Protocol Analysis and Evaluation

- Simulations allow for
  - large-scale and controlled evaluation
  - easy experiment management
  - but: hard to achieve realistic environments

- Real-world testbeds allow for
  - realistic environments, cross traffic, routing, …
  - provide realistic results
  - but: complex experiment management

- Focus of this talk
  - Experiment management for large-scale real-world distributed testbeds
Real-world testbeds

but: complex experiment management

• How we define experiment management
  1. Preparation – selection of machines, deployment of experiment package
  2. Running experiment, collecting runtime status
  3. Collecting of experiment results

• Why is experiment management complex?
  • Example 1: which machines are selected, how does load on the machine-set evolve?
  • Example 2: how are experiments deployed efficiently, status/results collected efficiently?
  • Example 3: very heavy load on control machine for deployment, status and result collection
    e.g. 500 parallel ssh sessions, DDoS when status is sent back

Problems in Detail

Which machines are selected, how does load on the machine-set evolve?

• Key challenges
  • select most appropriate nodes at experiment start
  • prevent running experiments from being degraded in quality

• Solution strategies
  • maintain pool of machines + quality estimation for selection at runtime → focus of this presentation
  • employ mechanism to replace degraded machines with more appropriate
General Selection Strategies

How does a researcher* select machines?

- **Random**
  - naive selection of nodes
  - don’t care for anything

- **TopNodes**
  - select ‘best’ nodes at experiment start (t₁)

- **Adaptive**
  - initially like TopNodes
  - maintain at runtime
  - replace degraded nodes

*the person intending to run an experiment in a testbed

Distributed Experiment Management for Large-scale Testbeds

Distributed Approach

- Distributed experiment management
  - define metric \( A_E(m,t) \) specific to experiment \( E \)
    - applicability of machine \( m \) for experiment \( E \) at time \( t \)
    - e.g. \( A_E(m,t) = \text{cpu-load}_t^{-1} + \text{free-mem}_t \)
  - all testbed machines form an overlay-based tree \( T \)
    that stays sorted at runtime with respect to \( A \)

Machine selection
- counter-based

Experiment deployment
- ALM-based distribution

Result/status collection
- concast-based aggregation

* e.g. Reliability-oriented Switching Trees, Tan and Jarvis, IEEE Transactions on Parallel and Distributed Systems, 2007
Adaptive Machine Selection

- **Initial selection of machines**
  - require 6 machines for experiment
  - send experiment package with counter to gateway

- **Runtime Adaptation**
  - machines report metric to parents
  - switch roles, exchange experiment package, and status context
  - prefer unused machines
  - unaggressive switching, prevent thrashing

Simulation Setup

- **Simulations based on OMNeT++**
  - simulation environment
    - simulation-time duration of 100h
    - 100 machines, 50 overall experiments
  - experiment environment
    - lifetime distributed uniformly between 10min and 10h
    - deadtime distributed uniformly between 10min and 10h
      - ~25 experiments alive at every time
    - experiment selects set of machines at each start
      - uniform between 10 and 90 machines
  - metric environment
    - $load_t(m)$ – load of machine $m$ defined as number of experiments run on that machine at time $t$
    - $A_{\text{t}}(m,t) = (load_t(m)+1)^{-1}$
Testbed Provider Perspective

Testbed Load Variance

- Lowest load variance with adaptive approach, provides best overall service

Difference TopNodes/ Adaptive: prevention of peaks
Using single Adaptive strategy against all experiments using Random strategies still lowers worst machine load.

Mean of all experiments load variances: smoothness increased.
Conclusion and Outlook

- Distributed Testbed Management
  - easier management, better experiment performance
    - smoother testbed utilization, better service
    - prevent heavily loaded machines
    - more efficient experiment deployment through ALM
    - more efficient status and result collection through concat
  - more realistic experiments

  Testbeds do provide realistic environments, but a machine not responding to a ping for 20s is not realistic, but rather a PlanetLab-effect

- Outlook
  - still very simple view – only partially reflects reality
  - need data on experiment duration, number of experiments
  - heterogeneous machine resources, different metrics
  - cost of moving experiment from one machine to another