The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware

Matthias Vallentin[1], *Robin Sommer*[2,3],
Jason Lee[2], Craig Leres[2]
Vern Paxson[3,2], and Brian Tierney[2]

[1] *TU München*
[2] *Lawrence Berkeley National Laboratory*
[3] *International Computer Science Institute*

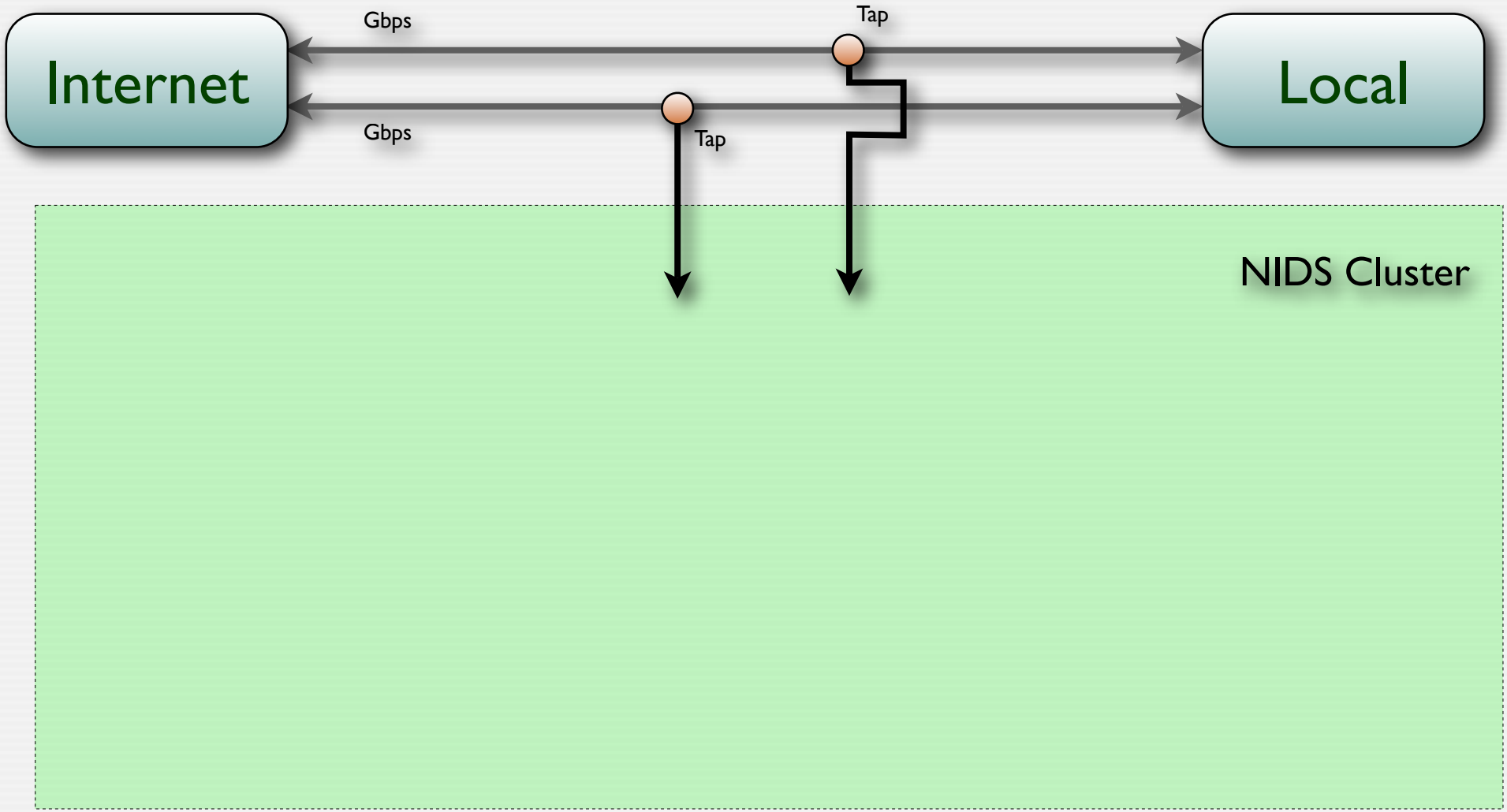
Motivation

- NIDSs have reached their limits on commodity hardware
  - Keep needing 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
- To overcome, we must either
  - Restrict the amount of analysis, or
  - Turn to expensive, custom hardware, or
  - Employ some form of load-balancing to split analysis across multiple machines
- 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

