Netbed: An Integrated Experimental Environment

Brian White, Shashi Guruprasad, Mac Newbold, Jay Lepreau
Leigh Stoller, Robert Ricci, Chad Barb, Mike Hibler, Abhijeet Joglekar

School of Computing, University of Utah

`www.cs.utah.edu/flux` `www.emulab.net` `www.netbed.org`

The diverse requirements of network and distributed systems research are not well met by any single experimental environment. Competing approaches remain popular because each offers different levels of *ease of use*, *control*, and *realism*. These include packet-level discrete event simulation, live network experimentation, and emulation, which subjects real hardware, protocols, and workloads to a synthetic network environment.

Netbed offers a new alternative. It is software that, when deployed on local and wide-area machines, provides a platform for research, education, or development in distributed systems and networks. Netbed's primary contribution is the seamless *integration* of the above disparate techniques into a common architectural framework that preserves the control and ease of use of simulation, without sacrificing the realism of emulation and live network experimentation. This framework provides common abstractions, namespaces, services, and user interfaces to all three environments. Netbed's integration allows tools such as topology and traffic generators that were originally targeted for one domain to apply to all.

Netbed leverages and extends its ancestor, *Emulab*, which focused on making *emulation* as easy to use and control as simulation. Our results show that Emulab speeds up experiment configuration on a cluster by two orders of magnitude, and through time- and space-sharing, improves cluster utilization by another two orders of magnitude.

Architecture: Netbed revolves around interacting state machines, monitored by a state management daemon. The central state machine is the "experiment." Subsidiary state machines include those handling node allocation, configuration, and disk reloading. This approach copes well with the reliability challenges of large-scale distributed systems, composed of often unstable commodity hardware. The state daemon catches illegal or tardy state transitions. For example, if a node hangs while rebooting, the state daemon times out and attempts an alternate reboot mechanism.

An experiment is generated from a user's *ns* script or via a GUI, resulting in hard state in Netbed's relational database. It represents a network configuration, such as switch VLAN mappings or IP tunnels, and node configurations, including OS images. A node's local disk is currently considered soft state; this allows an experiment to be "swapped out" and later regenerated from the database onto other hardware. The database provides a single namespace and abstractions for heterogeneous resources. The resulting resource **transparency** enables simulated, emulated, live, or hybrid deployments to be similarly realized.

Netbed's **efficient** mapping of a virtual topology to physical resources enables an interactive style of exploration. With the aid of abstraction techniques, our simulated annealing algorithm overcomes the NP-complete local mapping problem within seconds. If users request wide-area paths with specific characteristics (such as latencies), a genetic algorithm finds paths that best match the data provided by periodic path monitoring.

Netbed's **virtualization** allows links to be treated independently of their physical realization. VLANs control the topology between emulated links, while interposed nodes emulate link properties. Netbed obtains wide-area nodes by managing MIT's "RON" nodes; isolation and control of wide-area links is optionally provided by the automated establishment of IP tunnels and routing. Simulated resources are integrated through *nse*, a version of *ns* that interacts with real traffic. At runtime, local and simulated resources can be controlled via a distributed event system; wide-area control is not yet implemented.

Node management services load disk images, boot, configure, and monitor nodes. These services include a highly tuned diskloader which exploits reliable multicast and domain-specific compression to load 3GB images onto hundreds of local nodes in 100 seconds. Wide-area node support provides secure, robust auto-configuration and image update. Such **automated** configuration replaces hours of manual labor with 5 minutes of hands-off automation.

An administrative subsystem supports a hierarchical authorization model that minimizes staff time and maps to natural organizational structure. Netbed's authentication subsystem automatically distributes `ssh` keys. Project-specific NFS or SFS trees are exported appropriately.

Conclusion: Netbed is uniquely valuable for studying distributed systems and networks. Additionally, the concepts and software apply to many other areas, including: *Reliability and fault-injection studies*: an ideal platform. *Storage systems*: Netbed provides configurable network-attached "smart disks." *Cluster management*: Netbed flexibly partitions a cluster, providing "virtual clusters" that traditional cluster software can then manage. This provides unique features such as complete isolation and choice of OS. *Analysis and debugging of distributed systems*: since Netbed can provide a closed distributed system, it can be extended to provide a low-level distributed virtual machine, allowing the system under study to be checkpointed, traced, and rolled back and forward in time. *Security*: leveraging its closure and isolation, Netbed could quarantine cyberattacks too dangerous to study in the real world.

Sponsors: NSF ANI-0082493, Cisco, DARPA F30602-99-1-0503. www.cs.utah.edu/flux/papers/netbedtr02.pdf, netbed@flux.cs.utah.edu.