finding downloads of known malware.

# Analyzing Semantics

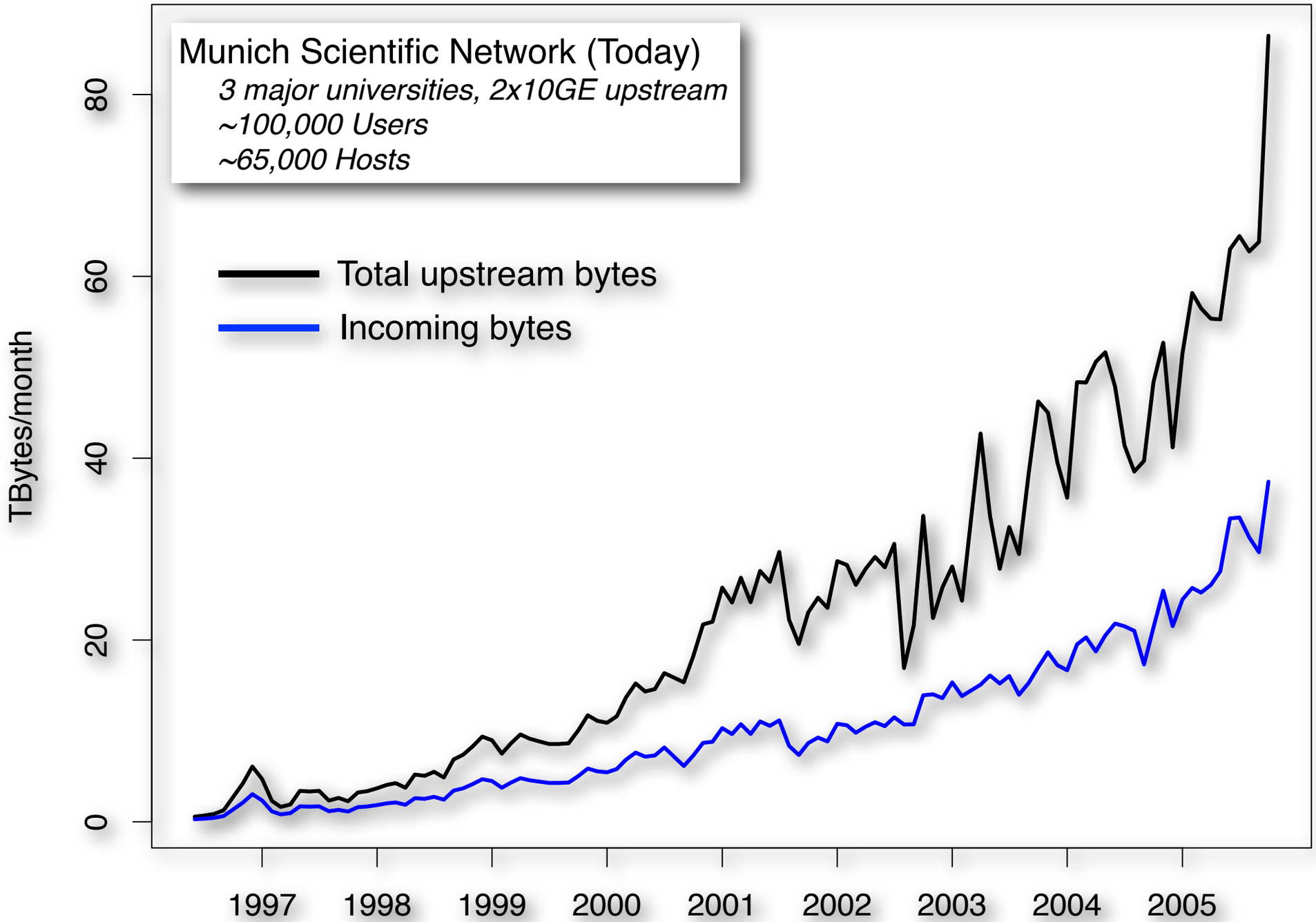



Example: Finding downloads of known malware.

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 ...

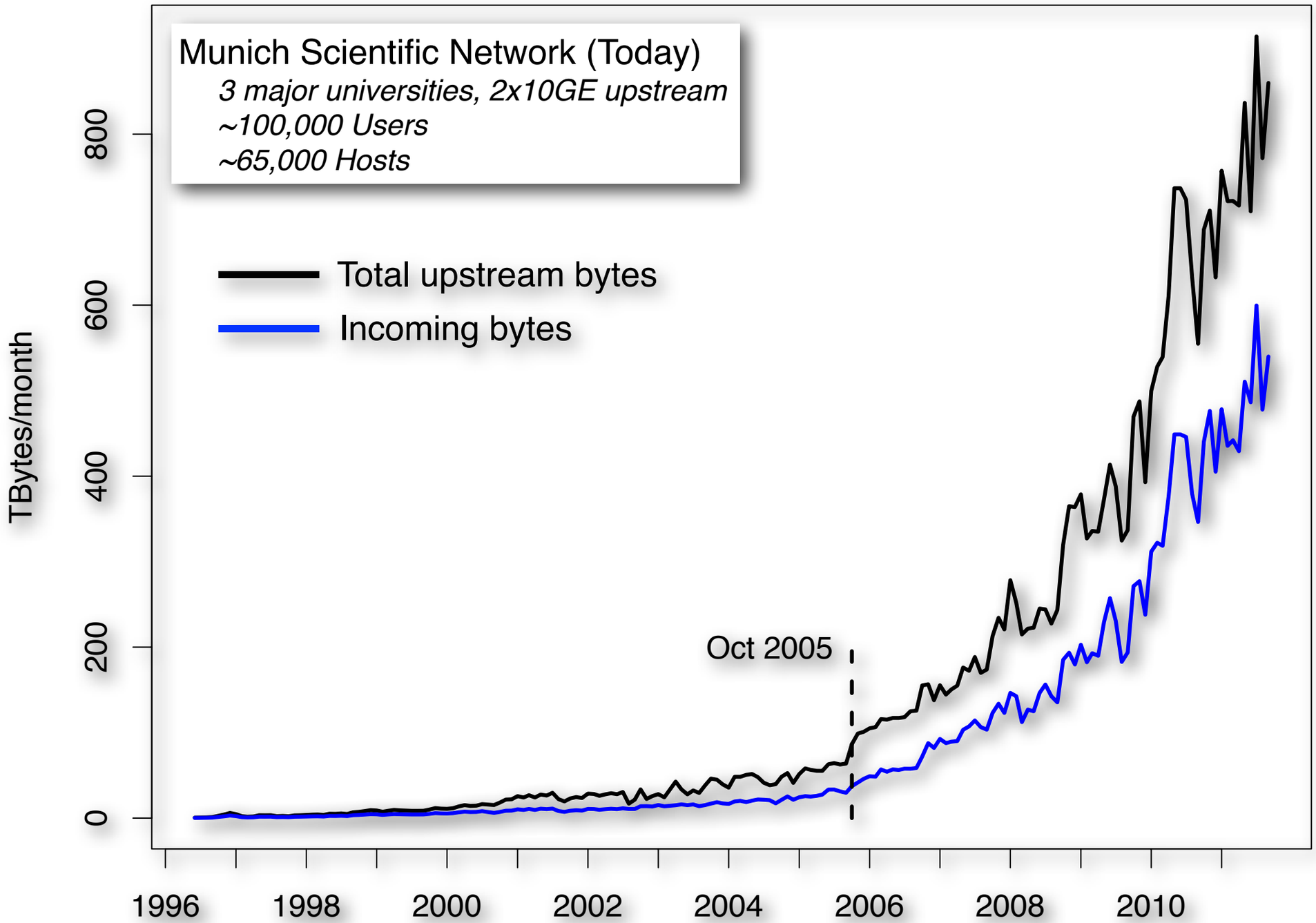
# Back in 2005 ...




Data: Leibniz-Rechenzentrum, München

# Today ...

# Today ...




Data: Leibniz-Rechenzentrum, München

# Traditional Gap: Research vs. Operations

# Traditional Gap: Research vs. Operations

Conceptually simple tasks can be hard in practice.

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

# Traditional Gap: Research vs. Operations

Conceptually simple tasks can be hard in practice.

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

We focus on working *with* operations.

- Close collaborations with several large sites.
- Extremely fruitful for both sides.

# Research Platform: Bro


# Research Platform: Bro

Originally developed by Vern Paxson in 1996.

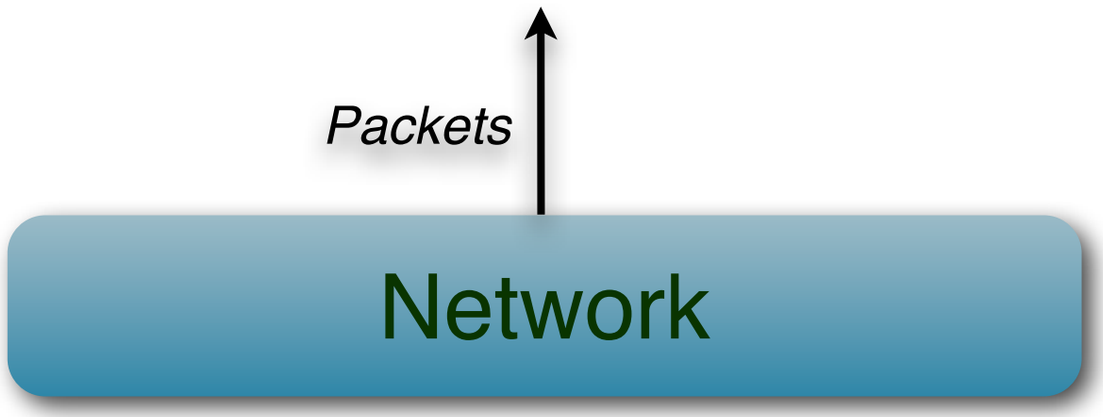
Open-source, BSD-license, maintained at ICSI.

In operational use since the beginning.

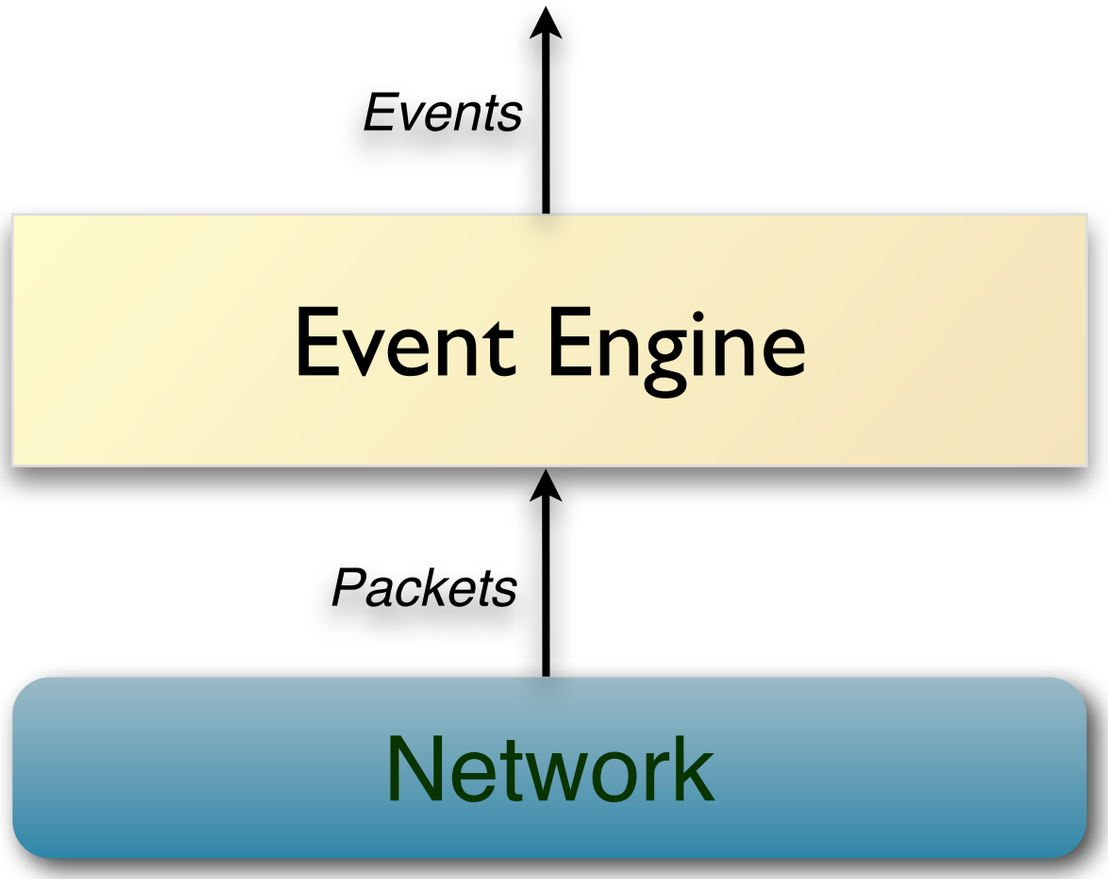
*Conceptually very different from other IDS.*


http://www.bro.org

# Architecture

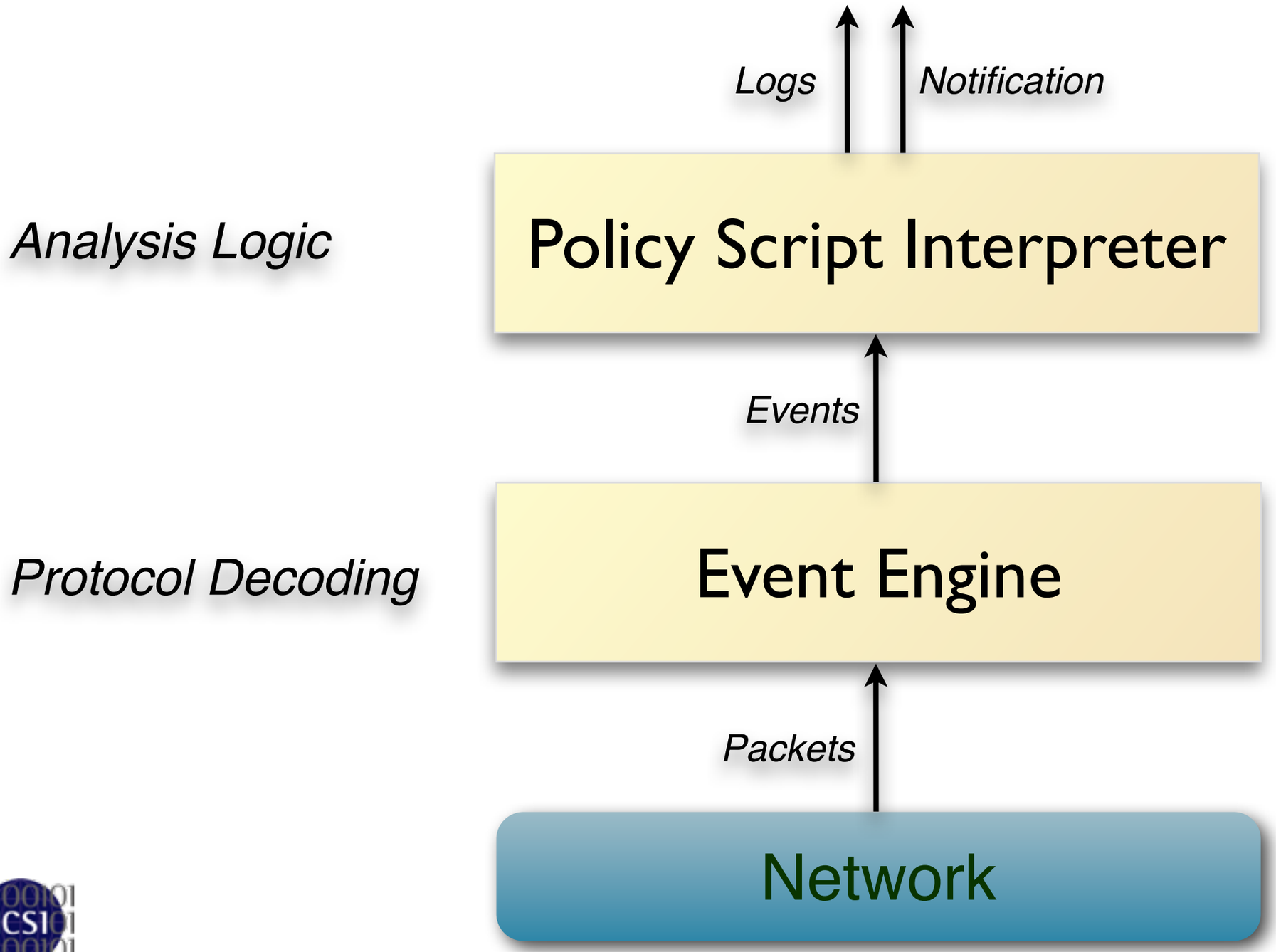


# Architecture

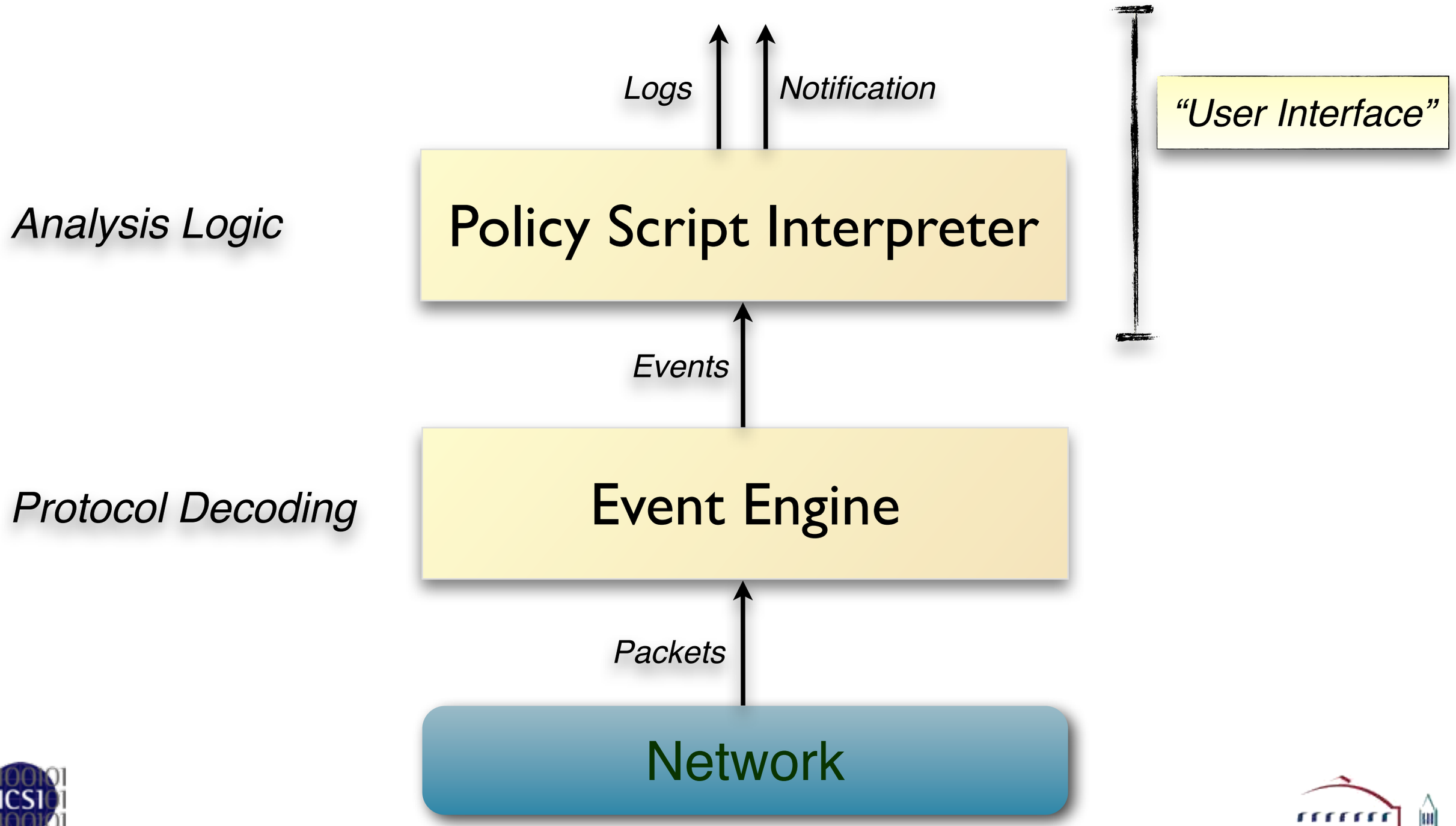



*Protocol Decoding*

# Architecture



# Architecture



# Script Example: Matching URLs

*Task: Report all Web requests for files called "`passwd`".*

# Script Example: Matching URLs

*Task: Report all Web requests for files called "`passwd`".*

```
event http_request(c: connection,           # Connection.
                   method: string,          # HTTP method.
                   original_URI: string,    # Requested URL.
                   unescaped_URI: string,   # Decoded URL.
                   version: string)         # HTTP version.
{
    if ( method == "GET" && unescaped_URI == /.*passwd/ )
        NOTICE(...); # Alarm.
}
```

