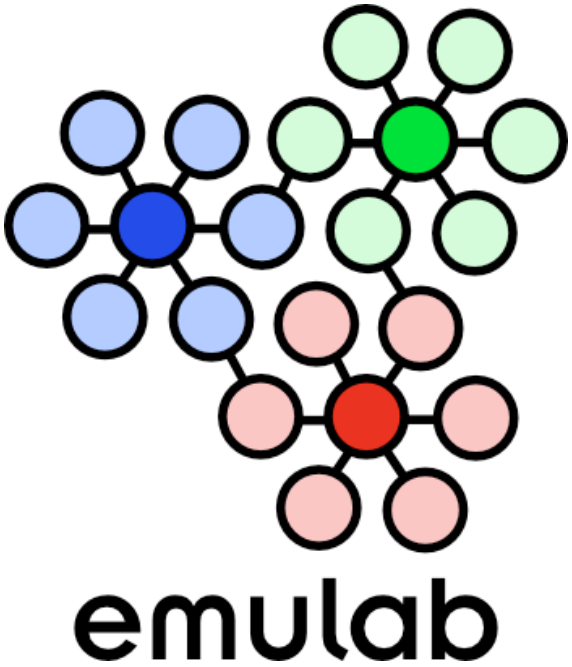
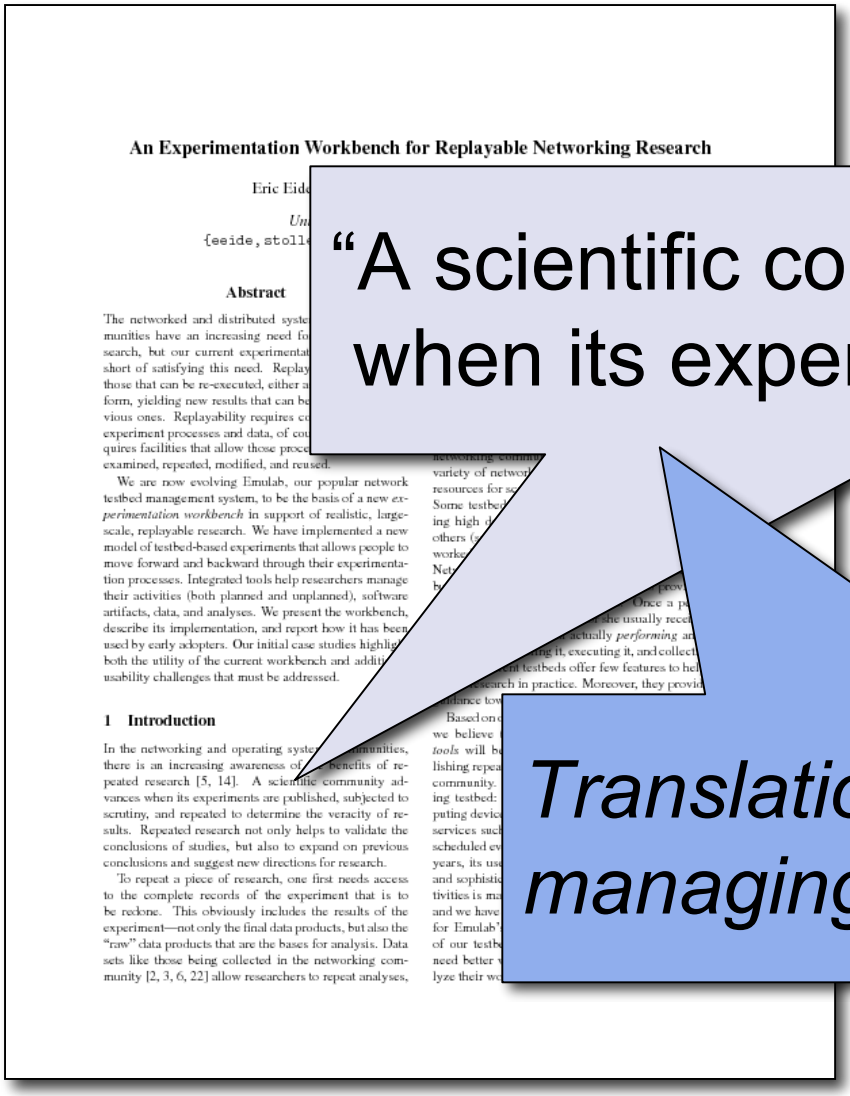
An Experimentation Workbench for Replayable Networking Research



Eric Eide, Leigh Stoller, and Jay Lepreau

University of Utah,
School of Computing
NSDI 2007 / April 12, 2007

Repeated Research



An Experimentation Workbench for Replayable Networking Research

Eric Eide

Un

{eeide, stoll

Abstract

The networked and distributed syste
munities have an increasing need fo
search, but our current experimenta
short of satisfying this need. Replay
those that can be re-executed, either a
form, yielding new results that can be
vious ones. Replayability requires co
experiment processes and data, of cou
quires facilities that allow those proce
examined, repeated, modified, and reused.

