Parallelizing Network Analysis

Robin Sommer

Lawrence Berkeley National Laboratory &
International Computer Science Institute

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

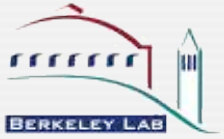
Motivation

- NIDSs have reached their limits on commodity hardware
  - Keep needing to do *more analysis* on *more data* at *higher speeds*
  - Analysis gets richer over timer, 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* task
  - Need to exploit parallelism to cope with load

Orthogonal Approaches

- The NIDS Cluster
  - Many PCs instead of one
  - Communication and central user interface creates the impression of one system
- Vision: Parallel operation within a single NIDS instance
  - In software: multi-threaded analysis on multi-CPU/multi-core systems
  - In hardware: compile analysis into a parallel execution model (e.g., on FPGAs)

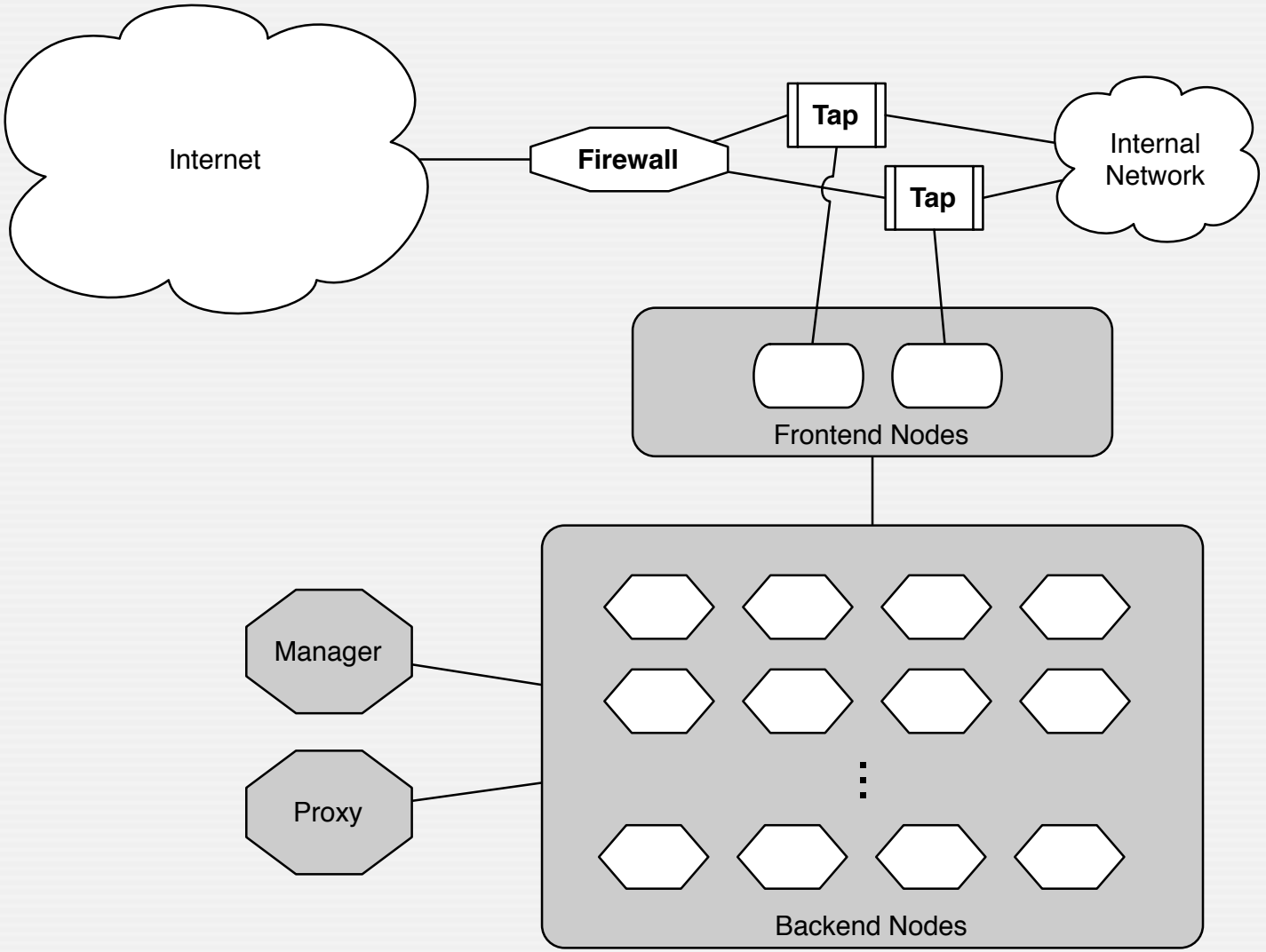