# Script Example: Scan Detector

*Task: Count failed connection attempts per source address.*

# Script Example: Scan Detector

*Task: Count failed connection attempts per source address.*

```
global attempts: table[addr] of count &default=0;

event connection_rejected(c: connection)
{
        local source = c.id.orig_h;      # Get source address.
        local n = ++attempts[source];    # Increase counter.
        if ( n == SOME_THRESHOLD )       # Check for threshold.
            NOTICE(...);                 # Alarm.
}
```

# "Who's Using It?"

## Installations across the US

Universities
Research Labs
Supercomputer Centers
Fortune 50 Industry

## Examples

Lawrence Berkeley National Lab
Indiana University
National Center for Supercomputing Applications
National Center for Atmospheric Research

*... and many more sites*

## Fully integrated into *Security Onion*

Popular security-oriented Linux distribution

## Recent User Meetings

Bro Workshop 2011 at NCSA
Bro Exchange 2012 at NCAR

Each attended by about 50 operators from from 30-35 organizations

# Bro History

1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013

Vern writes 1st line of code

# Bro History

1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013

Vern writes 1st line of code

LBNL starts using Bro operationally

v0.2
1st CHANGES entry

v0.4
HTTP analysis
Scan detector
IP fragments
Linux support

v0.6
RegExps
Login analysis

v0.7a48
Consistent CHANGES

v0.7a90
Profiling
State Mgmt

v0.7a175/0.8aX
Signatures
SMTP
IPv6 support
User manual

0.8a37
Communication
Persistence
Namespaces
Log Rotation

v0.8aX/0.9aX
SSL/SMB
STABLE releases
BroLite

v1.0
BinPAC
IRC/RPC analyzers
64-bit support
Sane version numbers

v1.1/v1.2
*when* Stmt
Resource tuning
Broccoli
DPD

v1.3
Ctor expressions
GeoIP
Conn Compressor

v1.4
DHCP/BitTorrent
HTTP entities
NetFlow
Bro Lite Deprecated

v1.5
BroControl

**Bro SDCI**

v2.0
New Scripts

v2.1
IPv6
Input Framew.

(v2.2)
File Analysis
Summary Stat.

# Bro History

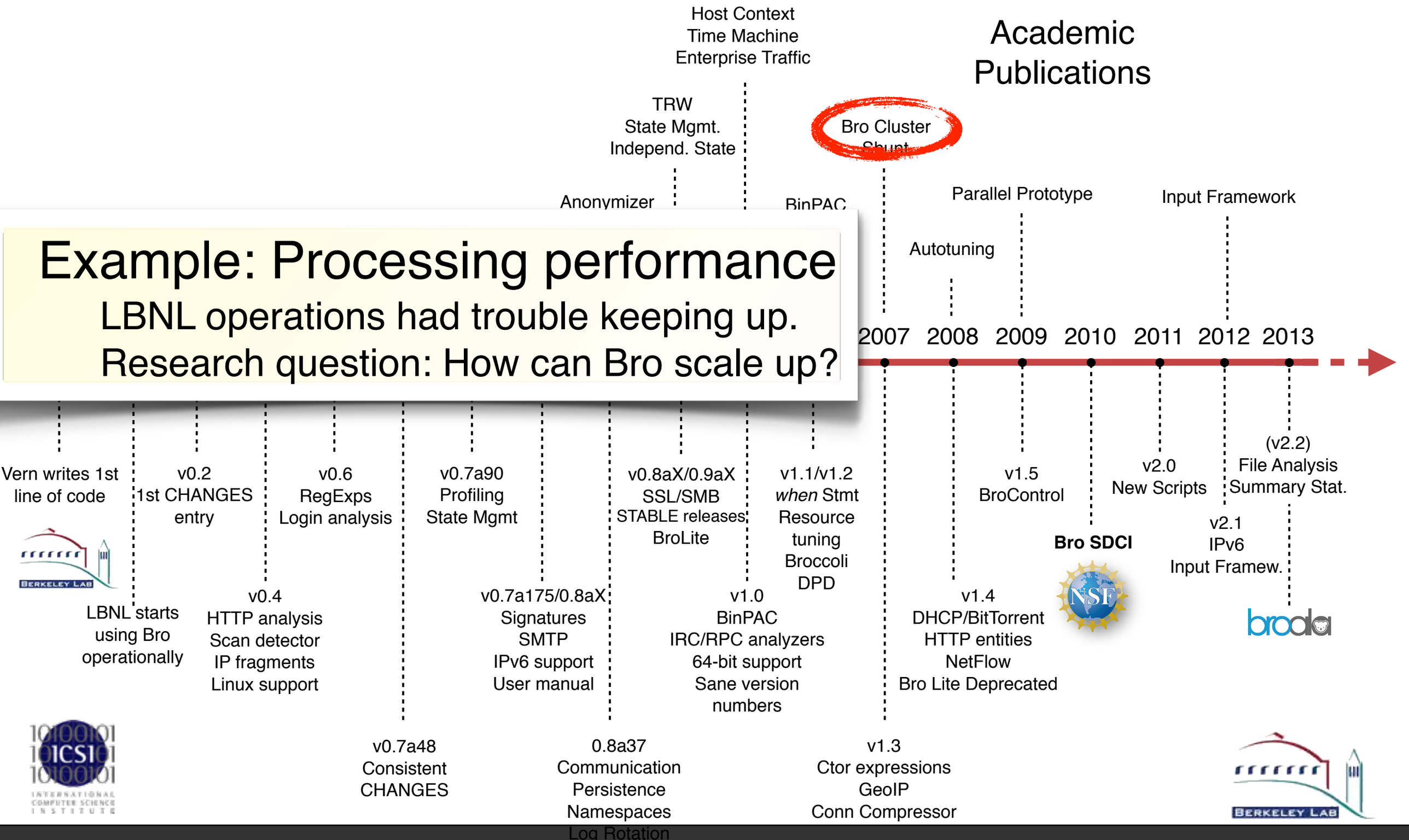
Academic Publications

1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013

- 1998: USENIX Paper
- 2000: Stepping Stone Detector
- 2003: Anonymizer, Active Mapping, Context Signat.
- 2004: TRW, State Mgmt., Independ. State
- 2005: Host Context, Time Machine, Enterprise Traffic
- 2006: BinPAC, DPD, 2nd Path
- 2007: Bro Cluster, Shunt
- 2008: Autotuning
- 2009: Parallel Prototype
- 2012: Input Framework

- Vern writes 1st line of code
- LBNL starts using Bro operationally
- v0.2 1st CHANGES entry
- v0.4 HTTP analysis, Scan detector, IP fragments, Linux support
- v0.6 RegExps, Login analysis
- v0.7a48 Consistent CHANGES
- v0.7a90 Profiling, State Mgmt
- v0.7a175/0.8aX Signatures, SMTP, IPv6 support, User manual
- 0.8a37 Communication, Persistence, Namespaces, Log Rotation
- v0.8aX/0.9aX SSL/SMB, STABLE releases, BroLite
- v1.0 BinPAC, IRC/RPC analyzers, 64-bit support, Sane version numbers
- v1.1/v1.2 *when* Stmt, Resource tuning, Broccoli, DPD
- v1.3 Ctor expressions, GeoIP, Conn Compressor
- v1.4 DHCP/BitTorrent, HTTP entities, NetFlow, Bro Lite Deprecated
- v1.5 BroControl
- **Bro SDCI**
- v2.0 New Scripts
- v2.1 IPv6, Input Framew.
- (v2.2) File Analysis, Summary Stat.

# Bro History



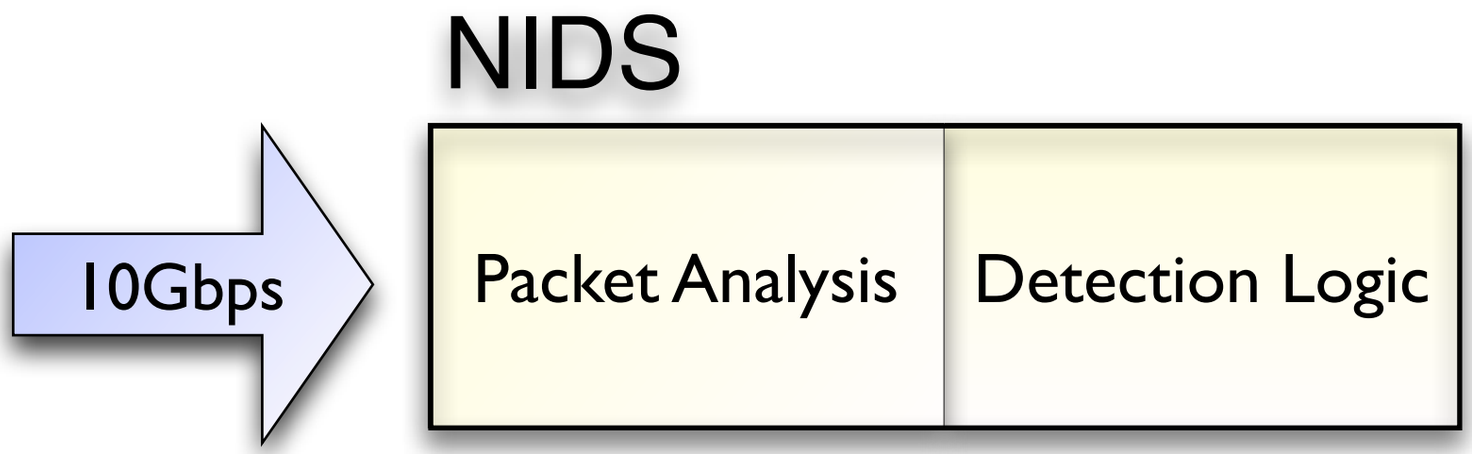

## Example: Processing performance

LBNL operations had trouble keeping up.

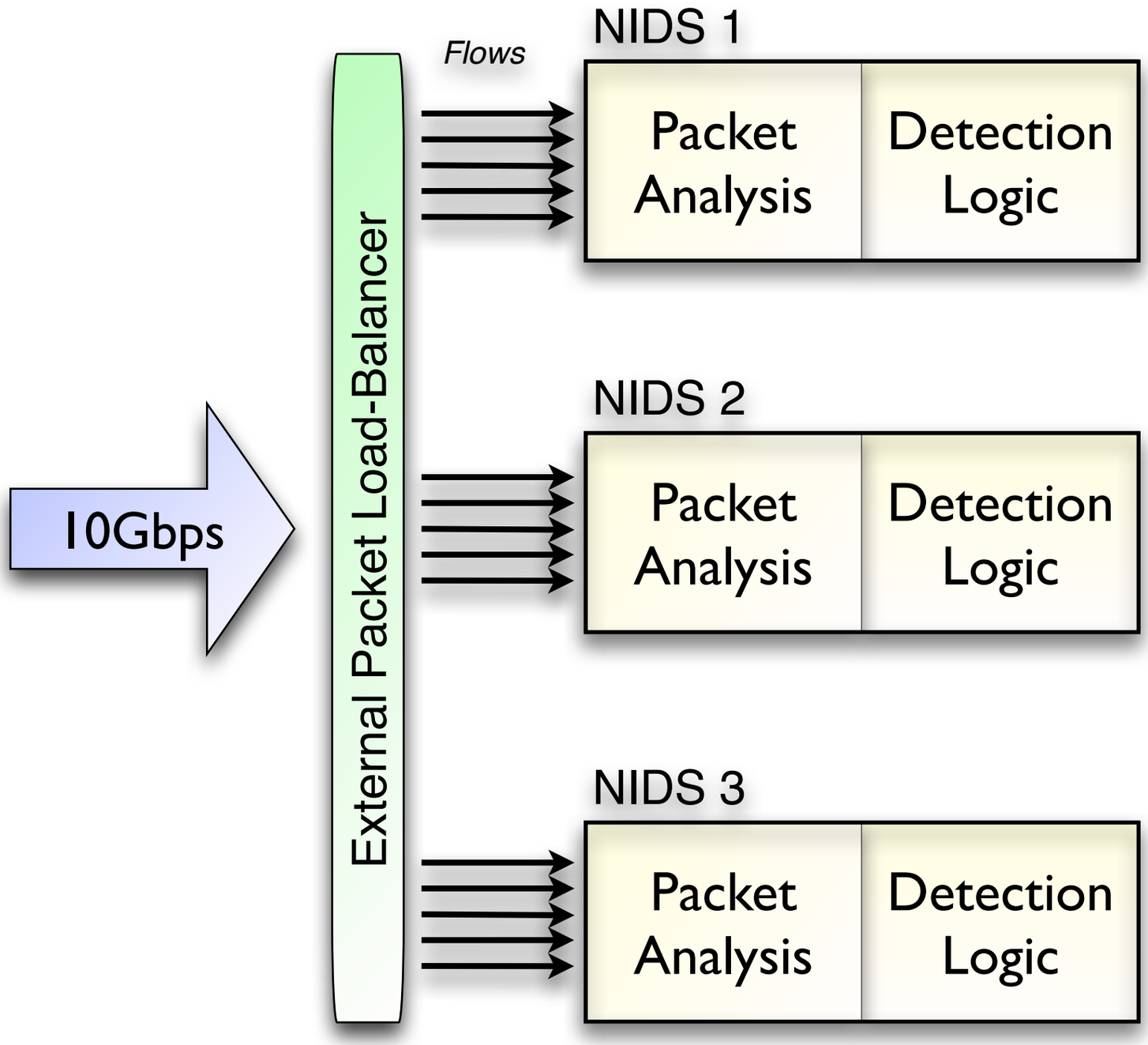
Research question: How can Bro scale up?

# Load-balancing Architecture

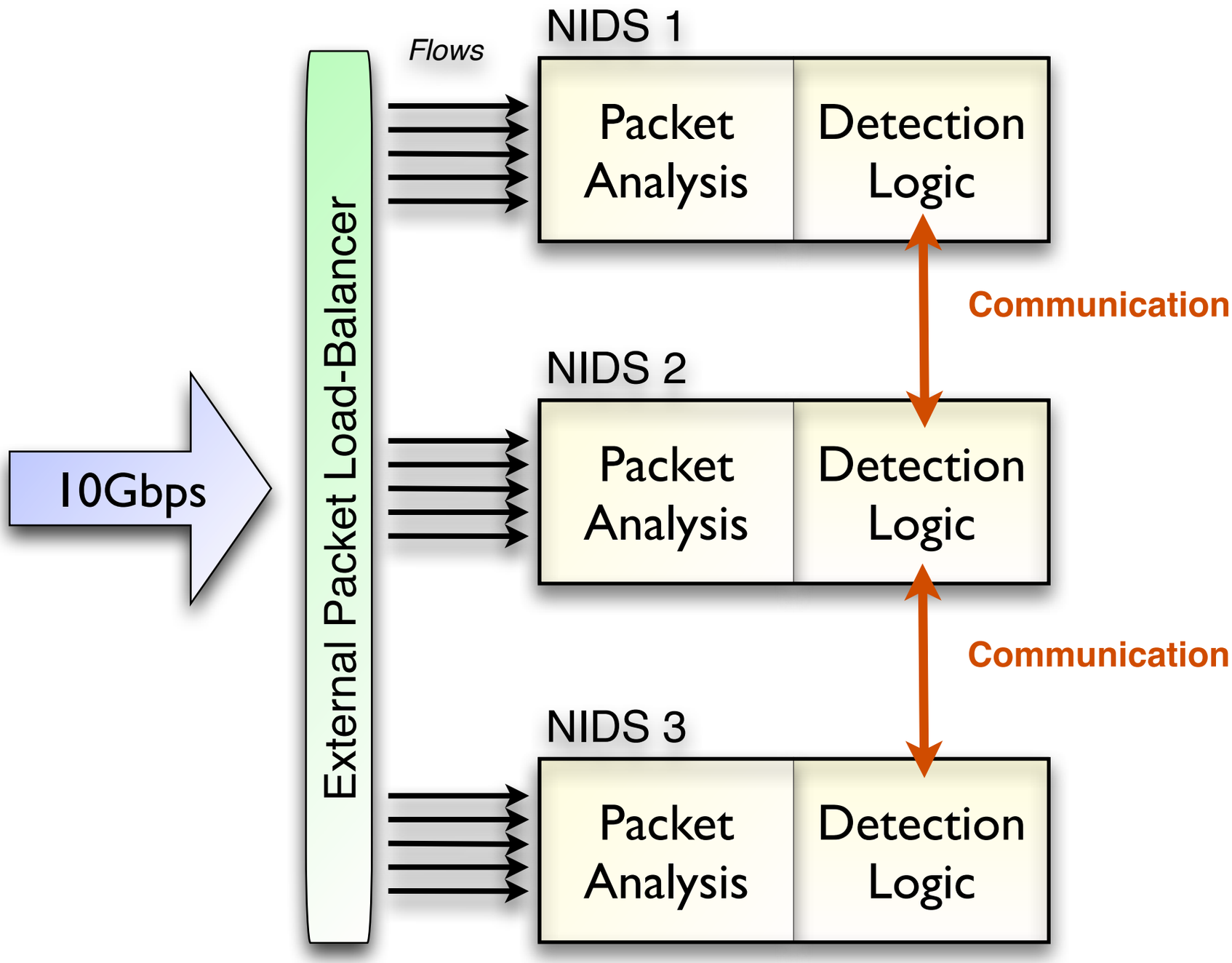

# Load-balancing Architecture



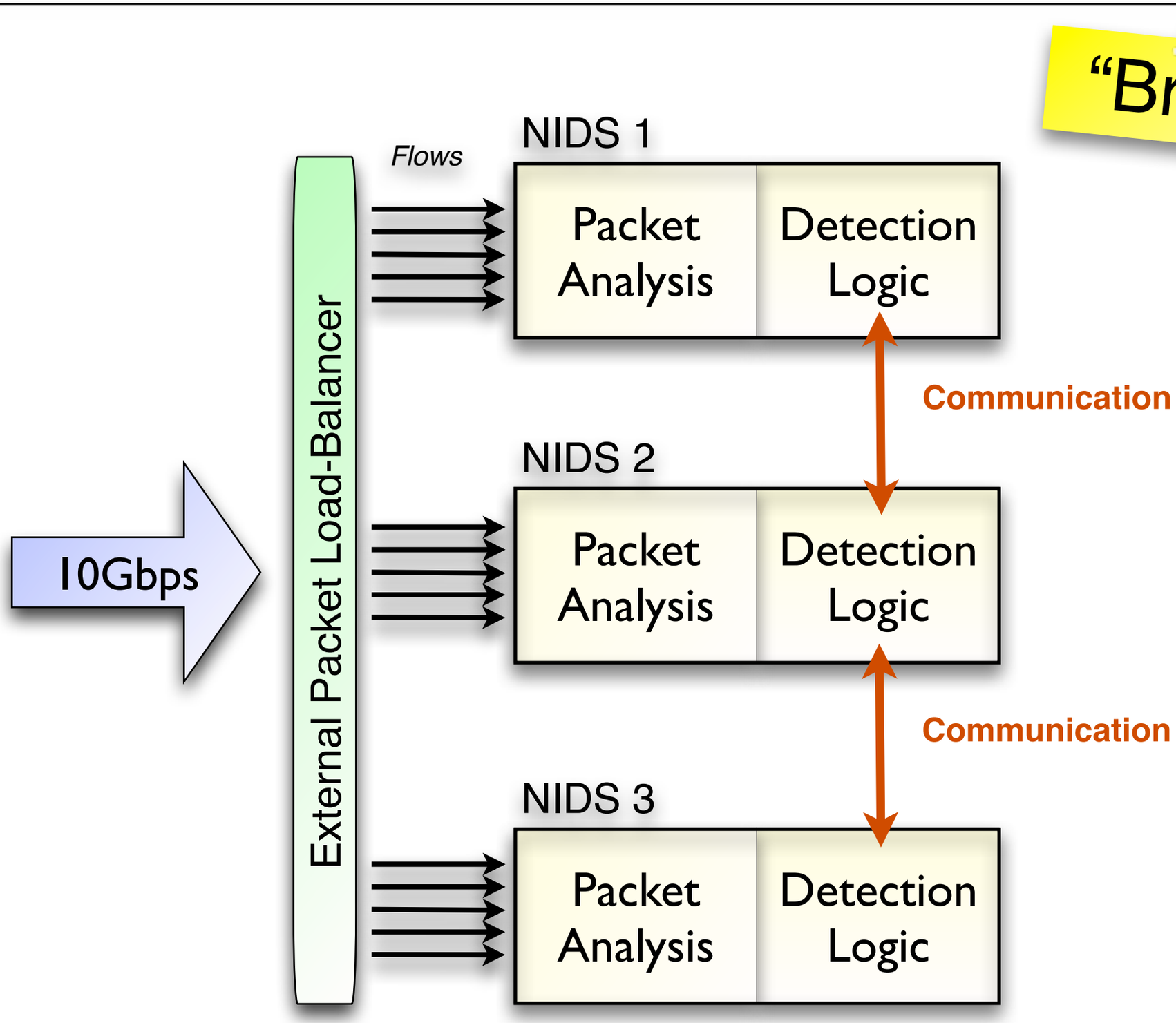
# Load-balancing Architecture



# Load-balancing Architecture



# Load-balancing Architecture



# 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.

No robust line-rate solution available in 2007.

