Intrusion Detection

and the Bro NIDS

Matthias Vallentin
vallentin@icsi.berkeley.edu

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

Acknowledgements

- Slides mixed from my **ICSI** / **ICIR** fellows
- Kudos to you, guys!

Robin Sommer
http://www.icir.org/robin

Christian Kreibich
http://www.icir.org/christian

Outline

- Intrusion Detection 101
- The Bro NIDS
- Port-independent Protocol Analysis
- Parallel Intrusion Detection
- Demo

Intrusion Detection 101

Detection vs. Prevention

- **Intrusion Detection**
  - passive
  - unobtrusive
- **Intrusion Prevention**
  - inline
  - critical

Deployment

- **Host-based**
  - Scope: single machine
    - Anti-{Virus, Rootkit, Phishing}

  + Access to system resources (memory, disk, periphals)

  - Expensive analysis decreases system performance
- **Network-based**
  - Scope: link-layer visibility

  + Analysis can incorporate data from multiple sources

  - Threats do not only come from the network

Detection Strategies

- Three analysis models
  - **Misuse Detection**
  - **Anomaly Detection**
  - **Specification-based Detection**

Detection Strategies (cont'd)

- **Misuse Detection**
  - Recognizes *known* attacks (pattern matching, blacklists)

  + Good attack libraries
  + Easy to understand results
  - Unable to detect new attacks or variants

Detection Strategies (cont'd)

- **Anomaly Detection**
  - Deviation from expected behavior raises an alert

+ Detects wide range of attacks, include novel

- High false positive rate
- Effectiveness depends on preliminary training

Detection Strategies (cont'd)

- **Specification-based Detection**
  - Codifies allowed behavior in policies (whitelists)

+ Detects wide range of attacks, including novel

+ Can accommodate signatures and anomalies

+ Directly supports implementing a site's policy

- Policies require significant development & maintanance

- Attack libraries difficult to construct

Trade-Offs and Limitations

- Cost ↔ Benefit
- False positives ↔ False negatives
- Stateful ↔ Stateless
- Evasion: attacks directed at the system itself
- Evalution: synthetic data ↔ real-world data
- Scalability: more traffic, more diversity

The Bro NIDS

System Philosophy

- Developed at ICSI & LBNL since 1996
- Real-time network analysis framework
  - Primary a **network intrusion detection system (NIDS)**
  - However it is also used for pure **traffic analysis**
  - Focus on **application-level** semantic analysis (rather than analyzing individual packets)
- Strong separation of mechanism and policy

System Philosophy

- Strong separation of **mechanism** and **policy**
  - *Policy-neutral* core (no notion of “good” or “bad”)
- Not restricted to a particular detection strategy
  - Typical: misuse detection
- Operators program their policy

System Philosophy (cont'd)

- Focus is not signature matching (like Snort)
- Focus is not anomaly detection
  - But scripting language allows to program in this model
- Thorough activity logging
  - Not just alerts
  - Policy-neutral logs are invaluable for forensics