We are now evolving Emulab, our popular network testbed management system, to be the basis of a new *experimentation workbench* in support of realistic, large-scale, replayable research. We have implemented a new model of testbed-based experiments that allows people to move forward and backward through their experimentation processes. Integrated tools help researchers manage their activities (both planned and unplanned), software artifacts, data, and analyses. We present the workbench, describe its implementation, and report how it has been used by early adopters. Our initial case studies highlig both the utility of the current workbench and additi usability challenges that must be addressed.

1 Introduction

In the networking and operating syste munities, there is an increasing awareness of benefits of repeated research [5, 14]. A scientific community advances when its experiments are published, subjected to scrutiny, and repeated to determine the veracity of results. Repeated research not only helps to validate the conclusions of studies, but also to expand on previous conclusions and suggest new directions for research.

To repeat a piece of research, one first needs access to the complete records of the experiment that is to be redone. This obviously includes the results of the experiment—not only the final data products, but also the "raw" data products that are the bases for analysis. Data sets like those being collected in the networking community [2, 3, 6, 22] allow researchers to repeat analyses,

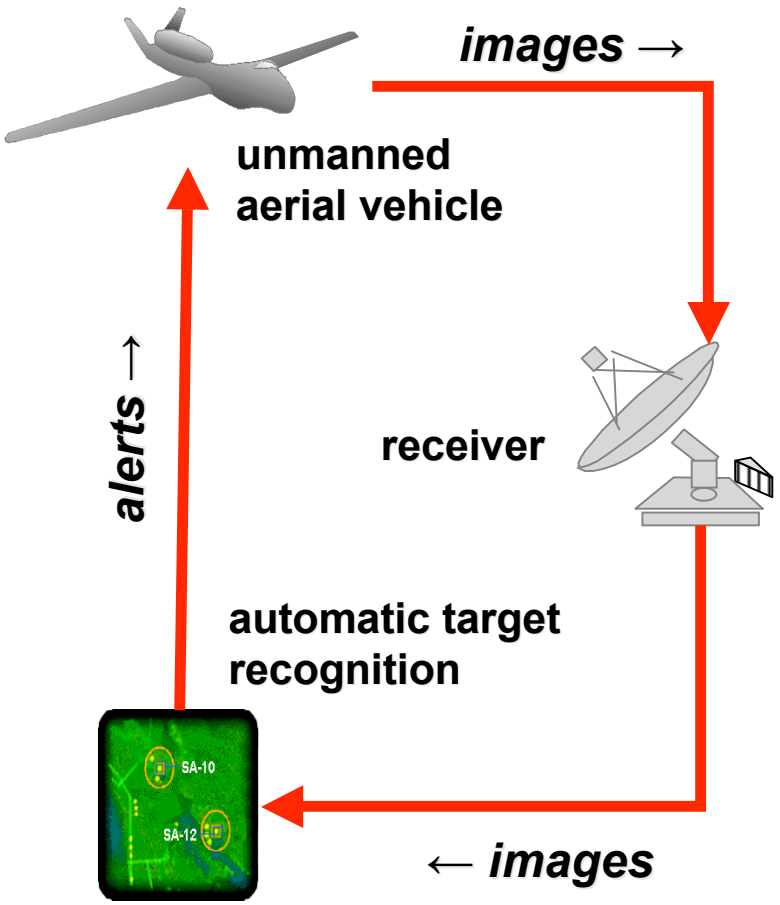
"A scientific community advances when its experiments are repeated…"

Translation: "I have trouble managing my own experiments."

Example From My Past

- a distributed, real-time application
- evaluate improvements to real-time middleware
  - *vs. CPU load*
  - *vs. network load*

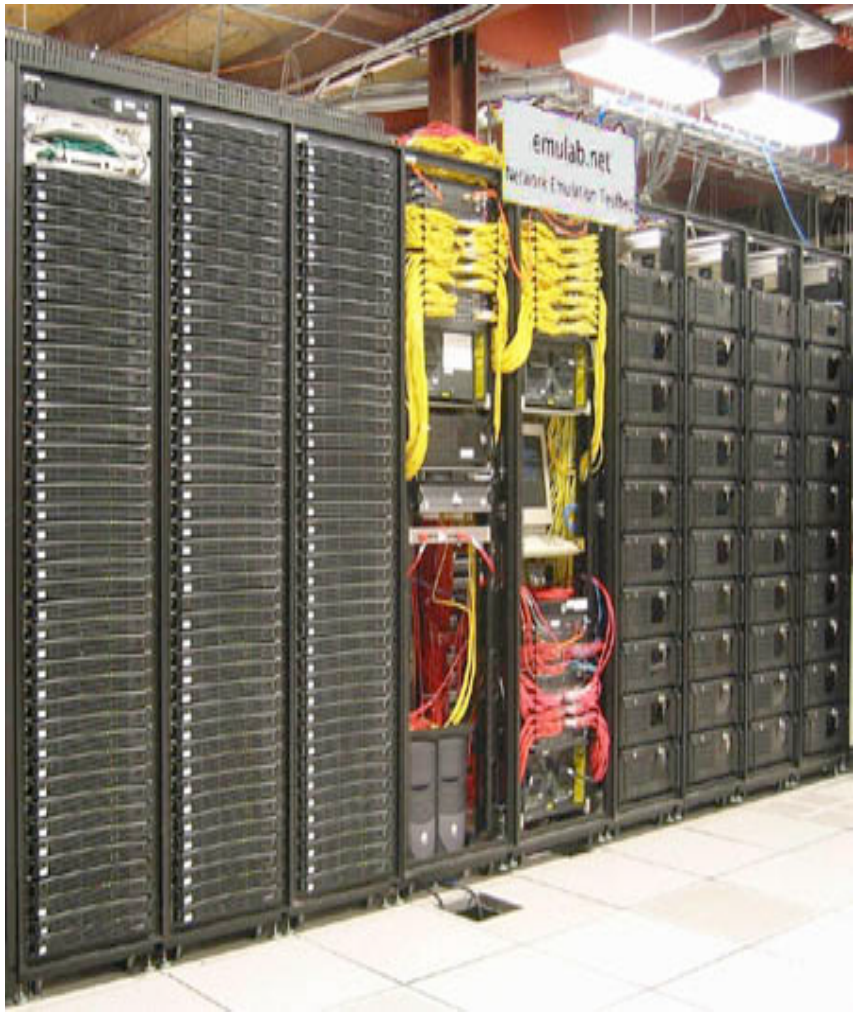
- ***4 research groups***
- ***x 19 experiments***
- ***x 56 metrics***
- ***use Emulab***