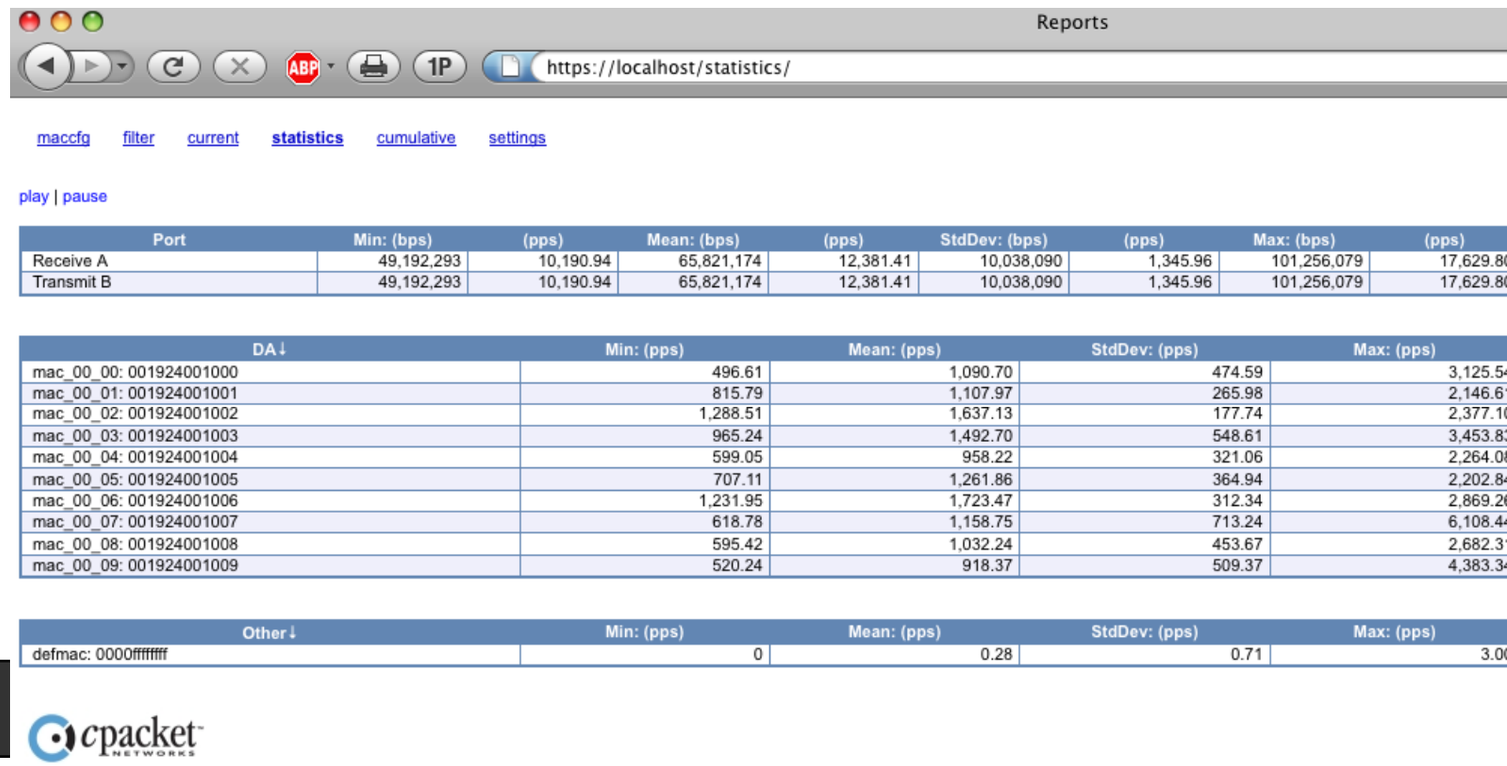
Eventually contracted vendor to build device.

# A Production Load-Balancer

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


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

*Available from cPacket*



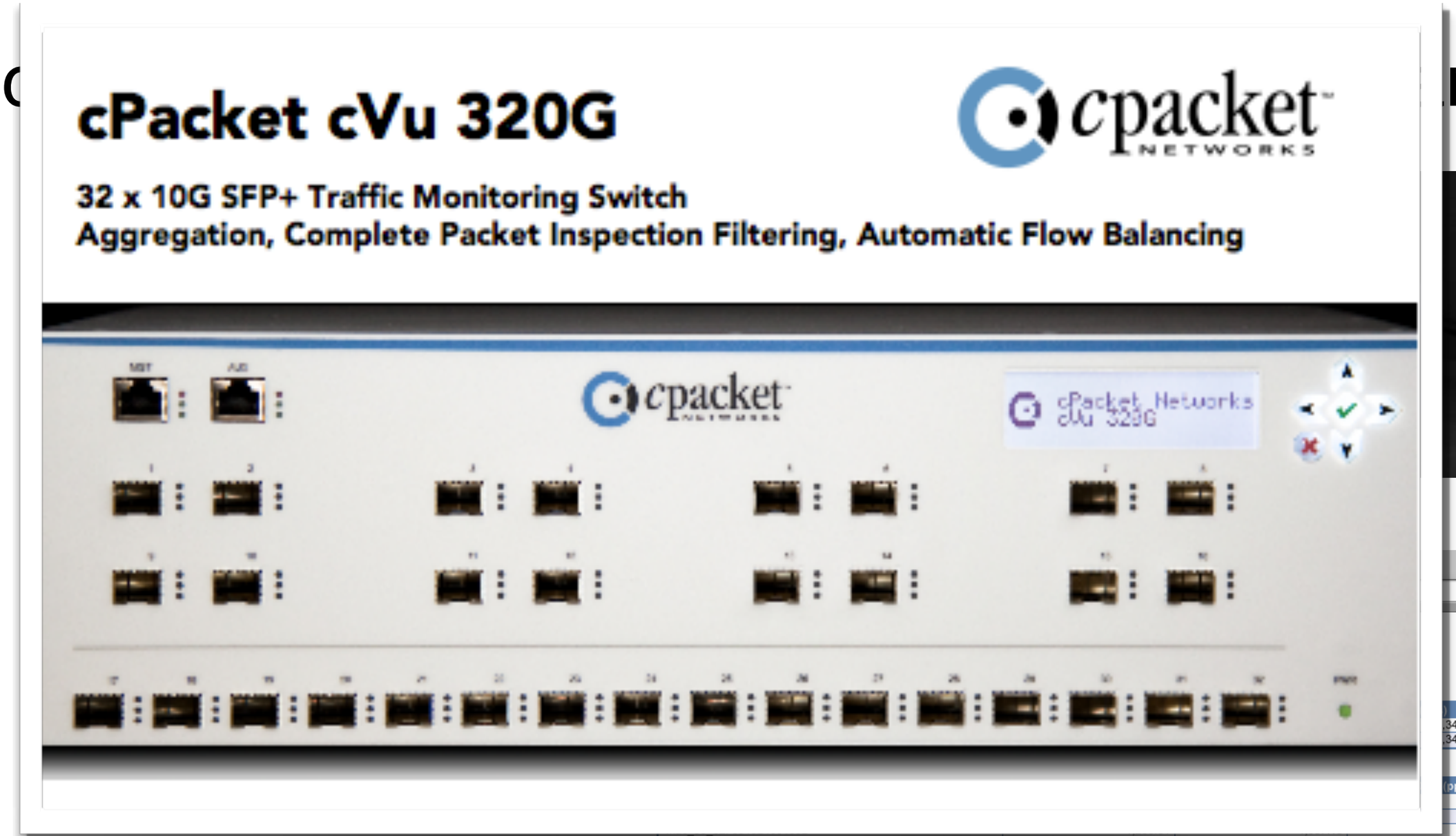

| Port | Min: (bps) | (pps) | Mean: (bps) | (pps) | StdDev: (bps) | (pps) | Max: (bps) | (pps) |
|---|---|---|---|---|---|---|---|---|
| Receive A | 49,192,293 | 10,190.94 | 65,821,174 | 12,381.41 | 10,038,090 | 1,345.96 | 101,256,079 | 17,629.8 |
| Transmit B | 49,192,293 | 10,190.94 | 65,821,174 |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

cpacket NETWORKS

# A Production Load-Balancer




*Available from cPacket*

| | | | Max: (bps) | (pps) |
|---|---|---|---|---|
| | | | 101,256,079 | 17,629.80 |
| | | | 101,256,079 | 17,629.80 |

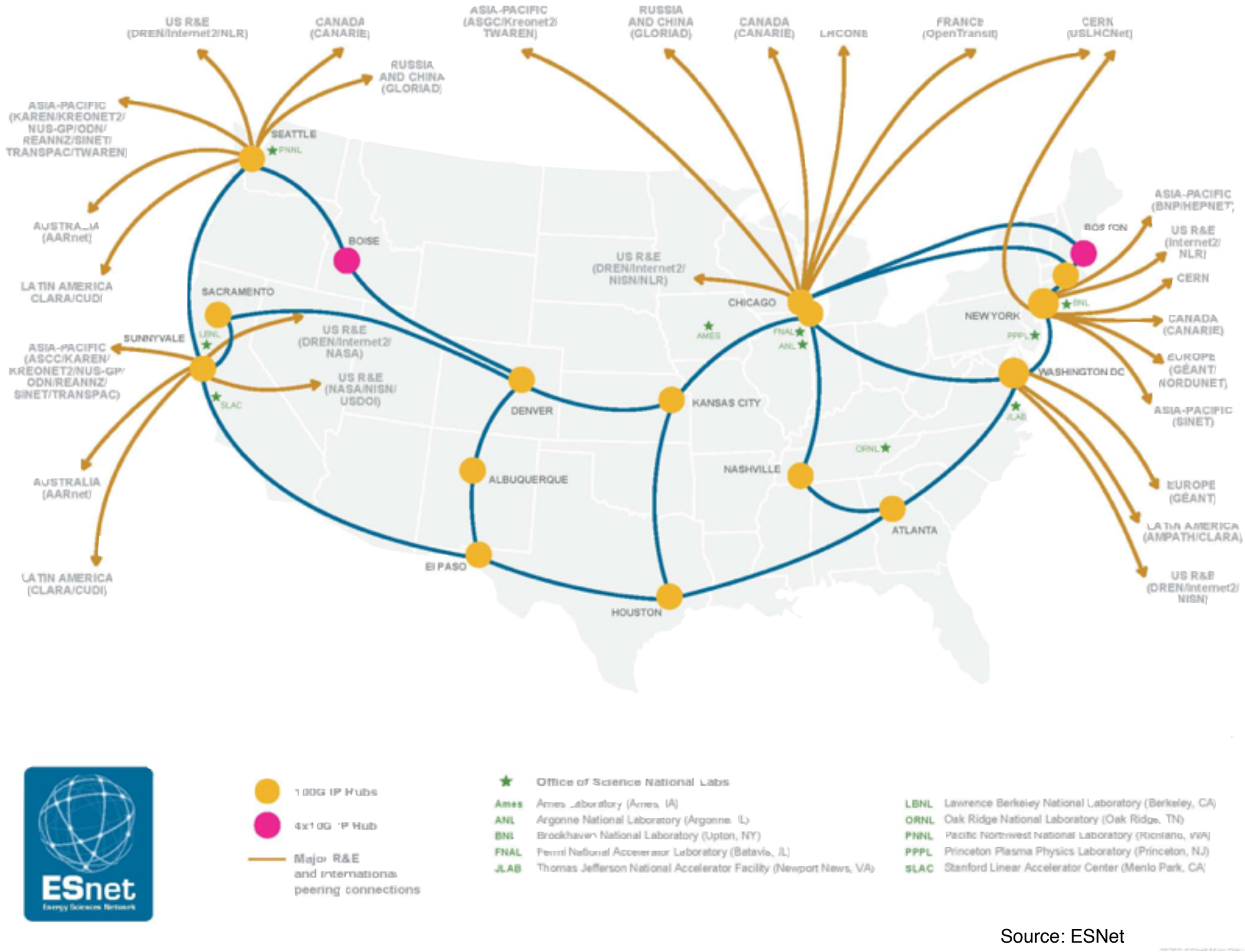
| | | | (pps) | Max: (pps) |
|---|---|---|---|---|
| | | | 474.59 | 3,125.54 |
| | | | 265.98 | 2,146.61 |
| | | | 177.74 | 2,377.10 |
| mac_00_03: 001924001003 | 965.24 | 1,492.70 | 548.61 | 3,453.83 |
| mac_00_04: 001924001004 | 599.05 | 958.22 | 321.06 | 2,264.08 |
| mac_00_05: 001924001005 | 707.11 | 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 |