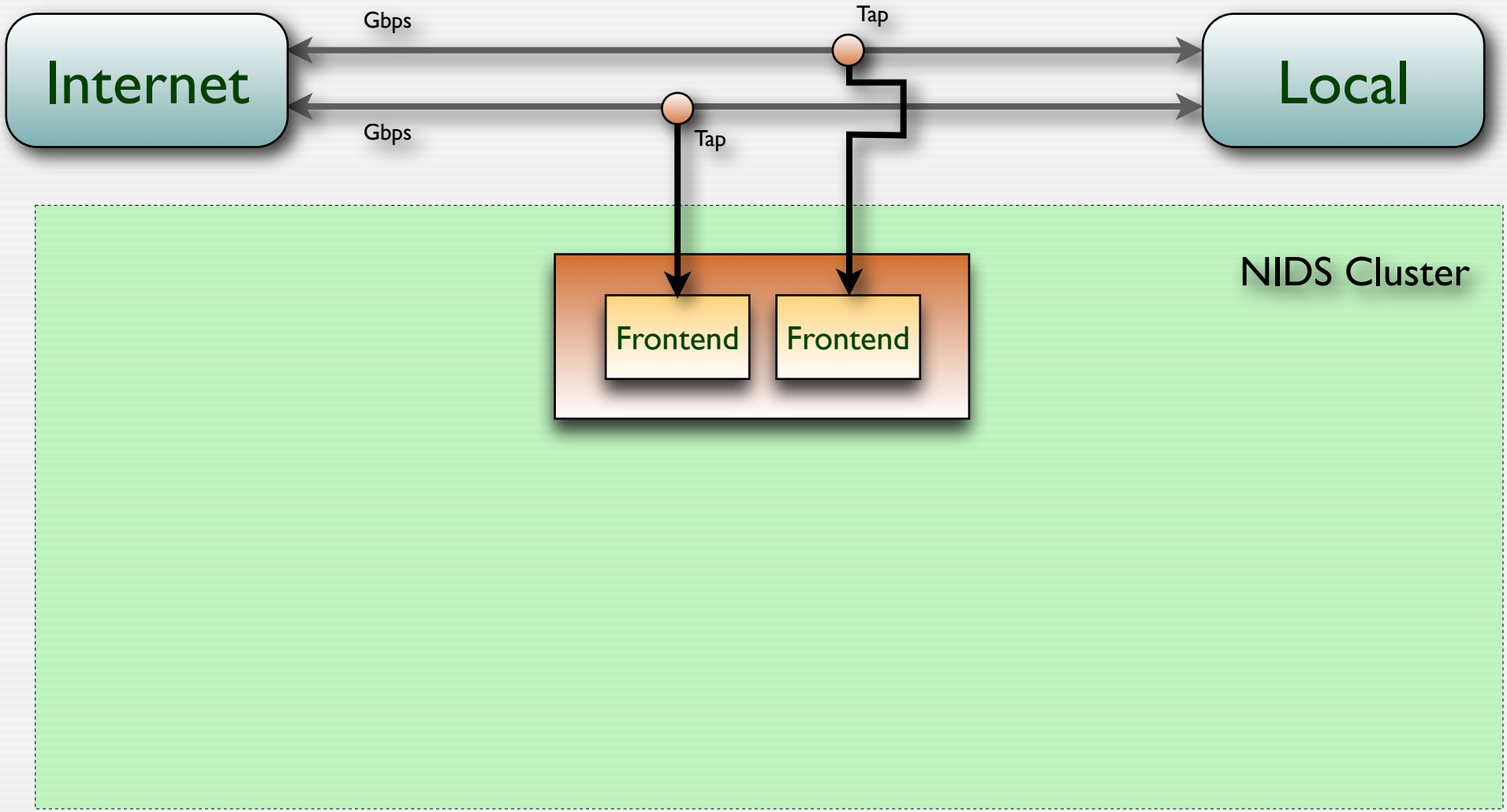
Correlation is Tricky ...

- Most NIDS provide support for multi-system setups
- However, instances tend to work independent
  - Central manager collects alerts of independent NIDS instances
  - *Aggregates* results instead of *correlating* analysis
- 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
  - Is scalable to large number of nodes
  - Still provides a single system as the user interface (logging, configuration updates)

