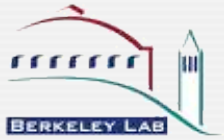
Seeking Visibility Into Network Activity for Security Analysis

Robin Sommer

Lawrence Berkeley National Laboratory &
International Computer Science Institute

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

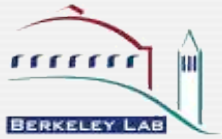
University of Adelaide - September 2007



Network Security at ICSI

- Our goal is to provide site operators with the capability to better understand network activity.
  - Many network operators do not have a good idea how their traffic looks like
  - Yet, providing effective security requires insight into network's activity
- Network security has no one-size-fits-all solution
  - Every network has different characteristics and policies
  - Operators need tools which *support* them in their local environment
- Three of our current projects
  - The *Bro Network Intrusion Detection System*
  - The *Time Machine*: A Long-Term Archive of Network Activity
  - A framework for distributed cooperative security monitoring

Policy-Controlled Monitoring with the Bro NIDS



System Philosophy

- 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 "good" or "bad")
  - User provides local site policy

System Philosophy (2)

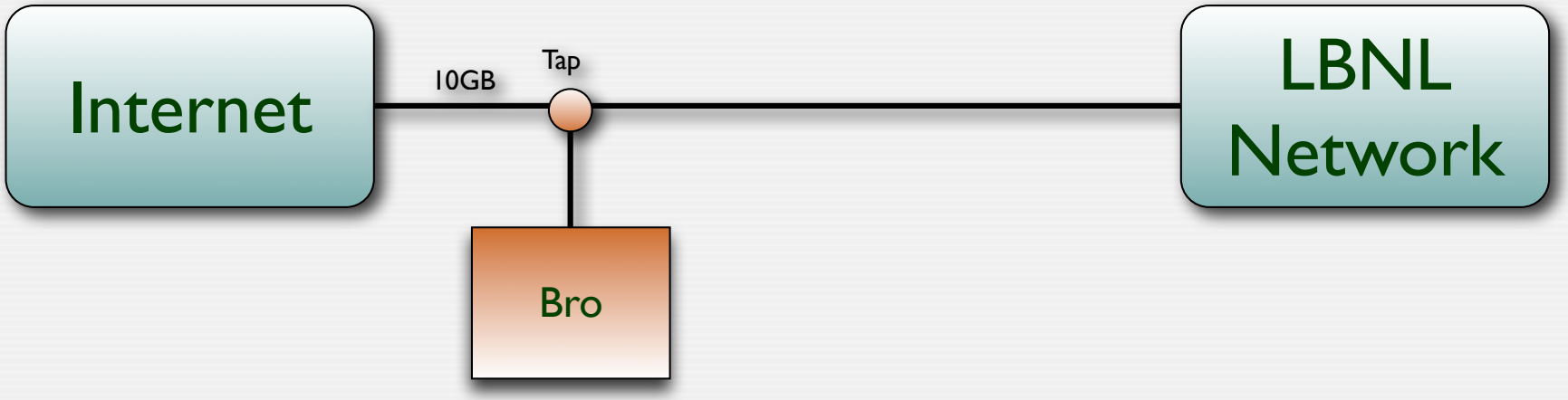
- Operators *program* their policy
  - Not really meaningful to talk about what Bro detects "by default"
- Bro comprehensively logs *all* activity
  - It does not just output alerts
  - Logs are invaluable for forensics
- Analysis model is *not* signature matching
  - Bro is fundamentally different from, 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

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 with BSD license
- 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
- We run Bro at several large-scale sites
  - UC Berkeley (UCB)
  - Lawrence Berkeley National Laboratory (LBNL)
  - National Energy Research Scientific Computing Center (NERSC)
  - TU München (TUM)

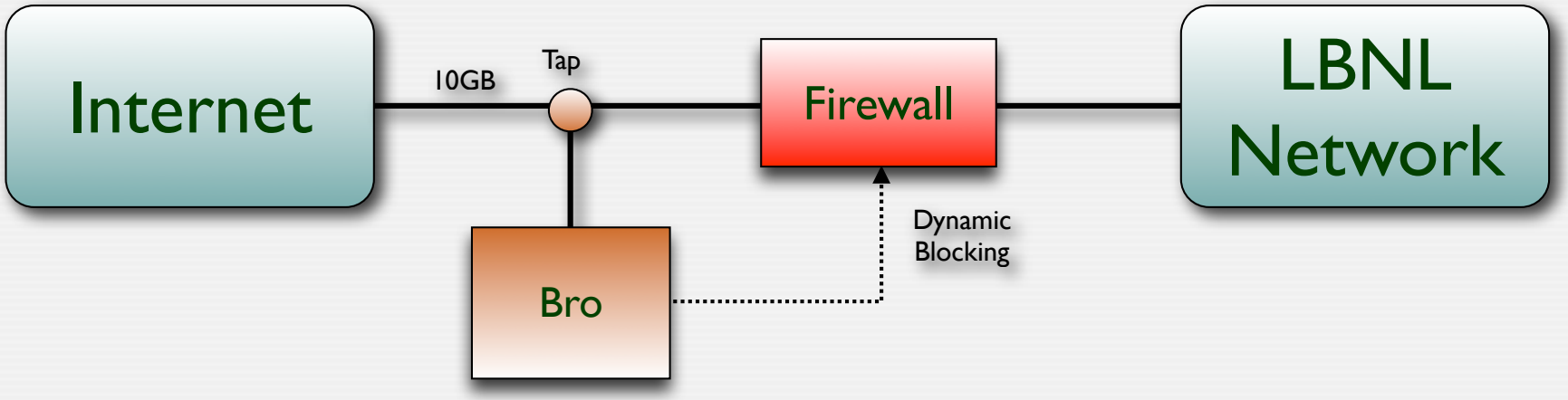
Bro at the Lawrence Berkeley Lab

- LBNL has been using Bro for >10 years
- It is is one of the main components of lab's security
- It monitors the 10 Gbps Internet uplink



Bro at the Lawrence Berkeley Lab

- LBNL has been using Bro for >10 years
- It is is one of the main components of lab's security
- It monitors the 10 Gbps Internet uplink

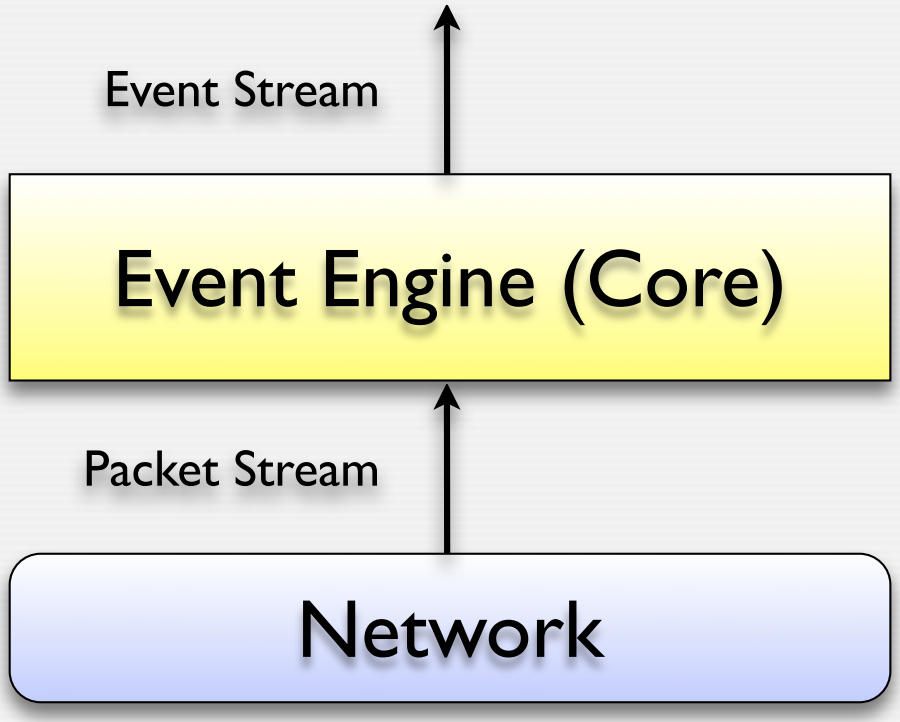


- Bro automatically drops connectivity for attackers
- It blocks about 4000 hosts/day at LBL!