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

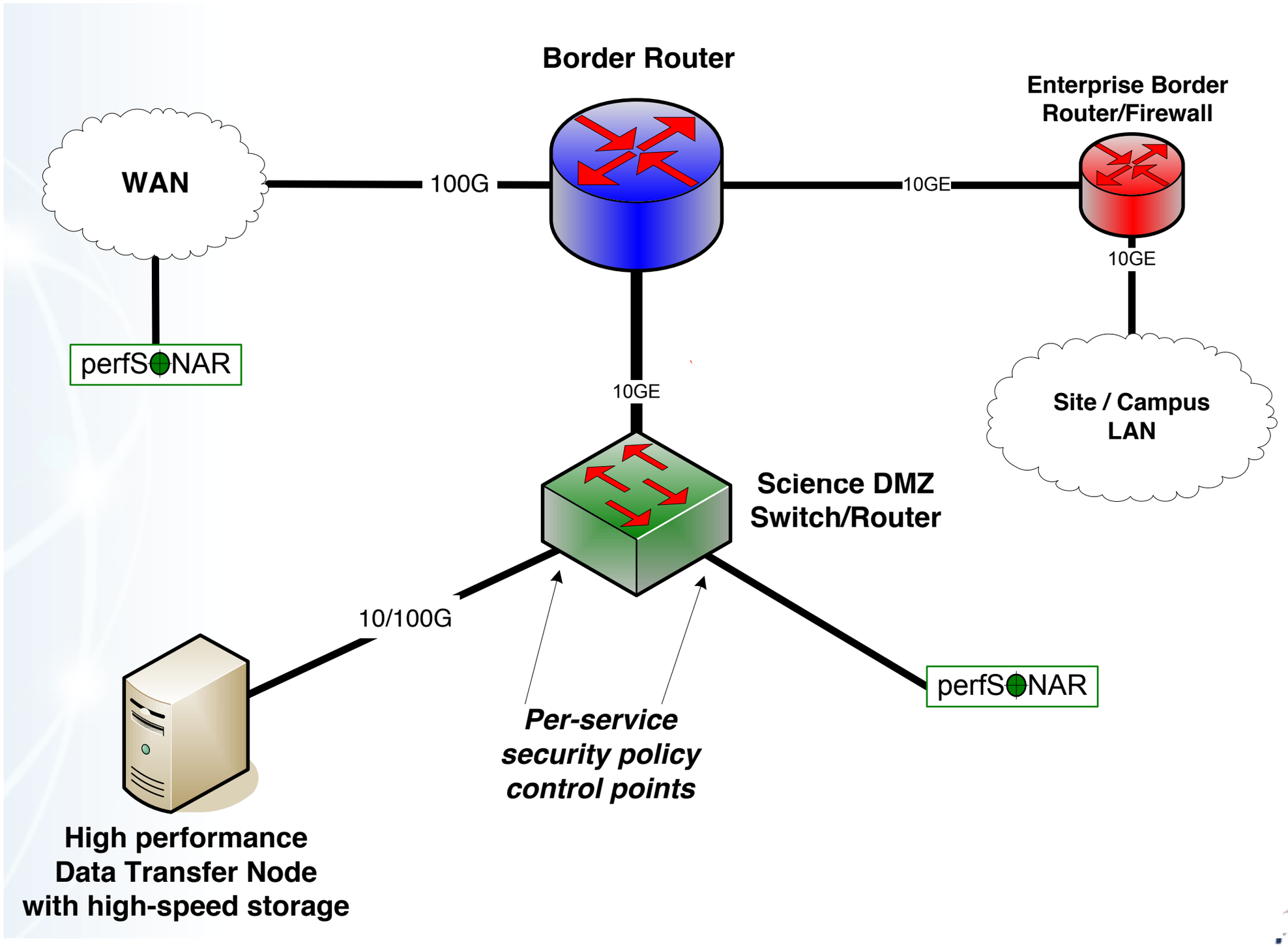
cpacket NETWORKS

# Next Stop: 100 Gb/s



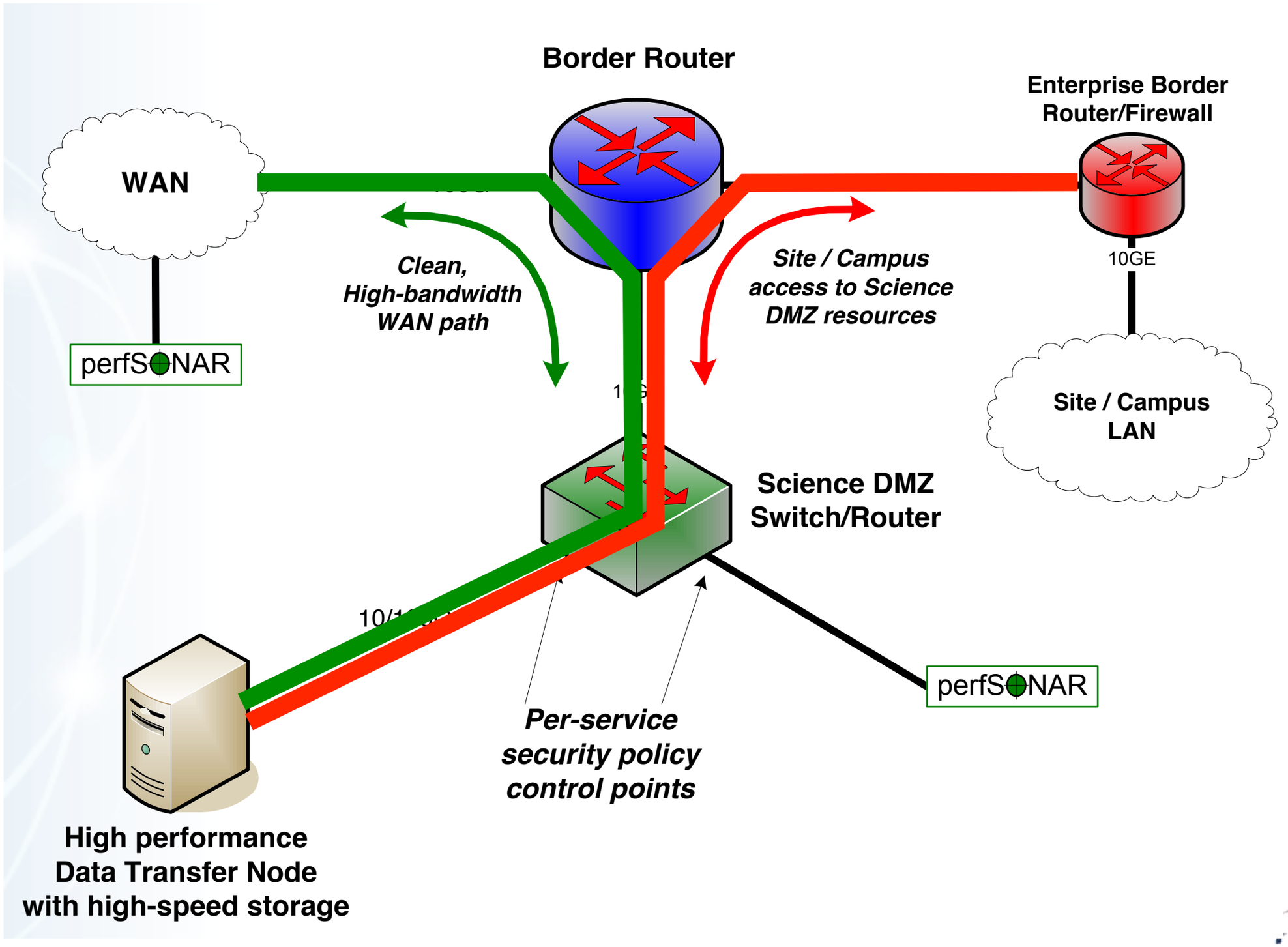

Source: ESNet

# Science DMZs



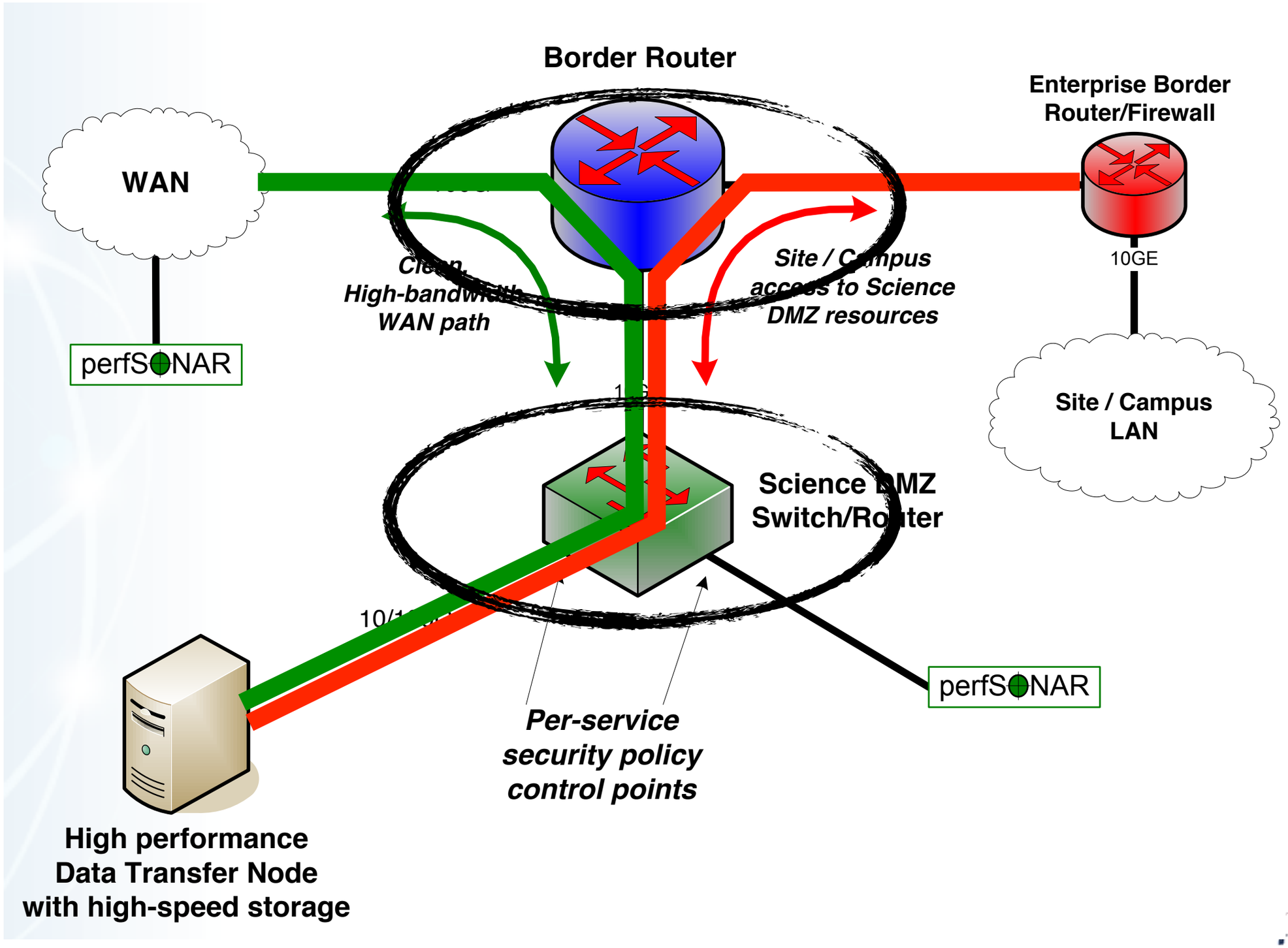

Source: ESNet

# Science DMZs



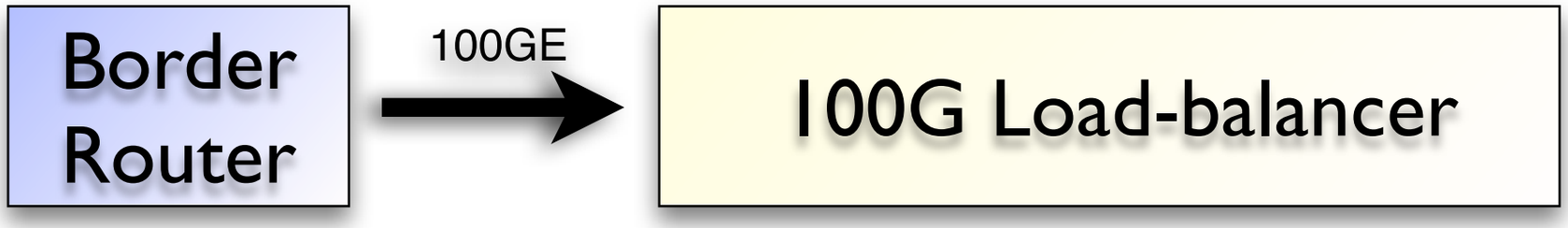

Source: ESNet

# Science DMZs

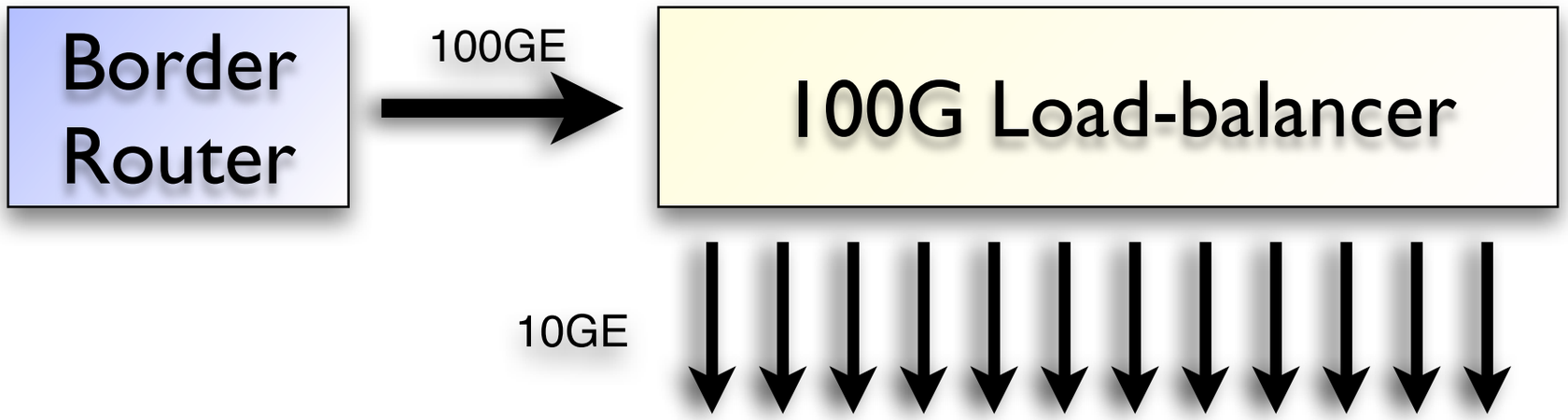



Source: ESNet

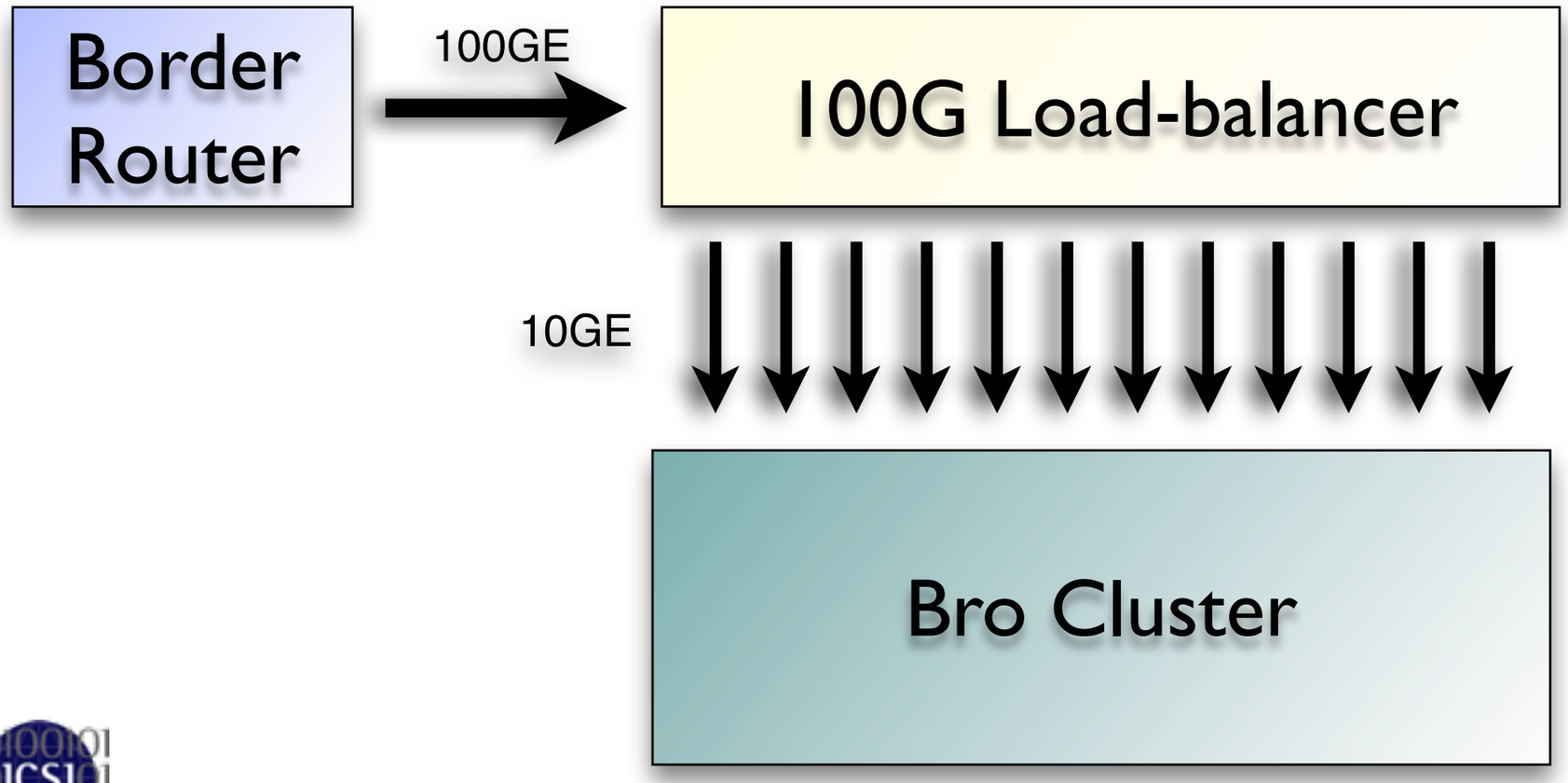
# 100 Gb/s Cluster



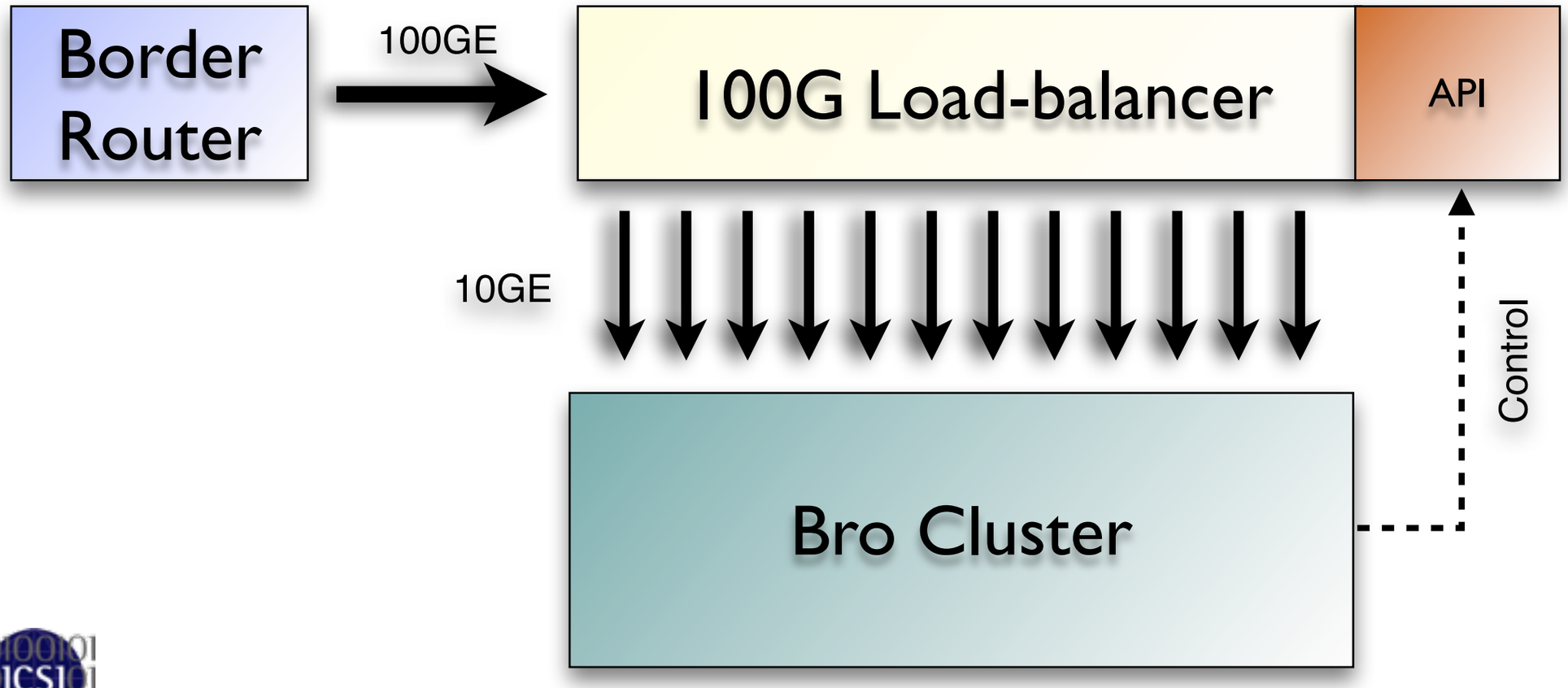
# 100 Gb/s Cluster



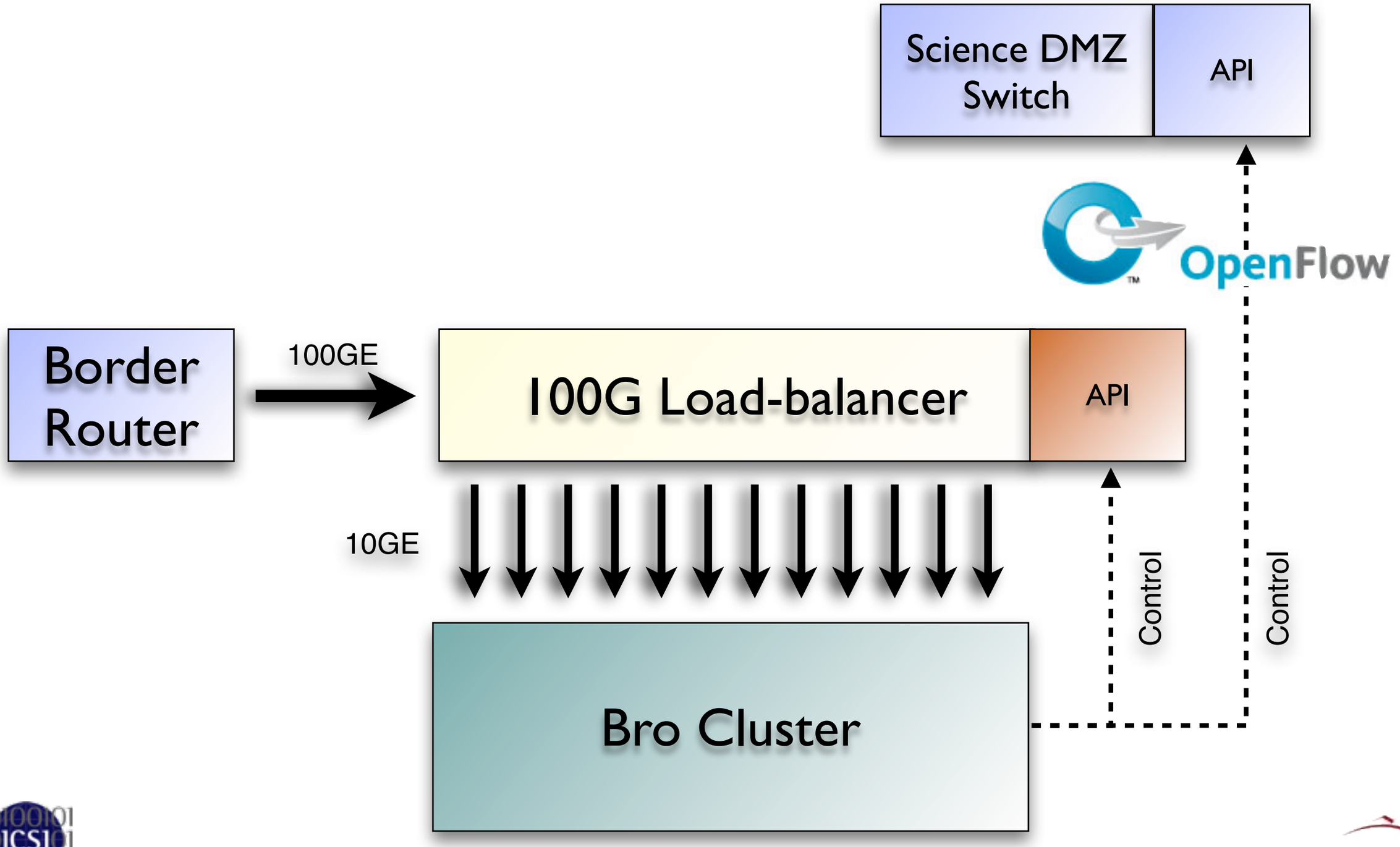
# 100 Gb/s Cluster



# 100 Gb/s Cluster



# 100 Gb/s Cluster



# Going Multi-Core

# Going Multi-Core

Bro is single-threaded

- Backends have multiple cores, which are mostly idling.
- Work-around: “Cluster in a box”

# Going Multi-Core

Bro is single-threaded

- Backends have multiple cores, which are mostly idling.
- Work-around: “Cluster in a box”

We *really* want multi-threading.