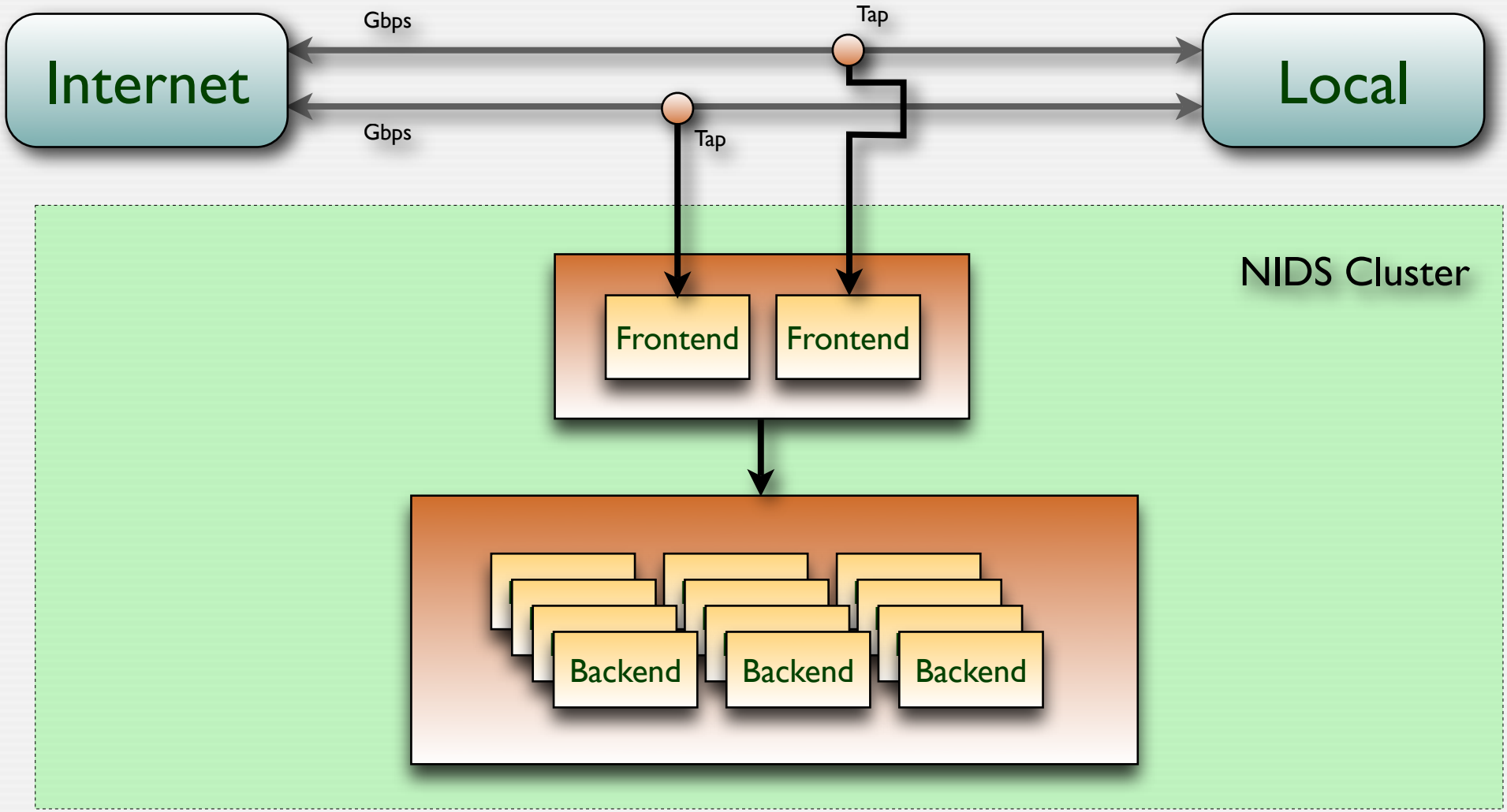
Architecture



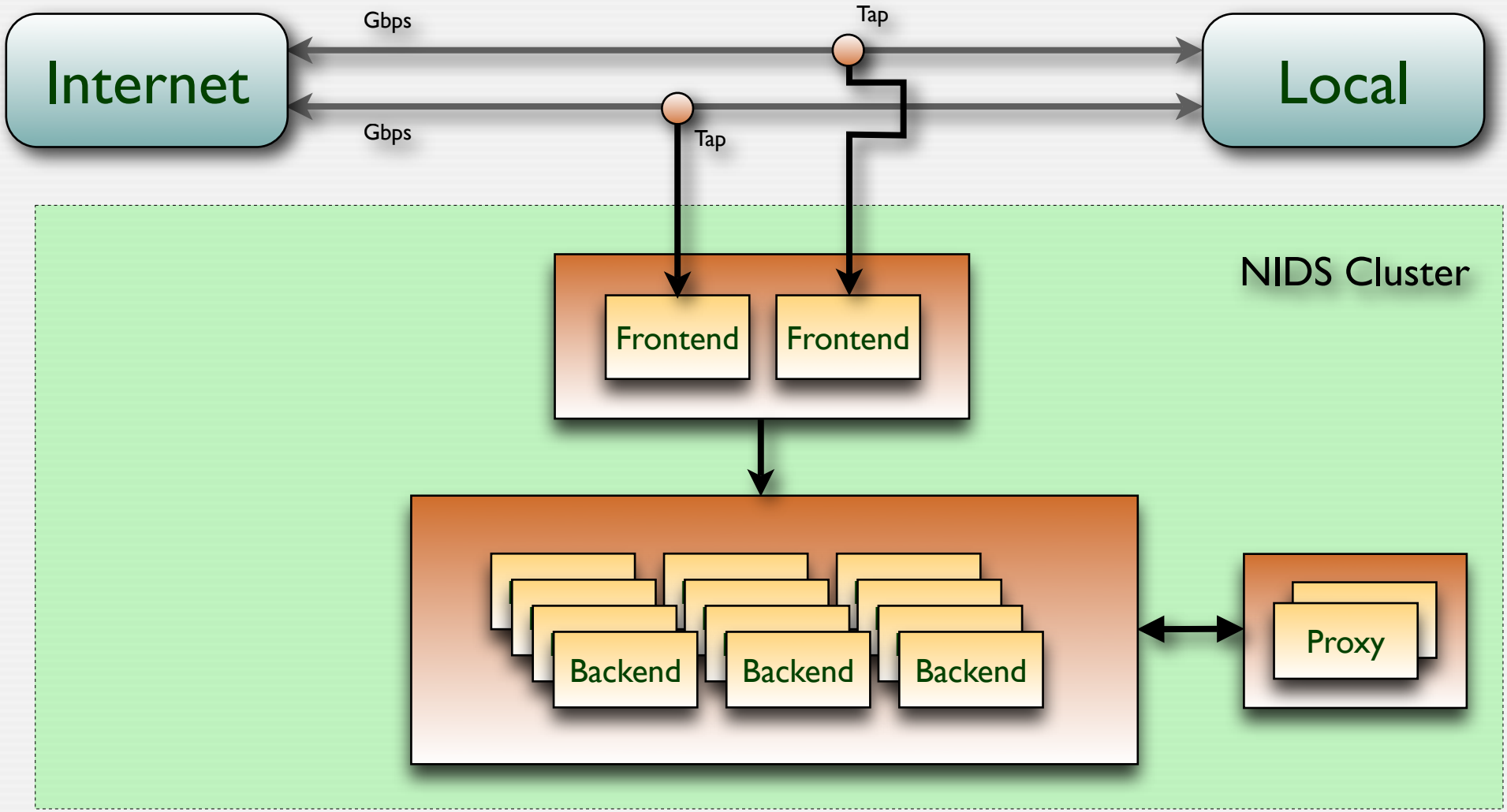
Architecture



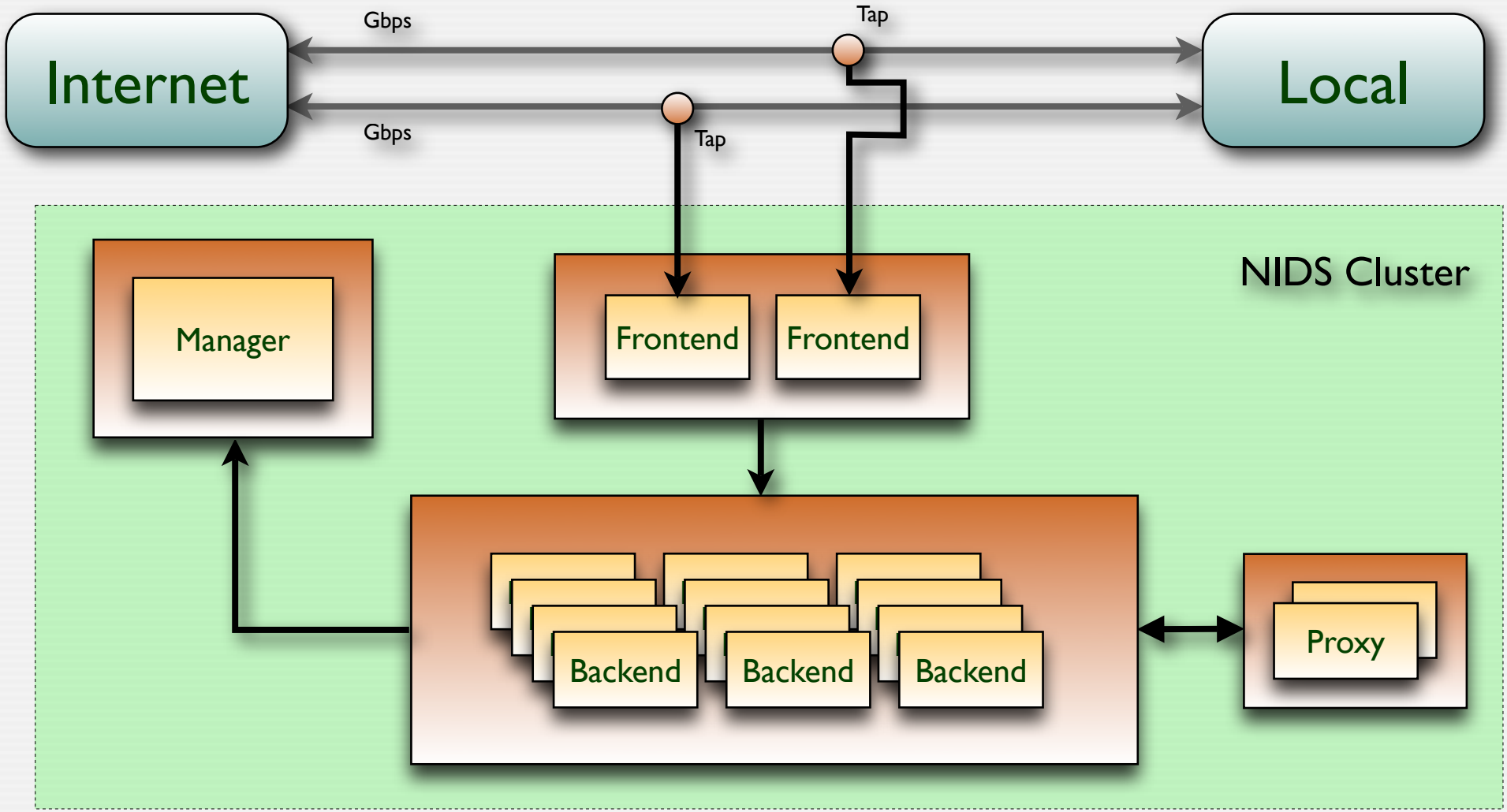
Architecture



Architecture



Architecture



Environments

- Initial target environment:
  Lawrence Berkeley National Laboratory
  - 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 with 1 frontend & 10 backends
- 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 / 2 backends (10 planned)
  - 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