The NIDS Cluster



The NIDS Cluster

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

Architecture



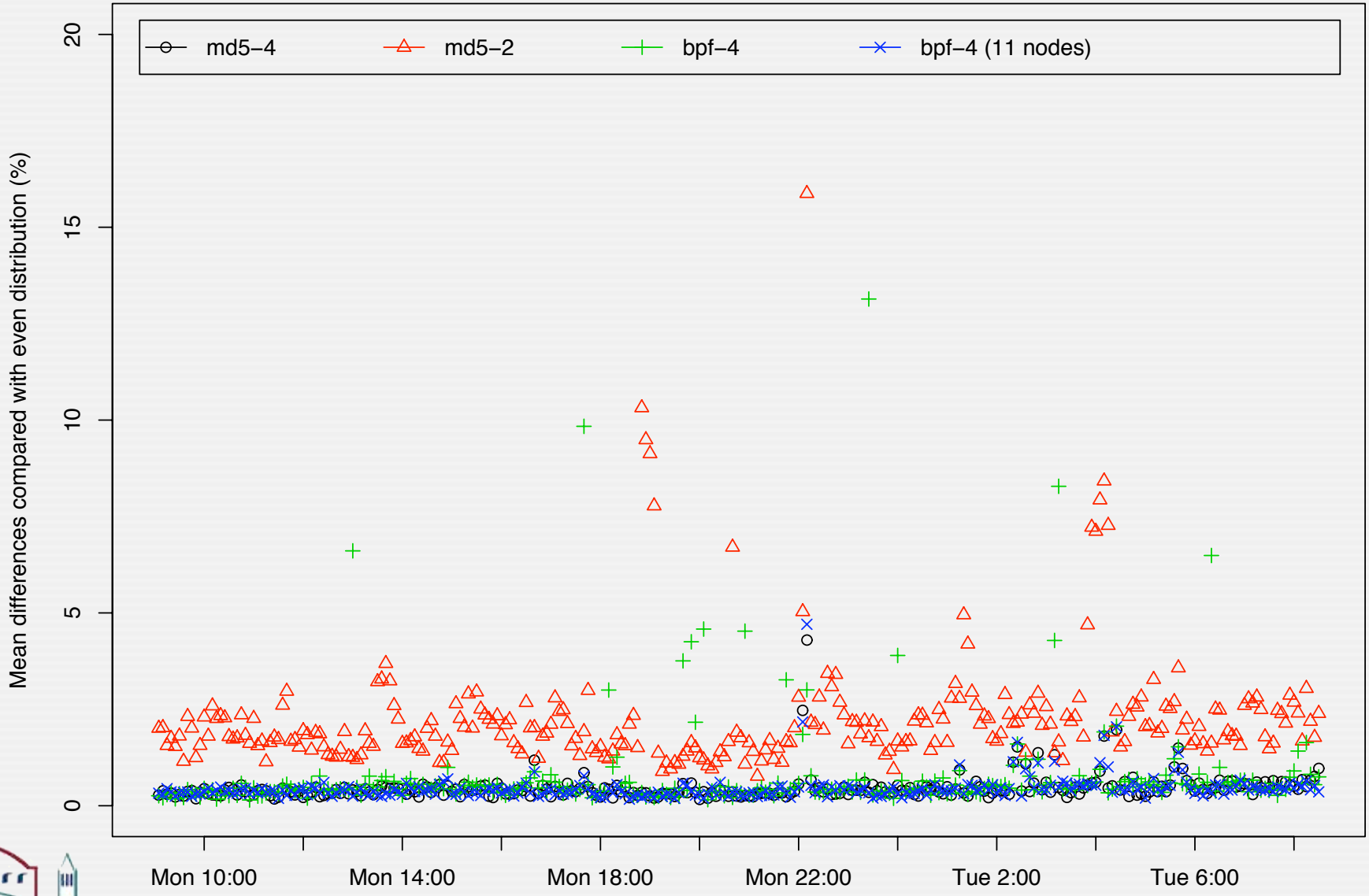
Prototype Setups

- Lawrence Berkeley National Laboratory
  - Monitors 10 Gbps upstream link
  - 1 front-end, 10 backends
- University of California, Berkeley
  - Monitors 2x1Gbps upstream links
  - 2 front-ends, 6 backends
- IEEE Supercomputing 2006
  - Conference's 1 Gbps backbone network
  - 100 Gbps *High Speed Bandwidth Challenge* network (partially)
- Goal: Replace current operational security monitoring

Front-Ends

- Distribute traffic to back-ends by rewriting MACs
  - In software via Click
  - In hardware via Force-10's P10 (prototype in collaboration with F10)
- Fault-tolerance
  - Easy to retarget traffic if a back-end node fails
- Per connection-hashing
  - Either 4-tuple (addrs,ports) or 2-tuple (addrs)
  - MD5 mod n, ADD mod n

