Case Studies in Experiment Design on a minimega Based Network Emulation Testbed

Brian Kocoloski
USC/ISI

Alefya Hussain
USC/ISI

Matthew Troglia
Sandia National Labs

Calvin Ardi
USC/ISI

Steven Cheng
Sandia National Labs

Dave DeAngelis
USC/ISI

Christopher Symonds
Sandia National Labs

Michael Collins
USC/ISI

Ryan Goodfellow
USC/ISI

Stephen Schwab
USC/ISI

2021 Workshop on Cyber Security Experimentation and Test (CSET ’21)
DARPA Searchlight Program

- Enterprises need better tools to (1) understand network activity, and (2) manage quality of service
- **TA1: Network traffic analysis**
  - What does a network look like? What is the topology, what are the paths, etc.
  - What protocols, and more specifically, what applications are on the network?
  - What are the network’s performance characteristics?
- **TA2: Network resource management**
  - Manage QoS between multiple applications of different priorities
  - Communicates with a TA1 to understand network behavior and state
Test and Evaluation Challenges

- Our objective: systematically evaluate TA1 and TA2 technologies on a variety of representative network conditions and applications

- TA1 evaluation challenges:
  - Evaluate classification accuracy of different applications on the network
    - Many combinations of applications make its job harder
  - Evaluate many different network topologies and network resource configurations
  - Understand impact of VPNs/tunnels on TA1 capabilities

- TA2 evaluation challenges:
  - Deploy high fidelity networks capable of supporting custom layer-2 forwarding applications
    - Virtual Openflow switches, DPDK applications, etc
Virtualization based network emulation tool from Sandia National Laboratories

- Define and launch VM-based networks through Qemu/KVM

- Layer-2 network virtualized with VLAN-based software bridge (OpenvSwitch), with VXL AN tunnels for emulating networks that span physical nodes

- Orchestrate runtime behavior through a custom command-and-control system

https://tip.minimega.org
Our experience with minimega

Features we liked and made use of:

- High resource utilization
  - We were somewhat resource limited on our physical testbed (DeterLAB) at the time
- Zero configuration for DHCP/DNS
- minirouter and its centralized routing interface
  - We used a combination of static and OSPF based routing
- Management of images through Qemu snapshots

Challenges we encountered:

- Topology modeling
- Configuration and deployment of traffic

Our experimental needs required us to run several hundreds of experiments with a mixture of different topologies and application traffic mixes
Topography Modeling

for i in $(seq 0 7); do
  vm config net foo$i
  vm config h$i
done
vm config net foo0,foo1,foo2,foo3,foo8
vm start c0
vm config net foo4,foo5,foo6,foo7,foo9
vm start c2
vm config net foo8,foo9
vm start c1

- Maintaining lists of VLAN tags can get cumbersome, particularly as topologies grow in size and complexity.

- Users of other network emulation testbeds (e.g., Emulab) are often familiar with link-centric specification methods:
  - Specify links with endpoints, instead of nodes with network interface cards.
Topology Modeling

for i in $(seq 0 7); do
  vm config net foo$i
  vm config h$i
done

vm config net foo0,foo1,foo2,foo3,foo8
vm start c0
vm config net foo4,foo5,foo6,foo7,foo9
vm start c2
vm config net foo8,foo9
vm start c1

Alternative Network Model

for i in $(seq 0 3); do
  connect_vms h$i c0
done

for i in $(seq 4 7); do
  connect_vms h$i c2
done

connect_vms c0 c1
connect_vms c2 c1
Minimega switches use source MAC address learning
This causes problems when guest software implements custom layer-2 forwarding mechanisms or performs asymmetric routing
We developed VM-to-VM links that remove MAC learning
More details in the paper
Traffic Modeling