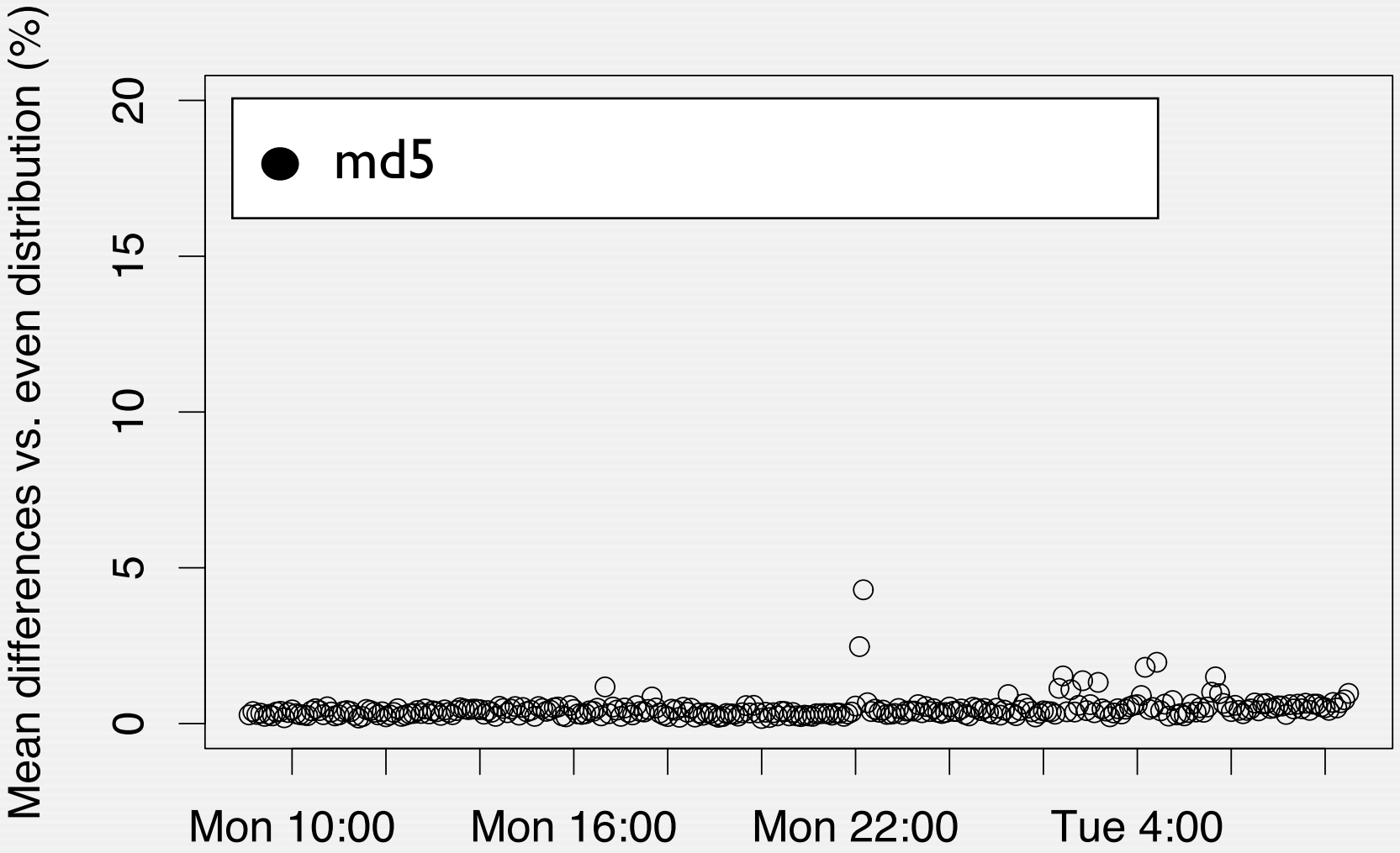
1. Distributing the traffic evenly while minimizing need for communication
2. Adapting the NIDS operation on the backend to correlate analysis with peers
3. Validating that the cluster produces sound results

Distributing Load

Distribution Schemes

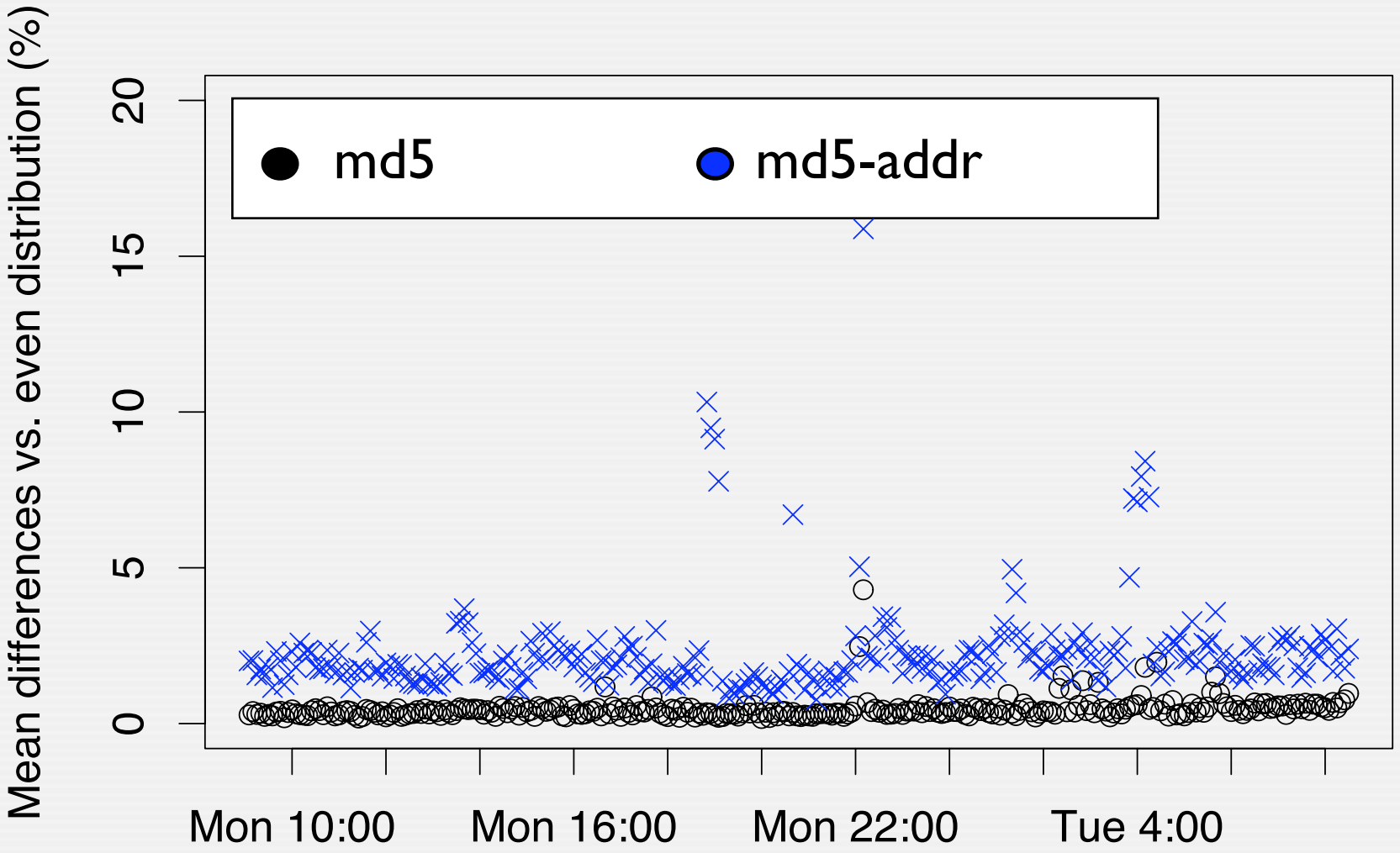
- Frontends need to pick a backend as destination
- Option 1: Route packets individually
  - Simple example: round-robin
  - Too expensive due to communication overhead (NIDS keep per-flow state)
- Option 2: Flow-based schemes
  - Send all packets belonging to the same flow to the same backend
  - Needs communication only for inter-flow analysis
- Simple approach: hashing flow identifiers
  - E.g., `md5(src-addr + src-port + dst-addr + dst-port) mod n`
  - Hashing is *state-less*, which reduces complexity and increases robustness
- But how well does hashing distribute the load?