- Needs to scale well with increasing numbers of cores.
- Needs to be transparent to the operator.

# Going Multi-Core

Bro is single-threaded

- Backends have multiple cores, which are mostly idling.
- Work-around: “Cluster in a box”

We *really* want multi-threading.

- Needs to scale well with increasing numbers of cores.
- Needs to be transparent to the operator.

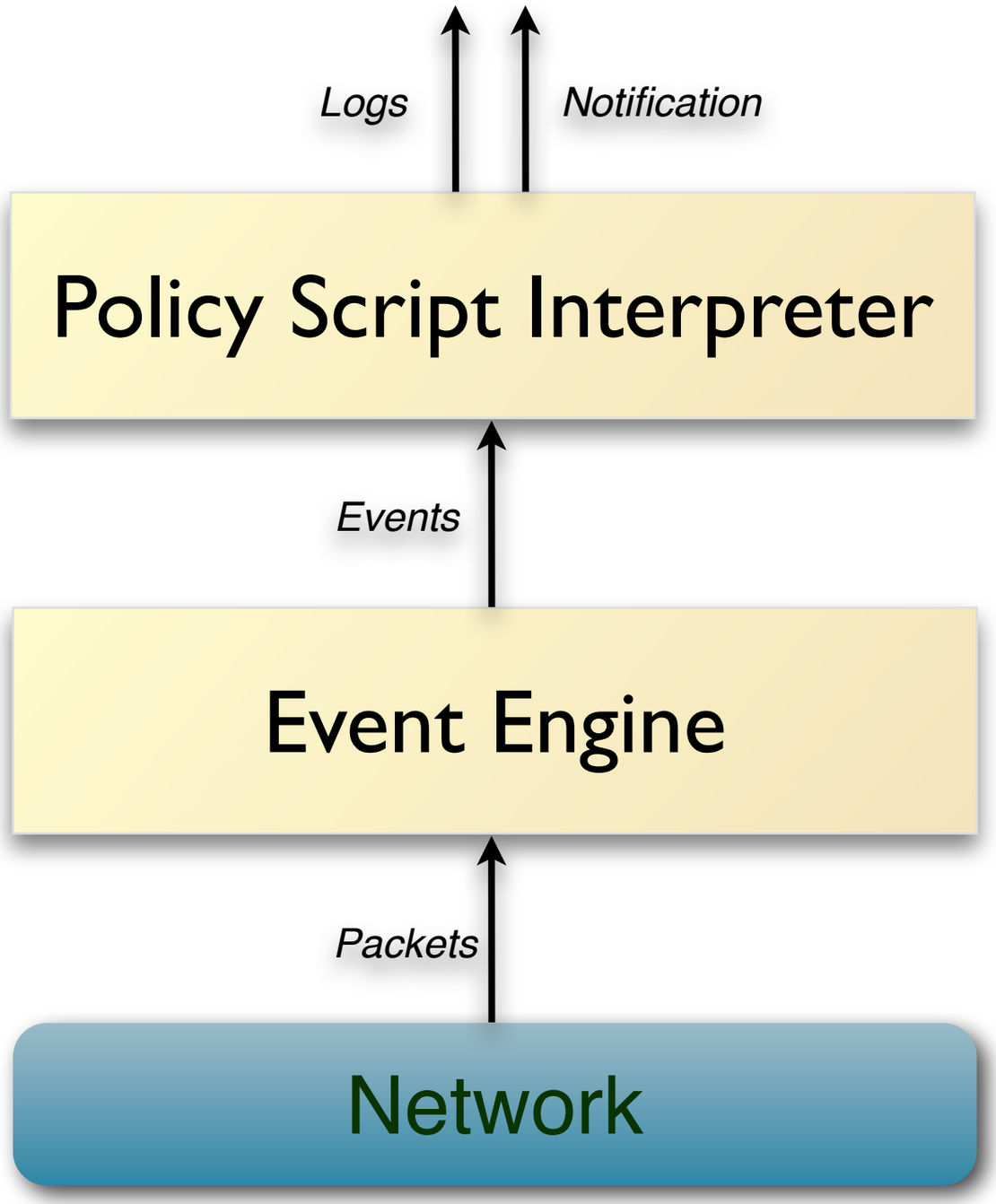
For some IDS, that’s not so hard.

- For others, it is ...