A Laboratory Is Not Enough

- testbeds give you lots of resources…
- …but offer little help in *using* those resources

  - *package / distribute / configure / instrument / init / execute / monitor / stop / collect / analyze / archive / revise / repeat*



What's Missing: Workflow

- current network testbeds
  - *…manage the "laboratory"*
  - *…not the experimentation process*
  - *i.e., scientific workflow*
- ➔ a big problem for large-scale activities

Need

- my experiment needs…
  - *encapsulation*
  - *automation*
  - *instrumentation*
  - *preservation*

package / distribute / configure / instrument / init / execute / monitor / stop / collect / analyze / archive / revise / repeat

- benefits
  - *verify previous results*
  - *establish base for new research*
  - *my own, or someone else's*

repeatable research

Opportunity

- get the lab manager to help us out!
  - ***integrated*** *support for experimental procedures*
  - ***resources + encapsulation + automation***
  - ***framework:*** *rapid start & common basis*

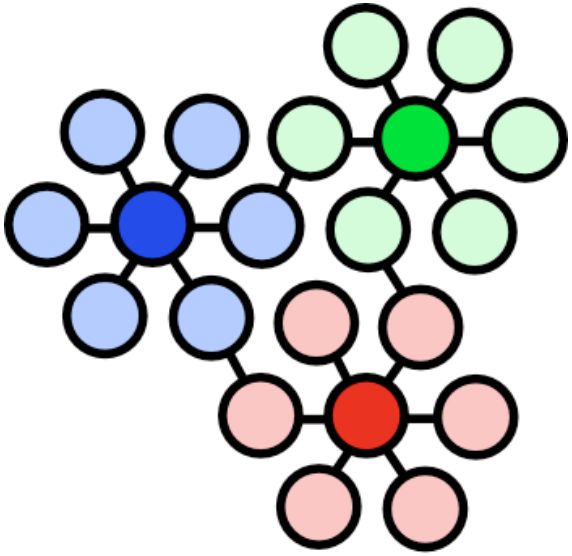
  - *manage scientific workflow, but also manage lab*

Experimentation Workbench

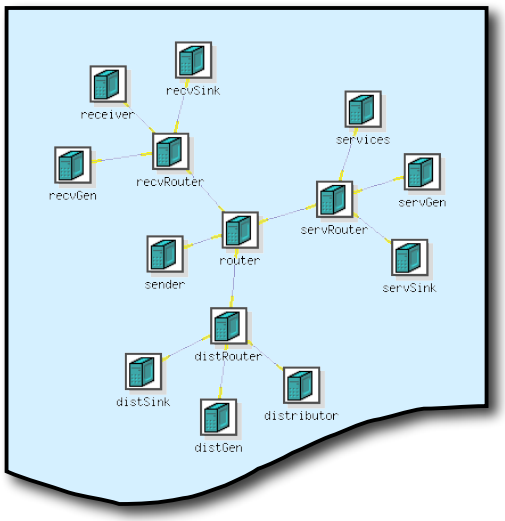
- an environment for “replayable research”
  - *experiment management + experiment execution*
  - *(but really: help me manage my work)*
  - *all Emulab-managed devices, incl. PlanetLab slivers, …*

- initial design, implementation, and evaluation
  - *new model of testbed-based experiments*
  - *prototype implementation*
  - *case studies*
  - *lessons learned*