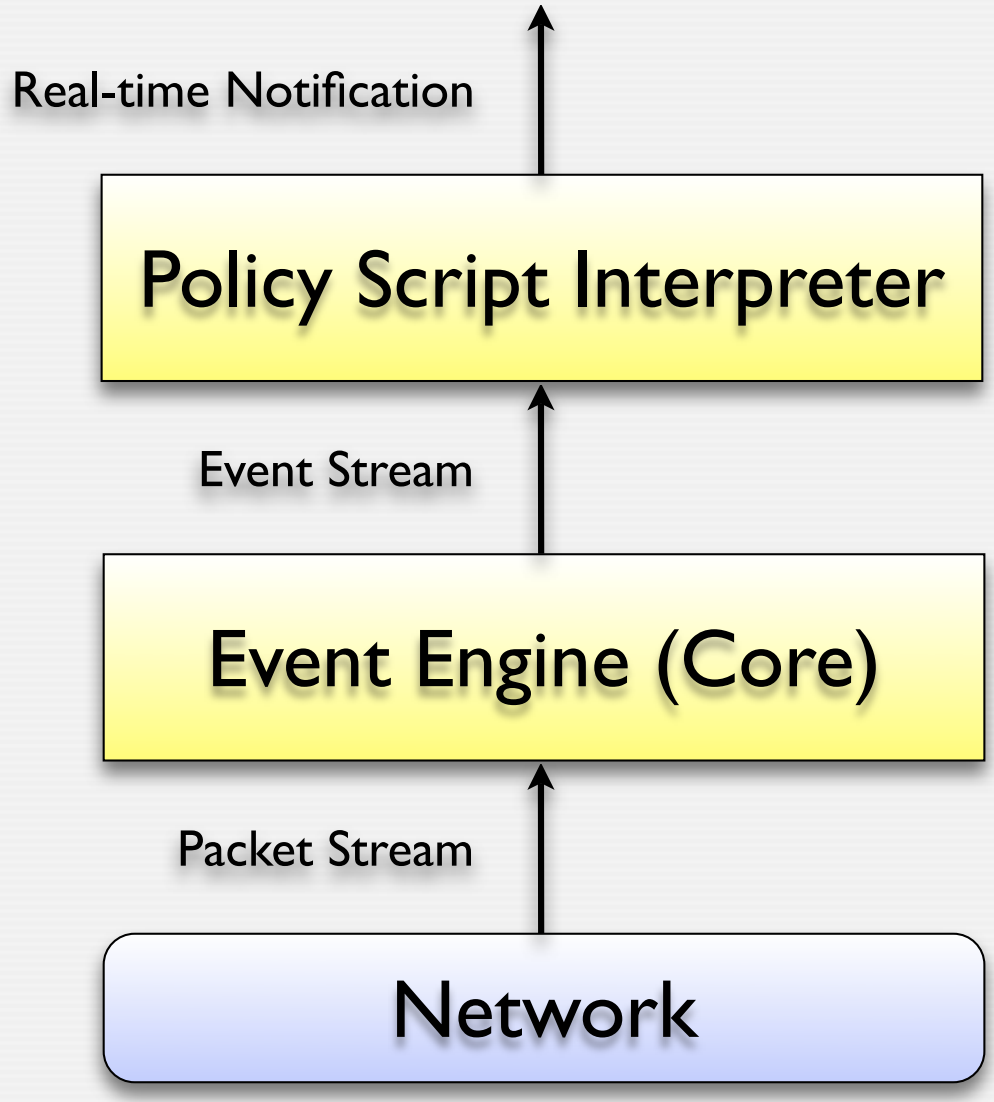
Architecture

Network

Architecture



Architecture



Event-Engine

- Event-engine is written in C++ for performance
- Performs low-level, *policy-neutral* analysis
  - Turns low-level activity into high-level events
  - Examples: `connection_established, http_request`
  - Events are policy-neutral (no notion "good" or "bad")
  - 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 events ...

Policy Scripts

- Scripts process event stream, incorporating ...
  - ... context from past events
  - ... site's local security policy
- Scripts take actions
  - Recording activity to disk
  - Generating alerts via syslog or mail
  - Executing program as a form of response
- Scripts are written in a custom language
  - Domain-specific, procedural, event-based, strongly typed
  - Bro ships with 20K+ lines of script code

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 "New SSH host found";
    }
```

Example: 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
```