# Concurrent Analysis

*Analysis Logic*

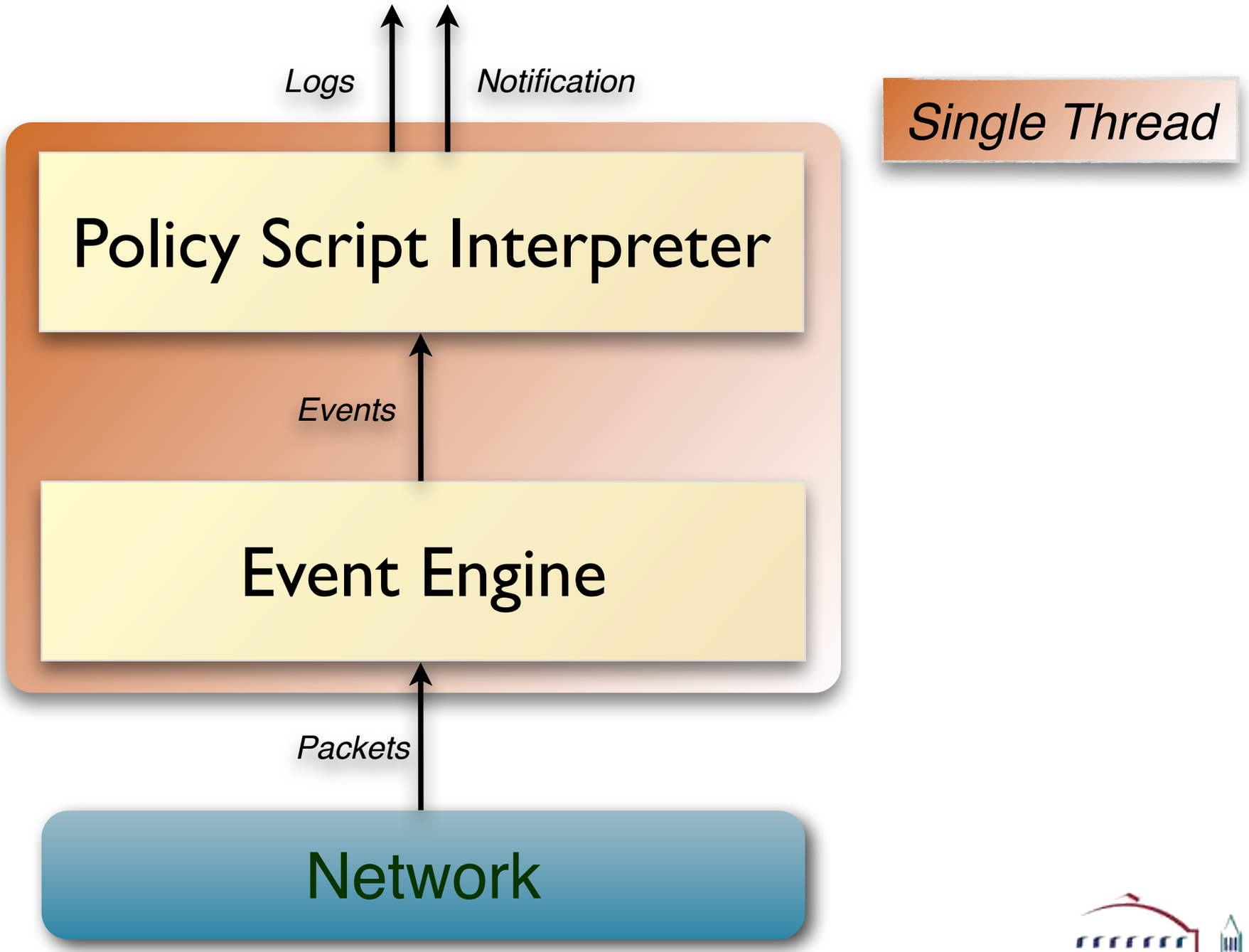
*Protocol Decoding*



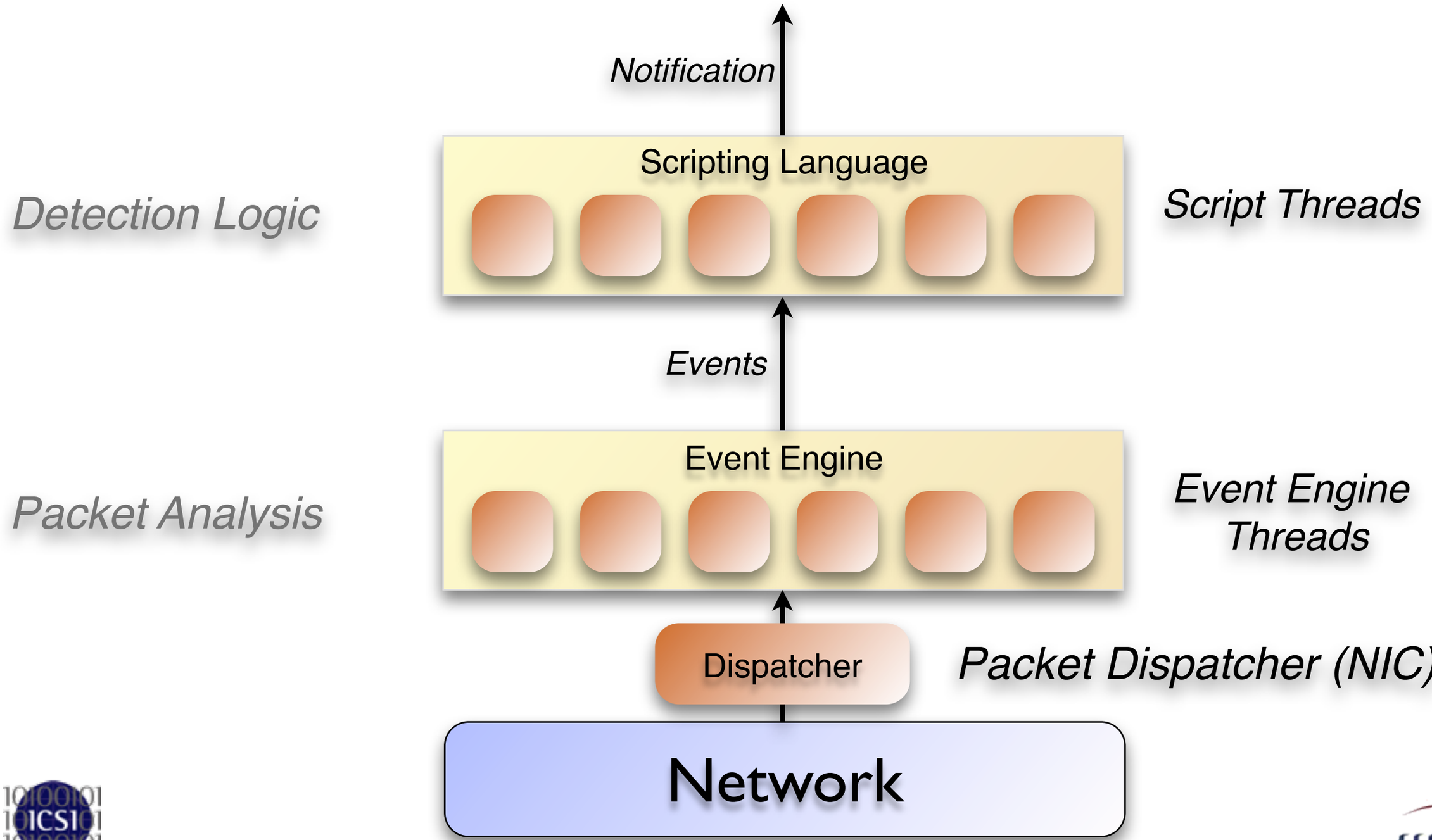
# Concurrent Analysis

*Analysis Logic*

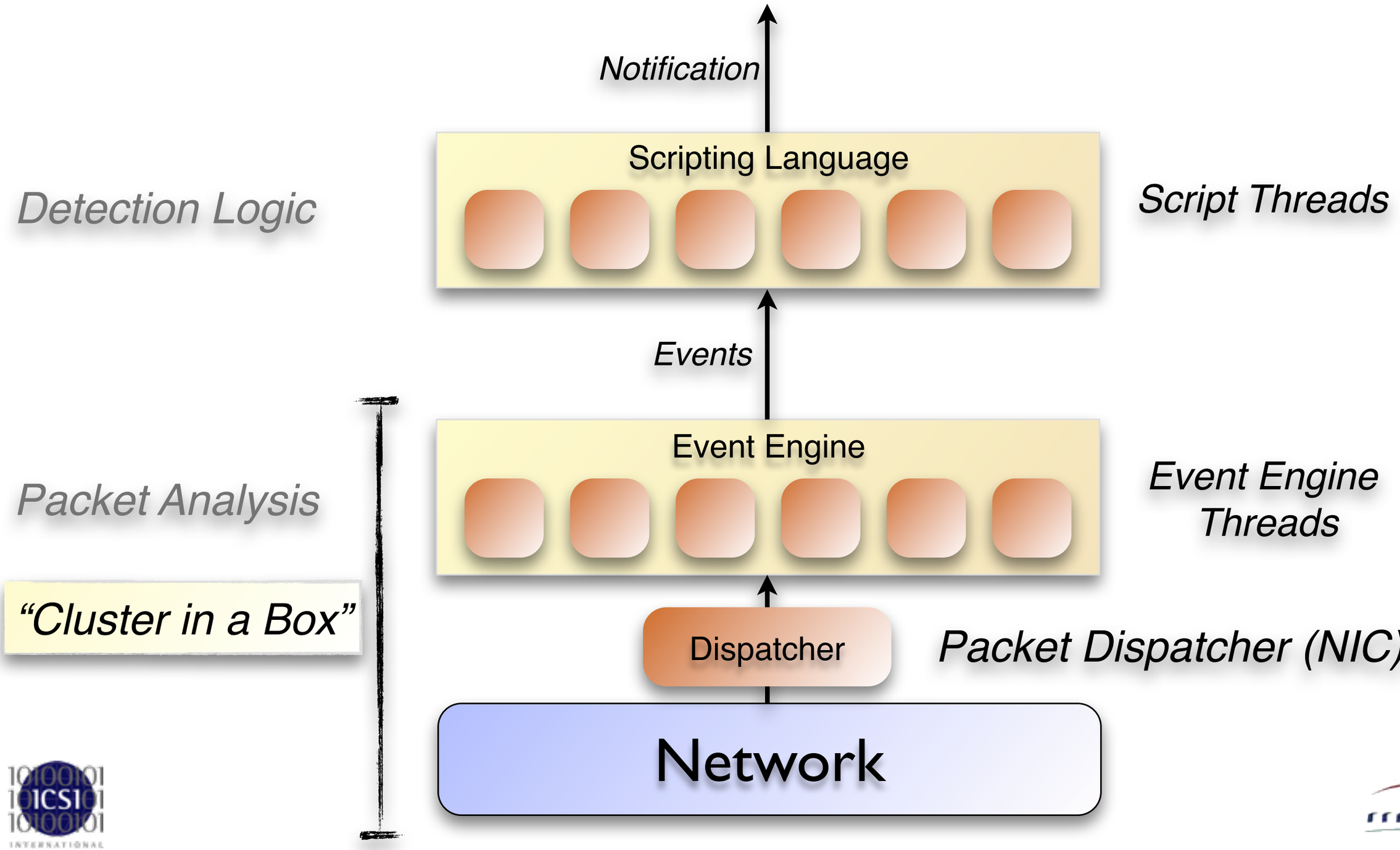
*Protocol Decoding*



# Architecture



# Architecture



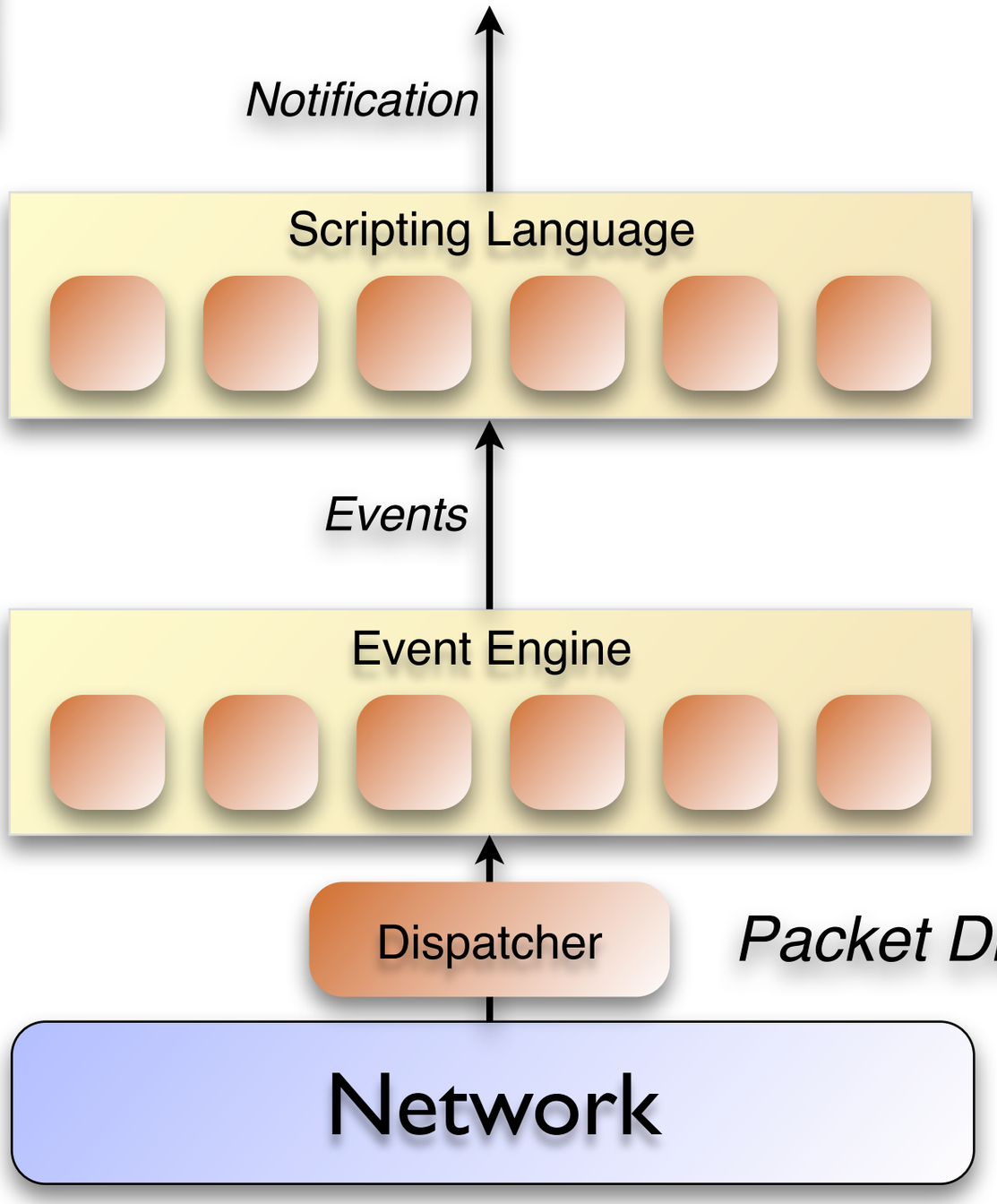
# Architecture

*How to parallelize a scripting language?*

*Detection Logic*

*Packet Analysis*

*"Cluster in a Box"*



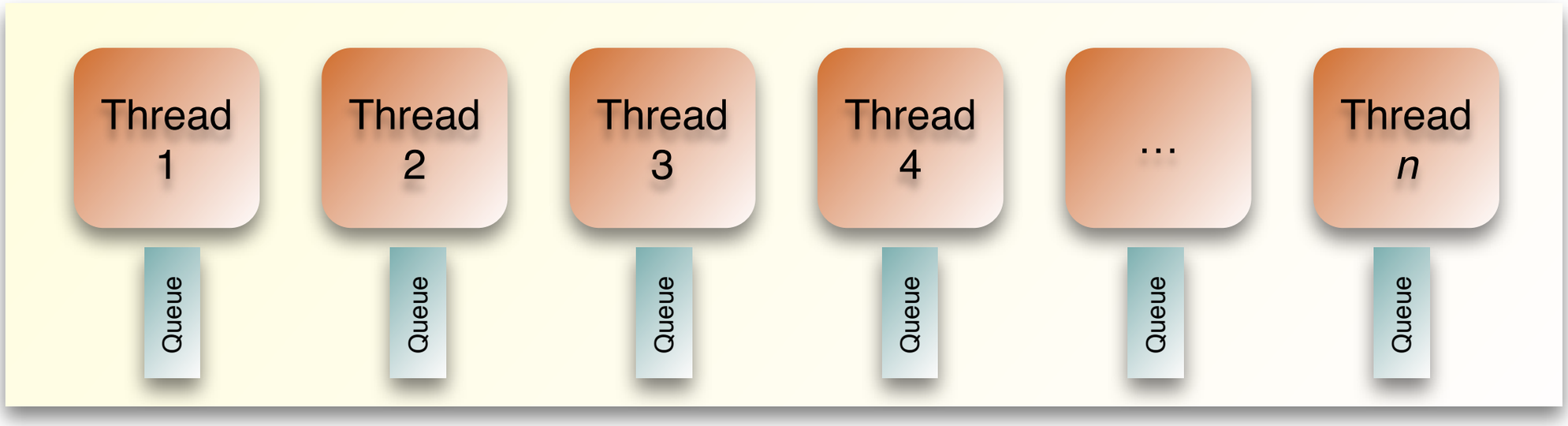

*Script Threads*

*Event Engine Threads*

*Packet Dispatcher (NIC)*

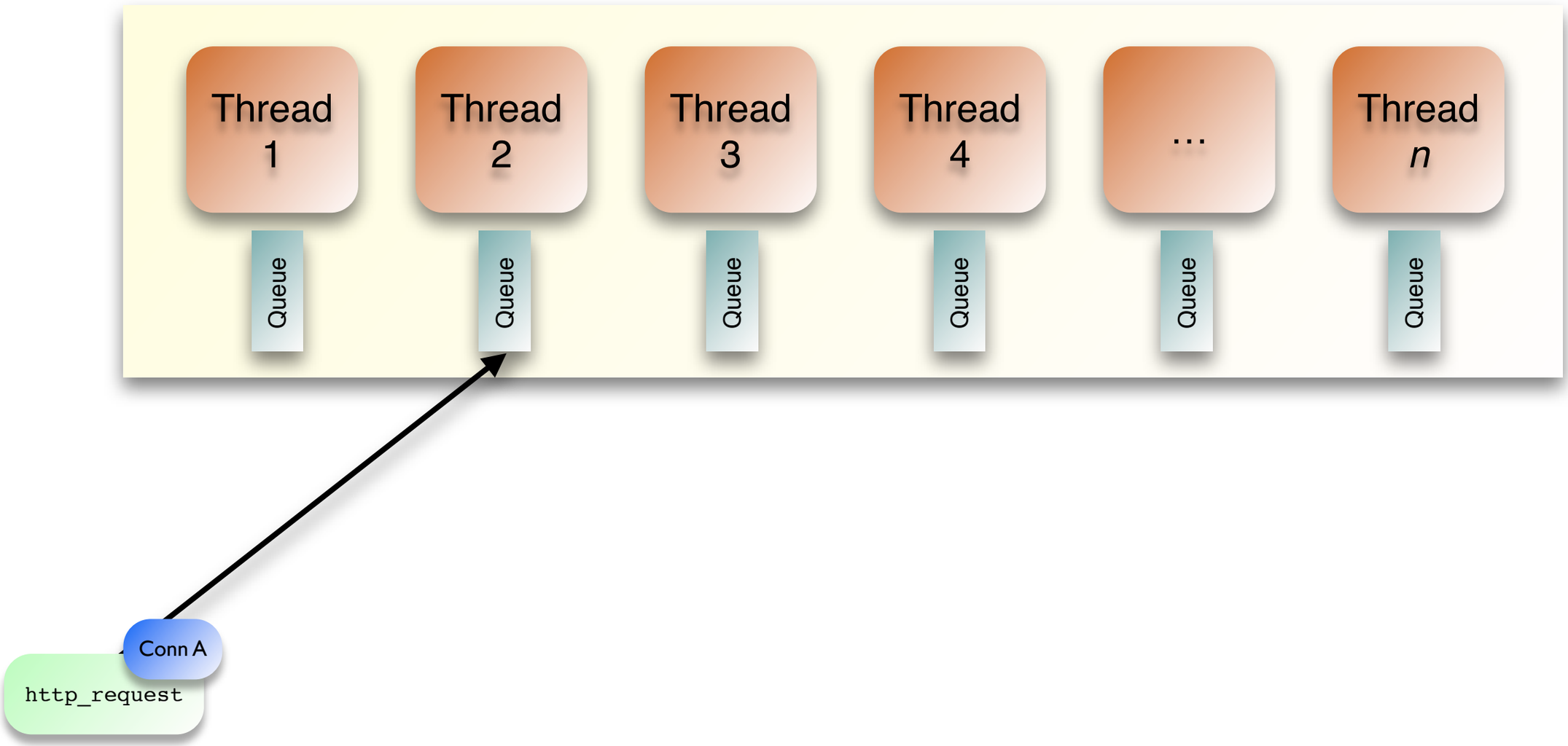
# Parallel Event Scheduling

## Threaded Script Interpreter



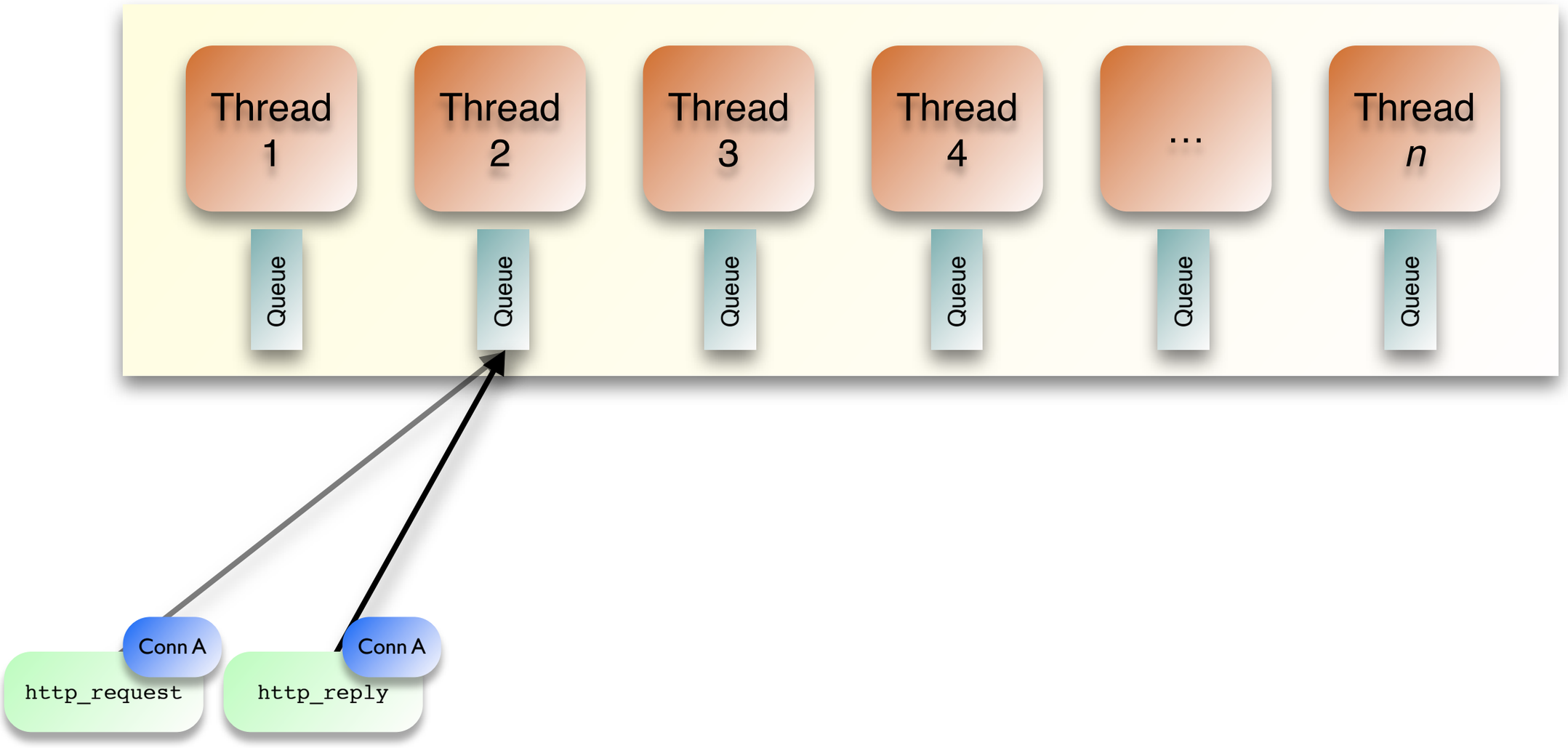
# Parallel Event Scheduling

## Threaded Script Interpreter



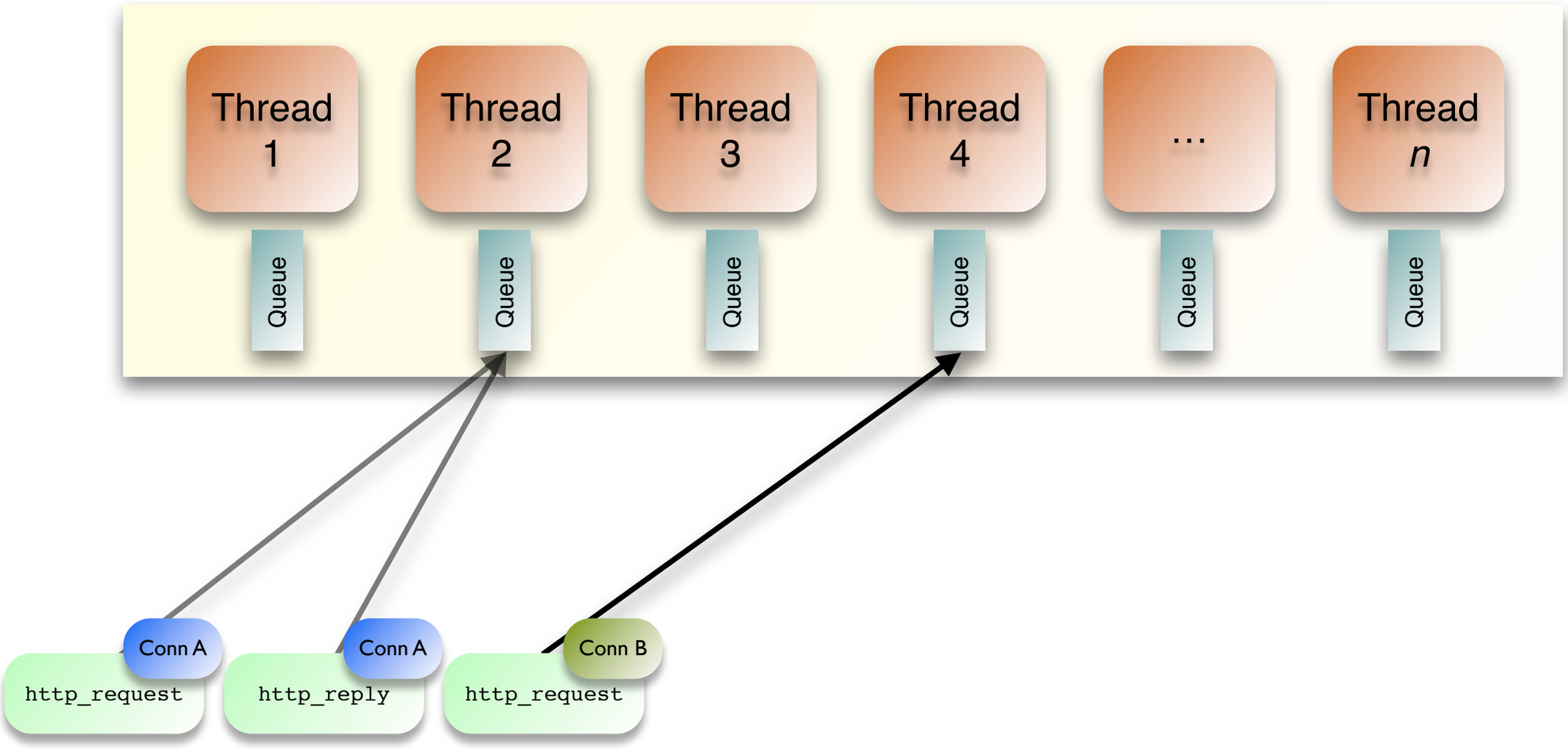
# Parallel Event Scheduling

## Threaded Script Interpreter



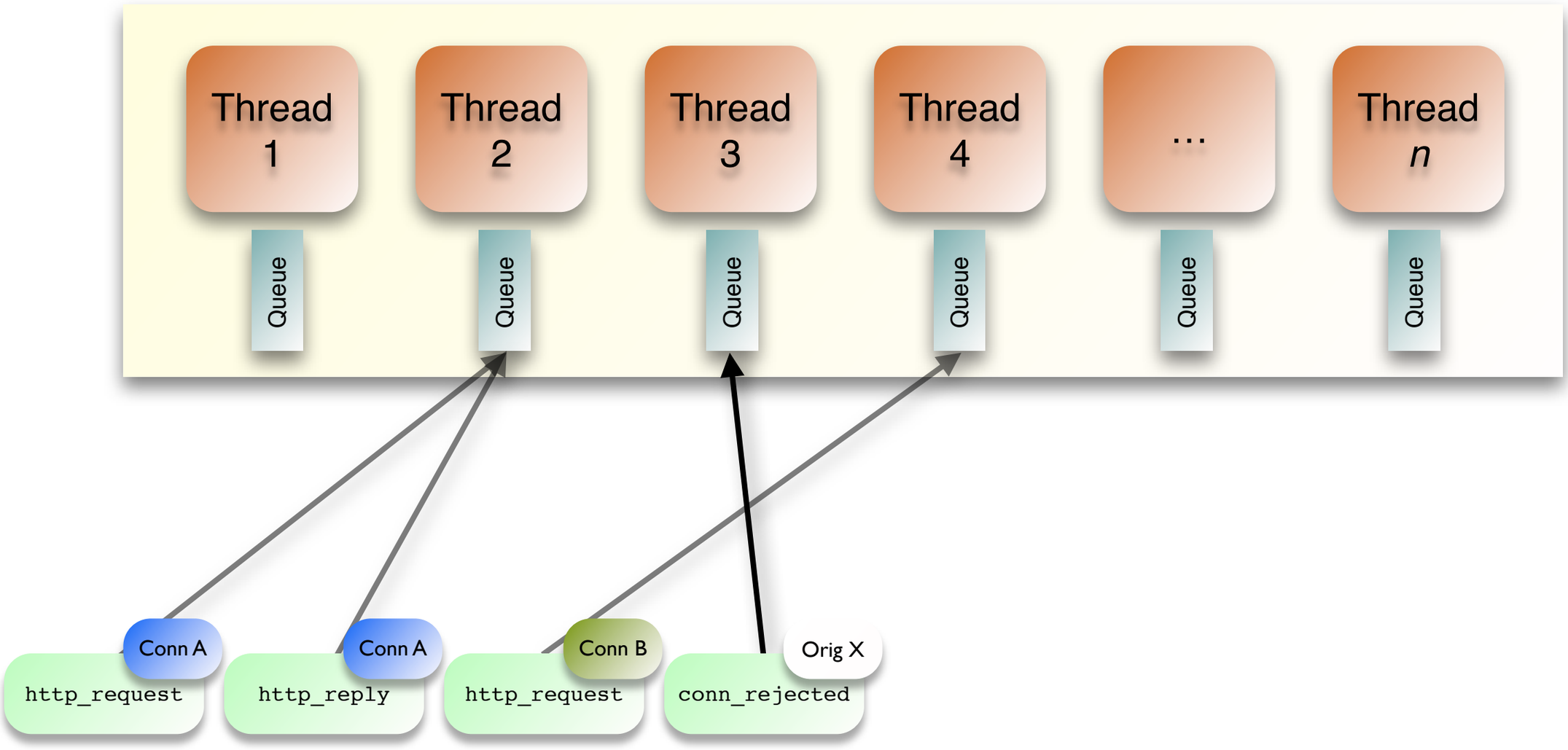
# Parallel Event Scheduling

## Threaded Script Interpreter



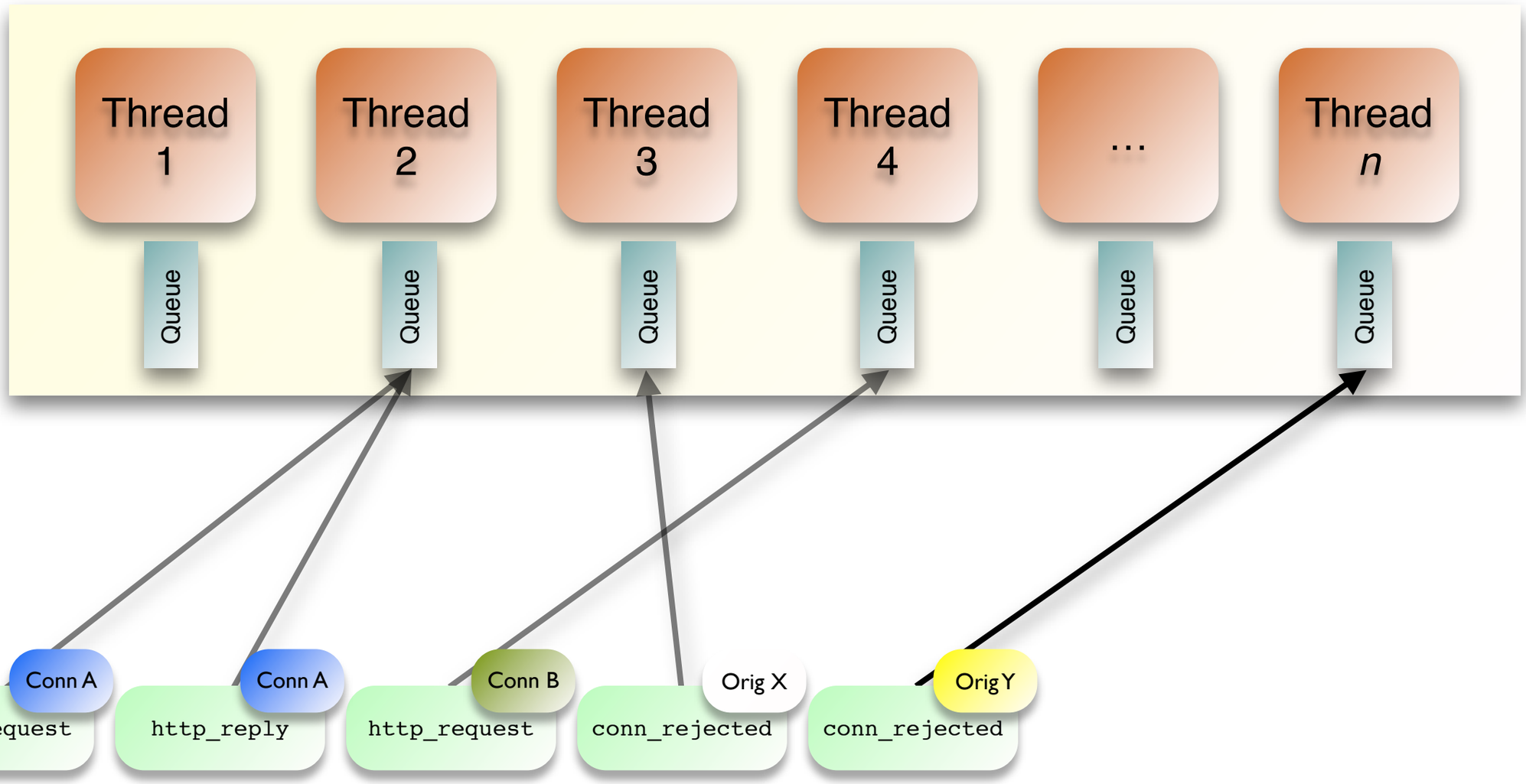
# Parallel Event Scheduling

## Threaded Script Interpreter



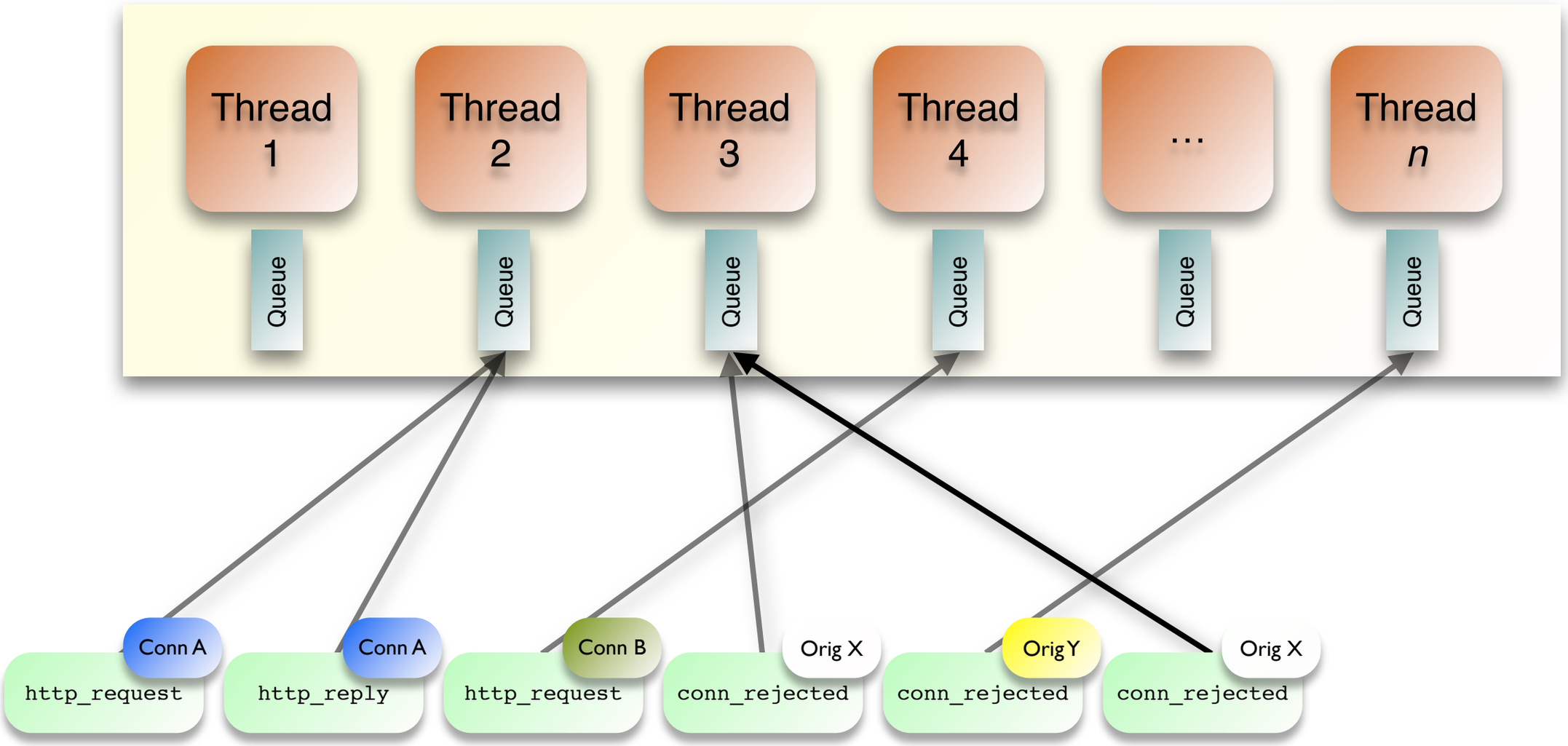