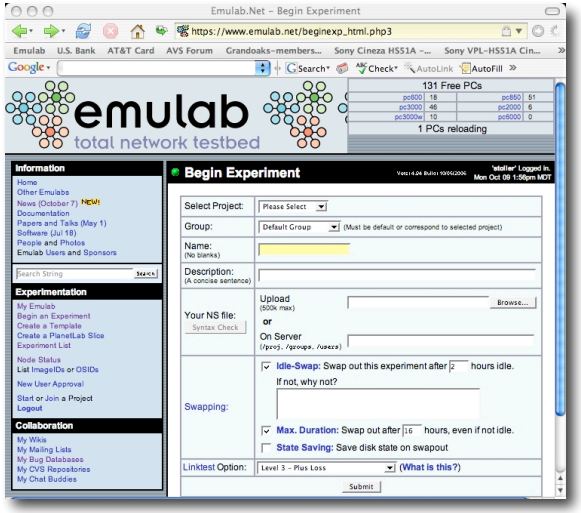
Workbench

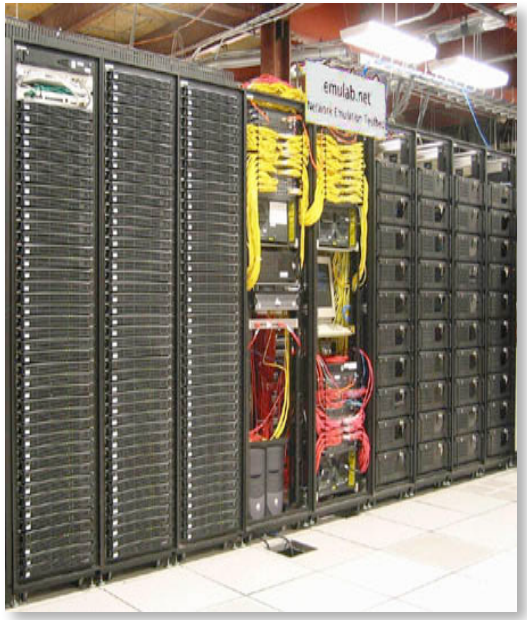


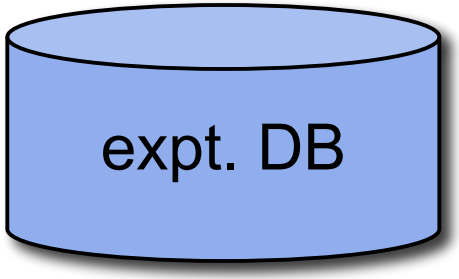
Classic “Experiments”



topology +
SW (by reference) +
events







Problems

- definition versus instance
- related experiments
- multiple trials per session
- data management
  - *instrumentation, collection, archiving, analyzing*
- ecosystem
  - *topology, software, config, input data, …*
- evolution over time

New Model

- template
- instance
- run
- activity
- record





- *divide and conquer*
- *separate the roles that an experiment plays*
- *evolve the new abstractions*
- *build on what testbed users already know and do*

