Monitoring Network Security with the Open-Source Bro NIDS

Robin Sommer
Lawrence Berkeley National Laboratory &
International Computer Science Institute

rsommer@lbl.gov
http://www.icir.org

DOE Network Security Monitoring Technical Summit at Jefferson Lab

Outline

- Overview of Bro's design & architecture
- Topics
  - Dynamic Protocol Detection
  - Bro Cluster
  - Time Machine with Bro interface
- Outlook

BERKELEY LAB

ICSI
INTERNATIONAL COMPUTER SCIENCE INSTITUTE

The Bro NIDS

System Philosophy

- Bro is being developed at LBNL & ICSI since 1996
  - LBNL has been using Bro operationally for >10 years
  - It is one of the main components of the lab's network security infrastructure
- Bro provides a real-time network analysis framework
  - Primary a network intrusion detection system (NIDS)
  - However it is also used for pure traffic analysis
- Focus is on
  - Application-level semantic analysis (rather than analyzing individual packets)
  - Tracking information over time
- Strong separation of mechanism and policy
  - The core of the system is policy-neutral (no notion of "good" or "bad")
  - User provides local site policy

System Philosophy (2)

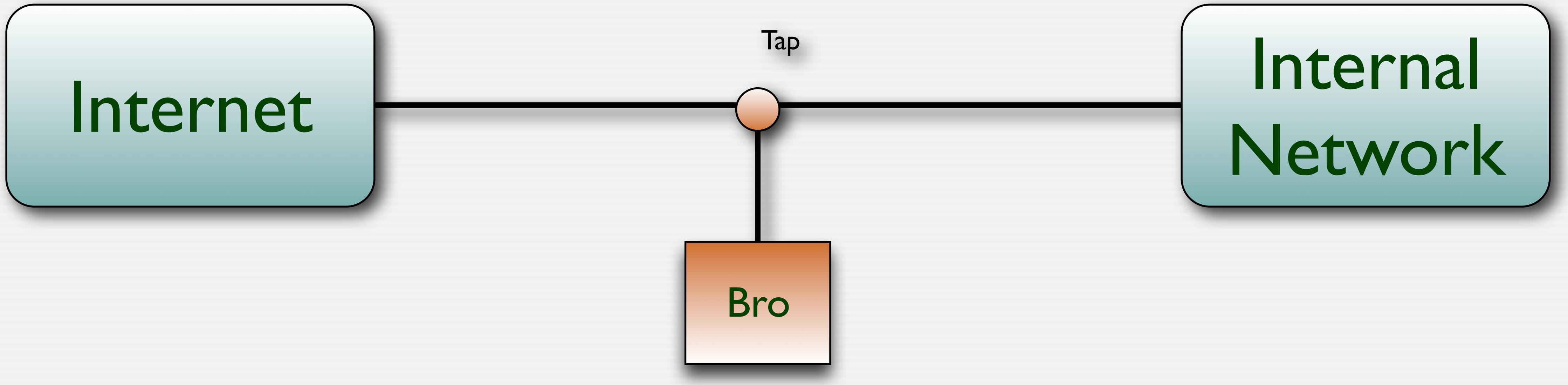
- Operators *program* their policy
  - Not really meaningful to talk about what Bro detects "by default"
- Analysis model is *not* signature matching
  - Bro is fundamentally different from, e.g., Snort (though it *can* do signatures as well)
- Analysis model is *not* anomaly detection
  - Though it does support such approaches (and others) in principle
- System thoroughly logs all activity
  - It does not just alert
  - Logs are invaluable for forensics

Target Environments

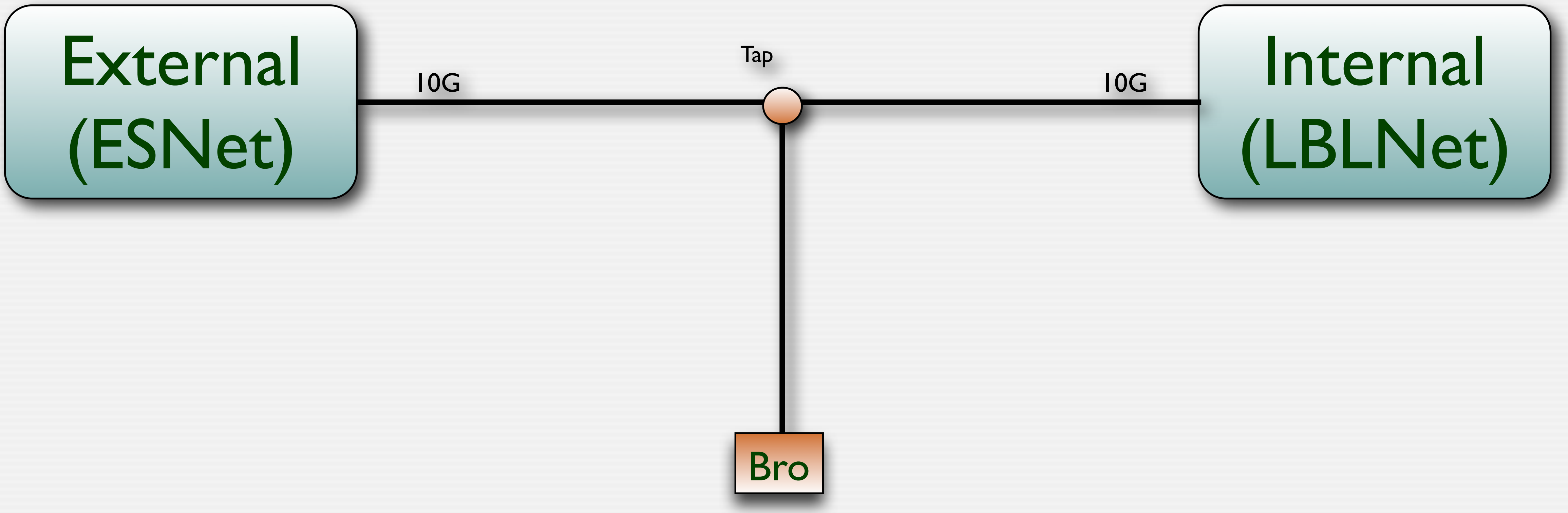
- Bro is specifically well-suited for scientific environments
  - Extremely useful in networks with liberal (“default allow”) policies
  - High-performance on commodity hardware
  - Supports intrusion prevention schemes
  - Open-source (BSD license)
- It does however require some effort to use effectively
  - Pretty complex, script-based system
  - Requires understanding of the network
  - No GUI, just ASCII logs
  - Only partially documented
  - Lacking resources to fully polish the system
- Development is primarily driven by *research*
  - However, our focus is operational use; we invest much time into “practical” issues
  - Want to bridge gap between research and operational deployment

Bro Deployment

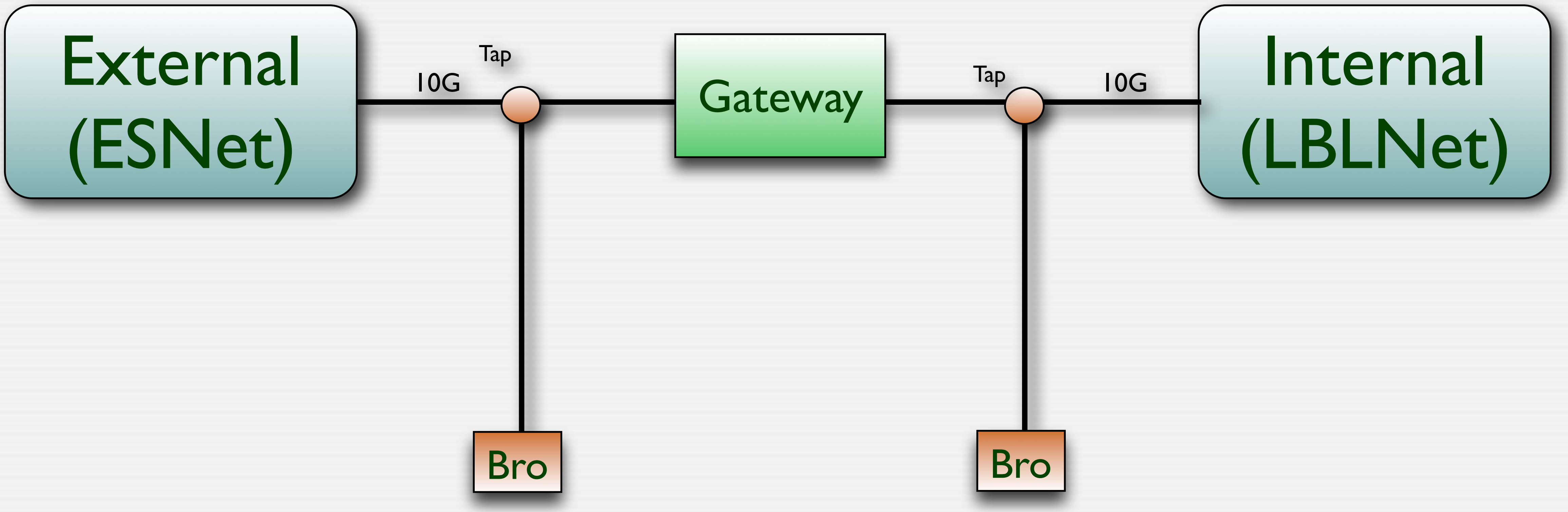
- Bro is typically deployed at a site's upstream link
  - Monitors all external packets coming in or going out
  - Deployment similar to other NIDS