# Parallel Event Scheduling

## Threaded Script Interpreter



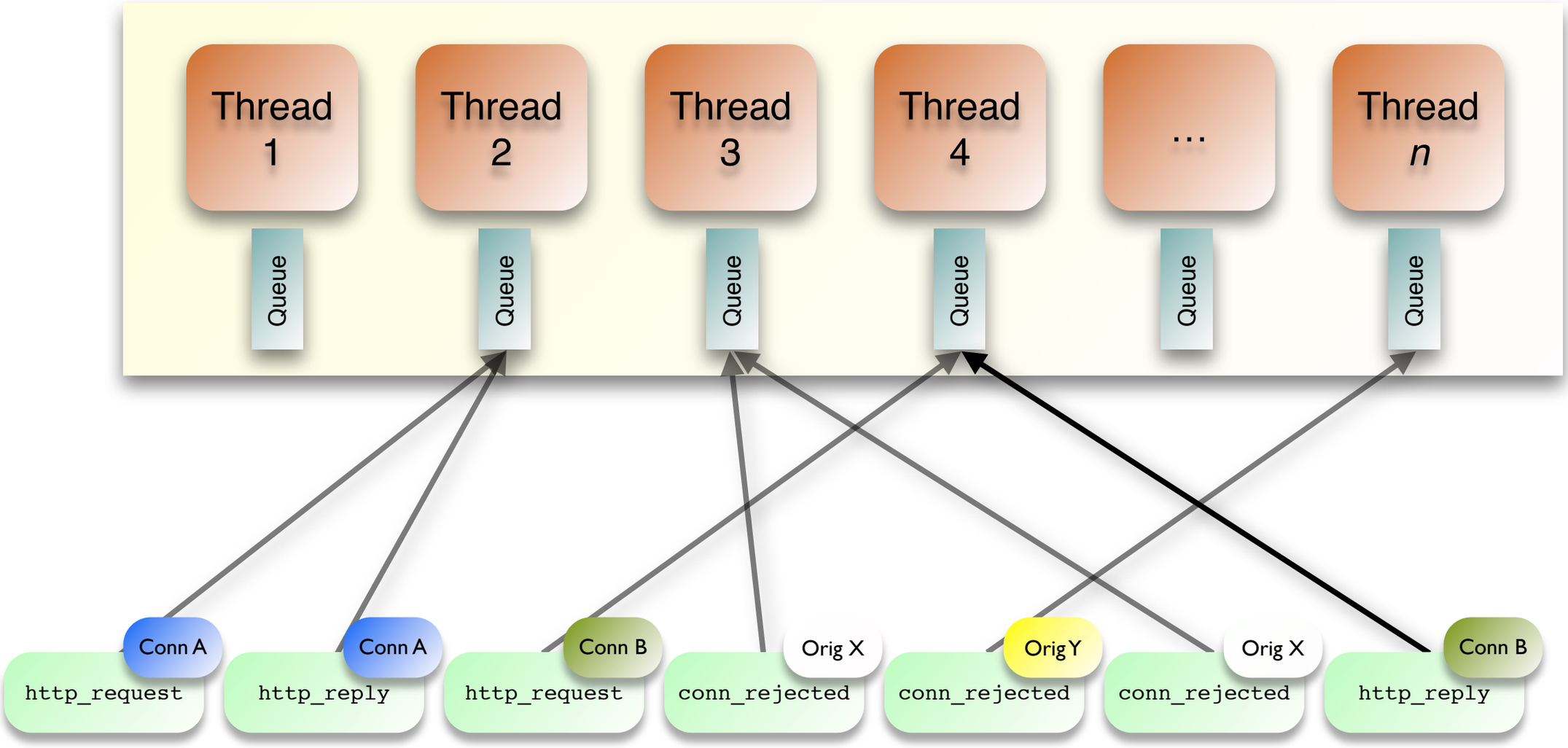
# Parallel Event Scheduling

## Threaded Script Interpreter



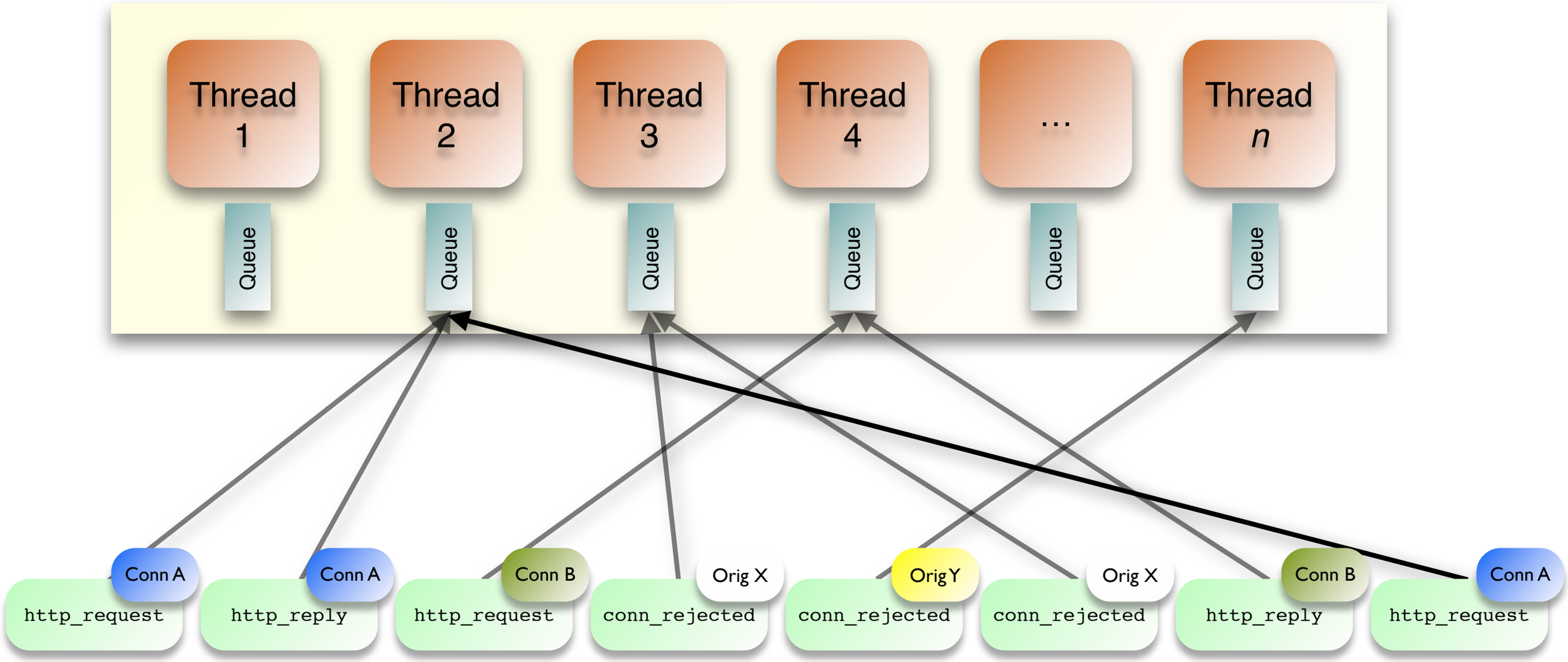
# Parallel Event Scheduling

## Threaded Script Interpreter



# Parallel Event Scheduling

## Threaded Script Interpreter



# New Platform: Abstract Machine

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

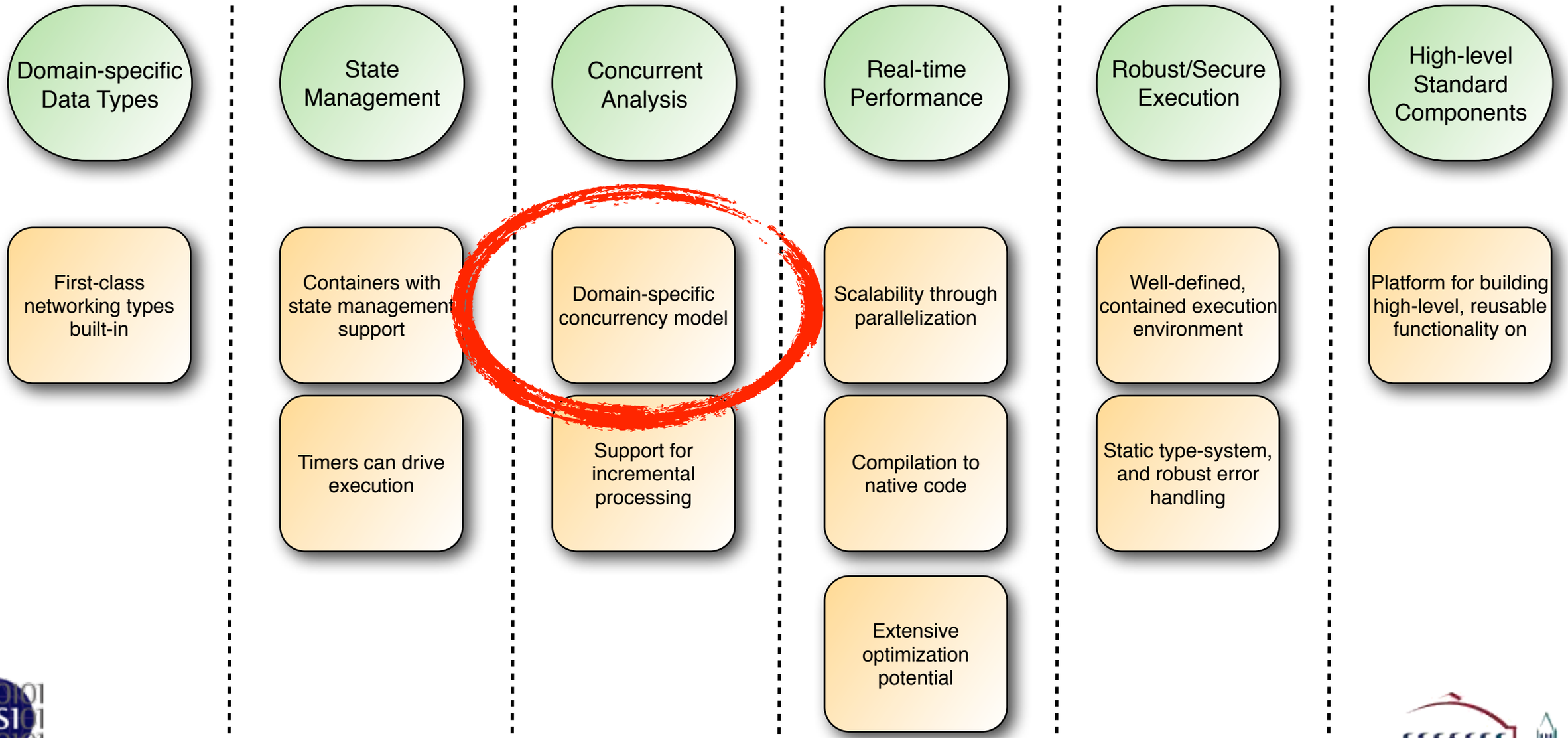
# New Platform: Abstract Machine

## 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 | | |

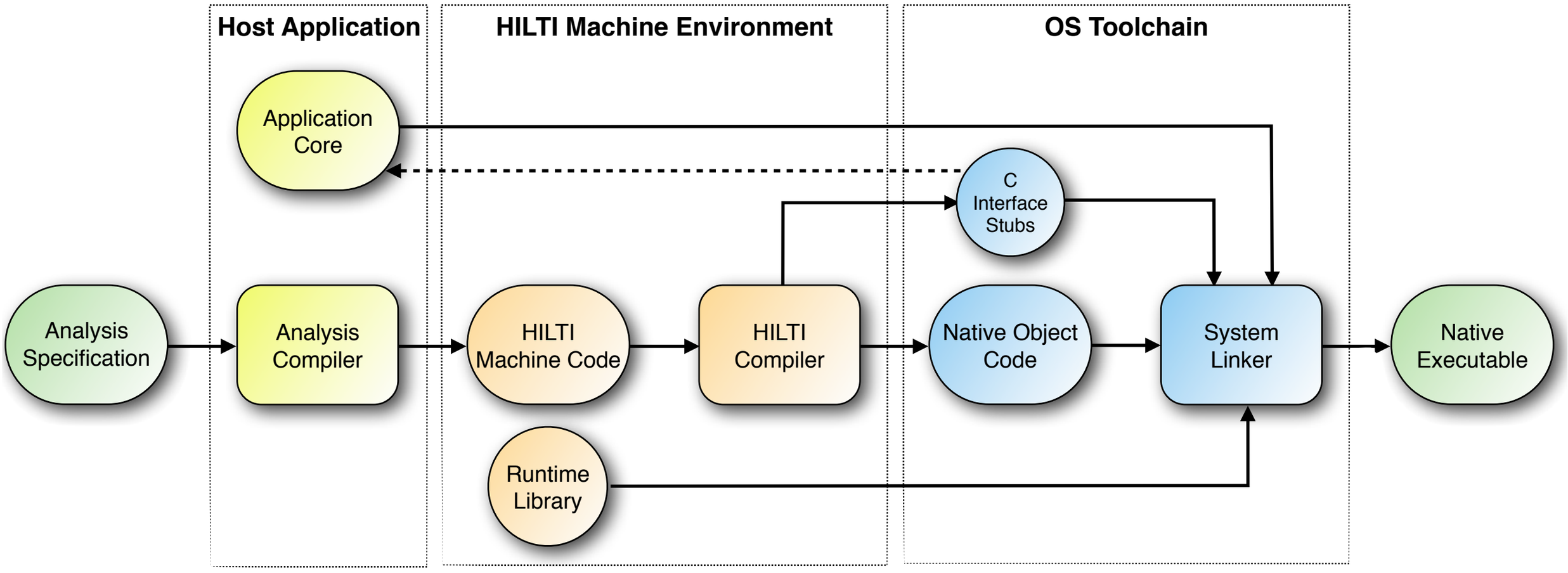
# New Platform: Abstract Machine

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



# HILTI Toolchain

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



# HILTI Goals

# HILTI Goals

Compiling analyses into native code.

Performance and concurrency.

Backend for parser generation.

A “yacc” for network protocols.

Secure execution environment.

Sandboxed execution.

Platform for optimization.

Whole program analysis just-in-time.

Abstraction-layer for hardware capabilities.