Template

- *template*

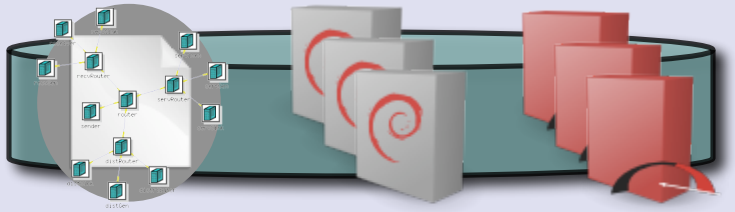
- instance

- run

- activity

- record

- *a "repository"*

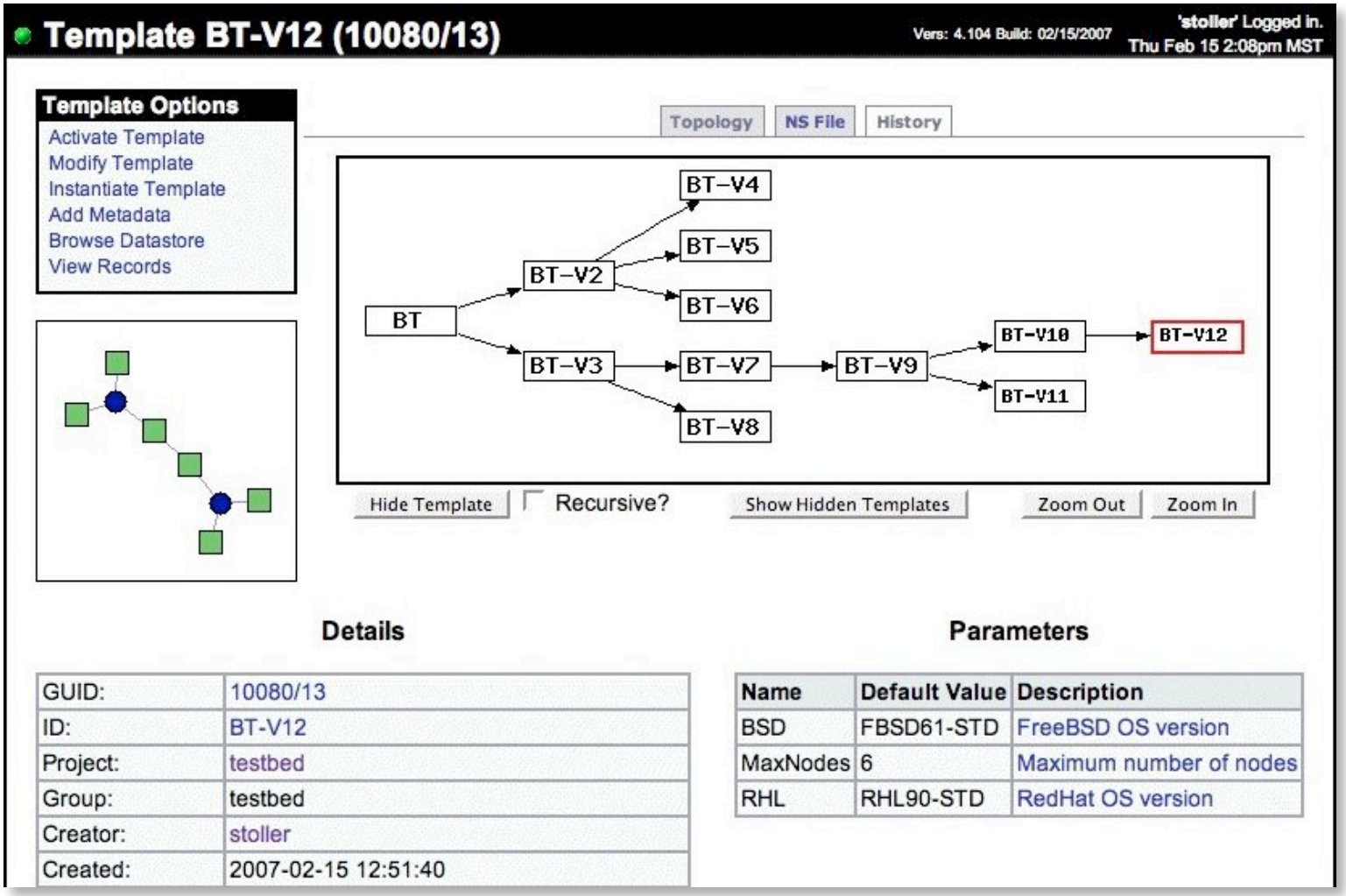


- *definition role of the classic "experiment" notion*
- resources by value
  - *files, scripts, DB tables, disk images, …*
- resources by reference
- prototype: implemented with Subversion *(user-hidden)*

Templates vs. Experiments

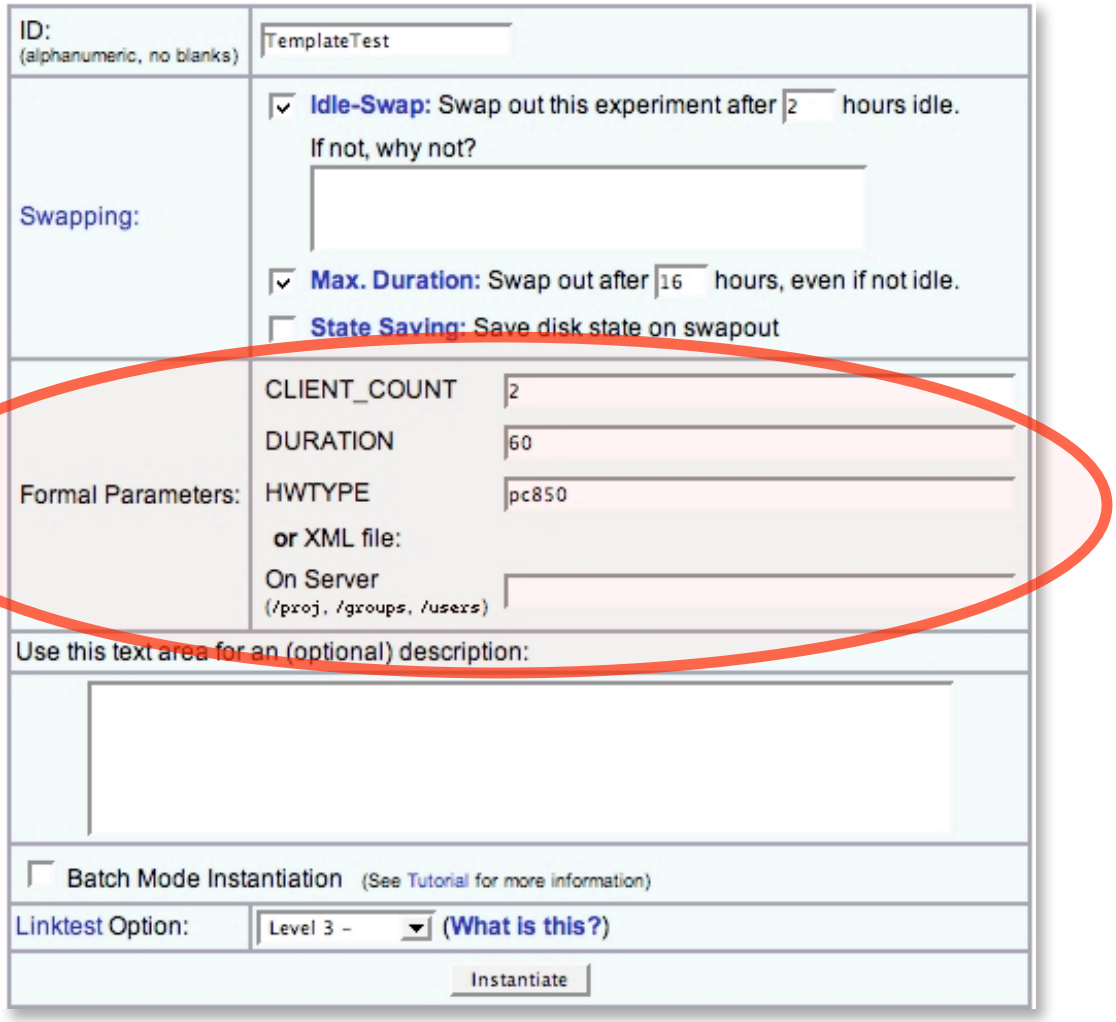
- a template is like a classic Emulab experiment, but a template has…
  - *datastore (file repository)*
  - *parameters*
  - *multiple instances*
  - *metadata*
  - *history*