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 ..."
[...]
```

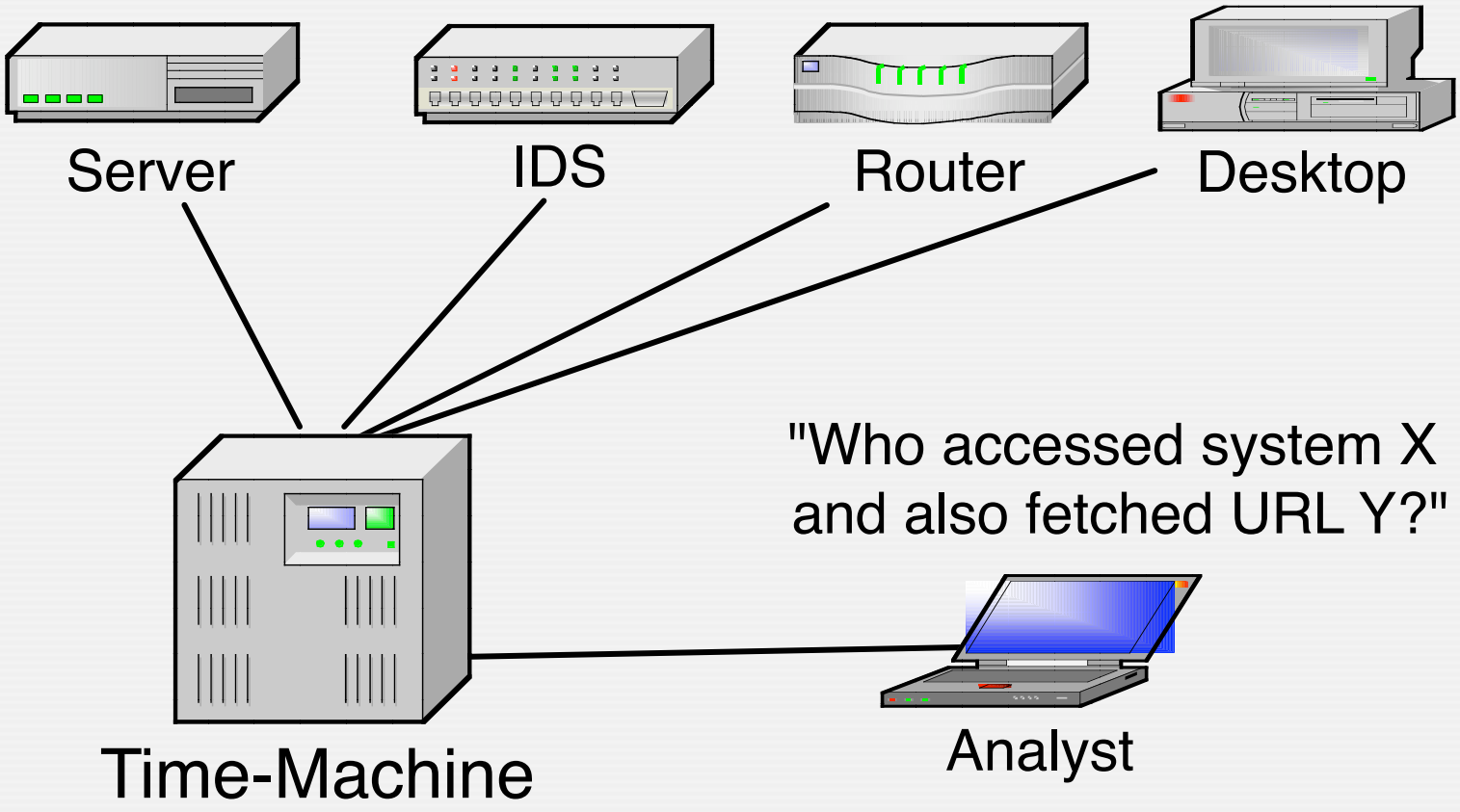
Going Back in Time
Archiving Network Activity

The Utility of Time Travel

- Often Bro's policy-neutral logs are the most useful
  - 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

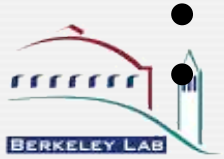
  "How did the attacker get in? What damage did he do? Did the guy access other hosts as well? How can we detect similar activity in the future?"
- This is when you need all information you can find
  - Administrators browse through the logs of their systems (NIDS, servers, firewall)
  - Manual, tedious, and usually incomplete process
  - Wouldn't it be nicer to just ask for "all" relevant activity?
- Goal: Building a comprehensive, long-term activity archive
  - Comprehensively archives information from heterogeneous sources
  - Operates fully policy-neutral
  - Provides an efficient, interactive query interface

A General Network Time Machine



Time Machine for Network Traffic

- We have already built a Time Machine for *network packets*
  - Efficient packet bulk-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
  - Proven to be extremely valuable for network forensics in operational use at LBL
- Heuristics for volume reduction
  - Cut-off: For each connection, TM stores only the first few KB
  - Expiration: Once available space is exhausted, TM expires oldest packets
- Simple yet very effective scheme
  - Leverages network traffic's "heavy-tails"
  - Even in large networks we can go back in time for several days
- Available online for download