- Minimega has limited traffic generation support through the **protonuke** traffic generator
- Deployment of traffic is done by developing shell scripts which minimega issues to each unique traffic generating end-host
- Challenges:
  - We found ourselves generating redundant shell scripts with significant overlap for things like managing process lifecycles and copying support bundles
  - Lack of traffic realism
- 3 contributions:
  - Development of a set of new traffic applications that extend minimega's native **protonuke** support.
  - Development of a common JSON-based structural abstraction that encodes application configuration
  - Compiler which automates the construction of shell scripts commands
Traffic Modeling: Video Streaming Example

```json
{
  "video-streaming": {
    "h0": [
      {
        "target": "h2",
        "params": {
          "client": {
            "resolution": "720",
            "protocol": "hls"
          },
          "server": {}
        }
      }
    ]
  }
}
```
Traffic Modeling: Video Streaming Example

```json
{
  "video-streaming": {
    "h0": [{
      "target": "h2",
      "params": {
        "client": {
          "resolution": "720",
          "protocol": "hls"
        },
        "protocol": "hls"
      },
      "server": {}
    }]
  }
}
```

```
c/cc exec bash -c "rm -rf /tmp/miniccc/files/miniccc_files/video-streaming/
cc send miniccc_files/protonuke
clear cc filter
c cc filter name=h2
c cc send miniccc_files/video-streaming/www.tar.gz
c cc send miniccc_files/video-streaming/extract-www-tar.sh
c cc exec bash -c "/tmp/miniccc/files/miniccc_files/video-streaming/extract-www-tar.sh"
c cc background /root/www-video/run.sh
clear cc filter
c c filter name=h0
c cc send miniccc_files/video-streaming/client-watch-video.tar.gz
c cc send miniccc_files/video-streaming/extract-client-watch-video.tar.sh
c cc exec bash -c "/tmp/miniccc/files/miniccc_files/video-streaming/extract-client-watch-video.tar.sh"
c cc background sudo -u searchlight /home/searchlight/client-watch-video/run.sh
cc --server h2:8001 --resolution 720 --protocol hls --time 3
```
Traffic Types

- Several types supported natively by protonuke
  - File transfer with many protocol variants (SCP, HTTP(s), FTP(s))
  - IRC
  - Email (SMTP)
  - Web Browsing
- Additional types developed by our team
  - SSH text editing
  - Video Streaming
    - HLS, DASH, and HTML5 based options supported
- Compilation
  - Our scripts compile a single JSON based traffic representation into a set of scripts for each client or server endpoint in the topology
  - Removes the need to develop a large collection of miniccc scripts
Case Study: Distributed Topology Discovery

Objective: evaluate how well the system could measure the topological characteristics

- Connectivity, routing (asymmetry + multipath), link criteria (delay/bandwidth/loss)
Case Study: Real-Time Traffic Classification

Objective: evaluate the system's ability to infer the set of applications on the network

- We developed a large collection of unique subsets of applications and determined classification accuracy
- Over 500 individual experiments using 26 different combinations of applications
- Structured, centralized interface to define traffic was instrumental in automating these experiments
Case Study: Distributed Traffic Engineering

- Objective: evaluate how well the system could achieve a target QoS specification (bandwidth for each application)
- Required VM-to-VM links due to the use of OpenFlow switches in the topology that could generate MAC address migration from the perspective of the minimega OVS switch
Takeaways and Conclusion

- Minimega is a useful, mature tool
  - Zero configuration DHCP/DNS
  - Centralized router configuration
  - VM lifecycles management
  - Image management

- We extended it to make it easier to run a lot of experiments that vary in subtle ways
  - Link centric models instead of VM centric in the minimega API
  - VM-to-VM links to address MAC learning problems
  - Centralized and automated traffic generation routines

- Models developed to run the experiments in the paper will soon be available online:
  [https://mergetb.org/projects/searchlight](https://mergetb.org/projects/searchlight)

- Thanks to the minimega community for a great testbed tool

Brian Kocoloski
bkocolos@isi.edu
Backup Slides
Topology Modeling: VM-to-VM Links
Switch uses a switch forwarding database (FDB) to determine how to forward packets between TAP devices.

FDB is constructed through MAC address learning. This leads to at least two issues for us when:

- Guest switches implement custom layer-2 forwarding mechanisms
- Guest routers construct asymmetric routes

MAC learning based forwarding becomes problematic, leading to excessive BUM traffic broadcast and/or packet loss.
Solution: program minimega switch FDB with known device endpoints; e.g.:
   connect_vms s0 d0
   mark_v2v s0 d0

Removes MAC learning from FDB construction