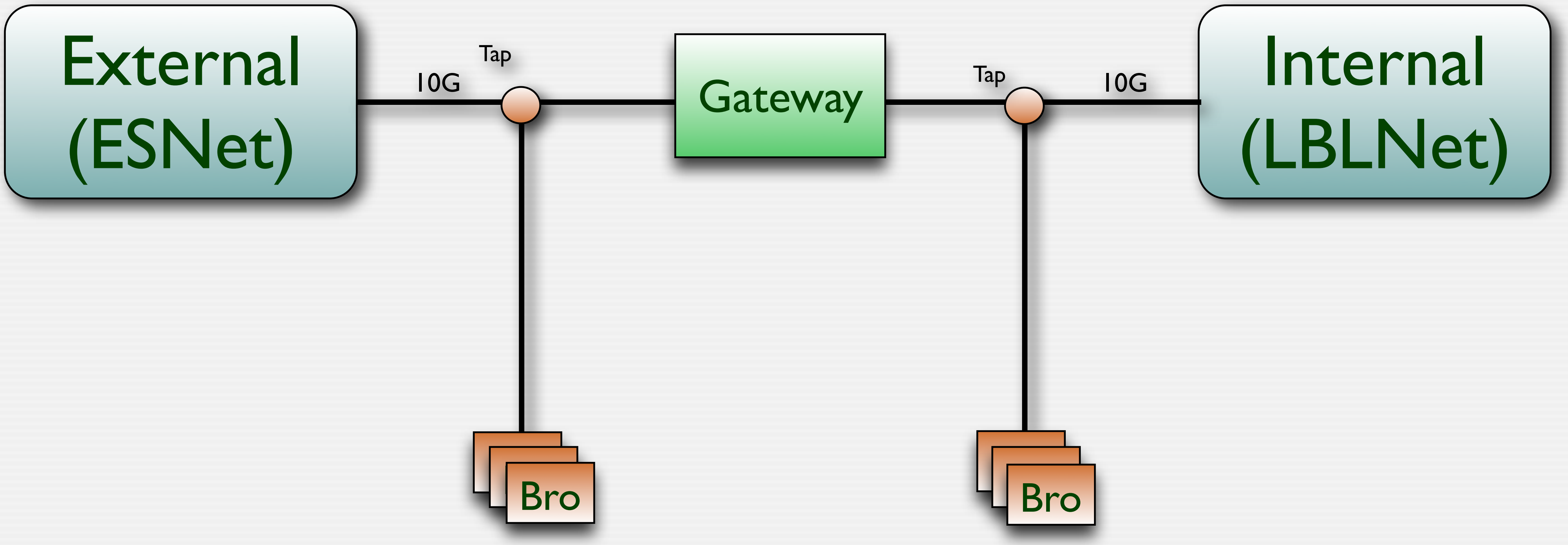
LBNL's Bro Setup



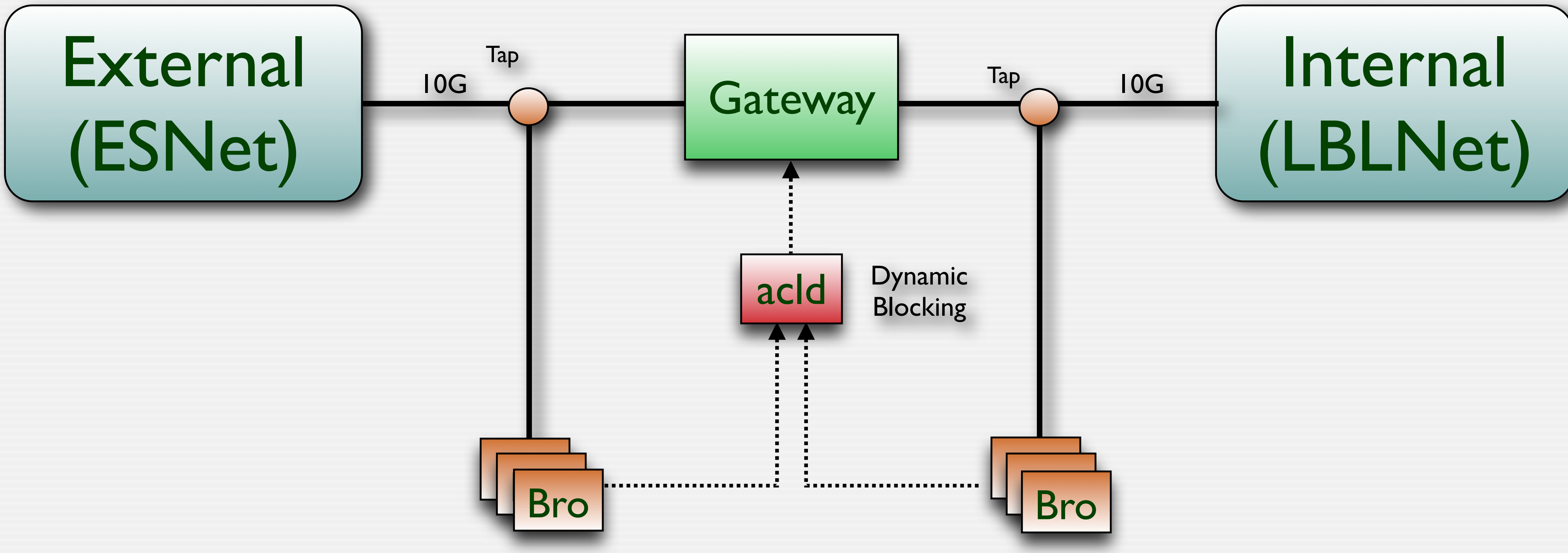
LBNL's Bro Setup



LBNL's Bro Setup

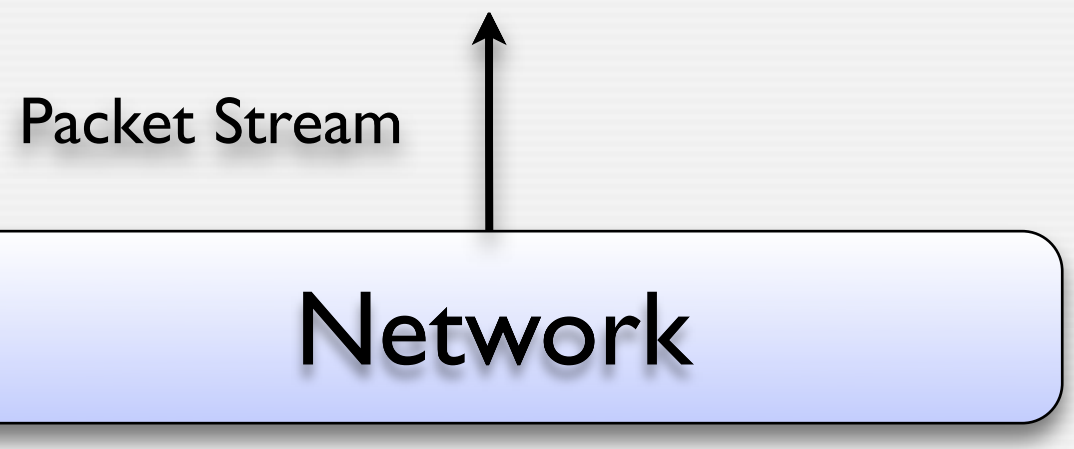


LBNL's Bro Setup

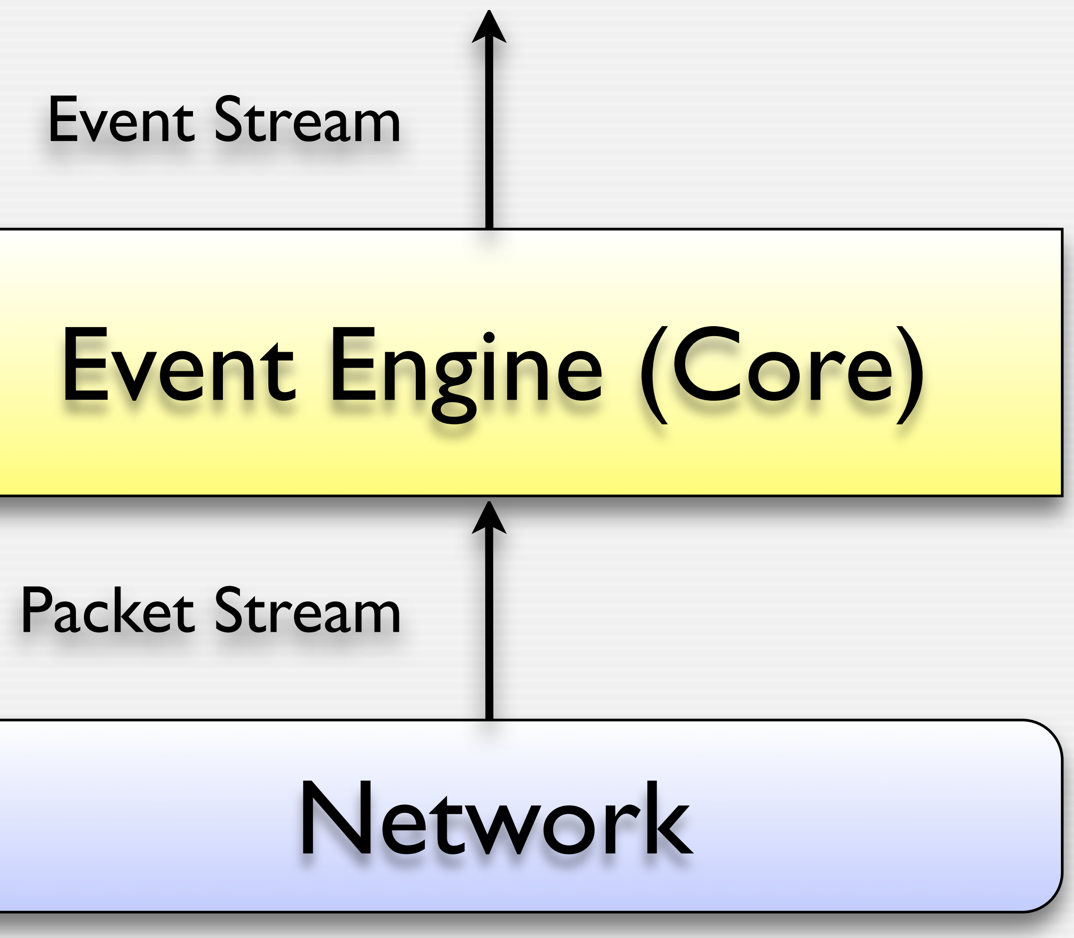


Bro blocks several thousands addresses per day!

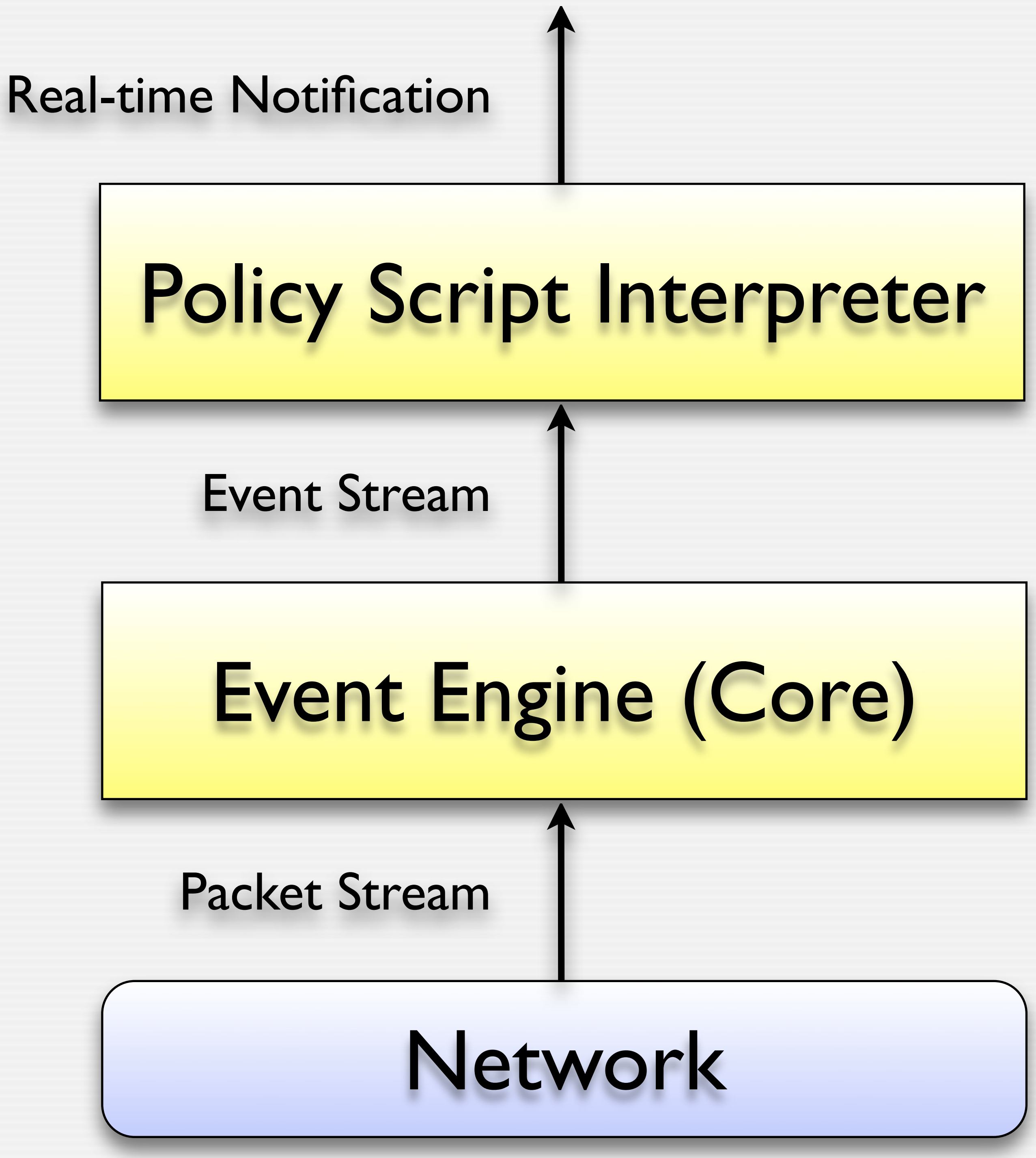
Architecture



Architecture



Architecture



Event-Engine

- Event-engine is written in C++
- Performs *policy-neutral* analysis
  - Turns low-level activity into high-level events
  - Examples: `connection_established, http_request`
  - Events are annotated with context (e.g., IP addresses, URL)
- Contains *analyzers* for >30 protocols, including
  - ARP, IP, ICMP, TCP, UDP
  - DCE-RPC, DNS, FTP, Finger, Gnutella, HTTP, IRC, Ident, NCP, NFS, NTP, NetBIOS, POP3, Portmapper, RPC, Rsh, Rlogin, SMB, SMTP, SSH, SSL, SunRPC, Telnet
- Analyzers generate ~300 types of events

Policy Scripts

- Scripts process event stream, incorporating ...
  - ... context from past events
  - ... site's local security policy
- Scripts take actions
  - Generating alerts via syslog or mail
  - Executing program as a form of response
  - Recording activity to disk

Example Log: Connection Summaries

- One-line summaries for all TCP connections
- Most basic, yet also one of the most useful analyzers

```
> bro -r trace tcp

Time                  Duration  Source           Destination
1144876596.658302     1.206521  192.150.186.169  62.26.220.2 \
  http  53052    80        tcp     874       1841      SF     X
  Serv  SrcPort  DstPort   Proto   SrcBytes  DstBytes  State  Dir
```

LBNL has connection logs for every connection attempt since June 94!