  - `http://www.net.t-labs.tu-berlin.de/research/tm`
  - Prototypical Bro interface for automated queries



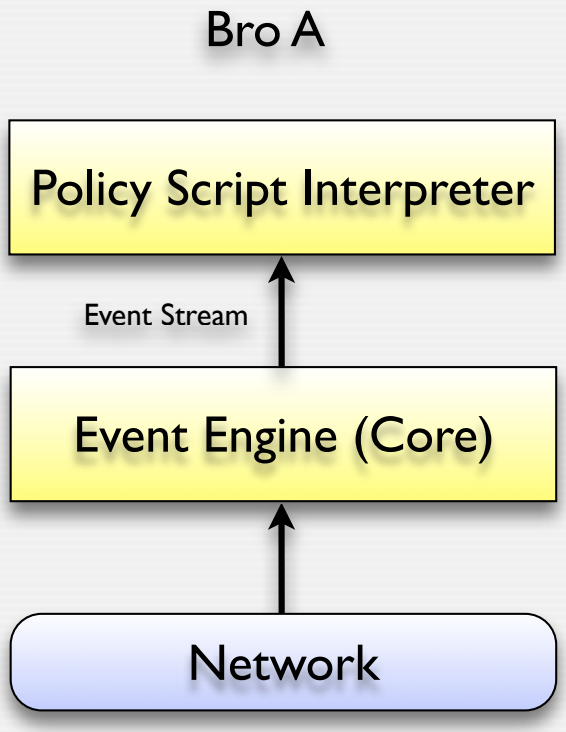
Generalizing the Time Machine

- We are now building a generalized Time Machine
  - Incorporates arbitrary network activity rather than just packets
  - Allows live queries for future activity
- The packet-based TM relies on three concepts
  1. A high-volume stream of input that we want to archive & query
  2. A rule how to separate more important input from less important input
  3. An aging mechanism to expire old information when storage fills up
- These concepts are independent of packets
- Need to find a suitable data model to network activity

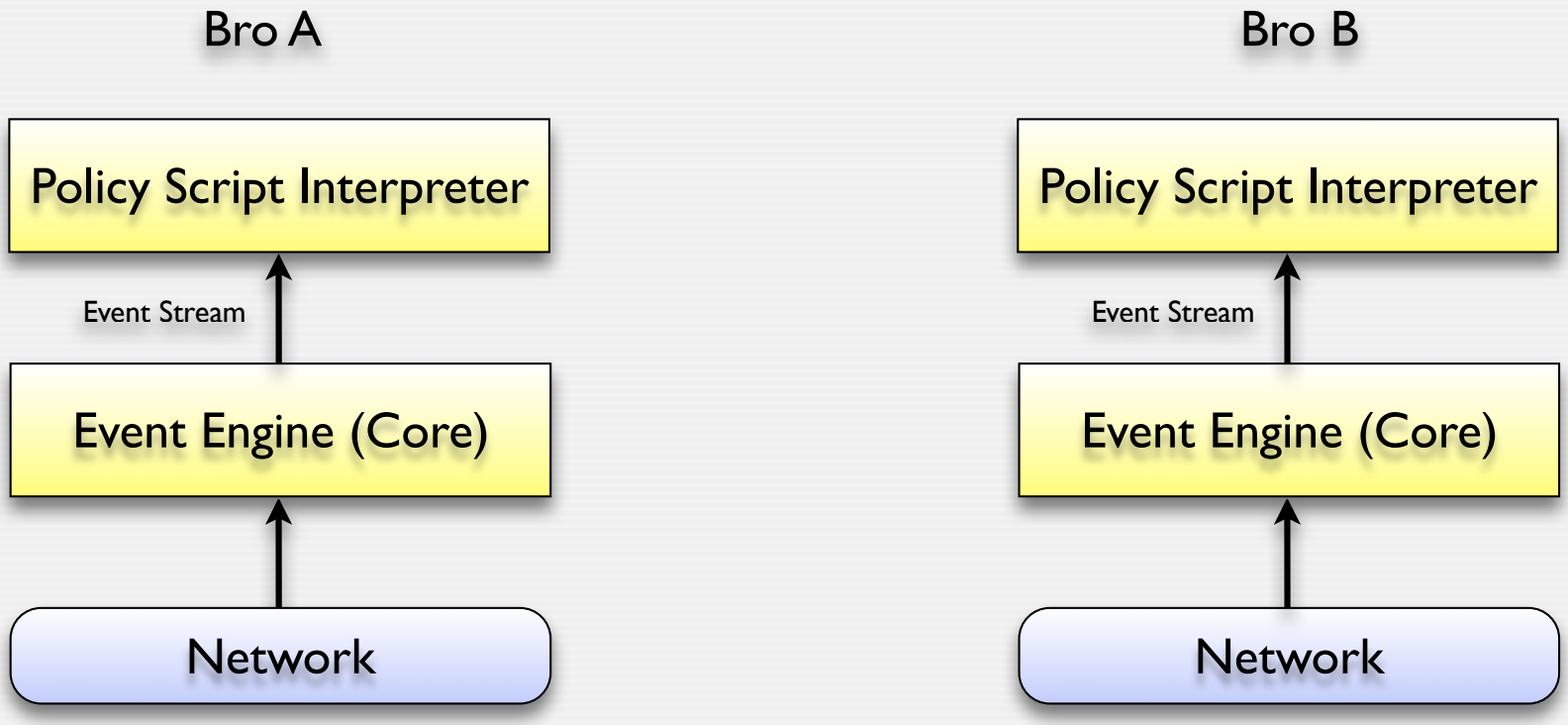
An Event-Based Data Model

- Bro system relies on an event model
  - Abstracts network packets into high-level, policy-neutral events
  - Events are policy-neutral (no notion "good" or "bad")
  - Security policies are specified on top of these event
- Events seem well-suited to represent network activity
- Network components can provide event streams
  - In previous work we have interfaced Bro to the external world

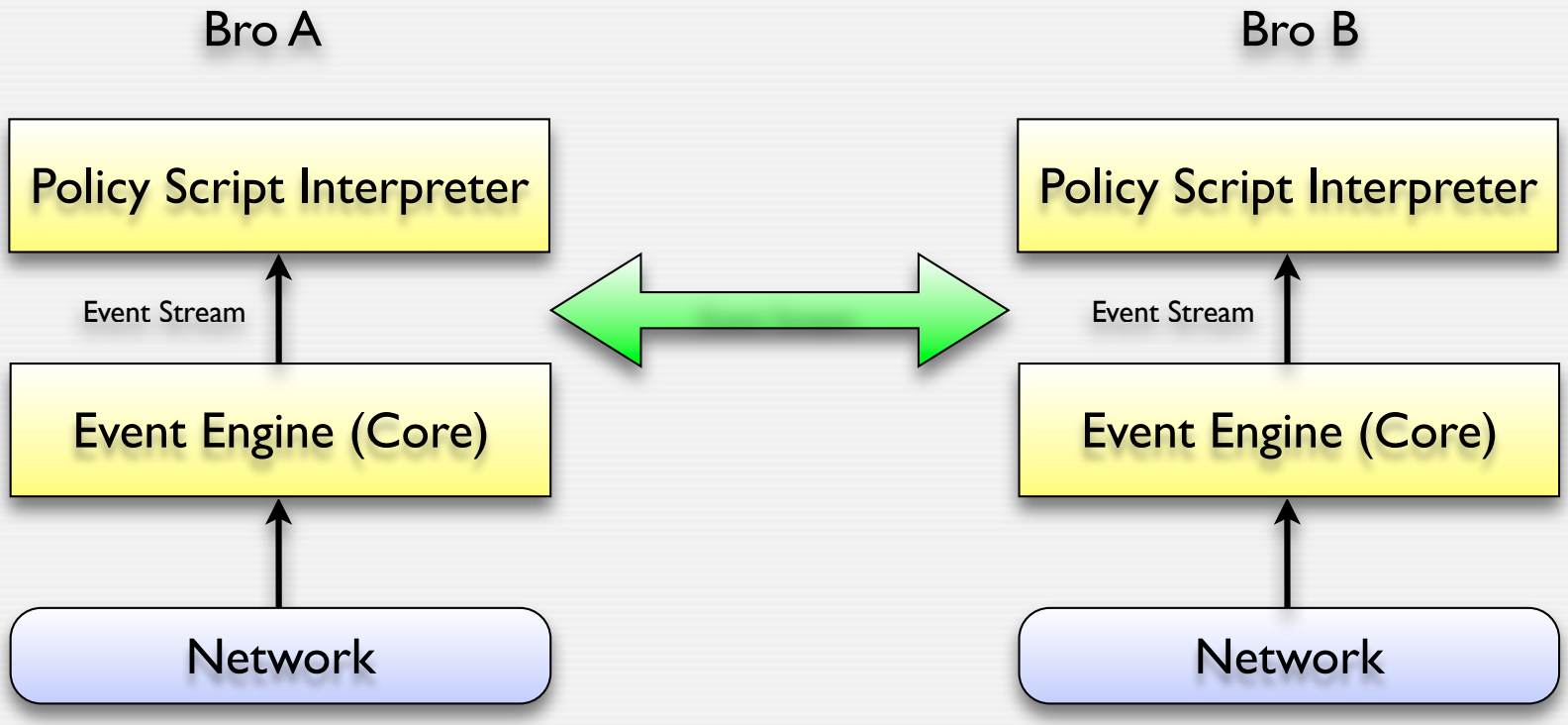
Detour: Bro's Communication System



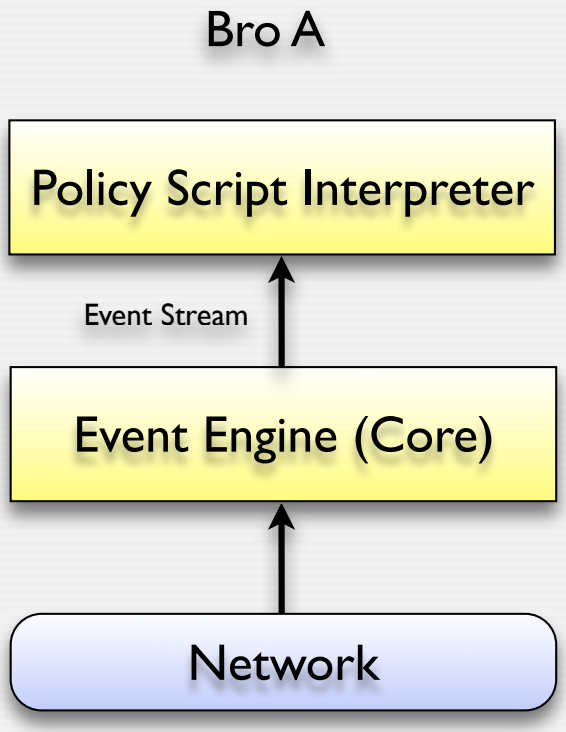
Detour: Bro's Communication System



Detour: Bro's Communication System

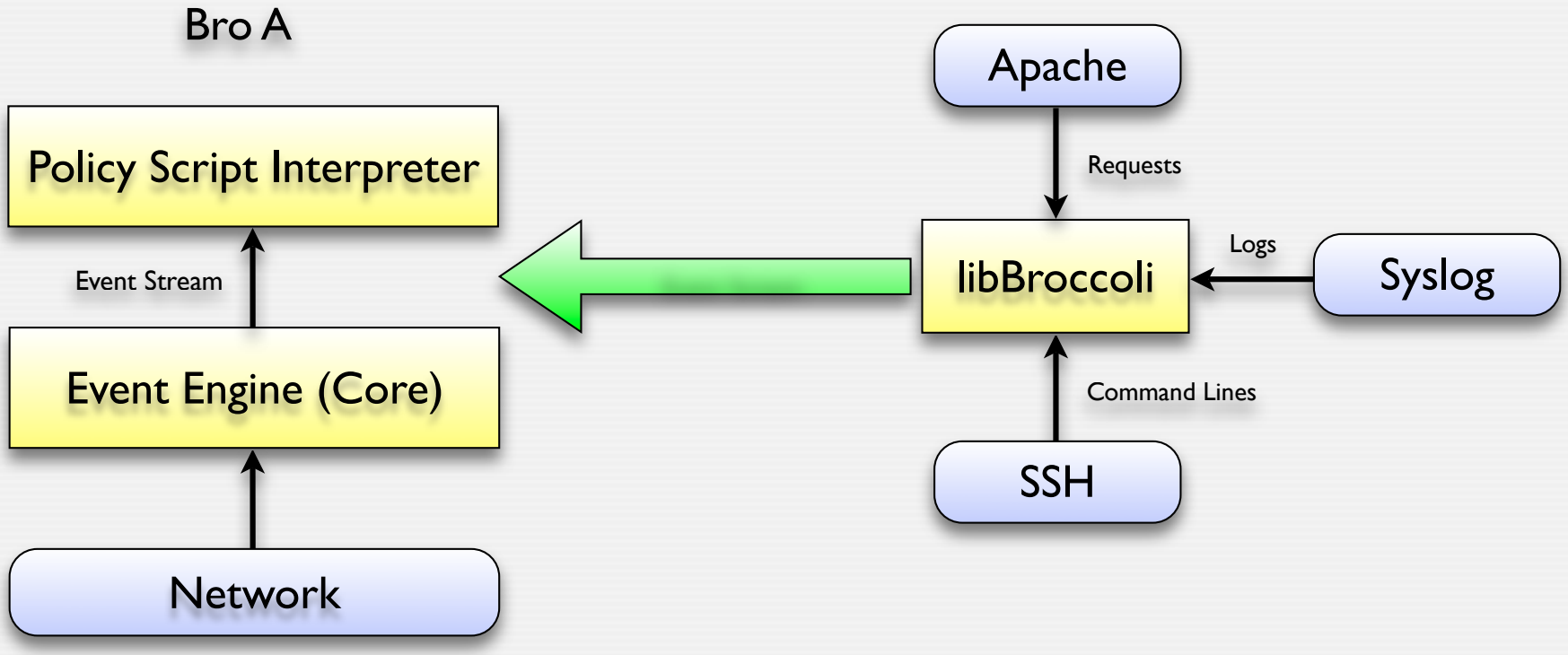


Detour: Bro's Communication System

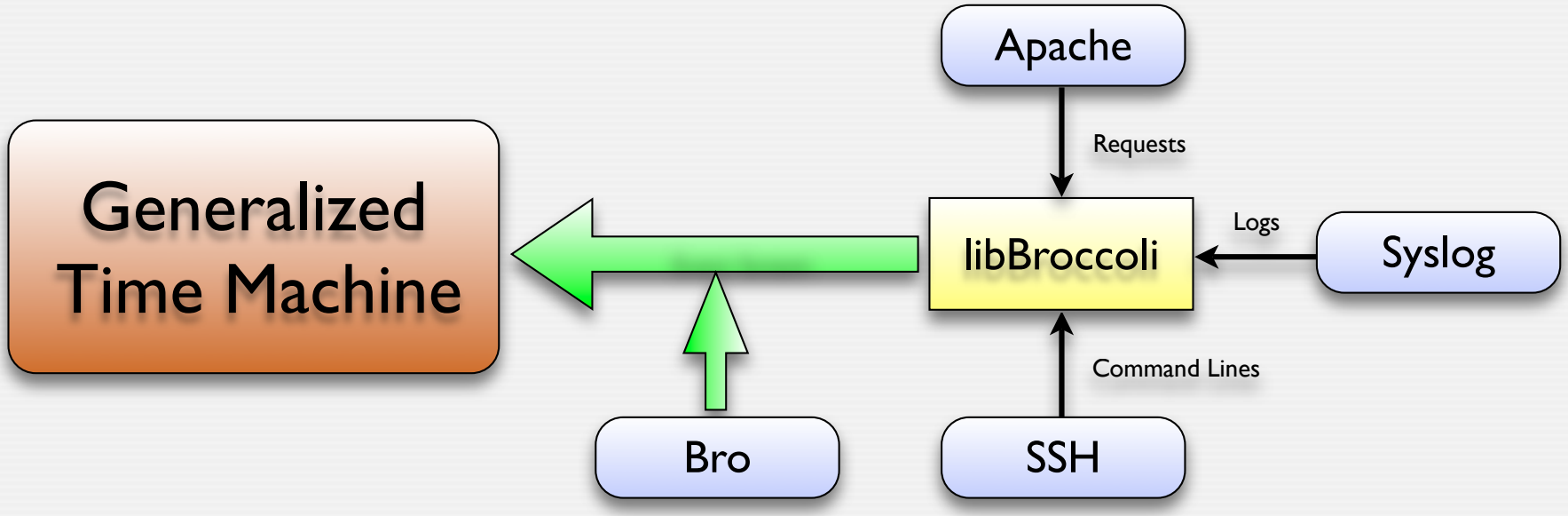


Detour: Bro's Communication System

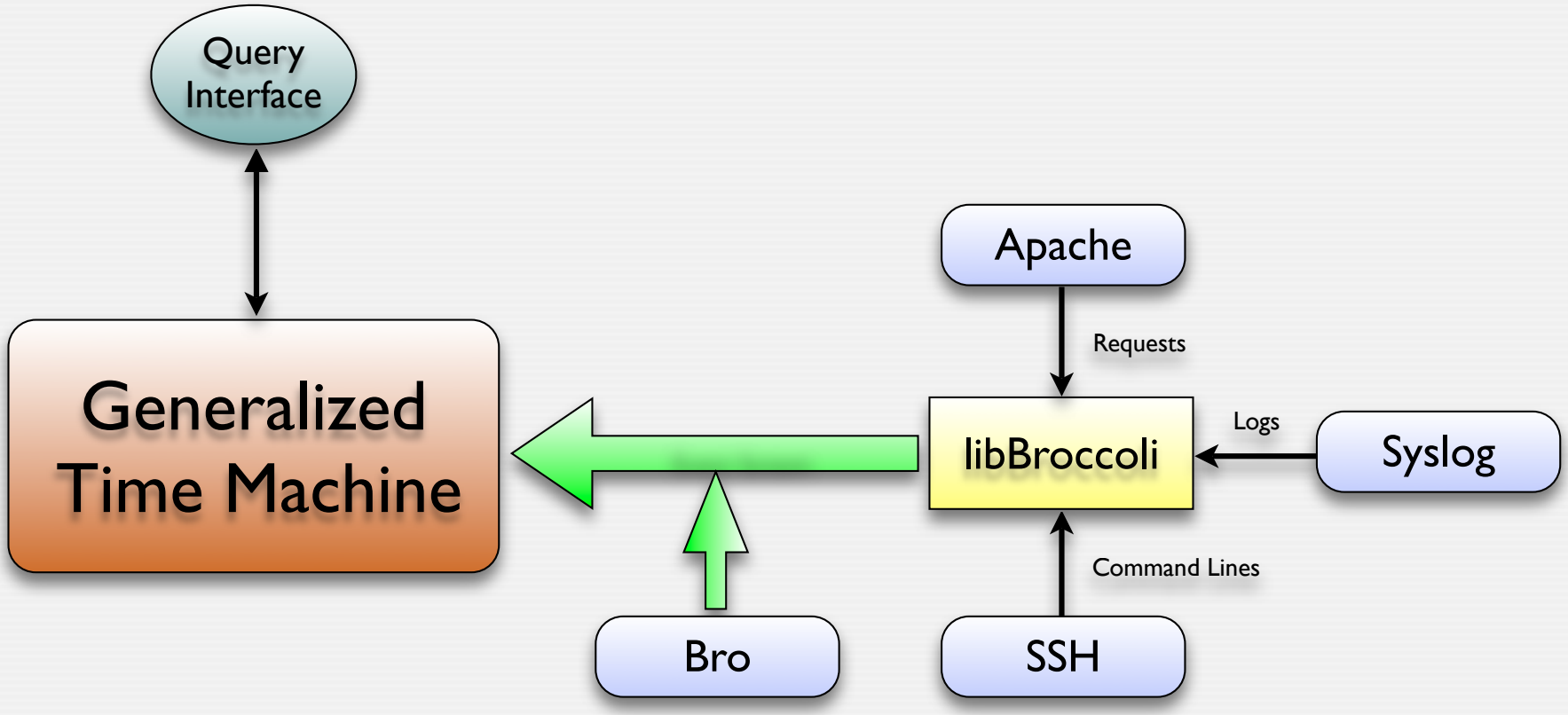
Detour: Bro's Communication System



An Archive of Network Activity