Target Environments

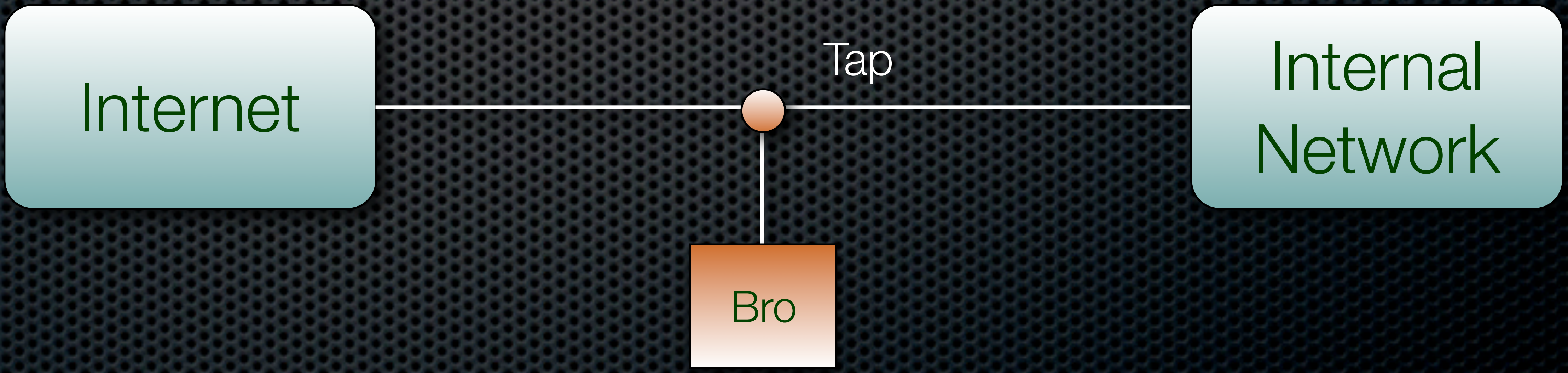
- Bro is specifically well-suited for scientific environments
  - Extremely useful in networks with liberal (“default allow”) policies
- Supports intrusion prevention schemes
- High-performance on commodity hardware
  - Runs on Unix-based systems (e.g., Linux, FreeBSD, MacOS)
  - Open-source (BSD license)

Target Environments (cont’d)

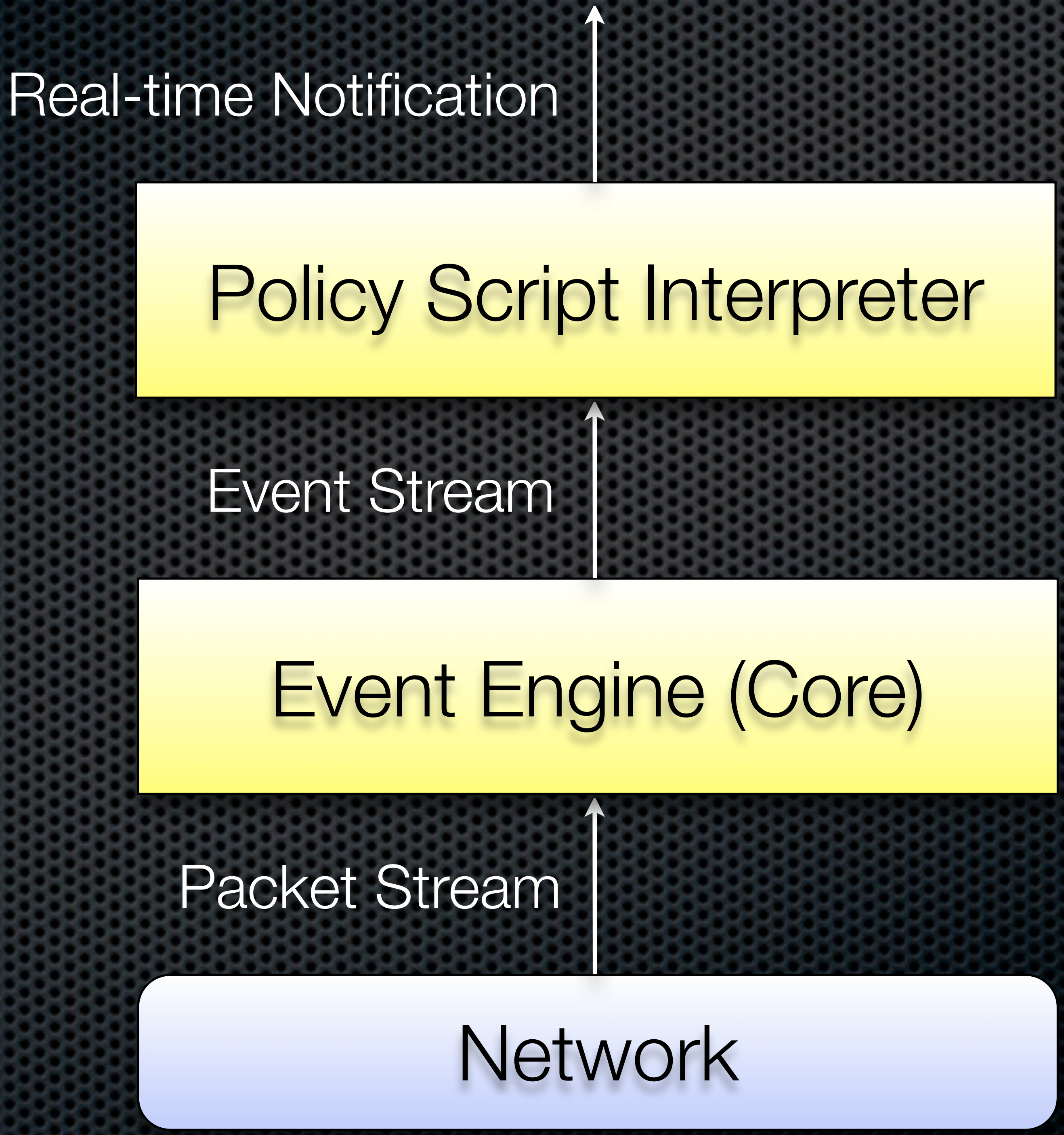
- Bro requires some effort to use it effectively
    - Pretty complex, script-based system
    - Requires understanding of the network
    - No GUI, just ASCII logs
    - Only partially documented
- Development is primarily driven by research
    - However, focus on operational use