Simulation of Hashing Schemes



1 day of UC Berkeley campus TCP traffic (231M connections), n = 10

Simulation of Hashing Schemes



1 day of UC Berkeley campus TCP traffic (231M connections), n = 10

Cluster Frontends

- We chose the address-based hash
  - Ports not always available (e.g., ICMP, fragments) & more complex to extract
  - Even with perfect distribution, load is hard to predict
- Frontends rewrite MAC addresses according to hash
- Two alternative frontend implementations
  - In software with Click (SHA1)
  - In hardware with a prototype of Force-10's P10 appliance (XOR)

Adapting the NIDS

Cluster Backends

- On the backends, we run the Bro NIDS
  - Bro is the NIDS used in our primary target environment LBNL
  - Bro already provides extensive, low-level communication facilities
- Bro consists of two layers
  - Core: Low-level, high-performance protocol analysis
  - Event-engine: Executes scripts which implement the detection analysis
- Observation: Core keeps only per-flow state
  - No need for correlation across backends
- Event-engine does all inter-flow analysis
  - The scripts needs to be adapted to the cluster setting

Adapting the Scripts ...

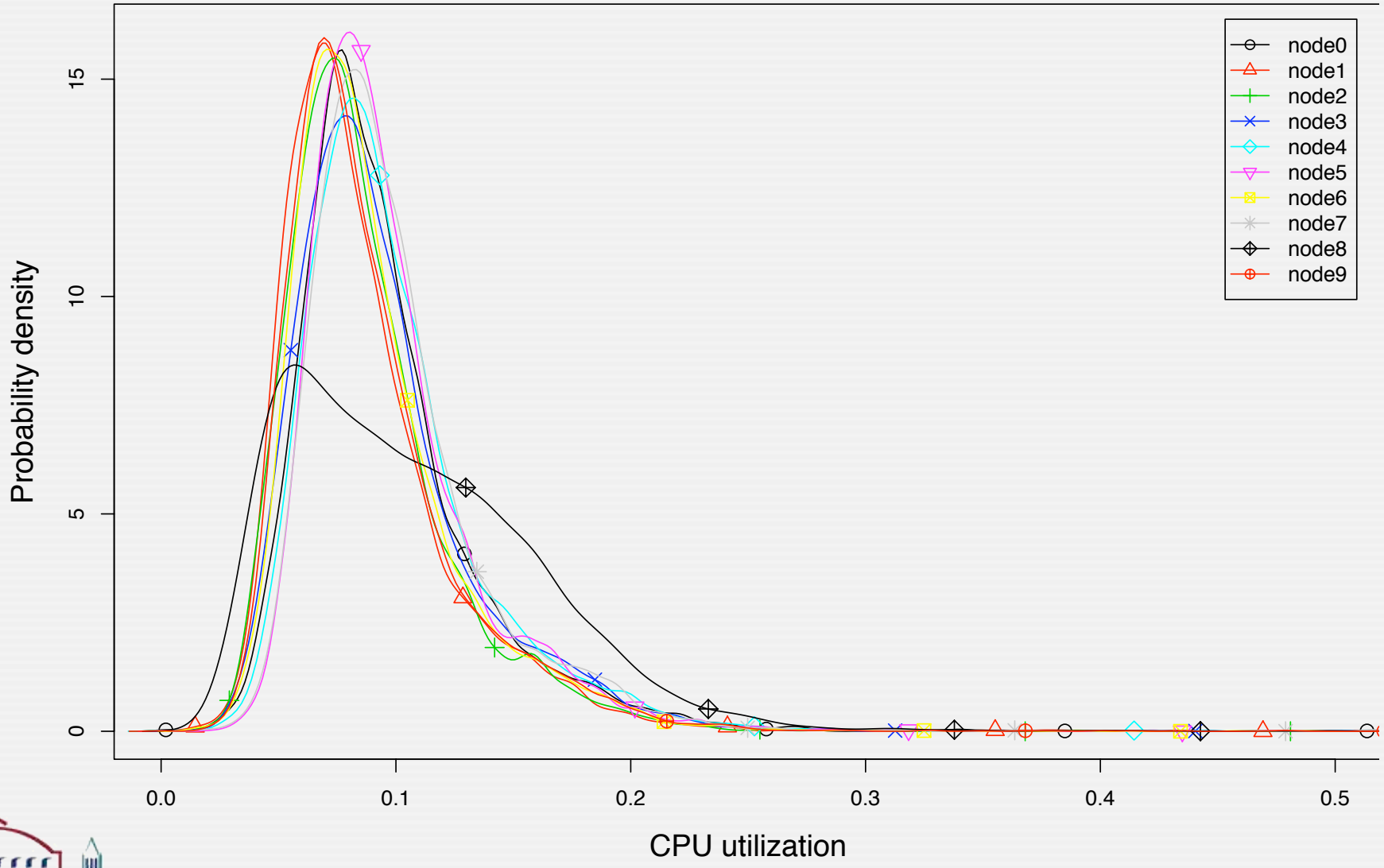
- Script language provides primitives to share state
  - Almost all state is kept in tables, which can easily be synchronized across peers
- Main task was identifying state related to inter-flow analysis
  - A bit cumbersome with 20K+ lines of script code ...
- Actually it was a bit more tricky ...
  - Some programming idioms do not work well in the cluster setting and needed to be fixed
  - Some trade-offs between gain & overhead exists are hard to assess
  - Bro's "loose synchronisation" introduces inconsistencies (which can be mitigated)
- Many changes to scripts and few to the core
  - Will be part of the next Bro release