Example Log: HTTP Session

```
1144876588.30 %2 start 192.150.186.169:53041 > 195.71.11.67:80
1144876588.30 %2 GET /index.html (200 "OK" [57634] www.spiegel.de)
1144876588.30 %2 > HOST:  www.spiegel.de
1144876588.30 %2 > USER-AGENT:  Mozilla/5.0 (Macintosh; PPC Mac OS ...
1144876588.30 %2 > ACCEPT:  text/xml,application/xml,application/xhtml ...
1144876588.30 %2 > ACCEPT-LANGUAGE:  en-us,en;q=0.7,de;q=0.3
[...]
1144876588.77 %2 < SERVER:  Apache/1.3.26 (Unix) mod_fastcgi/2.2.12
1144876588.77 %2 < CACHE-CONTROL:  max-age=120
1144876588.77 %2 < EXPIRES:  Wed, 12 Apr 2006 21:18:28 GMT
[...]
1144876588.77 %2 <= 1500 bytes: "<!-- Vignette StoryServer 5.0 Wed Apr..."
1144876588.78 %2 <= 1500 bytes: "r "http://spiegel.ivwbox.de" r..."
1144876588.78 %2 <= 1500 bytes: "icon.ico" type="image/ico">^M^J ..."
1144876588.94 %2 <= 1500 bytes: "erver 5.0 Mon Mar 27 15:56:55 ..."
[...]
```

Script Example: Tracking SSH Hosts

```
global ssh_hosts: set[addr];

event connection_established(c: connection)
    {
    local responder = c$id$resp_h; # Responder's address
    local service = c$id$resp_p;   # Responder's port

    if ( service != 22/tcp )
        return; # Not SSH.

    if ( responder in ssh_hosts )
        return; # We already know this one.

    add ssh_hosts[responder]; # Found a new host.
    alarm fmt("New SSH host found: %s", responder);
    }
```

Expressing Policy

- Scripts are written in custom, domain-specific language
  - Bro ships with 20K+ lines of script code
  - Default scripts detect attacks & log activity extensively
- Language is
  - Procedural
  - Event-based
  - Strongly typed
  - Rich in types
    - Usual script-language types, such as tables and sets
    - Domain-specific types, such as addresses, ports, subnets
  - Supporting state management (expiration, timers, etc.)
  - Supporting communication with other Bro instances

Port-independent Protocol Analysis with Dynamic Protocol Detection

Port-based Protocol Analysis

- Bro has lots of application-layer analyzers
- But which protocol does a connection use?
- Traditionally NIDS rely on ports
  - Port 80? Oh, that's HTTP.
- Obviously deficient in two ways
  - There's non-HTTP traffic on port 80 (firewalls tend to open this port...)
  - There's HTTP on ports other than port 80
- Particularly problematic for security monitoring
  - Want to know if somebody avoids the well-known port

Port-independent Analysis

- Look at the *payload* to see what is, e.g., HTTP
- Analyzers already know how a protocol looks like
  - Leverage existing protocol analyzers
  - Let each analyzer *try to parse* the payload
    - If it succeeds, great!
    - If not, then it's actually another protocol
- Ideal setting: *for every connection, try all analyzers*
- However, performance is prohibitive
  - Can't parse 10000s of connections in parallel with all analyzers

Making it realistic ...

- Bro uses byte patterns to *prefilter* connections
  - An HTTP signature looks for *potential* uses of HTTP
  - Then the HTTP analyzer verifies by trying to parse the payload
  - Signatures can be loose because false positives are inexpensive (no alerts!)
- Other NIDS often ship with protocol signatures
  - These directly generate alerts (imagine reporting all non-80 HTTP conns!)
  - These do not trigger protocol-layer semantic analysis (e.g., extracting URLs)
- In Bro, a match triggers further analysis
- Main internal concept: analyzer trees
  - Each connection is associated with an analyzer tree

Application Example: Finding Bots

- IRC-based bots are a prevalent problem
  - Infected client machines accept commands from their "master"
  - Often IRC-based but not on port 6667
- Just detecting IRC connections not sufficient
  - Often there is legitimate IRC on ports other than 6667
- DPD allows to analyze all IRC sessions *semantically*
  - Looks for typical patterns in NICK and TOPIC
  - Reports if it finds IRC sessions showing both such NICKs and TOPICs
- Very reliable detection of bots
  - Munich universities use it to actively block internal bots automatically

Application Example: FTP Data (2)

```
xxx.xxx.xxx.xxx/2373 > xxx.xxx.xxx.xxx/5560 start
response (220 Rooted Moron Version 1.00 4 WinSock ready...)
USER ops (logged in)
SYST (215 UNIX Type: L8)
[...]
LIST -al (complete)
TYPE I (ok)
SIZE stargate.atl.s02e18.hdtv.xvid-tvd.avi (unavail)
PORT xxx,xxx,xxx,xxx,xxx,xxx (ok)
STOR stargate.atl.s02e18.hdtv.xvid-tvd.avi, NOOP (ok)
ftp-data video/x-msvideo `RIFF (little-endian) data, AVI'
[...]
response (226 Transfer complete.)
[...]
QUIT (closed)
```

The Bro Cluster

Scalable, Stateful Detection on Commodity Hardware

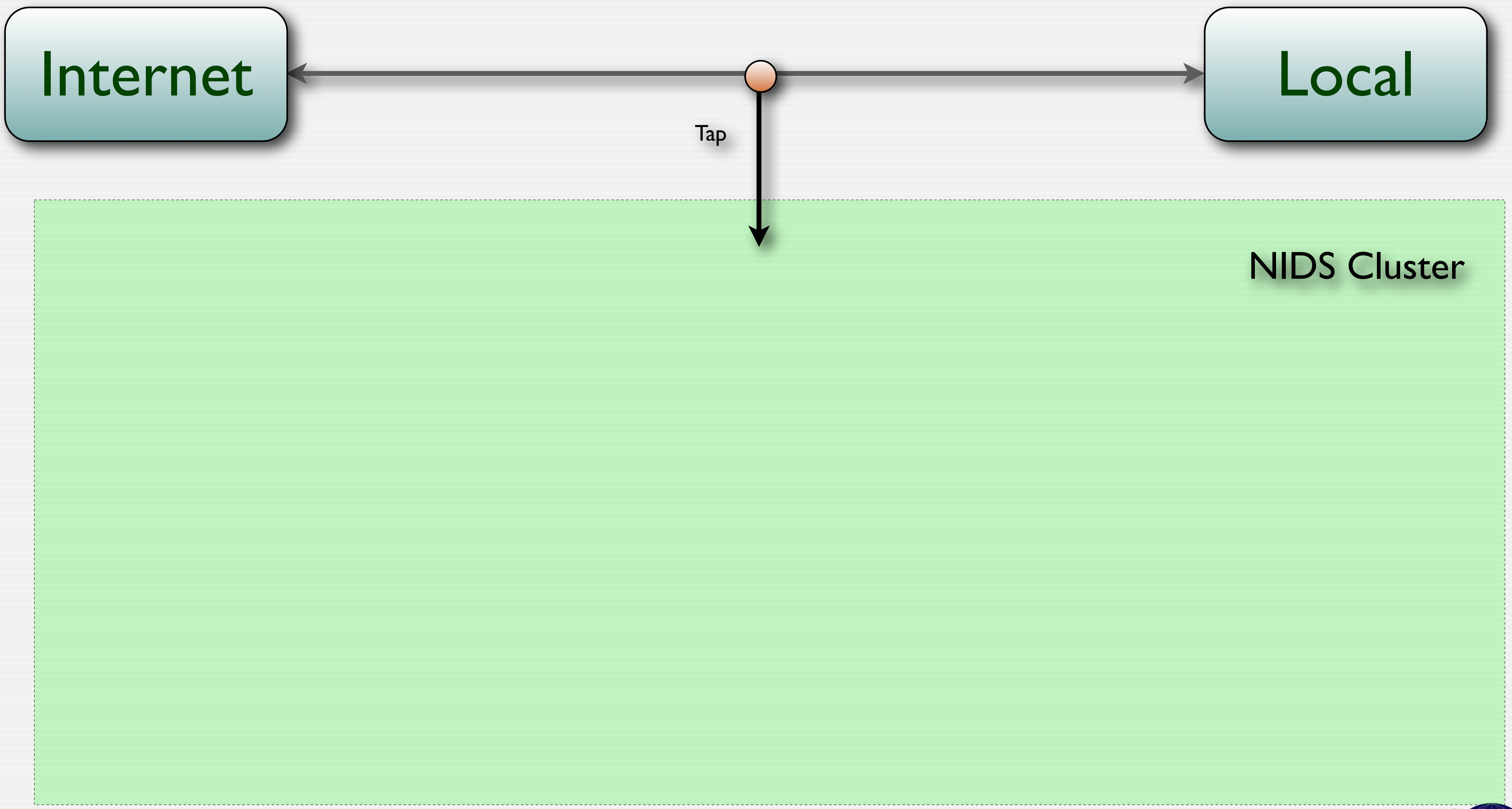
Motivation

- NIDSs have reached their limits on commodity hardware
  - Keep needing to do *more analysis* on *more data* at *higher speeds*
  - Analysis gets richer over time, as attacks get more sophisticated
  - However, single CPU performance is not growing anymore the way it used to
  - Single NIDS instance (Snort, Bro) cannot cope with >=1Gbps links
- Key to overcome current limits is *parallel analysis*
  - Volume is high but composed of many *independent* tasks
  - Need to exploit parallelism to cope with load