Simulation of Hashing Schemes



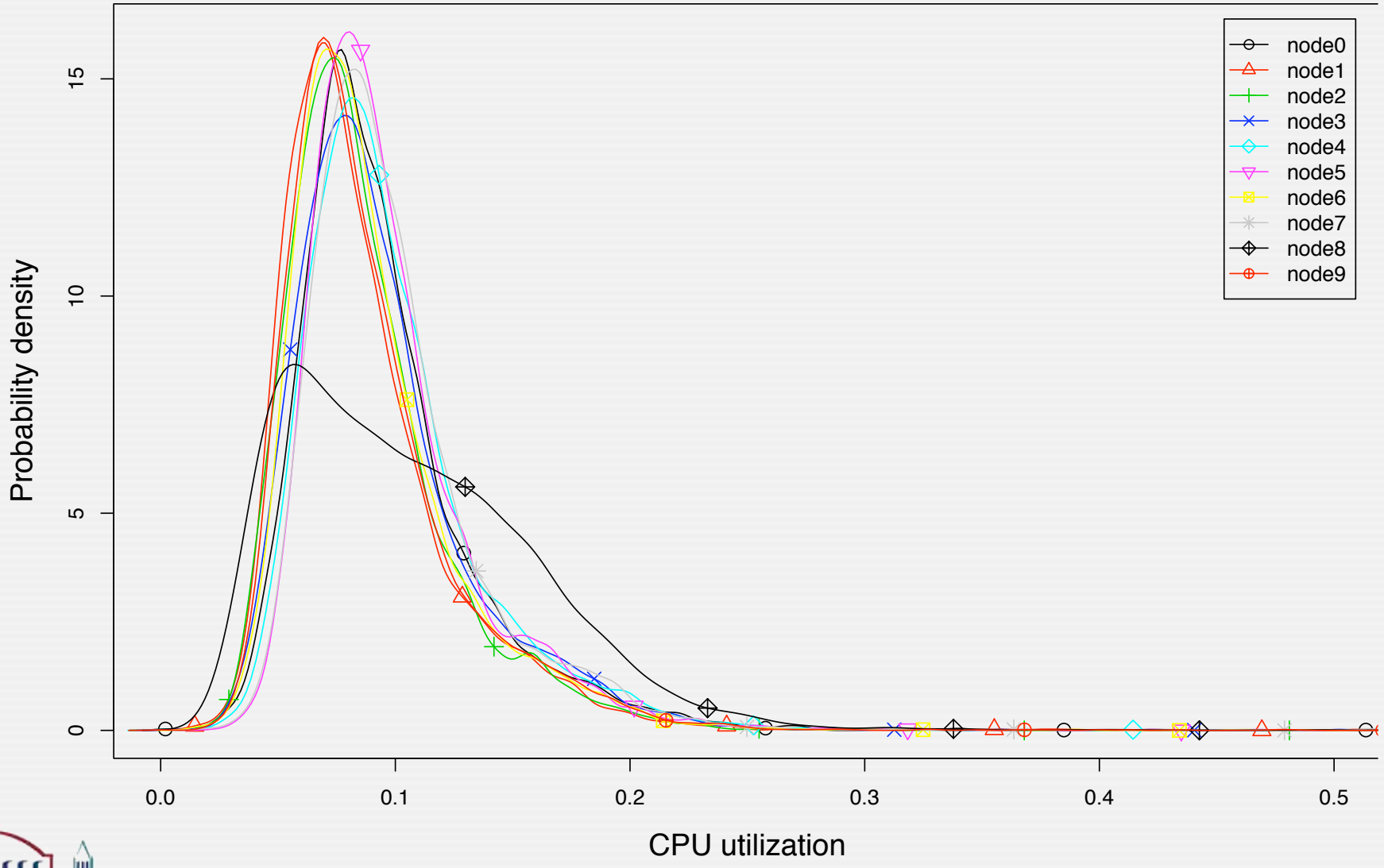
Back-ends

- 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

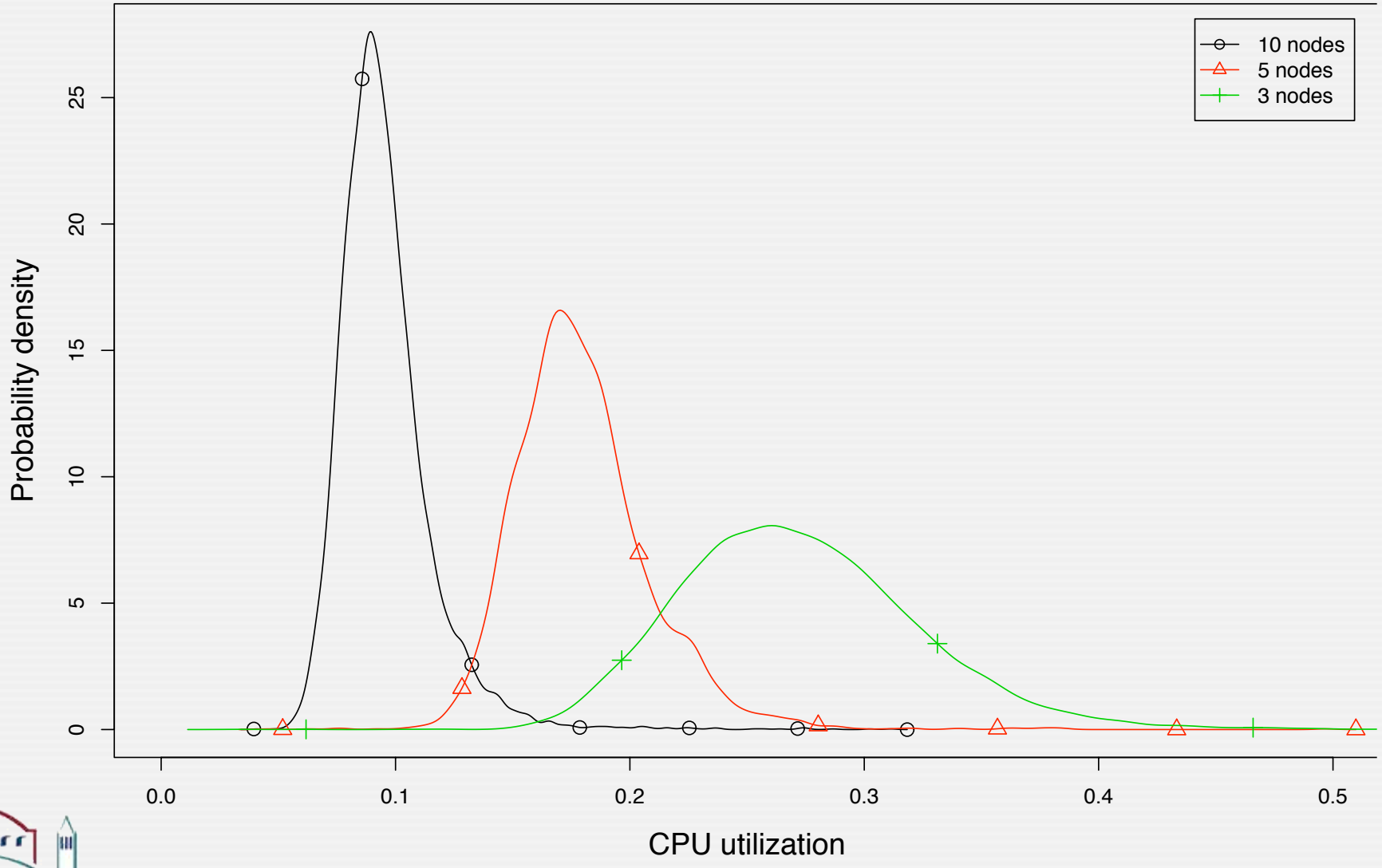
Evaluation & Outlook

- Prototypes are running nicely
  - Are able to perform analysis not possible before
  - E.g., full HTTP analysis & Dynamic Protocol Detection/Analysis
- Now in the process of making it production quality
- Evaluation
  - Verified accuracy by comparing against single Bro instance
  - Evaluated performance wrt load-balancing quality, scalability, overhead