Bro Deployment

- Bro is typically deployed at a site's upstream link
  - Monitors all external incoming or outgoing packets
  - Deployment similar to other NIDS
  - By default, purely passive monitoring



Architecture



Real-time Notification

Policy Script Interpreter

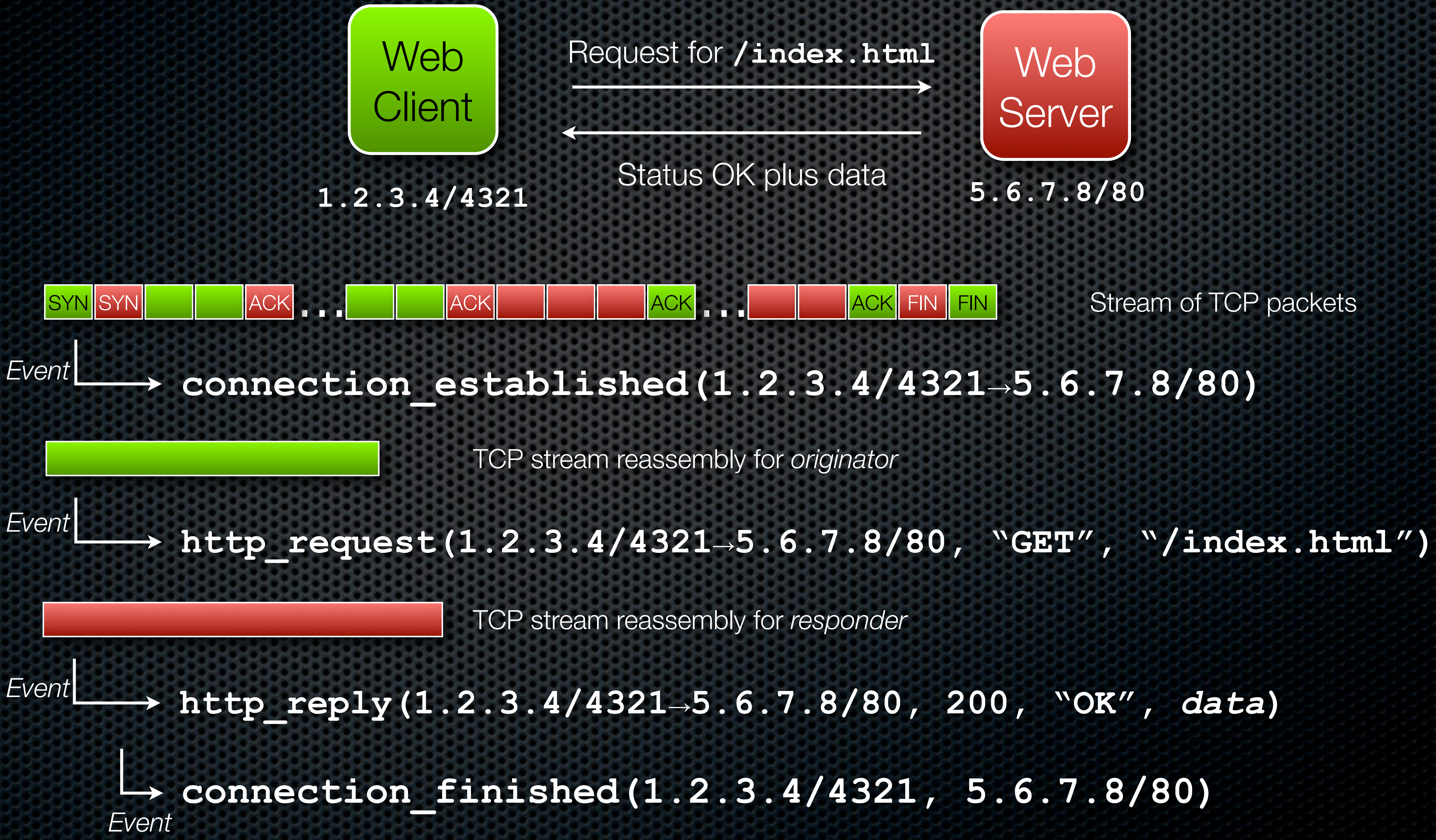
Event Stream

Event Engine (Core)

Packet Stream

Network

Event Model - Example



Web Client

Request for **/index.html**

Web Server

Status OK plus data

1.2.3.4/4321

5.6.7.8/80

SYN SYN 🟩 🟩 ACK ... 🟩 🟩 ACK 🟥 🟥 🟥 ACK ... 🟥 🟥 ACK FIN FIN

Stream of TCP packets

Event → **connection_established(1.2.3.4/4321→5.6.7.8/80)**

TCP stream reassembly for *originator*

Event → **http_request(1.2.3.4/4321→5.6.7.8/80, “GET”, “/index.html”)**

TCP stream reassembly for *responder*

Event → **http_reply(1.2.3.4/4321→5.6.7.8/80, 200, “OK”, *data*)**

→ **connection_finished(1.2.3.4/4321, 5.6.7.8/80)**

Event

Event Engine

- 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)
- Event-engine is written in C++ for performance
  - Performs work per packet

Event Engine (cont’d)

- Contains analyzers for **>30** protocols, including
  - ARP, IP, ICMP, TCP, UDP
  - BitTorrent, DCE-RPC, DNS, FTP, Finger, Gnutella, HTTP, IRC, Ident, NCP, NFS, NTP, NetBIOS, NetFlow, POP3, Portmapper, RPC, Rsh, Rlogin, SMB, SMTP, SSH, SSL, SunRPC, Telnet, XML w/ XQuery
- Analyzers generate **~300** types of events

Expressing Policy with Scripts

- Scripts are written in a **domain-specific language**
  - Bro ships with **20K+** lines of script code
  - Default scripts detect attacks & log activity extensively
- Scripts take actions
  - Generate alerts via syslog or mail
  - Execute program as a reactive form of response
  - Record activity to disk

Bro’s Scripting Language

- Bro’s scripting language is
  - Procedural
  - Event-based
  - Strongly typed
  - Rich in types (tables/sets, address, port, subnet, ...)
- State management (persistence, expiration, timers, ...)
  - Supporting communication with other Bro instances

Script Example: Matching URLs

```
event http_request(c: connection, method: string, path: string)
{
    if ( method == “GET” && path == “/etc/passwd” )
    NOTICE(SensitiveURL, c, path);
}
```

Code simplified. See policy/http-request.bro.