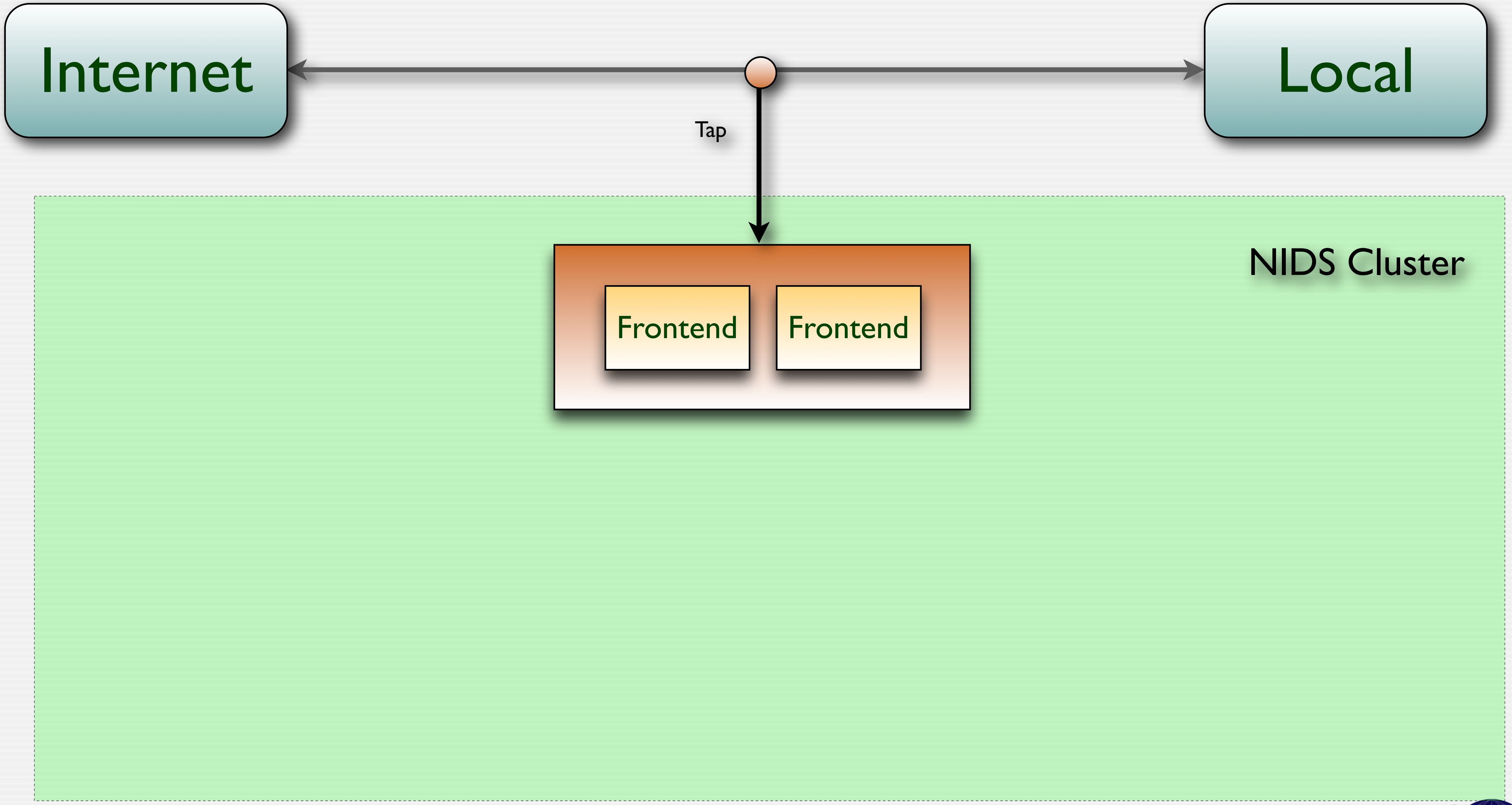
The Bro Cluster

- Load-balancing approach: use many boxes instead of one
- The Bro cluster works *transparently* like a single NIDS
  - Gives same results as single NIDS would if it could analyze all traffic
  - Correlation of low-level analysis
  - No loss in detection accuracy
  - Scalable to large number of nodes
  - Single system for user interface (log aggregation, configuration changes)
- Most NIDS provide support for multi-system setups
- However instances tend to work independently
  - Central manager collects alerts of independent NIDS instances
  - Aggregates results instead of correlating analysis