An Archive of Network Activity



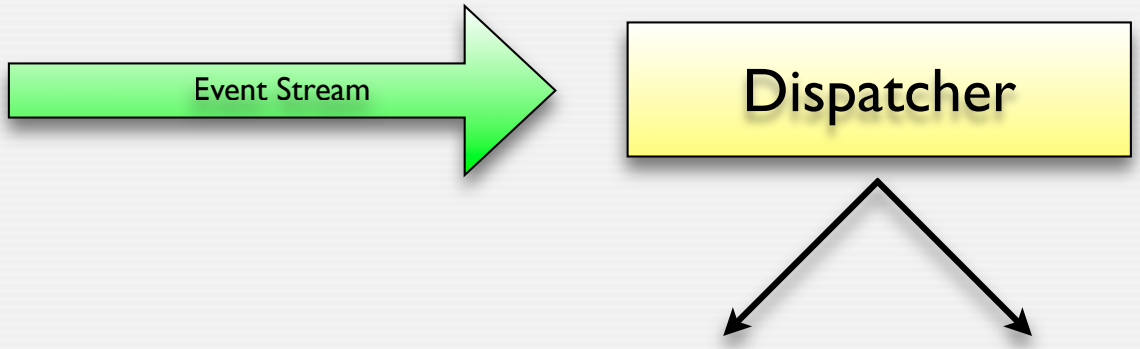
An Event-Based Data Model

- *Events* are well-suited to represent network activity
- Bro system already relies on an event model
  - Abstracts network packets into high-level, policy-neutral events
  - Events are policy-neutral (no notion "good" or "bad")
  - Security policies are specified on top of these event
- Network components can provide event streams
  - In previous work we have interfaced Bro to the external world
- We rely on events as input for the new Time Machine
  - Event archive provides a picture of the network's activity
  - We already have much of the machinery in place (Bro & Broccoli)

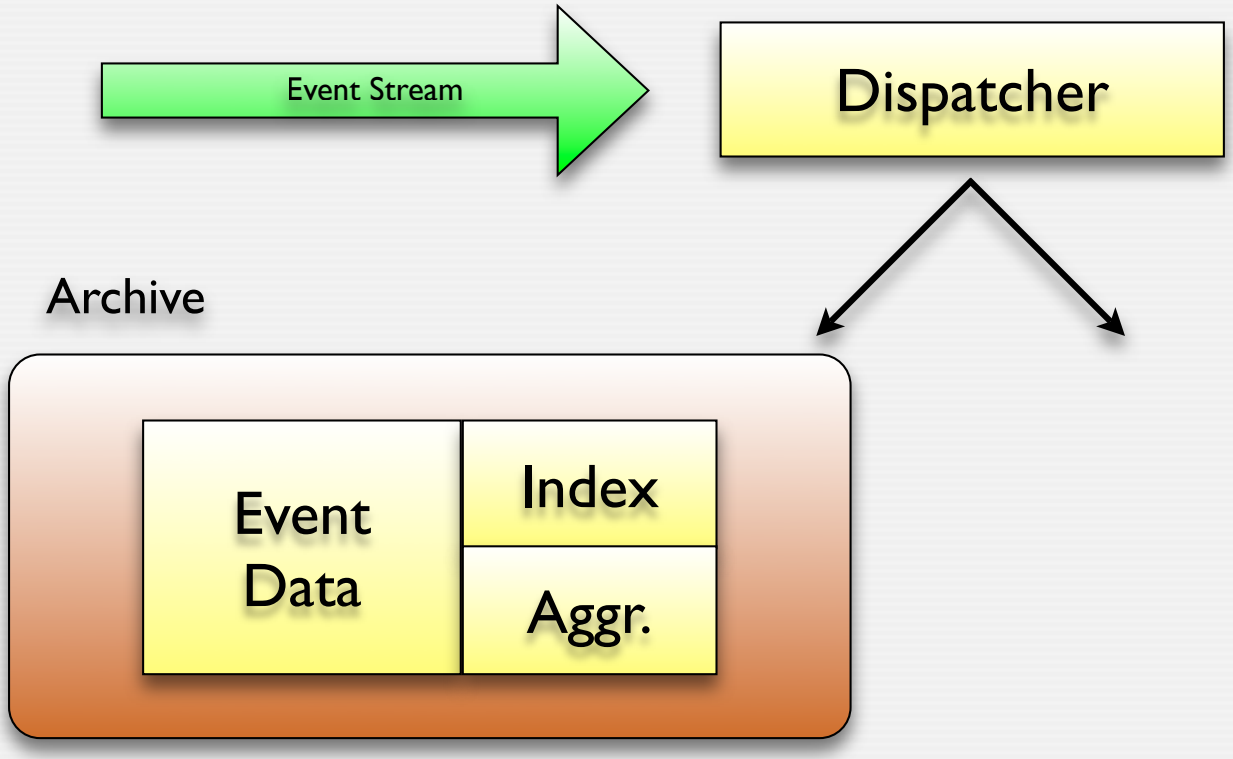
Event Model: More Advantages

- *Aggregation* gives us a powerful aging scheme
- Instead of deleting old data, we can aggregate
  - From low-level & high-volume to high-level & low-volume
  - E.g., from NetFlow records to traffic matrices
- Aggregation schemes are event-specific
  - Time Machine will provide extensive hooks for flexibility
- Events also allow sanitizing data for archival
  - E.g., anonymization before storage

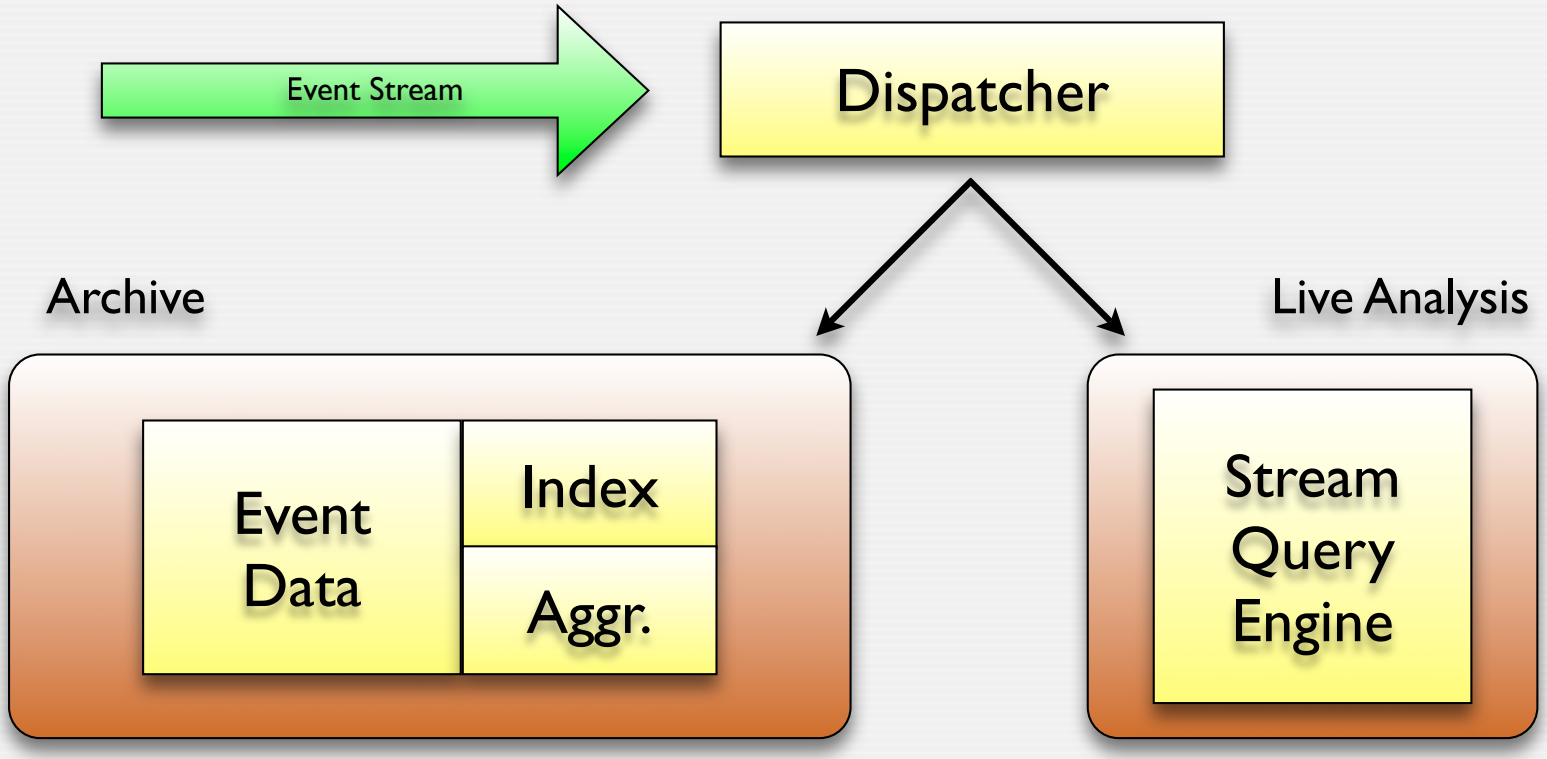
Time Machine Architecture



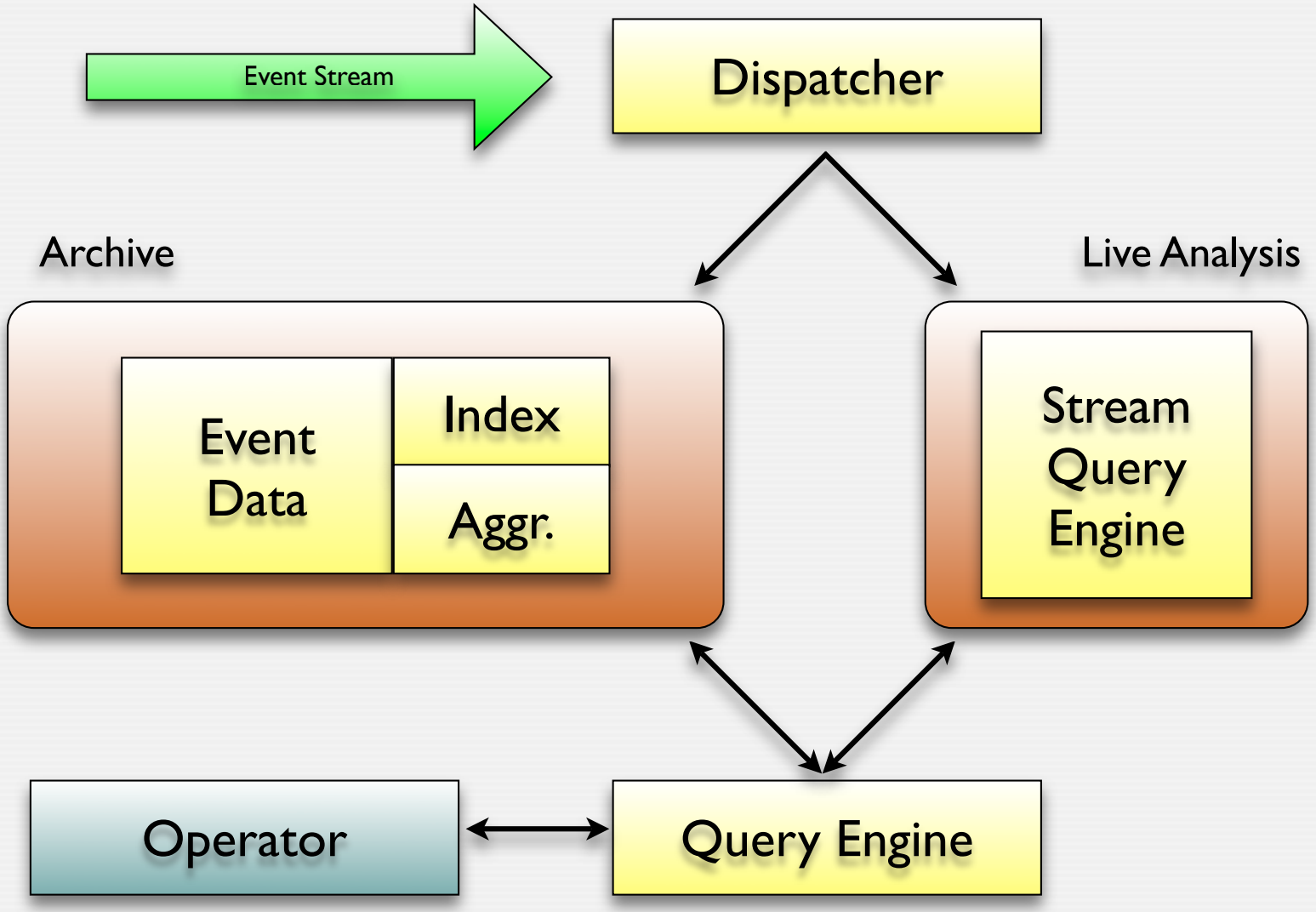
Time Machine Architecture



Time Machine Architecture



Time Machine Architecture



Distributed Cooperative Security Monitoring

When Security is Breached ...

- Lots of questions when host has been compromised
  - *"How did the attacker get in? What damage did he do? Did the guy access other hosts as well? How can we detect similar attactivy in the future?"*