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.
    print "New SSH host found", responder;
}
```

Policy-neutral Logging

- Bro’s default scripts perform two main tasks
  - Detecting malicious activity (mostly misuse-detection)
  - Logging activity comprehensively without any actual assessment
- In practice, policy-neutral logs are often most useful
  - Form of new attacks typically unknown
  - Detailed information highly useful when incidents happen

Example Log: HTTP Session

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

Port-independent Protocol Analysis

with Dynamic Protocol Detection (DPD)

Shortcut

Port-based 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.

Port-based Analysis (cont'd)

- 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
- Ideal setting: for every connection, try all analyzers
- Performance penalty: can't parse 10 000s of connections in parallel with all analyzers enabled

Making it realistic ...

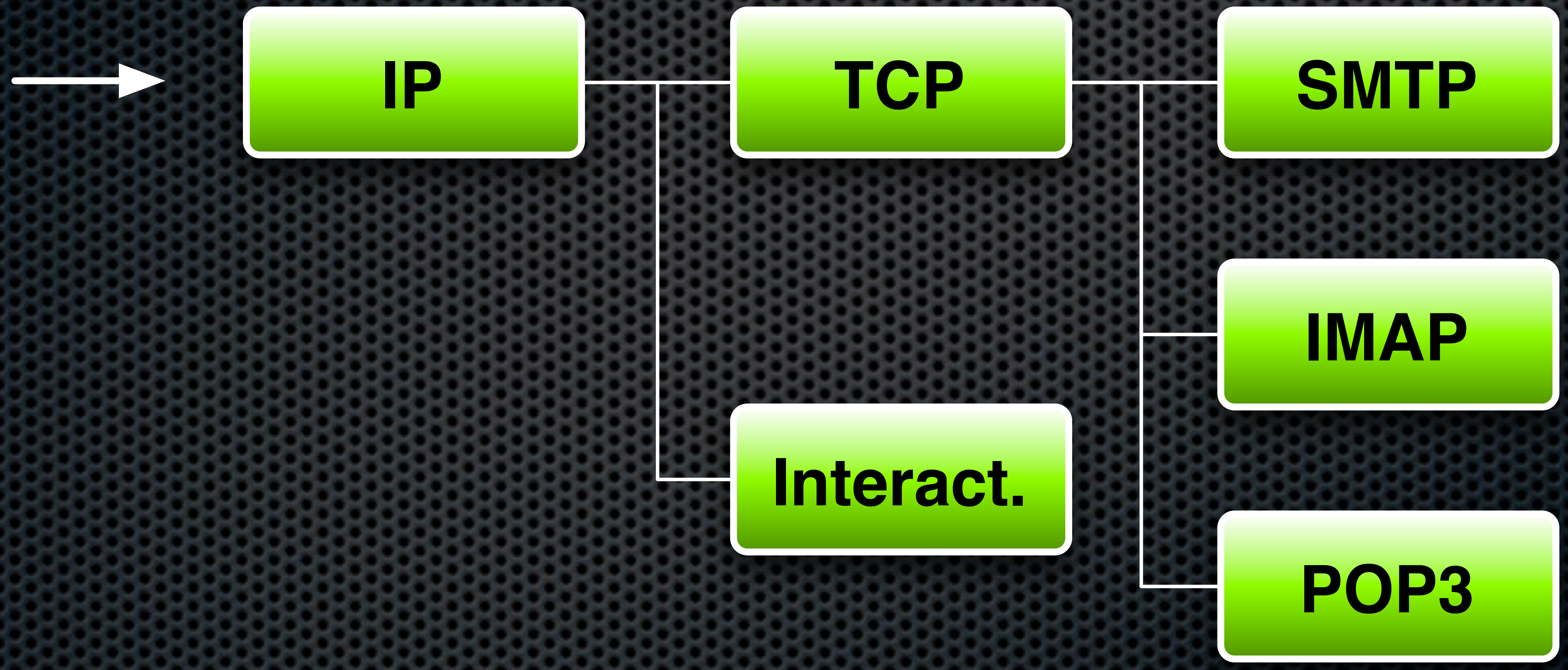
- Bro uses byte patterns to **prefilter** connections
    - An HTTP signature looks for **potential** uses of HTTP
    - HTTP analyzer then verifies by trying to parse the payload
- Signatures can be loose because false positives are inexpensive (no alerts!)

Making it realistic ...

- 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

Example: Analyzer Tree

A connection looks like mail, but what is it?



Application Example: FTP Data

- FTP data sessions can’t be analyzed by port-based NIDSs
- Bro’s DPD has a notion of “expected connections”
  - Can be told in advance which analyzer to use for an upcoming connection
- Bro also has a File Analyzer
  - Determines file-type (via libmagic)

Application Example: FTP Data (cont’d)