Template History



Instantiating a Template

parameters



template instances can also be created programmatically

Template Instance

- template

- *instance*

- run

- activity

- record

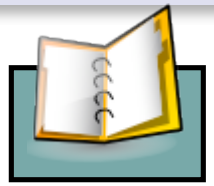
- *a container of testbed resources*
- *resource-owner role of classic "experiment" notion*
- a transitory object
  - *created and destroyed by users and activities*
- nodes & network
  - *files from the datastore*
- database for user

Run & Activity

- template
- instance
- *run*
- *activity*
- record



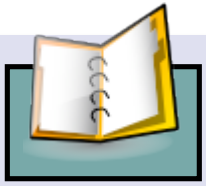




- *run: a container of a user-defined "unit of work"*
  - *defines a context*
  - *a "trial"*
  - *one / serial / parallel*
- *activity: a process, script, workflow, … within a run*
  - *events & data come from activities in a run*
- runs and activities can be scripted or interactive
- prototype: implemented via agents & events

Record

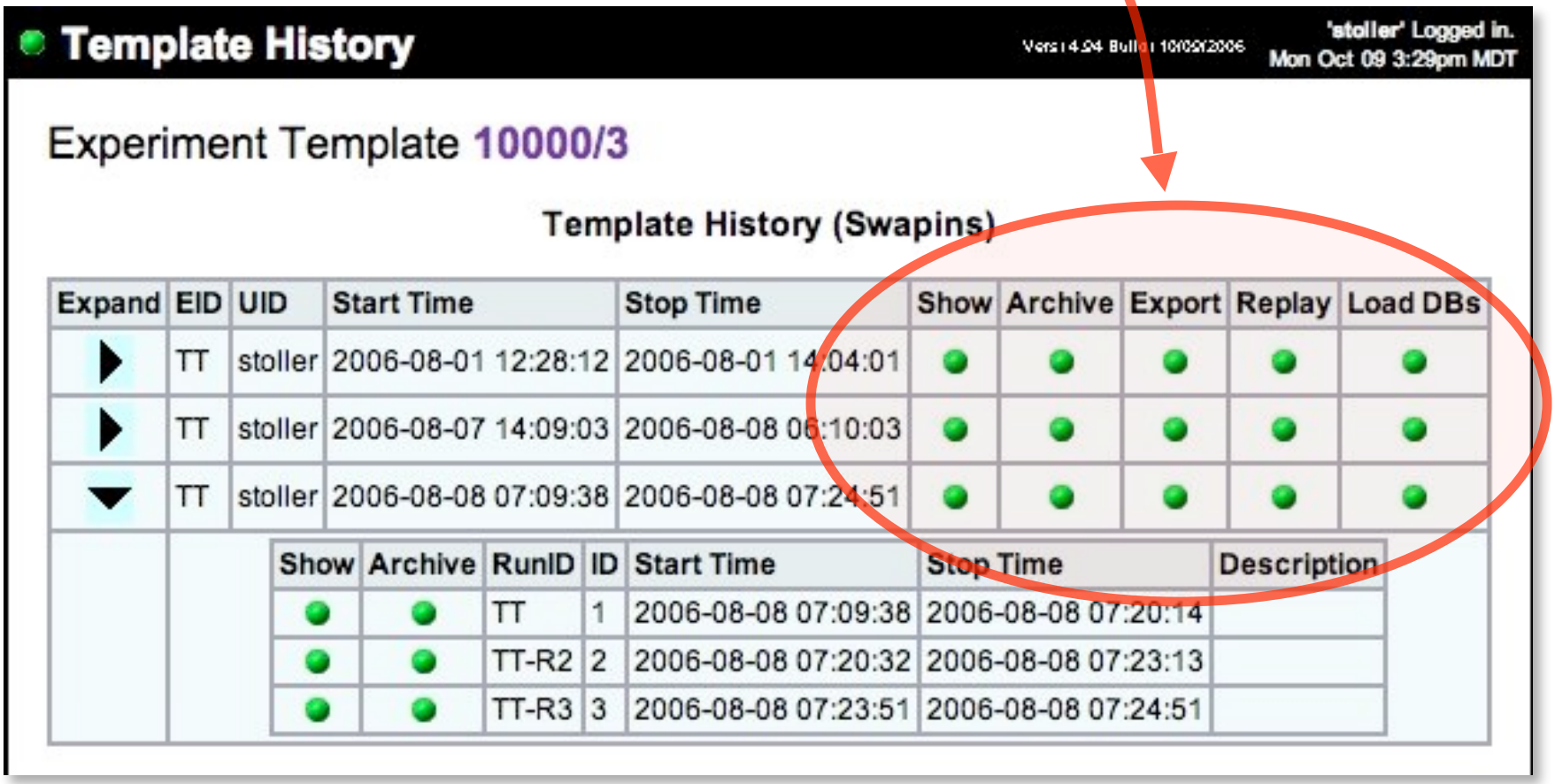
- template
- instance
- run
- activity
- *record*