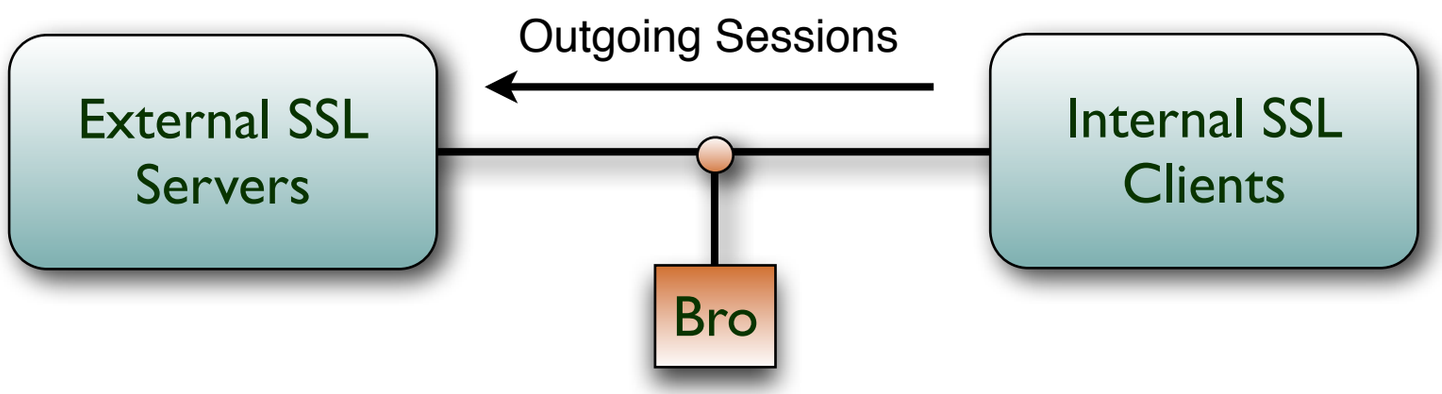
Integrate custom hardware transparently.

# Gaining a Global Perspective

## Using Bro for Large-scale Data Collection

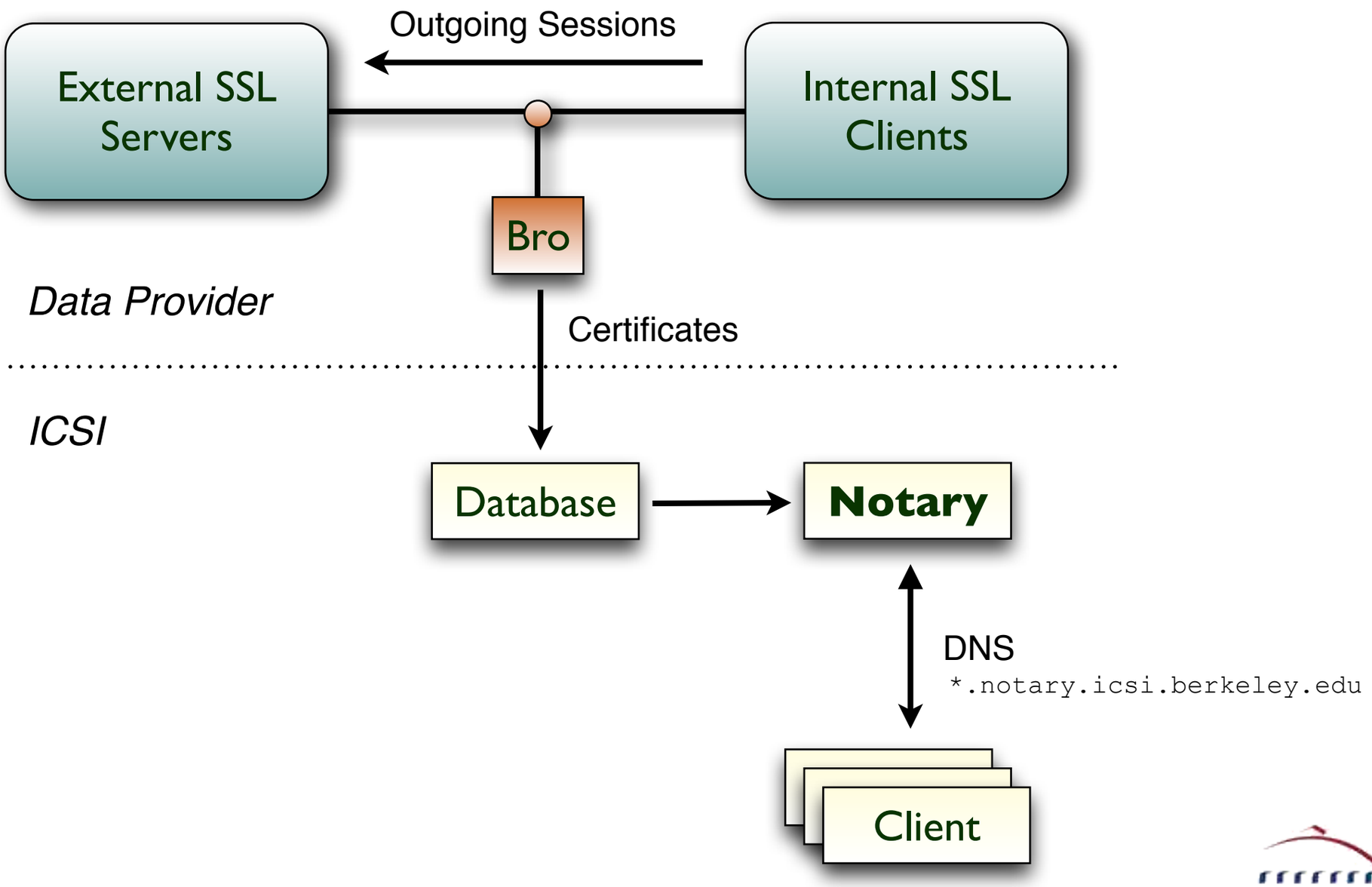
# ICSI SSL Notary

A global perspective on the SSL ecosytem.



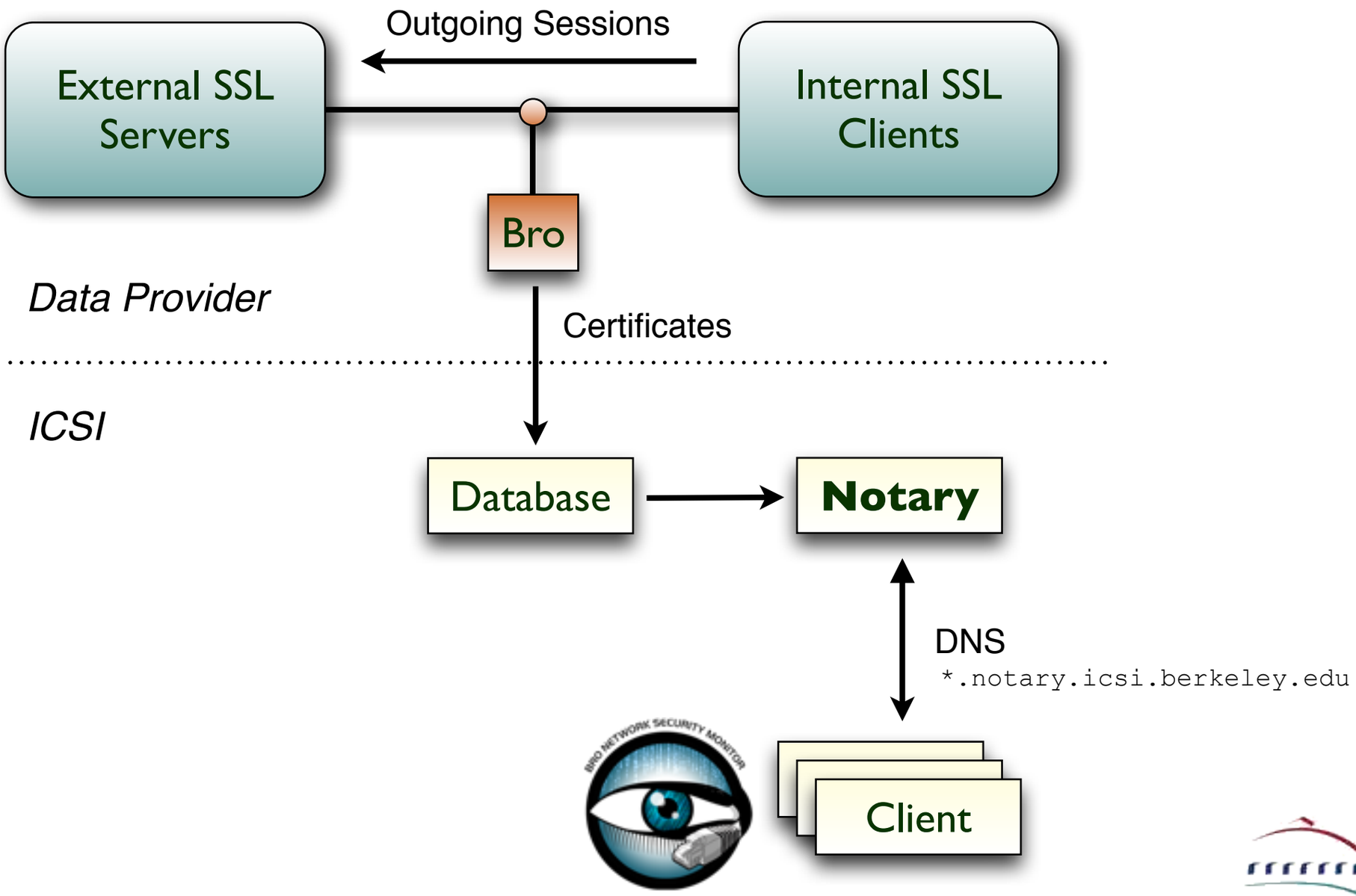
# ICSI SSL Notary

A global perspective on the SSL ecosytem.



# ICSI SSL Notary

A global perspective on the SSL ecosytem.



# SSL Certificate Notary

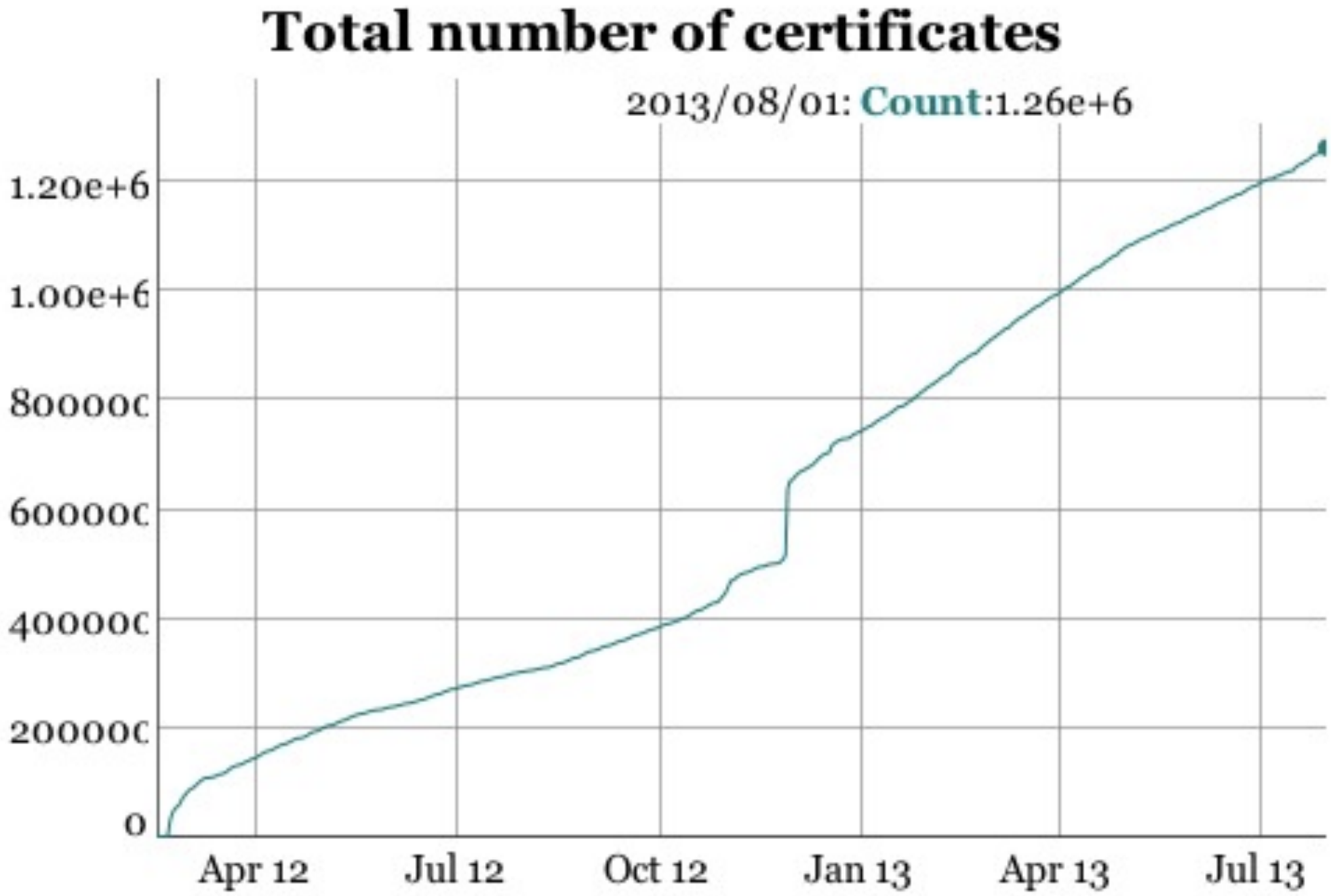
```
# dig txt C1956DC8A7[...]3358.notary.icsi.berkeley.edu
  [...]
  ;; ANSWER SECTION:
  C1956DC8A7[...]3358.notary.bro-ids.org. 2100 IN TXT \
      "15387 15450 64"
  [...]
```

*Days first seen*

*Most recent day seen*

*Number of days reported*

# Data Set


Total number of certificates
2013/08/01: Count:1.26e+6
1.20e+6
1.00e+6
800000
600000
400000
200000
0
Apr 12
Jul 12
Oct 12
Jan 13
Apr 13
Jul 13

# Notary: Data Providers

# Notary: Data Providers

| Site | Users | Certificates Total | Certificates Notary | Sessions | Duration (days) |
|---|---|---|---|---|---|
| University 1 | 90,000 | 50M | 972K | 13.1G | 579 |
| University 2 | 50,000 | 993K | 383K | 6.8G | 441 |
| University 3 | 3,000 | 14K | 8.8K | 10M | 142 |
| University 4 | 30,000 | 305K | 176K | 864M | 287 |
| University 5 | 100,000 | 699K | 330K | 7.9G | 291 |
| University 6 | 10,000 | 99K | 69K | 1.2G | 273 |
| Research Lab 1 | 250 | 564K | 44K | 163M | 511 |
| Research Lab 2 | 4,000 | 236K | 141K | 1.2G | 476 |
| Gov. Network | 50,000 | 166K | 161K | 720M | 318 |
| Backbone Net. | 30,000 | 34K | 32K | 636M | 141 |
| **Total** (Unique) | **314,250** | 52.2M | **1.3M** | **32.7G** | - |

August 2013

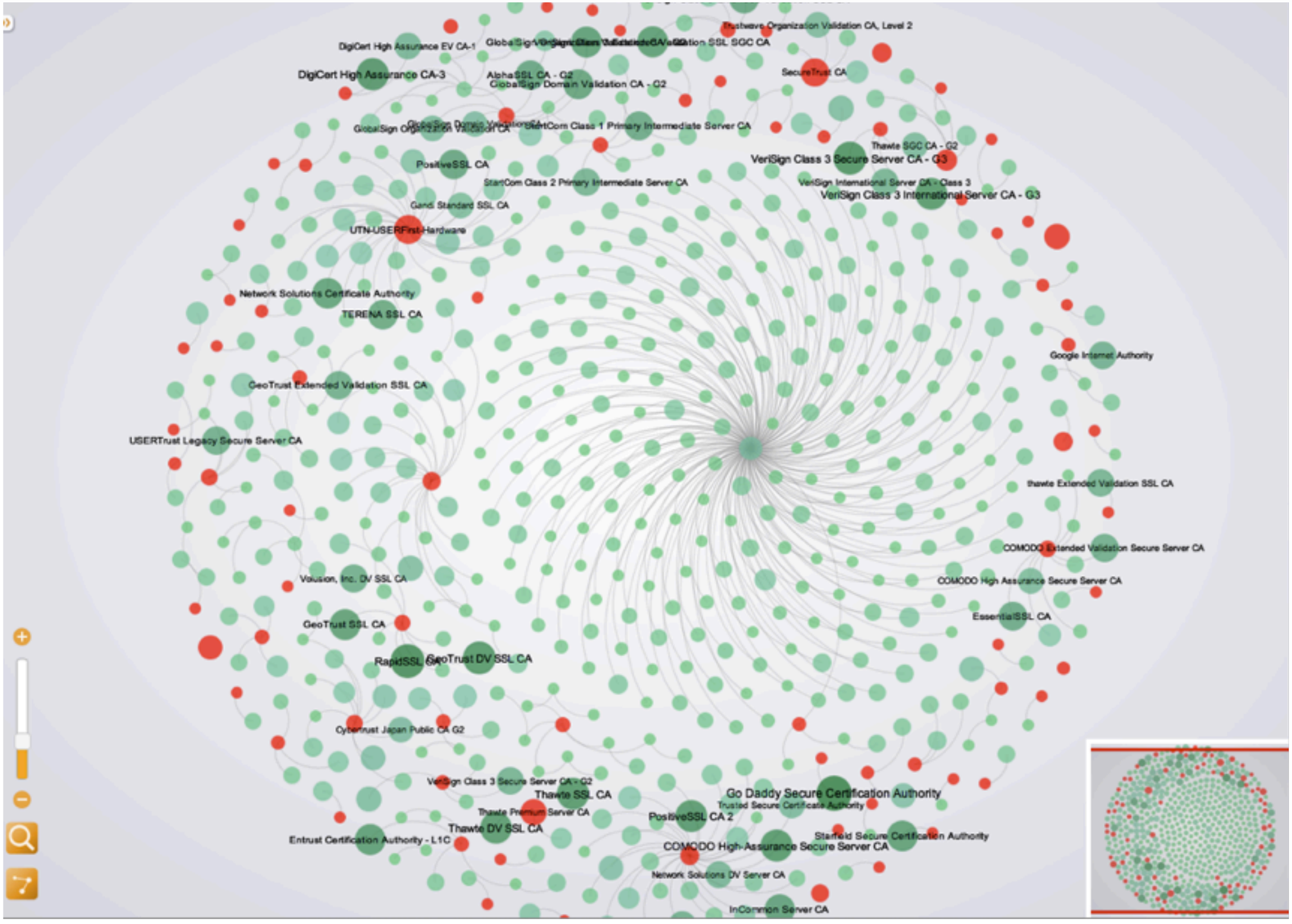
# Notary: Data Providers

| Site | Users | Certificates Total | Certificates Notary | Sessions | Duration (days) |
|---|---|---|---|---|---|
| University 1 | 90,000 | 50M | 872K | 13.1G | 579 |
| University 2 | 50,000 | | | | 441 |
| University 3 | 3,000 | | | | 142 |
| University 4 | 30,000 | | | | 287 |
| University 5 | 100,000 | | | | 291 |
| University 6 | 10,000 | | | | 273 |
| Research Lab 1 | 250 | | | | 511 |
| Research Lab 2 | 4,000 | | | | 476 |
| Gov. Network | 50,000 | | | | 318 |
| Backbone Net. | 30,000 | | | | 141 |
| **Total** (Unique) | **314,250** | | | | - |

**Collected Features**

- Server Certificate
- Available ciphers
- Client SSL Extensions
- Server SSL Extensions
- Hash(Client, Server)
- Hash(Client, SNI)
- Hash(Client Session ID)
- Hash(Server Session ID)
- Selected Cipher
- Server Name Indication
- Ticket Lifetime Hint
- Timestamp
- SSL Protocol Version

# http://notary.icsi.berkeley.edu



# Summary

# Summary

New Attack Trends.

Underground economy, targetted attacks.

# Summary

## New Attack Trends.

Underground economy, targetted attacks.

## Bro.

From research to operations.

# Summary

## New Attack Trends.

Underground economy, targetted attacks.

## Bro.

From research to operations.

## Performance.

Scaling Bro Clusters to 100 Gbits/sec.

# Summary

## New Attack Trends.

Underground economy, targetted attacks.

## Bro.

From research to operations.

## Performance.

Scaling Bro Clusters to 100 Gbits/sec.

## Large-scale data collection.

Analyzing the global SSL landscape.

# Thanks for your attention.


Robin Sommer

*International Computer Science Institute, &*
*Lawrence Berkeley National Laboratory*

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