  - The Time Machine setup can answer these
- But there is more: *"Was it just us?"*
  - Many attacks are global: DDOS floods, bots, scanners, phishing, etc.
- Incident analysis involves contacting other sites
  - Seeking additional information (*"Did you guys see this? What else did you find?"*)
  - Distributing results (*"You better look for this!"*)
- Manual nature of this process makes it expensive
  - Time and resources are limited; only high-priority incidents can be examined
- Goal: A platform to support sharing of information

Building a Platform For Cooperation

- However, data sharing is *really* hard ...
  - Security policies differ across sites so it is not clear what to share
  - Privacy and trust concerns severely limits what can be shared
- Existing efforts for inter-site data sharing fall short
  - Operational systems are often of limited use, e.g.,
    - DShield (data quality unclear)
    - Internet Motion Sensor (restricted to darknet space)
  - Research systems often ignore operational deployment issues
    - E.g., require automatic propagation of sensitive information
- Primary objective for us:

  A system which addresses operational constraints
- Operators need to remain in control

Restricting the Scope ...

- Data sharing is a two-edged sword
  - Administrators would like to share data
  - But are reluctant to share *(i)* with everybody, and *(ii)* in an automated fashion
- Addressing operational constraints by limiting scope
  - Exchange information across a small set of "similar" sites (e.g., DOE labs, UCs)
  - Keep the human in the loop for all policy questions
- Gives us a beneficial trust model