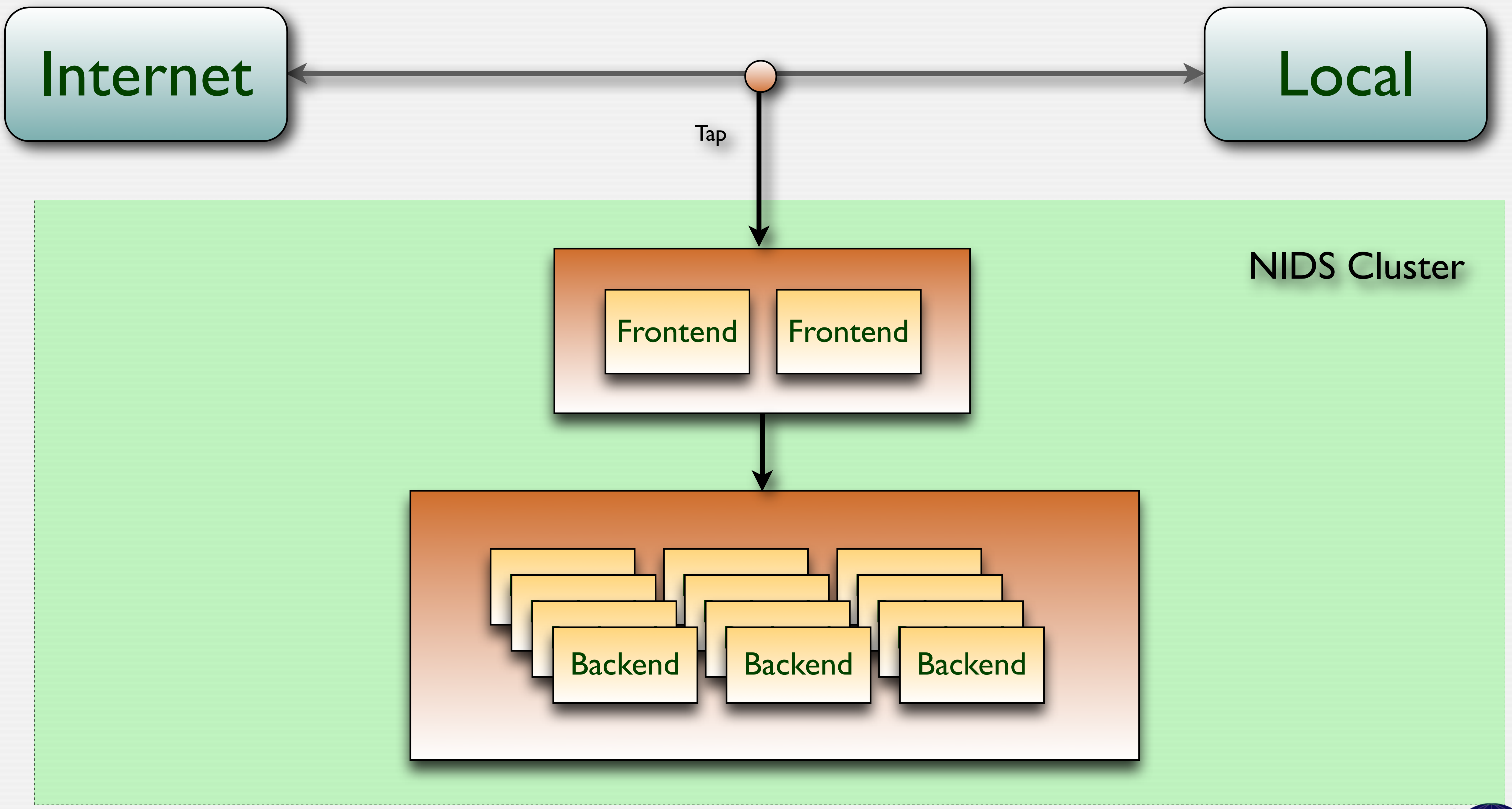
Architecture



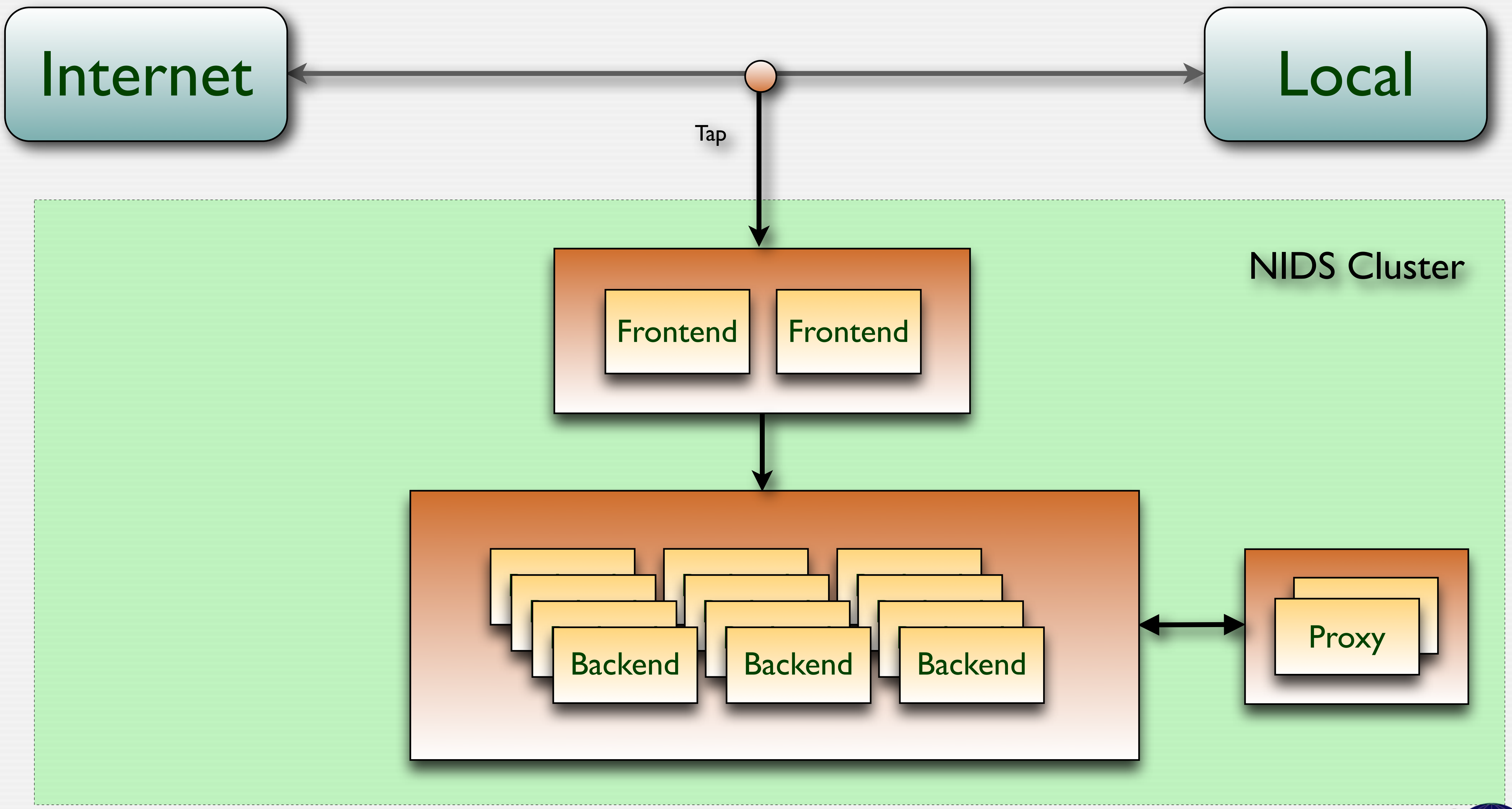
Architecture



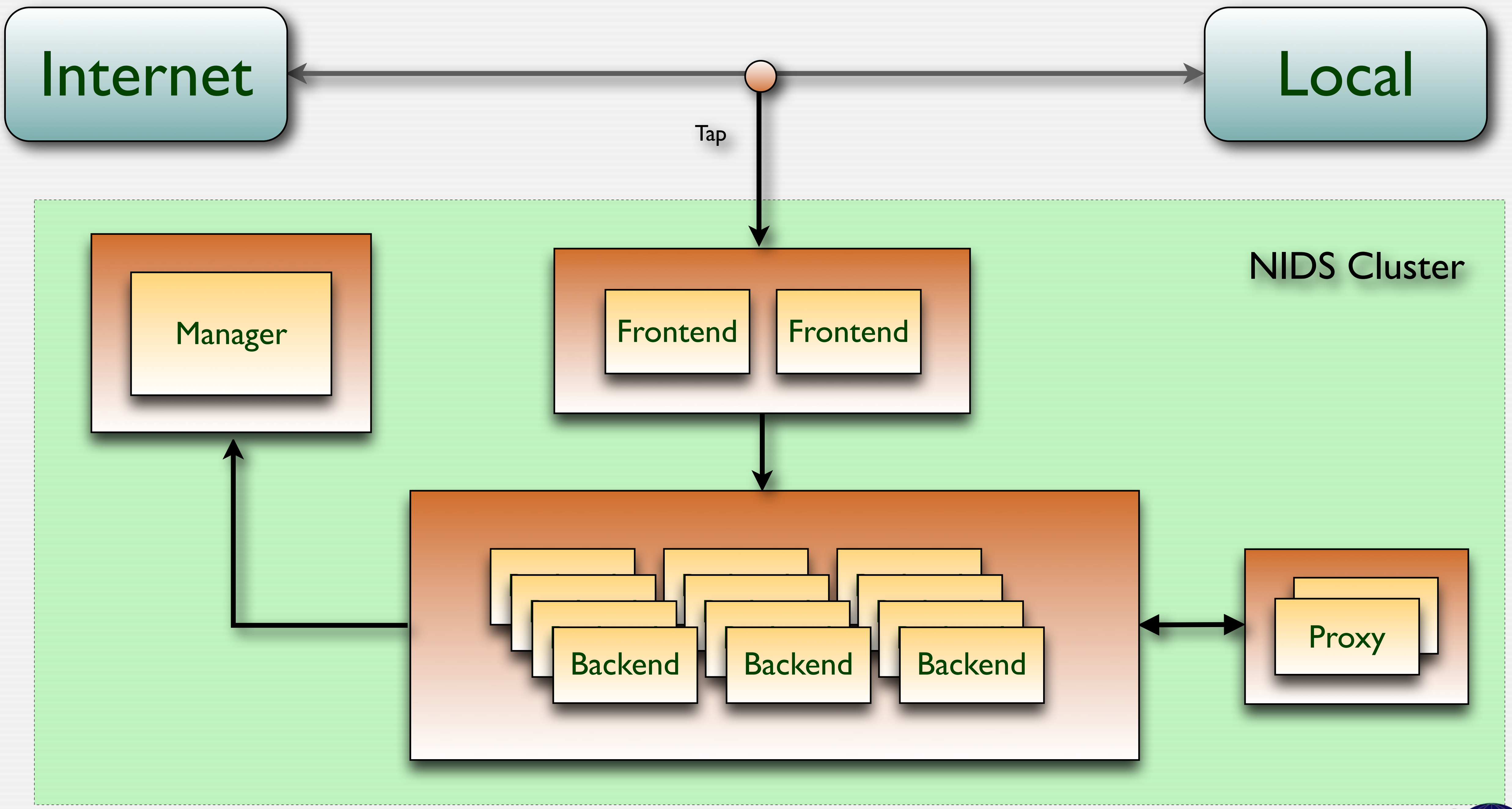
Architecture



Architecture



Architecture



Prototype Setups

- Lawrence Berkeley National Laboratory
  - Monitors 10 Gbps upstream link
  - 1 frontend, 10 backends
- University of California, Berkeley
  - Monitors 2x1Gbps upstream links
  - 2 frontends, 6 backends
- IEEE Supercomputing 2006
  - Monitored conference's 1 Gbps backbone network
  - 10 Gbps *High Speed Bandwidth Challenge* network
- Goal: Replace operational security monitoring at LBNL

Frontends

- Slicing the traffic connection-wise
  - Hashing based on either 4-tuple (addrs, ports) or 2-tuple (addrs)
- Distributing traffic to backends by rewriting MACs
  - In software via Click (open-source "modular router")
  - In hardware via Force-10's P10 (prototype in collaboration with F10)
- LBNL is contracting a hardware vendor
  - Will build production frontends operating at 10Gbps line-rate
  - Available be available in 4-5 months

Backends and Manager

- Running Bro as their analysis engine
- Bro provides extensive communication facilities
  - Independent state framework
  - Sharing of *low-level* state
  - Script-layer variables can be *synchronized*
- Basic approach: pick state to be synchronized
  - A few subtleties needed to be solved
- Central manager
  - Collects output of all instances
  - Raises alerts
  - Provides dynamic reconfiguration facilities
  - Working on interactive *cluster shell*