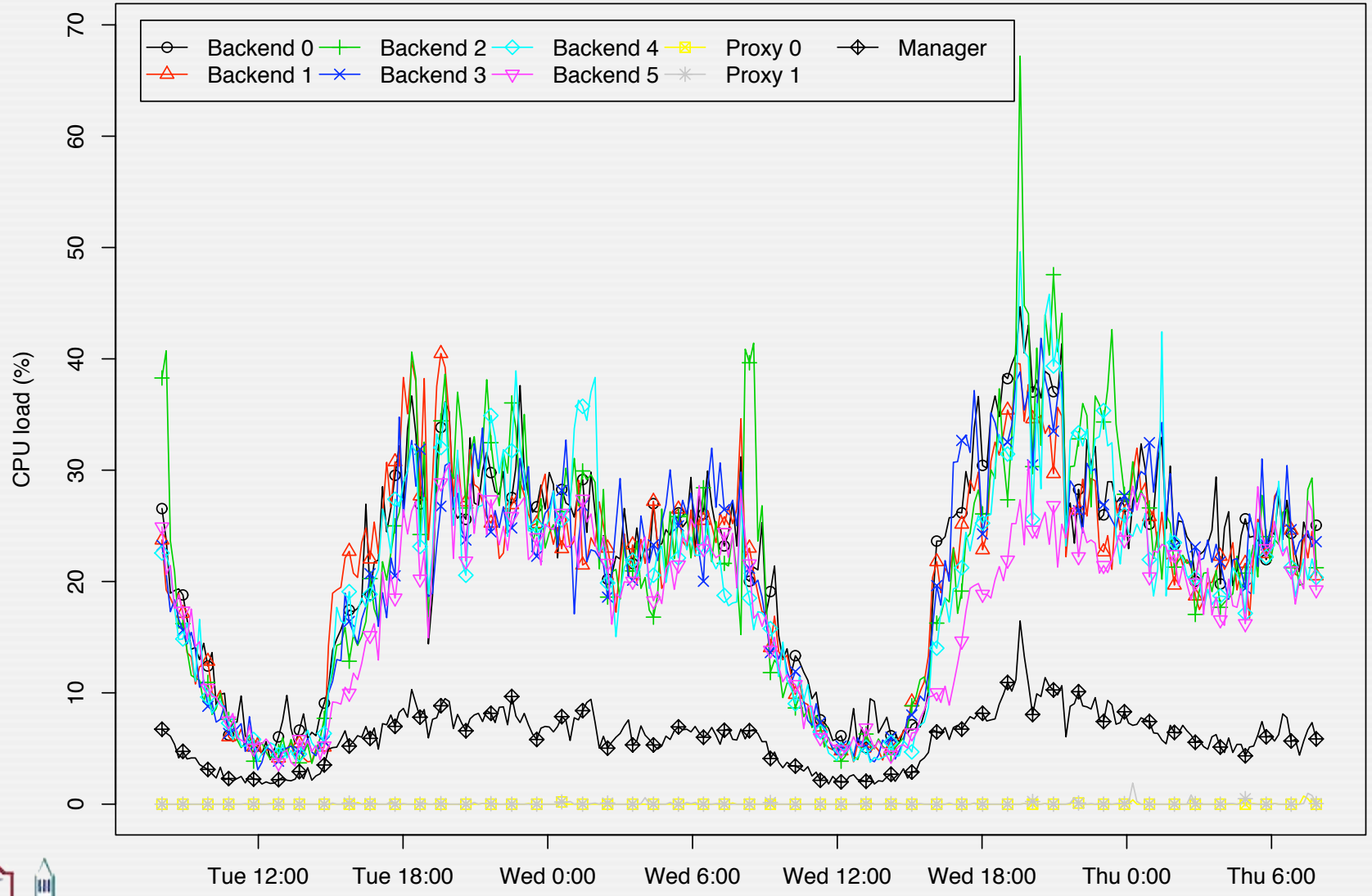
CPU Load per Node



Scaling of CPU



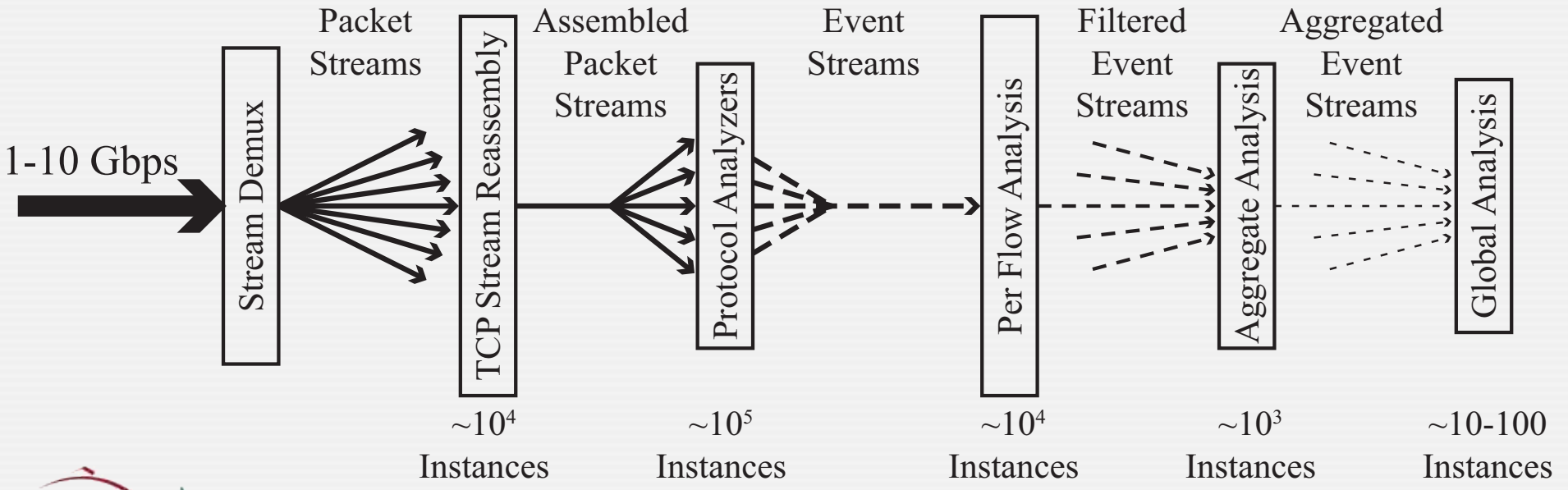
Load on Berkeley Campus