  - Assume that sites *usually* behave in a responsible manner
  - Yet failure of trust does not compromise security (but may require time)
- Initial focus on incident analysis
  - Provide support for the steps currently performed manually
  - E.g., *"have you seen this?"* mails

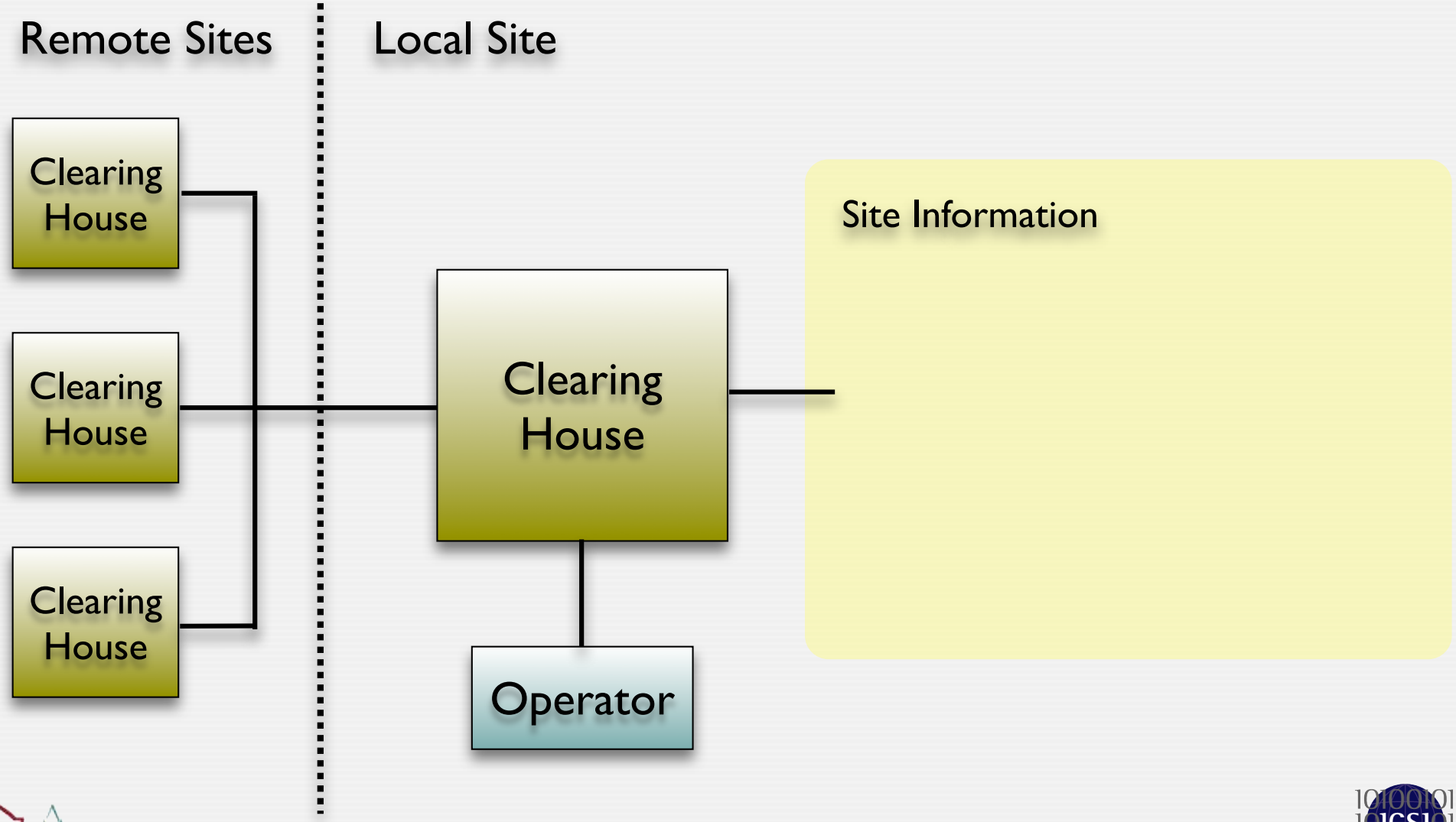
Informal Activity Description

"Attackers compromised SSH credentials, perhaps grabbed at one of **these sites**. They then fetched their botware from two different places, with filenames ending in **.jpg** even though the files are executables. They opened up backdoors on the **following ports**. These other sites connected to **those backdoors**. One signature of the activity is outbound email from a compromised host with **this subject**, or—maybe—Web activity fetching **these URLs**, but unfortunately we're not sure. If you find evidence of infection and can tell us about other URLs or attacking sites, we would much appreciate that so we can check them against our logs."

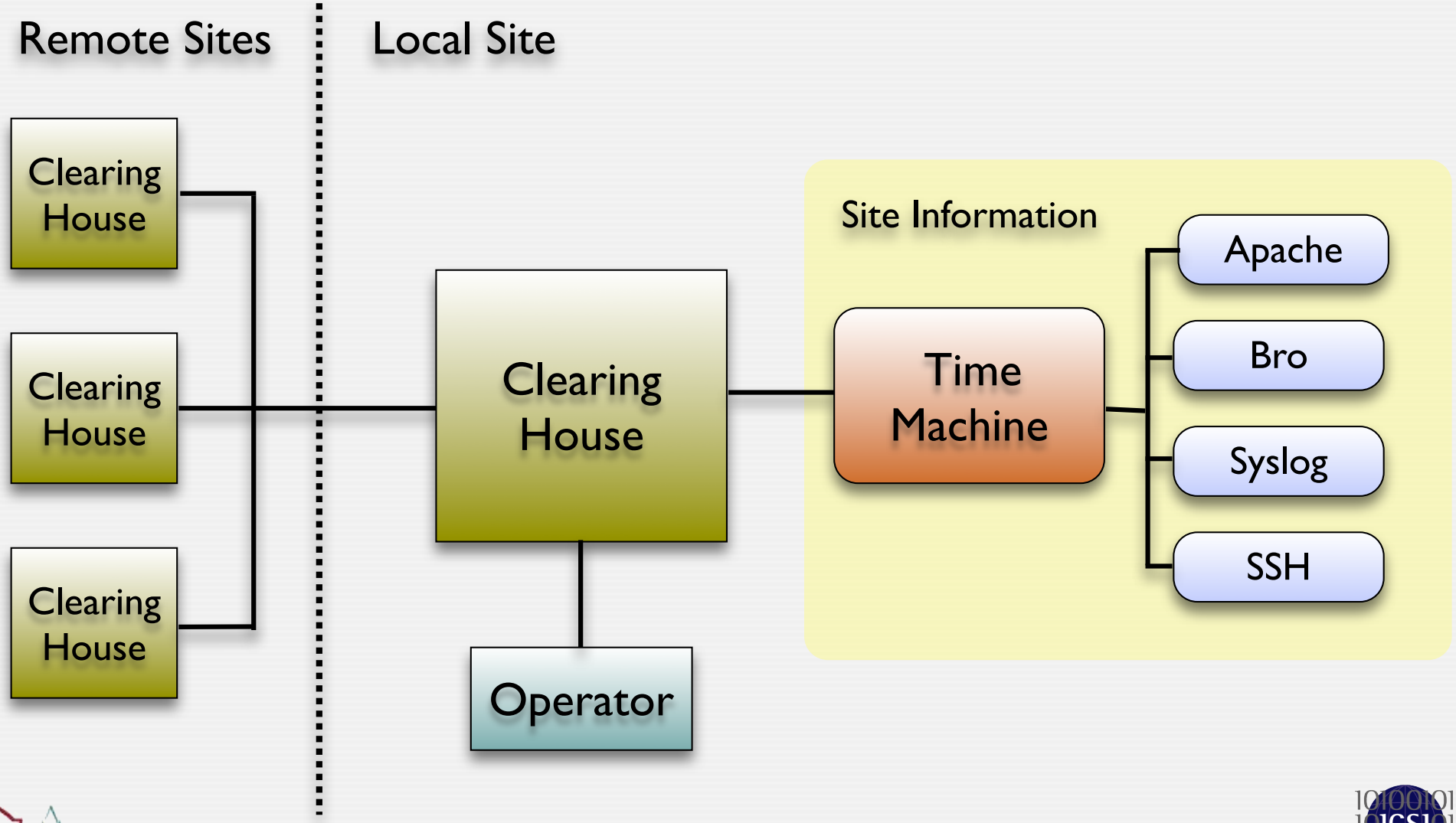
(Semi-)Automating the Process

- Once an incident is understood, a site codifies the relevant activity into a high-level *script*
  - Describes how to locate activity in *past* logs and in *future* activity
  - Describes what kind of information site would like to know from its partners
- Script is send to partners
- Receiving analyst manually decides whether
  - Is of interest
  - Confirms to local policy (e.g., privacy constraints)
  - Sufficient resources are available to run the script
- If so, the receiver executes the script to
  - Find past/future attacks on the local network