The Cluster Shell

```
homer ~
robin@homer:~>cluster

Welcome to BroCluster 0.1

Type "help" for help.

[BroCluster] > status
Name         Type        Status      Host       Pid     Peers    Started
manager      manager     homer       running    3743    9        07 Oct 16:49:53
proxy-1      proxy       homer       running    3781    9        07 Oct 16:50:02
worker-2a    worker      lisa        running    86072   2        07 Oct 16:11:18
worker-2b    worker      lisa        running    86110   2        07 Oct 16:11:19
worker-3a    worker      bart        running    93591   2        07 Oct 16:11:21
worker-3b    worker      bart        running    93629   2        07 Oct 16:11:23
worker-4a    worker      maggie      running    92713   2        07 Oct 16:11:24
worker-4b    worker      maggie      running    92751   2        07 Oct 16:11:26
worker-5a    worker      abraham     running    17416   2        07 Oct 16:11:27
worker-5b    worker      abraham     running    17453   2        07 Oct 16:11:29
[BroCluster] > capstats
Host           mbps       kpps       (10s avg)
192.168.1.5    113.1      20.4
192.168.1.4    186.0      27.1
192.168.1.3    131.4      30.7
192.168.1.6    114.5      21.4
[BroCluster] > analysis
            dns is enabled    -  DNS analysis
            ftp is enabled    -  FTP analysis
      http-body is enabled    -  Analysis of HTTP bodies
    http-header is disabled   -  Analysis of HTTP headers
     http-reply is enabled    -  Server-side HTTP analysis
   http-request is enabled    -  Client-side HTTP analysis
           scan is enabled    -  Scan detection
           smtp is enabled    -  SMTP analysis
[BroCluster] >
```

Going Back in Time with the *Time Machine*

The Utility of Time Travel

- Bro's policy-neutral logs are often the most useful output
  - Typically we do not know in advance how the next attacks looks like
  - But when an incident occurred, we need to understand *exactly* what happened

    "How did the attacker get in? What damage did he do? Did the guy access other hosts as well? How can we detect similar activity in the future?"

- This is when you need all information you can find

- The most comprehensive resource are *packet traces*
  - Wouldn't it be cool if you had a packet trace of that incident?

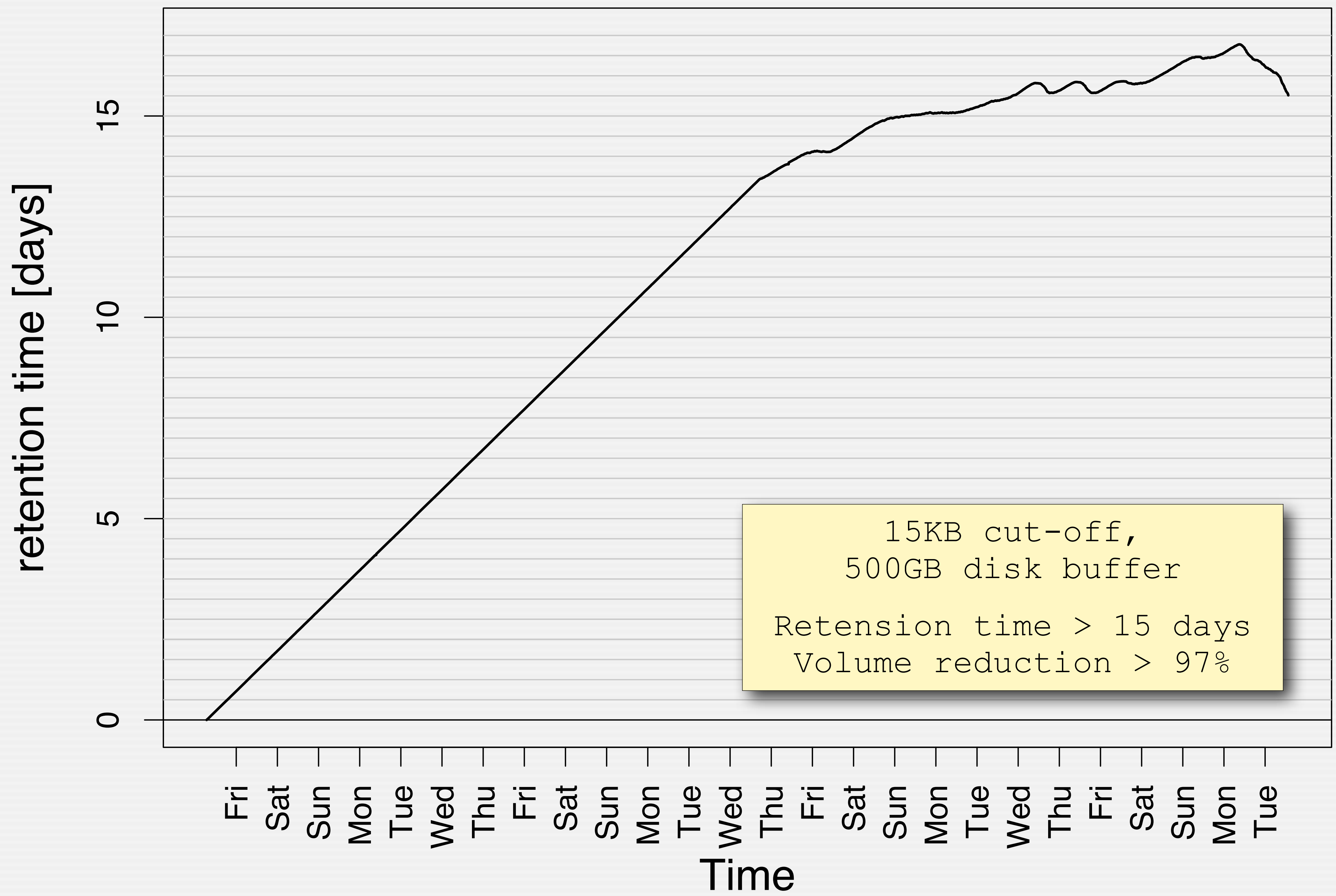
The Time Machine

- The *Time Machine*, a bulk-recorder for network traffic
  - Efficient packet recorder for high-volume network streams
  - Taps into a network link and records packets in their entirety
  - Provides efficient query interface to retrieve past traffic
- Storing *everything* is obviously not feasible
- TM uses heuristics for volume reduction
  - *Cut-off*: For each connection, TM stores *only the first few KB*
  - *Expiration*: Once space is exhausted, TM expires oldest packets automatically
- Simple yet very effective scheme
  - Leverages network traffic's "heavy-tails"
  - Even in large networks we can go back in time for several days
  - Proven to be extremely valuable for network forensics in operational use at LBL

BERKELEY LAB

ICSI
INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Example: Retension Time at LBNL



10Gbps upstream link, 10,000 hosts, 100-200Mbps average, 1-2TB/day

Query Interface

- Interactive console interface

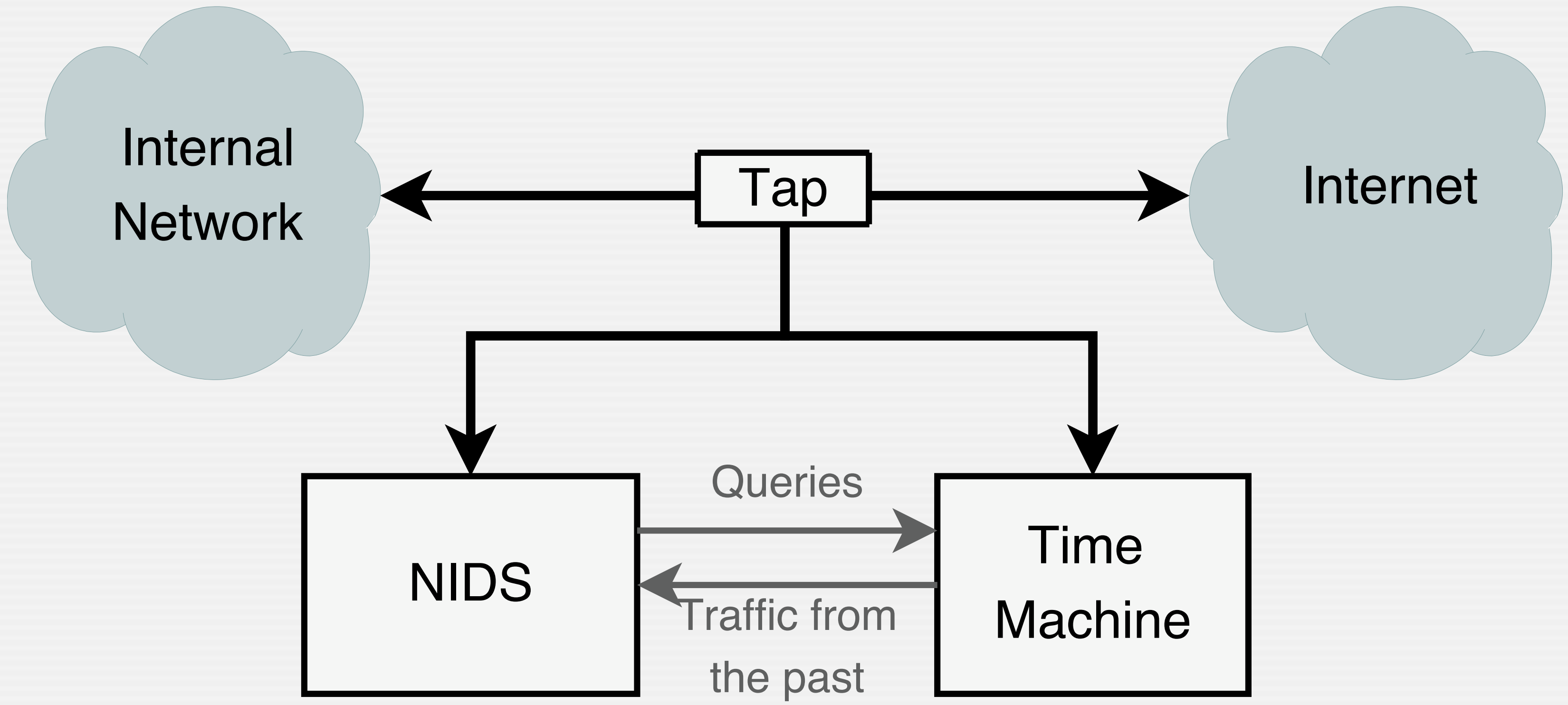
```
# An example query. Results are stored in a file.
query to_file "trace.pcap" index ip "1.2.3.4"

# Dynamic class. All traffic of IP 5.6.7.8 is
# assigned to class alarm
set_dyn_class 5.6.7.8 alarm
```