Parallelizing Analysis

Potential

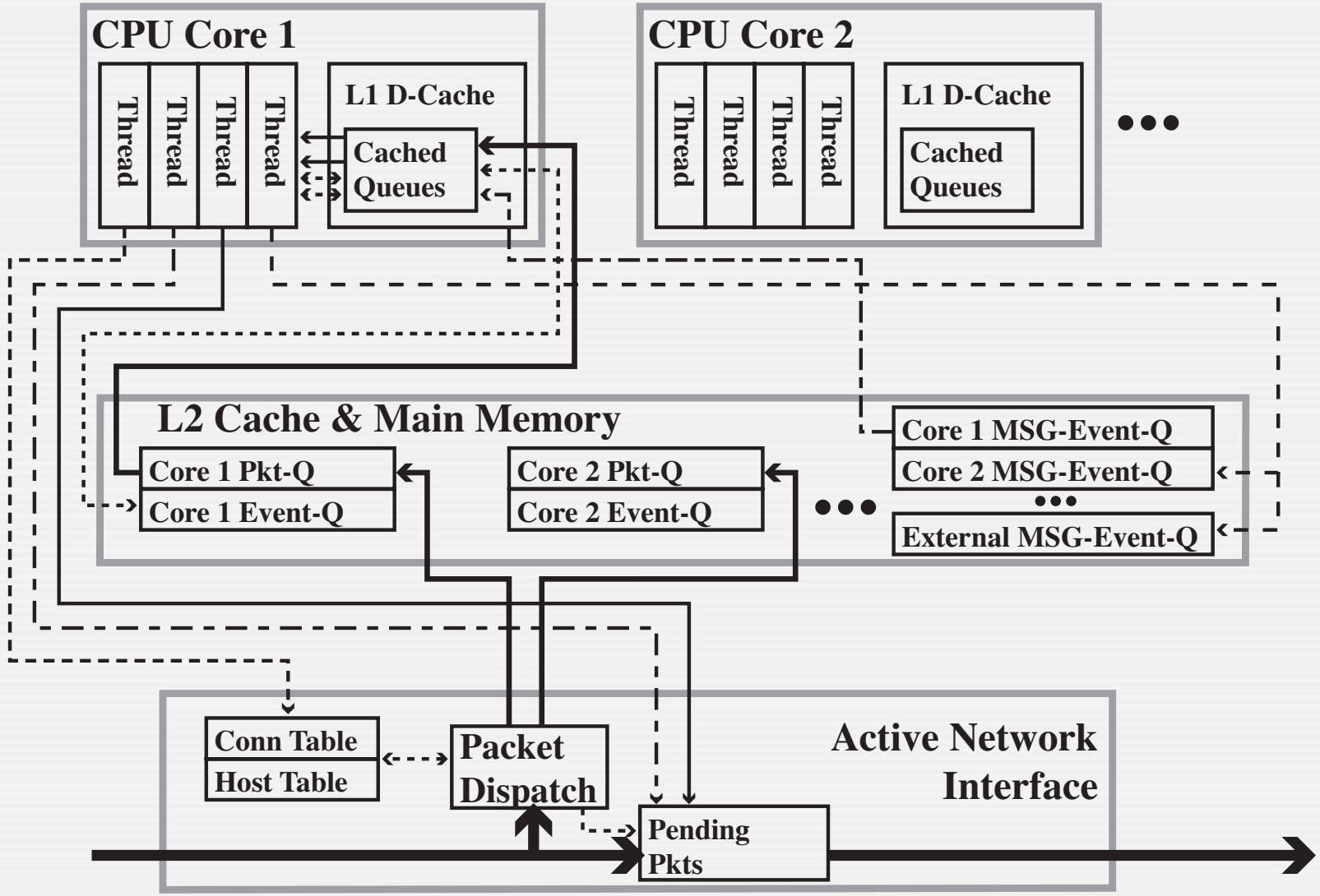
- Observation
  - Much of the processing of a typical NIDS instance can be done in parallel
  - However, existing systems do not exploit the potential
- Example: Bro NIDS
  - Assume Gbps network with 10,000 concurrent connections