```
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)
```

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

Application Example: Finding Bots

- 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: Finding Bots (cont'd)

```
Detected bot-servers:
IP1 - ports 9009,6556,5552 password(s) <none> last 18:01:56
 channel #vec:
 topic ".asc pnp 30 5 999 -b -s|.wksescan 10 5 999 -b -s|[...]"
 channel #hv:
 topic ".update http://XXX/image1.pif f"
[...]
Detected bots:
IP2 - server IP1 usr 2K-8006 nick [P00|DEU|59228]
IP4 - server IP1 usr XP-3883 nick [P00|DEU|88820]
[...]
```

DPD: Summary

- Port-independent protocol analysis
  - Idea is straight-forward, but Bro is the only system which does it
- Bro now has a very generic analyzer framework
  - Allows arbitrary changes to analyzer setup during lifetime of connection
  - Is not restricted to any particular approach for protocol detection

DPD: Outlook

- Main performance impact: need to examine all packets
  - Well, that's pretty hard to avoid
- Potential extensions
  - More protocol-detection heuristics (e.g., statistical approaches)
  - Analyze tunnels by pipelining analyzers (e.g., to look inside SSL)
  - Hardware support for pre-filtering (e.g., on-NIC filtering)

Parallel Network Intrusion Detection

Problem

- NIDSs reached their limits on commodity hardware
  - Need to do more analysis on more data at higher speeds
  - However, CPU performance is not growing anymore the way it used to