Validating the Cluster

Accuracy

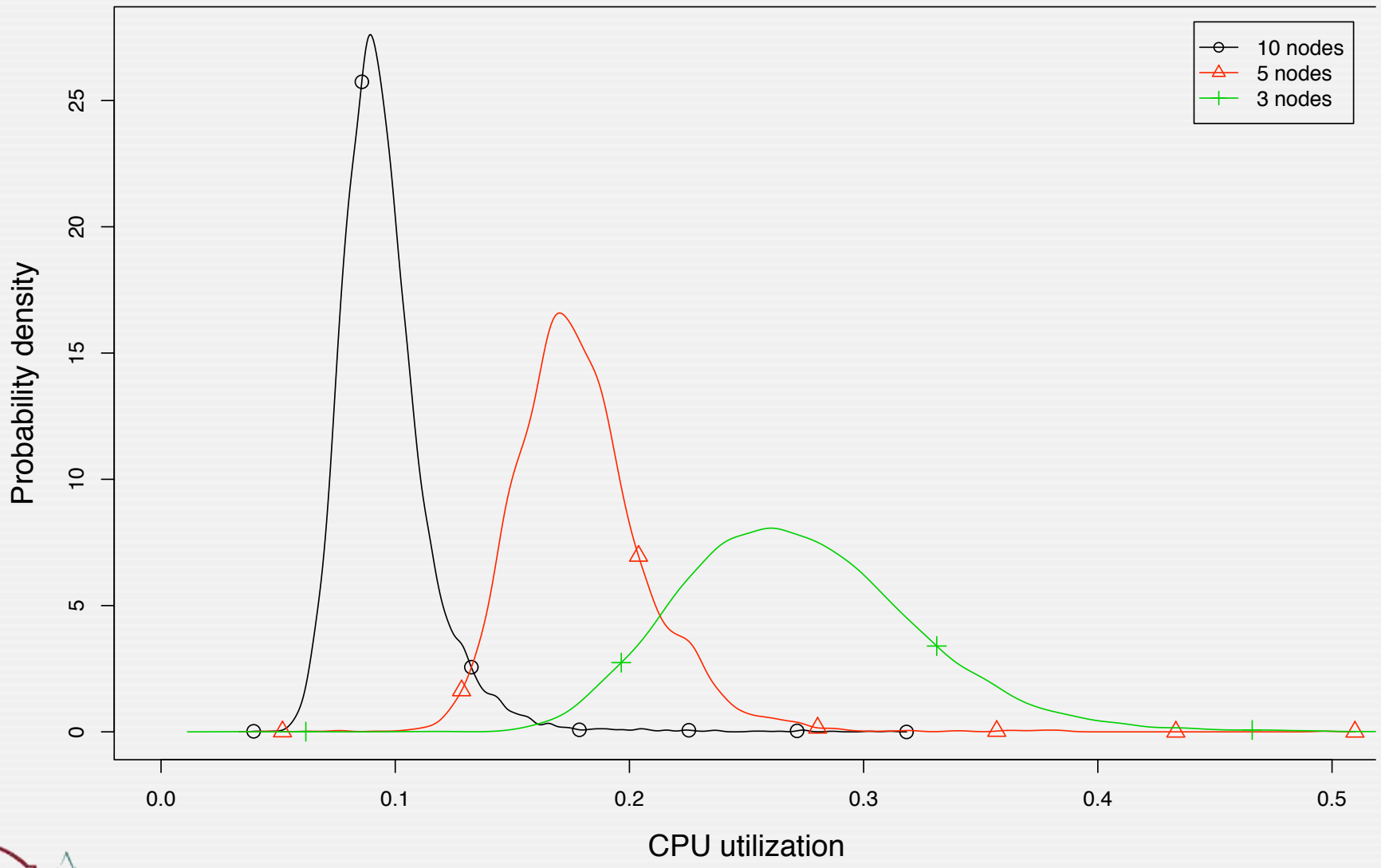
- Goal: Cluster produces same result as a single system
- Compared the results of cluster vs. stand-alone setup
  - Captured a 2 hour trace at LBNL's uplink (~97GB, 134M pkts, 1.5 M host pairs)
  - Splitted the trace into slices and copied them to the cluster nodes
  - Setup the cluster to examine the slices just as if it would process live traffic
  - Compared output of the manager with the output of a single Bro instance on the trace
- Found excellent match for the alarms & logs
  - Cluster reported all 2661 alarms of the singe instance as well
  - Slight differences in timing & context due to latency and synchronization semantics
  - Some artifacts of the off-line measurement setup