Commodity Hardware

- Multi-thread/multi-core CPU provide necessary power
  - Inexpensive commodity hardware
  - *Aggregated* throughput does in fact still follow Moore's law
- Need to structure applications in highly parallel fashion
  - Do not get the performance gain out of the box
  - Need to tructure processing into separate low-level threads
- Need to address
  - Intrusion *prevention* functionality
  - Exchange of state between threads for global analysis
  - Yet minimize inter-thread communication
  - Factor in memory locality (within one core / across several cores)
  - Provide performance *debugging* tools

Proposed Architecture

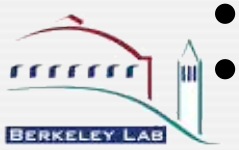


Active Network Interface

- Only non-commodity components currently
  - Prototype to be based on NetFPGA platform ($2000)
  - Commodity hardware might actually be suitable later
    (E.g., Sun's Niagara 2 has 8 CPU cores plus 2 directly attached 10GE controller!)
- Thread-aware Routing
  - ANI copies packet directly into thread's memory (cache)
  - ANI keeps per-flow table of routing decisions
  - Dispatcher thread takes initial routing decision per flow
- Selective packet forwarding
  - ANI holds packets until it gets the clearance (might use caching per e.g. flow/ip)
- Normalization

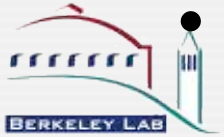
Parallelized Network Analysis

- Architecturally-aware Threading
  - Need to identify the right granularity for threads
  - Protocol analysis consists of fixed blocks of functionality
  - Event processing needs to preserve *temporal* order
    → Multiple independent event queues (e.g., one per core)
- Scalable Inter-thread Communication
  - Can use shared memory
  - Need to consider nonuniformities in system's cache hierarchy
  - Potentially restructure detection algorithms to minimize communication (e.g., loosing semantics via probabilistic algorithms)
- Prevention Functionality
  - Only forward packet once all events are processed
- Evaluation, profiling & debugging
  - Race conditions & memory access patterns
  - Trace-based reproducability



Going Further: Custom Hardware

- Goal: *custom platform for highly parallel, stateful network analysis*
- Custom hardware (e.g., FPGAs) is ideal for parallel tasks
- Expose the parallelism and map it to hardware
- We can identify three types of functionality in Bro
  - Fixed function blocks →Handcraft (e.g., robust reassembly)
  - Protocol analyzers → Use BinPAC with new backend
  - Policy scripts → Compile into parallell computation model
- Envision using MIT's *Transactor* model
  - Many small self-contained units communicating via message queues
- Ambitious but highly promising
  - Generic network analysis beyond network intrusion detection



Thanks for your attention.

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.