  - Single NIDS instance (e.g., Snort, Bro) cannot cope with Gbps links

Motivation

- To overcome, we must either
  - Restrict the amount of analysis
  - Turn to expensive, custom hardware
  - Employ parallelization of the processing across
    - Machines
    - CPUs

Orthogonal Approaches

- The NIDS Cluster
  - Many PCs instead of one
  - Communication and central user interface creates the impression of one system
  - First installations up and running

- Parallel operation within a single NIDS instance
  - In **software**: multi-threaded analysis on multi-core systems
  - In **hardware**: compile analysis into a parallel execution model (e.g., on FPGAs)

The NIDS Cluster

Overview

- We do load-balancing with the “NIDS Cluster”
  - Use many boxes instead of one
  - Every box works on a slice of traffic
  - Correlate analysis to create the impression of a single system

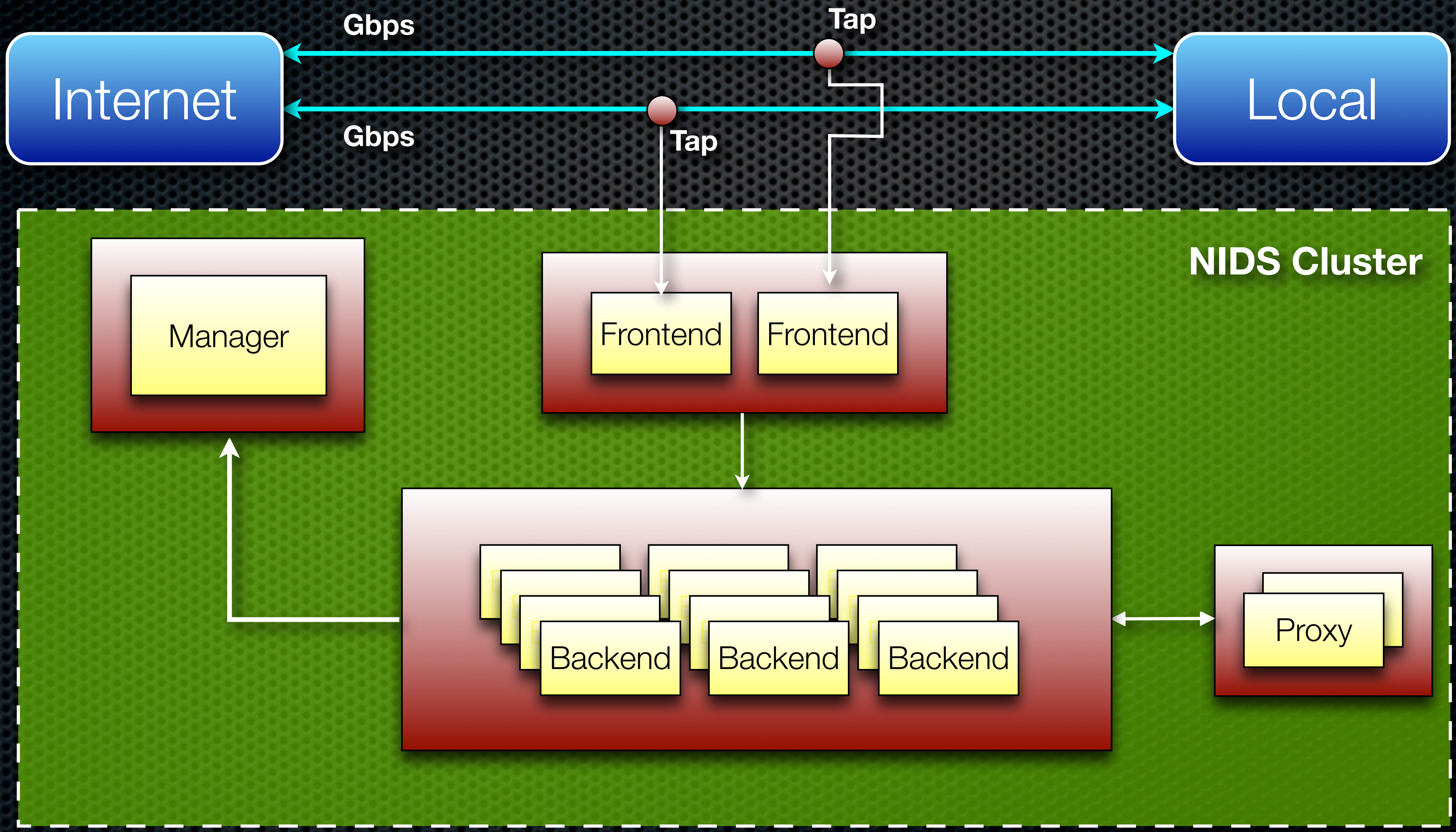
Traditional Approach

- 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

Our Approach

- Our NIDS cluster works **transparently** like a single NIDS
  - Gives same results as single NIDS would if it could analyze all traffic
  - Does not sacrifice detection accuracy
  - Scalable to large number of nodes
  - Still provides a single system as the user interface
    - logging, configuration updates

Architecture



Environments

- Initial target environment:
  Lawrence Berkeley National Laboratory (LBNL)