CPU Load per Node



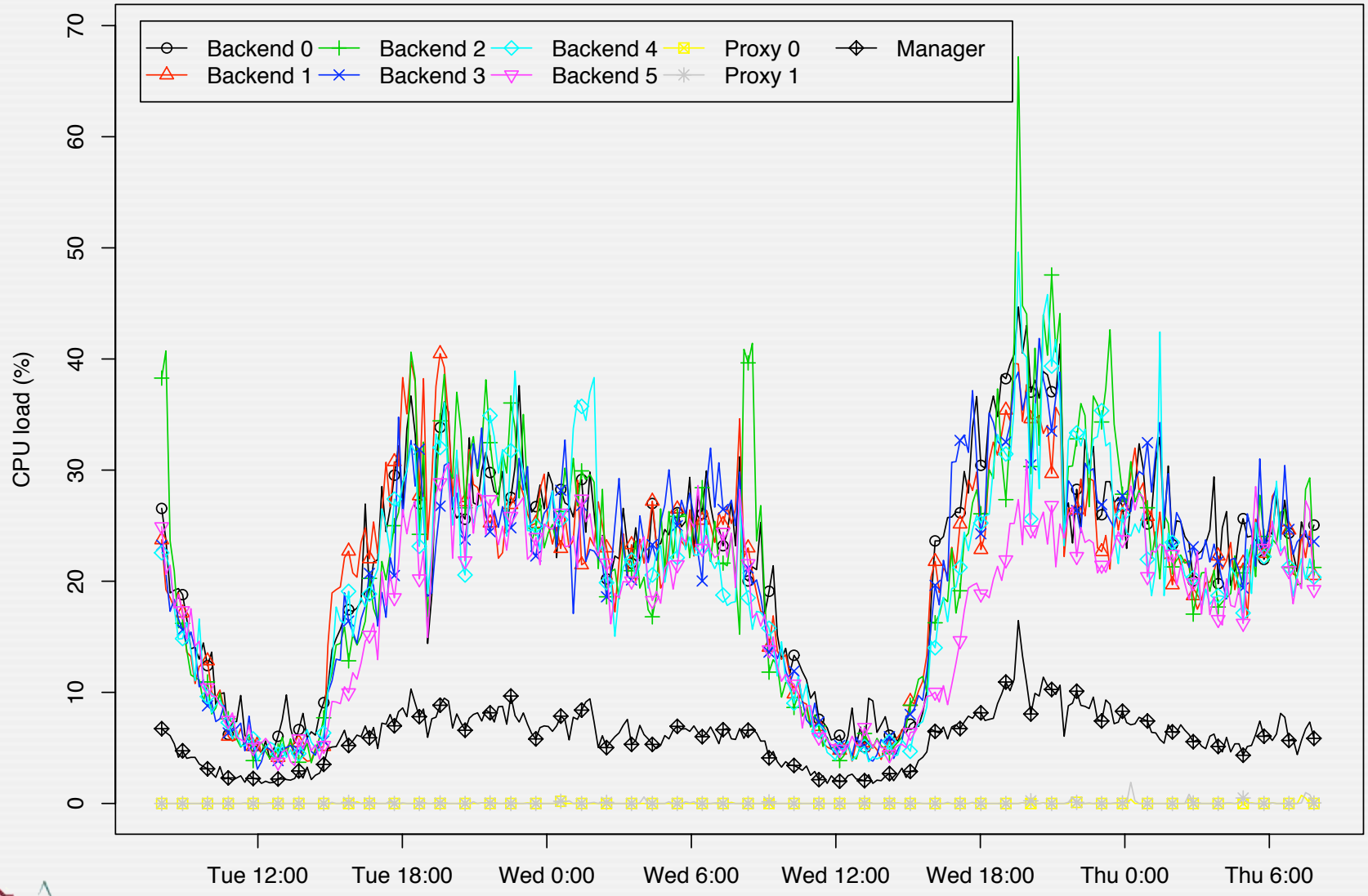
10 backends, ext. LBNL config, 2hr full trace, (~97GB, 134M pkts)

Scaling of CPU



ext. LBNL config, 2hr full trace, (~97GB, 134M pkts)

Load on Berkeley Campus



With 1 frontend = 50% of the total traffic

Conclusion & Outlook

- Cluster monitors Gbps networks on commodity hardware
  - Provides high-performance, stateful network intrusion detection
  - Correlates analysis across its nodes rather than just aggregating results
- When building the cluster we
  - Examined different load distribution schemes
  - Adapted an open-source NIDS to the cluster setting
  - Evaluated correctness & performance in a real-world setting
- Challenge was to build something which works
  - Less to lead into fundamentally new research directions
- Now in the process of making it production quality
- We will soon release the *Cluster Shell*
  - An interactive shell running on the manager

Any questions ... ?

Robin Sommer

Lawrence Berkeley National Laboratory &
International Computer Science Institute

robin@icir.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.