- Command-line client for common queries

```
tm-query --ip 1.2.3.4 localhost host.pcap --time 12h
```

Interfacing the TM with Bro

- The Time Machine can provide a NIDS with historic traffic



- Applications
  - Bro controls the TM (change cut-offs, switch storage class)
  - Bro permanently stores attack traffic
  - Bro analyzes traffic *retrospectively*

Augmenting Bro Alerts with Traffic

01/25/08 12:23:03 HTTP_SensitiveURI [redacted]:55899/tcp = [redacted]
[redacted]:80/tcp = [redacted]

[redacted]/55899 > [redacted]/http %worker-8-1708639: GET /index.php?content=/etc/passwd (200 "OK" [1469] www[redacted])

⊟ Tcpdump of connection's packets (file size: 2725 bytes)

```
12:23:03.048404 IP [redacted]5899 > [redacted].80: S 1104981131:1104981131(0)
12:23:03.048635 IP [redacted].80 > [redacted]899: S 1655847285:1655847285(0) ack 11
12:23:03.236799 IP [redacted]5899 > [redacted].80: . ack 1 win 5840
12:23:03.237600 IP [redacted]5899 > [redacted].80: P 1:110(109) ack 1 win 5840
12:23:03.237841 IP [redacted].80 > [redacted]899: . ack 110 win 5840
12:23:03.239162 IP [redacted].80 > [redacted]899: . 1:1461(1460) ack 110 win 5840
12:23:03.239165 IP [redacted].80 > [redacted]899: P 1461:1699(238) ack 110 win 5840
12:23:03.239167 IP [redacted].80 > [redacted]899: F 1699:1699(0) ack 110 win 5840
12:23:03.426493 IP [redacted]5899 > [redacted].80: . ack 1461 win 8760
12:23:03.426495 IP [redacted]5899 > [redacted].80: . ack 1699 win 8760
12:23:03.426497 IP [redacted]5899 > [redacted].80: F 110:110(0) ack 1700 win 8760
12:23:03.426990 IP [redacted].80 > [redacted]899: . ack 111 win 5840
```

⊞ Strings in connection's packets (file size: 2725 bytes)

⊞ Tcpdump of host's traffic (file size: 14414 bytes)

⊞ Strings in host's traffic (file size: 14414 bytes)

⊞ Reassembled originator contents (file size: 109 bytes)

⊟ Reassembled responder contents (file size: 1698 bytes)

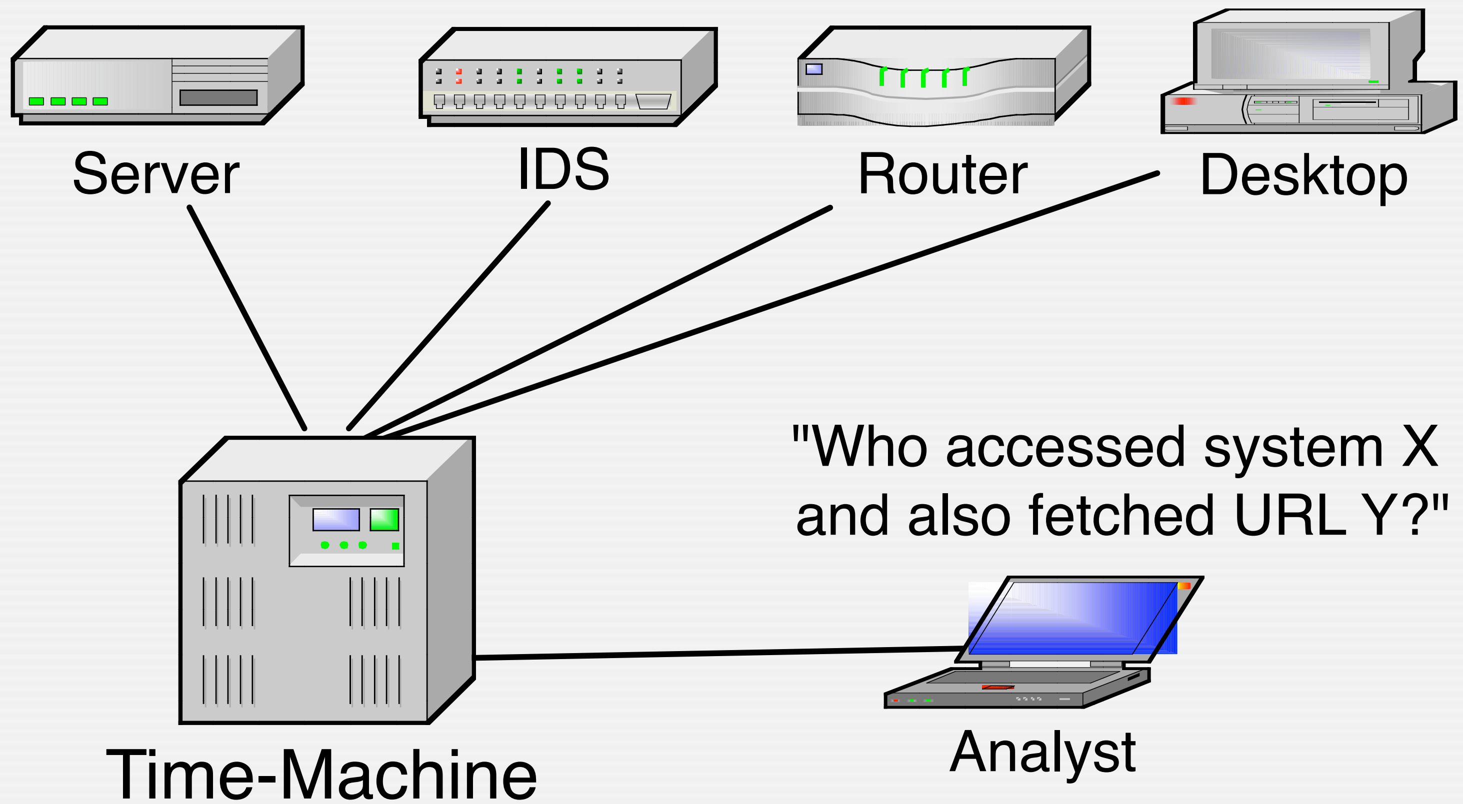
```
HTTP/1.1 200 OK
Date: Fri, 25 Jan 2008 20:23:03 GMT
Server: Apache/2.2.3 (Unix) mod_ssl/2.2.3 OpenSSL/0.9.8a PHP/5.1.6 mod_jk/1.2.19
X-Powered-By: PHP/5.1.6
Content-Length: 1469
Connection: close
```

BERKELEY LAB

ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Current Work: “Time Machine NG”

- We are now building a generalized Time Machine
  - Incorporates arbitrary network activity rather than just packets
  - Allows live queries for future activity



One goal: Facilitate cross-site information sharing

Summary & Outlook

The Bro NIDS

- Bro is one of the most powerful NIDS available
  - Open-source and runs on commodity hardware
  - While primarily a research system, it is well suited for operational use
  - One of the main components of LBNL's network security monitoring
- Working a various extensions
  - New analyzers for NetFlow, BitTorrent, SIP, XML w/ XQuery support
  - Multi-core support
- Turning cluster prototype into production at LBNL
  - Reimplementing frontends on new platforms

Any questions ... ?

Robin Sommer

Lawrence Berkeley National Laboratory &

International Computer Science Institute

rsommer@lbl.gov

http://www.icir.org

BERKELEY LAB

ICSI

INTERNATIONAL COMPUTER SCIENCE INSTITUTE