Toward Orchestration of Complex Networking Experiments

Alefiya Hussain, Prateek Jaipuria, Geoff Lawler, Stephen Schwab, Terry Benzel

Long Experience Paper
What is an Networking Experiment?

- Create meso-scale representations of the internet
- Understand how the system behaves
What is an Networking Experiment?

- Create meso-scale representations of the internet
- Understand how the system behaves
What makes experiments complex?
Complexity in networking experiments

The system is mapped to different configuration
Complexity in networking experiments

Each configuration is overlapped with rich set of application mixes
Complexity in networking experiments

The configuration is converted to execution
Experiment Orchestration

Definition: Sequence of steps required to execute the representative scenarios on the testbed
Related Work: Tools and Testbeds

Shell or Ssh-based Scripts:
+ most popular
- limited feedback and error handling

Ansible and other configuration management Tools:
+ rich toolkit
- limited expressibility

Emulab: first emulation testbed
* Tevc
* Experimenters workbench

PlanetLab: first globally distributed testbed
* Plush

GENI: Federated collection of testbeds
* ansible
* Labwiki
* ODEL

DETER: first cyber security testbed
* SEER
* MAGI

Emerging Testbeds:
* Fabric
* Chameleon
* EdgeNet
* MergeTB
Experiment Orchestration in MAGI

- **Design:** agents for wide range of scenarios
- **Execute:** orchestrator and daemons
MAGI: Montage AGent Infrastructure

Conceptual: Sequence of Steps

Specification: agent activation language

Execution: Orchestrator and node daemons, agents

Design: agents for wide range of scenarios

Execute: orchestrator and daemons
MAGI Specification

**Group**: mapping of behavior roles to physical and virtual machines

**Agent**: implementation of the behavior roles

**Event**: a method that can be invoked in the agent

**Eventstreams**: ordered collection of events that formulate the experiment behaviors

**Triggers**: time- or condition based synchronization points
**Specification**

**Group**: mapping of behavior roles to physical and virtual machines

**Agent**: implementation of the behavior roles

**Event**: a method that can be invoked in the agent

**Eventstreams**: ordered collection of events that formulate the experiment behaviors

**Triggers**: time- or condition based synchronization points
Specification

**Group**: mapping of behavior roles to physical and virtual machines

**Agent**: implementation of the behavior roles

**Event**: a method that can be invoked in the agent

**Eventstreams**: ordered collection of events that formulate the experiment behaviors

**Triggers**: time- or condition based synchronization points
Specification

**Group**: mapping of behavior roles to physical and virtual machines

**Agent**: implementation of the behavior roles

**Event**: a method that can be invoked in the agent

**Eventstreams**: ordered collection of events that formulate the experiment behaviors

**Triggers**: time- or condition based synchronization points
Specification

Group: mapping of behavior roles to physical and virtual machines

Agent: implementation of the behavior roles

Event: a method that can be invoked in the agent

Eventstreams: ordered collection of events that formulate the experiment behaviors

Triggers: time- or condition based synchronization points
MAGI: Montage AGent Infrastructure

**Design:** agents for wide range of scenarios

**Conceptual:** Sequence of Steps

**Specification:** agent activation language

**Execution:** Orchestrator and node daemons, agents
Orchestration

**Parser:** Reads specification

**Scheduler:** handles each eventstream concurrently, sends events to node daemons.

**Evaluator:** receives return values from the node daemons to satisfy triggers
Daemons and Agents

**Daemons**: lightweight control conduit
Received events to launches and controls agents
Returns values from agents to orchestrator for trigger evaluation

**Agent Modules**: implementations on nodes in Python
Case Studies:

**Education**
- Development and assessment of multi-user text-based chat client and server system
- 40-75 students for undergraduate class, Introduction to Computer Networks;
  - Student client with instructor server
  - Random client with student server
  - Upto 30 clients with student server

**Feedback Loops**
- Different teams interact in an experiment; while limiting access to parts of the scenario
- 2000 webclients, 1000 control clients, 50 apache2 servers in webfarm
  - **Sense** traffic on network
  - **Compute** devise control action to increase, decrease or maintain traffic
  - **Actuate** executes control action
Case Studies:

Integrated system development
- Five teams develop adversary-resistant communication to circumvent censorship in Tor
- Configure, deploy, manage Tor and technologies
  - Multi-scale experiments, 10 machines to 100 machines
  - Tor agents to start relays, bridges, and clients
  - Large scale- 5120 client processes, microblogging, VoIP, file sharing apps

Cyber physical systems
- Distributed optimization control algorithms for monitoring power flow oscillations in presence of DDoS attacks
- IEEE 39 bus power system overlaid on a 18 node communication topology
- High volume attacks and study impact on damping the oscillations.
Retrospective Takeaways

● Specification is topology agnostic
  ○ allows direct scaling experiments

● Unordered events and with synchronization triggers
  ○ enables exploiting concurrency and asynchronous execution in experiment

● Error handling and logging
  ○ Errors and failures forwarded from nodes to orchestrator
Conclusion

The MAGI tool makes it easier to run large and complex experiments on testbeds by providing a wide range of traffic agents and automating the experiment execution.

MAGI is general
- runs on most testbeds
- open source

Available at https://github.com/deter-project/magi
Documented with examples at https://montage.deterlab.net/magi
Thank you

Contact: Alefiya Hussain

hussain@isi.edu