Network Debugging for Experiments using CREASE

Alexander Wolosewicz (IL Tech), Raffay Atiq (SRI), Nishanth Shyamkumar (IL Tech), Vinod Yegneswaran (SRI), Ashish Gehani (SRI), Nik Sultana (IL Tech)

Technical Assistance (FABRIC/Fablib) from Komal Thareja and Mert Cevik (RENCI)

Assistance (FABRIC PTP) from Mami Hayashida, Hussamuddin Nasir, and Jim Griffioen (University of Kentucky)

First – Follow-Along Notebook

- https://crease.cs.iit.edu/knit11/artifact
- Run the setup cell – we'll come back to it later

Why build debugging tools?

- Survey – many use tools like ping, tcpdump, traceroute
  - Others tend to be vendor/system specific
- Using these can be tedious at scale
  - Many SSH terminals
  - Writing extra code
- More efficient tools > more time researching, less debugging

Approach

- Extend Fablib with new Node types and API
  - Backwards-compatible
  - Easy to convert existing notebooks to use CREASE
- User-level, no special permissions required
- Ability to debug issues like drops, misconfigurations, latency
  - Packet histories

Before/After Slice Setup for CREASE

```python
from fabrictestbed_extensions.fablib.fablib import FablibManager

slice_name = "crease-knit11-small"
fablib = FablibManager(log_level="DEBUG")

host_cores = 2
host_ram = 2
host_disk = 10
host_image = "default_ubuntu_22"
site = fablib.get_random_site()

vslice = fablib.new_slice(name=slice_name)

h1 = vslice.add_node(name="h1", site=site, cores=host_cores, ram=host_ram, disk=host_disk, image=host_image)
h1_iface0 = h1.add_component(model="NIC_Basic", name="h1p0").get_interfaces()[0]
h2 = vslice.add_node(name="h2", site=site, cores=host_cores, ram=host_ram, disk=host_disk, image=host_image)
h2_iface0 = h2.add_component(model="NIC_Basic", name="h2p0").get_interfaces()[0]
h3 = vslice.add_node(name="h3", site=site, cores=host_cores, ram=host_ram, disk=host_disk, image=host_image)
h3_iface0 = h3.add_component(model="NIC_Basic", name="h3p0").get_interfaces()[0]

# Attestable switch (pre-loaded BMv2 on Ubuntu 22.04, in main fablib 1.9.3)
r1 = vslice.add_attestable_switch(name="r1", site=site, ports=["p0", "p1", "p2"])

vslice.add_l2network(name="h1p0", interfaces=[h1_iface0, r1.get_port_interface("p0")])
vslice.add_l2network(name="h2p0", interfaces=[h2_iface0, r1.get_port_interface("p1")])
vslice.add_l2network(name="h3p0", interfaces=[h3_iface0, r1.get_port_interface("p2")])

vslice.submit()
```

```python
from fabrictestbed_extensions.fablib.fablib import FablibManager

slice_name = "crease-knit11-small"
fablib = FablibManager(log_level="DEBUG")

host_cores = 2
host_ram = 2
host_disk = 10
host_image = "default_ubuntu_22"
site = fablib.get_random_site()

# Replaces new_slice()
vslice = fablib.new_crinkle_slice(name=slice_name)
# New command
vslice.add_analyzer(site=site, cores=8, ram=16, disk=100)

h1 = vslice.add_node(name="h1", site=site, cores=host_cores, ram=host_ram, disk=host_disk, image=host_image)
h1_iface0 = h1.add_component(model="NIC_Basic", name="h1p0").get_interfaces()[0]
h2 = vslice.add_node(name="h2", site=site, cores=host_cores, ram=host_ram, disk=host_disk, image=host_image)
h2_iface0 = h2.add_component(model="NIC_Basic", name="h2p0").get_interfaces()[0]
h3 = vslice.add_node(name="h3", site=site, cores=host_cores, ram=host_ram, disk=host_disk, image=host_image)
h3_iface0 = h3.add_component(model="NIC_Basic", name="h3p0").get_interfaces()[0]

# Attestable switch (pre-loaded BMv2 on Ubuntu 22.04, in main fablib 1.9.3)
r1 = vslice.add_attestable_switch(name="r1", site=site, ports=["p0", "p1", "p2"])

# Replaces add_l2network to add monitor nodes to links
vslice.add_monitored_l2network(name="h1p0", interfaces=[h1_iface0, r1.get_port_interface("p0")], cores=4)
vslice.add_monitored_l2network(name="h2p0", interfaces=[h2_iface0, r1.get_port_interface("p1")], cores=4)
vslice.add_monitored_l2network(name="h3p0", interfaces=[h3_iface0, r1.get_port_interface("p2")], cores=4)

vslice.submit()
```

CREASE – Causal Reasoning and Attestation for Scientific Experimentation

- **Attestable Switch:** preloaded Ubuntu 22 with custom BMv2 v1.15
  - Python API for easy switch management
  - Optional remote attestation/provenance support for security research *(INDIS'23)*
- **Crinkle:** extended slice for easier debugging
  - Use of transparent middleboxes for packet-level provenance
  - Like running tcpdump on every node, but without the command hassle or the requirement the node support tcpdump
- **Choir:** traffic replayer *(INDIS '25)*
  - Quickly reproduce issues for debugging

Using CREASE

User Eval

- Compared FABRIC users using any tool they want to others using CREASE to debug issues in a medium-scale topology
- Some had significant time increases
  - Medium experience, up to 37.5% faster
- Since then: further improvements to tool usability

Want to Help?

- Website: https://crease.cs.iit.edu
- Survey
- Large Artifact: https://crease.cs.iit.edu/knit11/large-artifact
- Share your feedback – this tool is to help users like you!