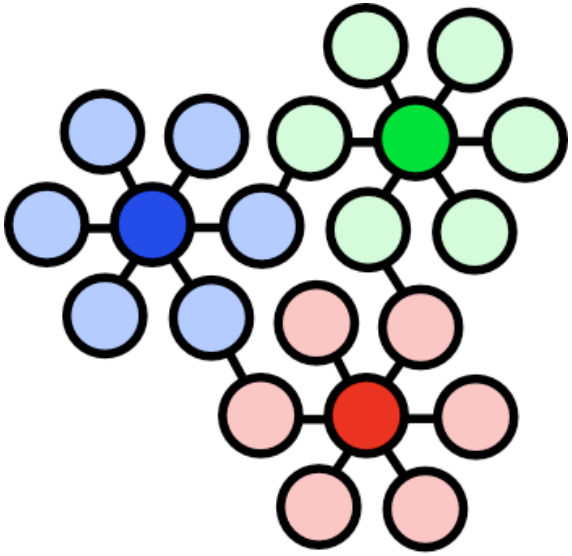
- *the "flight recorder" of a run*
  - *parameter values*
  - *input & output files, DBs*
  - *raw data & derived data*
  - *template's by-reference resources*
  - *dynamically recorded events*

Record Repository

view, export, and replay



Evaluation and Lessons Learned



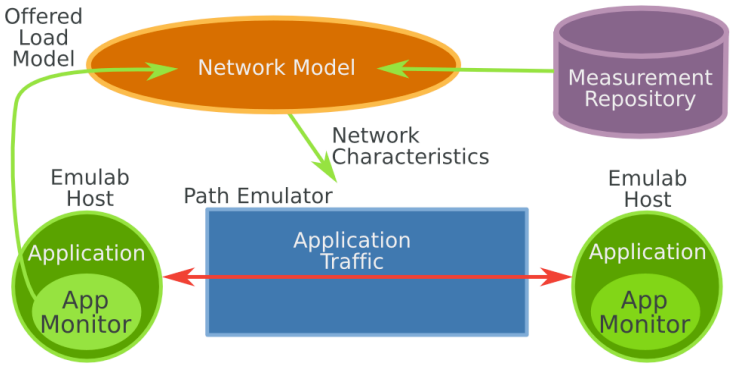
Evaluation

- how to evaluate?
  - *new capabilities ➔ user studies*

- **goal:** early feedback about design & impl.
- **approach:** three case studies
- **outcome:** specific & general lessons learned

Study 1: Flexlab Development

- replace ad hoc experiment management



- originally:
  - *a configurable ns file*
  - *start/stop trial scripts*
  - *"scaffold" in CVS*
  - *manual archiving*
  - *destructive modification*
- now:
  - *templates & params*
  - *runs, start/stop hooks*
  - *scaffold & results in WB*
  - *automatic archiving*
  - *preserved history*

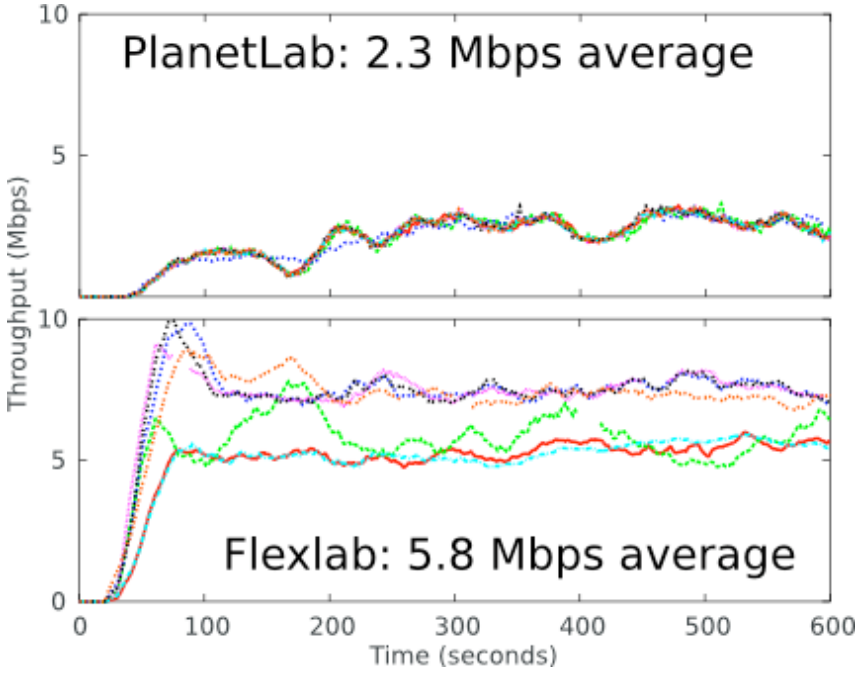
Conclusion: the new model "fits" developers' model