  - LBNL monitors 10 Gbps upstream link with the Bro NIDS
  - Setup evolved into many boxes running Bro independently for sub-tasks
  - Cluster prototype now running at LBNL
    - 1 frontend and 10 backends

Environments (cont’d)

- Further prototypes
  - **University of California, Berkeley**
    2 x 1 Gbps uplink, 2 frontends / 6 backends for 50% of the traffic
  - **Ohio State University**
    450 Mbps uplink, 1 frontend / 12 backends
  - **IEEE Supercomputing Conference 2007**
    Conference’s 1 Gbps backbone / 10 Gbps “High Speed Bandwidth Challenge” network
- Goal: Replace operational security monitoring

Challenges

- Main challenges when building the NIDS Cluster
  - Distributing the traffic evenly while minimizing need for communication
  - Adapting the NIDS operation on the backend to correlate analysis with peers
  - Validating that the cluster produces sound results

Summary

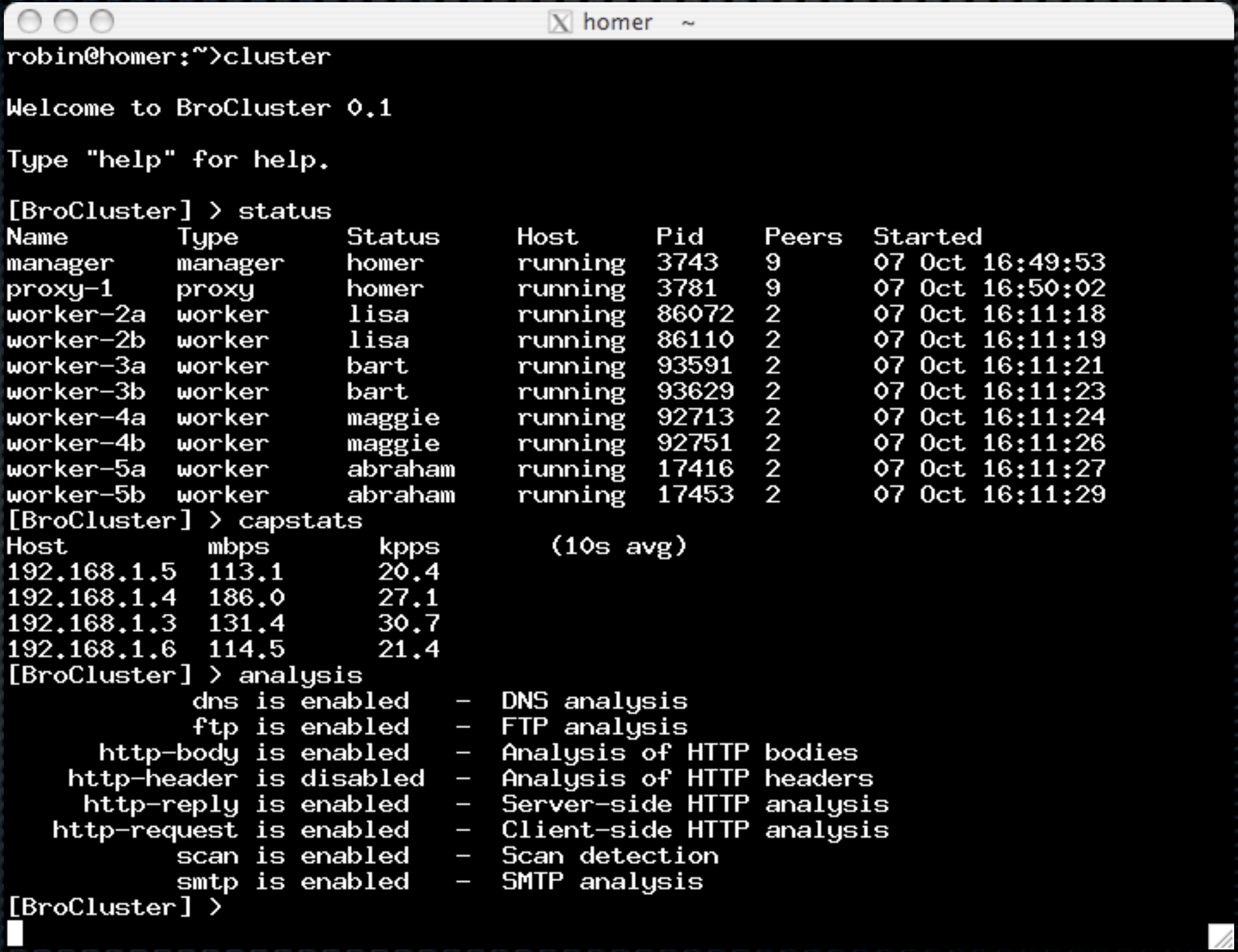
Summary

- 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
  - Deployed at large universities and labs

Current Work

- Interactive Cluster Shell for easy installation/operation of a Bro Cluster
- **Time Machine** interface
    - see http://www.net.t-labs.tu-berlin.de/research/tm
- Turning cluster prototype into production
- Multi-core support
- Inter-site data sharing

Cluster Shell



```
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] >
```

FIN

Matthias Vallentin
vallentin@icsi.berkeley.edu

INTERNATIONAL COMPUTER SCIENCE INSTITUTE