  - Answer the queries of sender

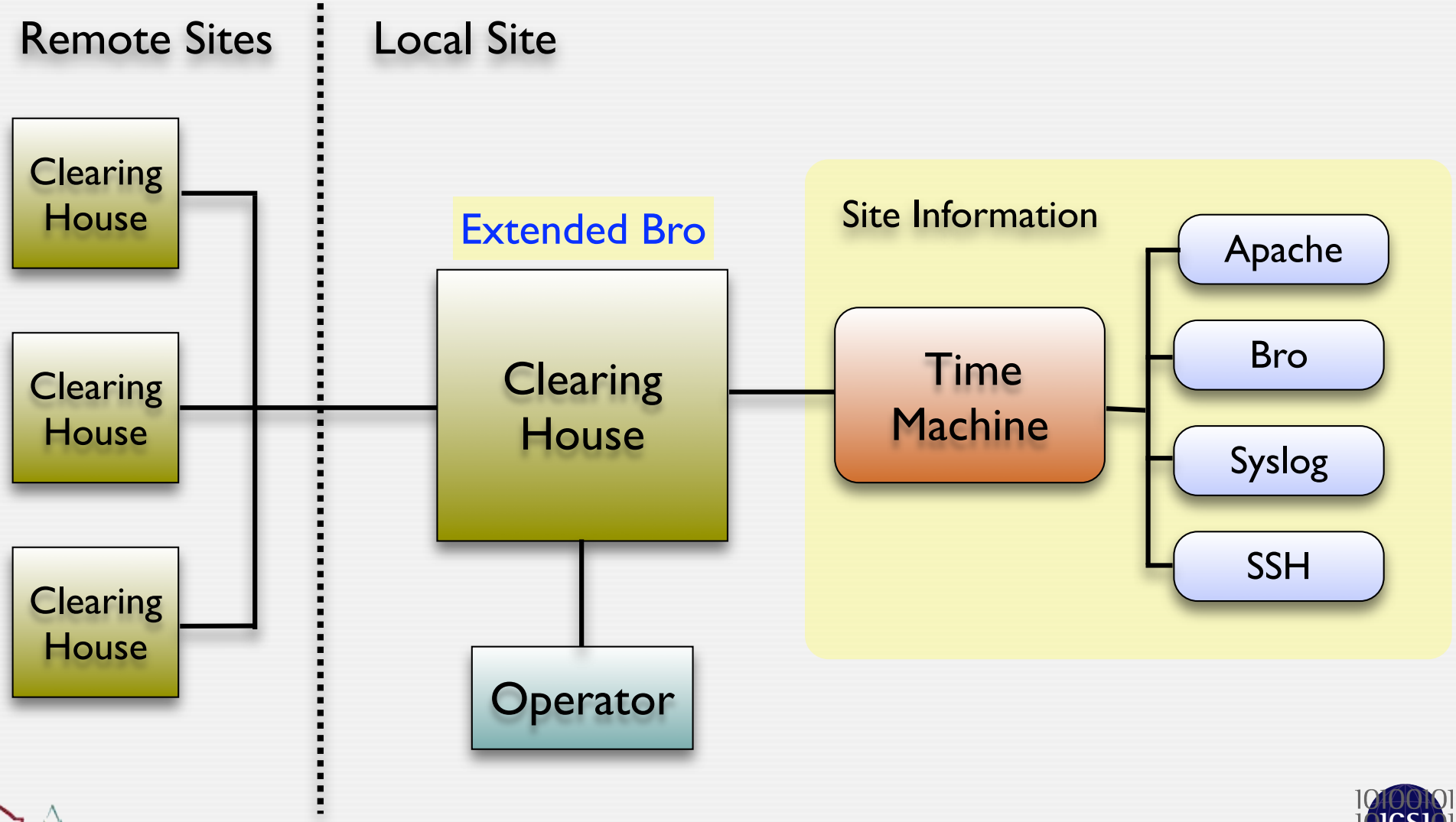
Clearing House Architecture



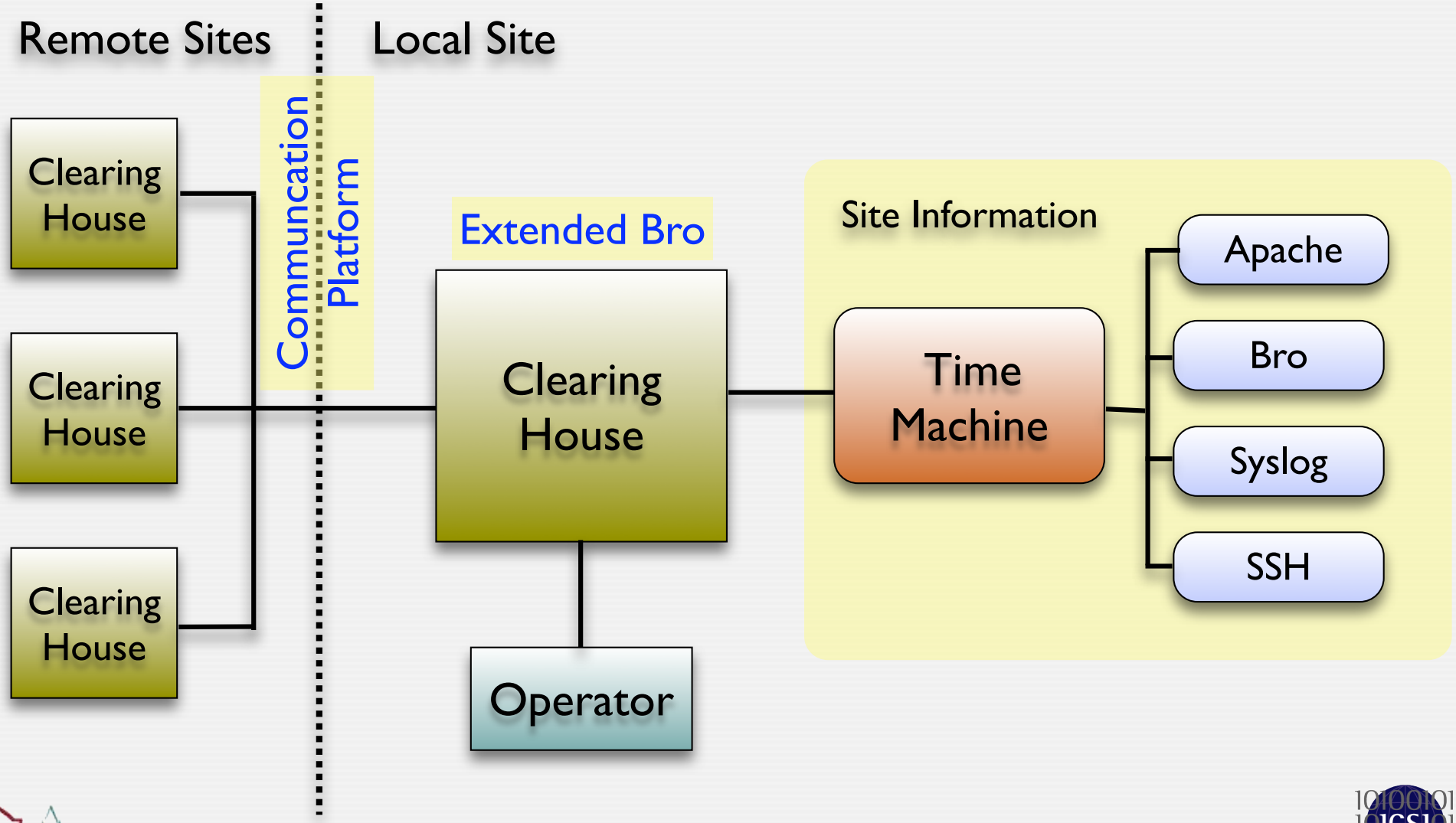
Clearing House Architecture



Clearing House Architecture



Clearing House Architecture



Building a Communication Platform

- Designing a communication protocol
  - With support for publish/subscribe and point-to-point communication
  - With security in mind
- Building a C++ library which implements the protocol
  - Provides overlay functionality
  - Has plenty of hooks to control the data flow
  - Application-independent
- Simple first application: Hot-list sharing via Bro
  - As test of the library in a real setting
- Prototype should be available in 2-3 months

Summary

Summary

- Three current projects
  - The *Bro Network Intrusion Detection System*
  - The *Time Machine*: An Archiver for Long-term Storage of Activity
  - A framework for distributed cooperative security monitorin
- The projects will supplement another eventually
- Focus is on real-world deployment
  - Need to ensure that systems are deployable in large-scale networks
- Other projects:
  - The Bro Cluster - Transparent load-balancing for intrusion detection
  - Network intrusion detection on multi-core platforms

Any questions ... ?

Robin Sommer

Lawrence Berkeley National Laboratory &
International Computer Science Institute

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

This work is supported by the Office of Science and Technology at the Department of Homeland Security. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Office of Science and Technology.