Study 2: Flexlab Use

- study BitTorrent on Flexlab and PlanetLab

- outcome:
  - *parameterization*
  - *utilized per-run database*
  - *team communication*
  - *results for publication*

  - *stress point: latency*
  - *stress point: node failure*



Lessons: Storage

- initial philosophy: “store everything”
  - *templates + results + history + metadata + …*
- space efficiency + group commits
  - ➔ *Subversion*
- cognitive overload
  - ➔ *careful UI design*

Space and Time

	Record size (MB)	Stored in repo. (MB)	Elapsed time (m)
BitTorrent	31.0	18.9	7.0
GHETE	70.6	19.8	14.4

- **solution:** pipeline record-making with user activities
- **new problem:** isolation
- **new approach:** branching file systems

What Users Want

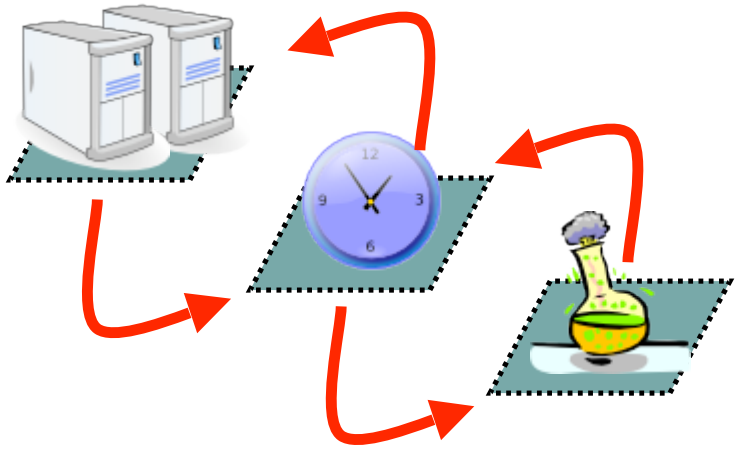
- deletion!
  - *not a space concern*
  - *cognitive clutter — "junk"*
  - *privacy — "mistakes"*
- a range of options is required
- "true deletion" is a new requirement

Lessons: The Model

- initial philosophy: “divide and conquer”
  - *more kinds of entities*
  - *describe notions and relationships*

- experience:
  - *new model does map to users' abstractions*
  - *captures separations and connections...*
  - *...but not “life cycle” concerns*

“Life Cycle” Concerns



- multiple levels of abstraction
  - *instance: “the lab”*
  - *run & activity: “the work”*
- intertwined & concurrent
  - *workbench must manage experiments and the lab*
  - *a key difference with “ordinary” scientific workflow systems*
- approach: further refine and enhance our model
  - *e.g., adopt features of Plush [Albrecht et al., OSR 40(1)] or SmartFrog [Sabharwal, ICNS ‘06]*

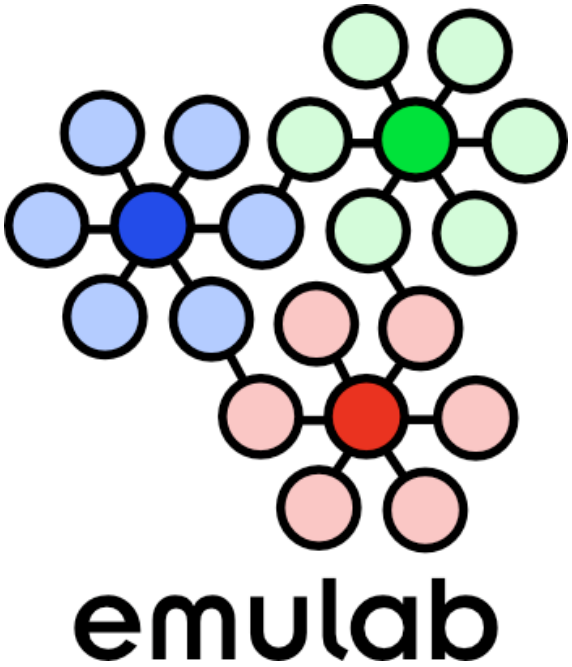
Summary

- **goal: better research ← better process tools**
- **experiment management + experiment execution**
- **prototype** builds on existing testbed infrastructure
  - *model maps pretty well to user notions*
- **experience:** strong integration is required
  - *…for overlapping activities safely*
  - *…for lab management + experiment management*
  - *…for making it user-friendly in practice*

Status

- “alpha test” stage
- internal users
- select external users…
  - *mail to testbed-ops@emulab.net*

http://www.emulab.net/



Thank you!
Questions?