ILLINOIS TECH
College of Computing

A Network Debugger for FABRIC Experiments

Alexander Wolosewicz — Illinois Tech
Vinod Yegneswaran — SRI
Ashish Gehani — SRI
Nik Sultana — Illinois Tech

Try CREASE out on FABRIC!

http://crease.cs.iit.edu/

Introduction

- FABRIC, as a testbed, affords significant flexibility compared to other deployment environments.
- This flexibility allows for new techniques that are much more general, supporting any components, topology, or protocols.

Motivation

- Debugging is a significant barrier to developing experiments on FABRIC.
- A research-friendly debugger should not be reliant on hardware or protocol support, since that is often the focus of the research.
- Data collection could pave the way for automated assistance in tracking down and solving bugs.

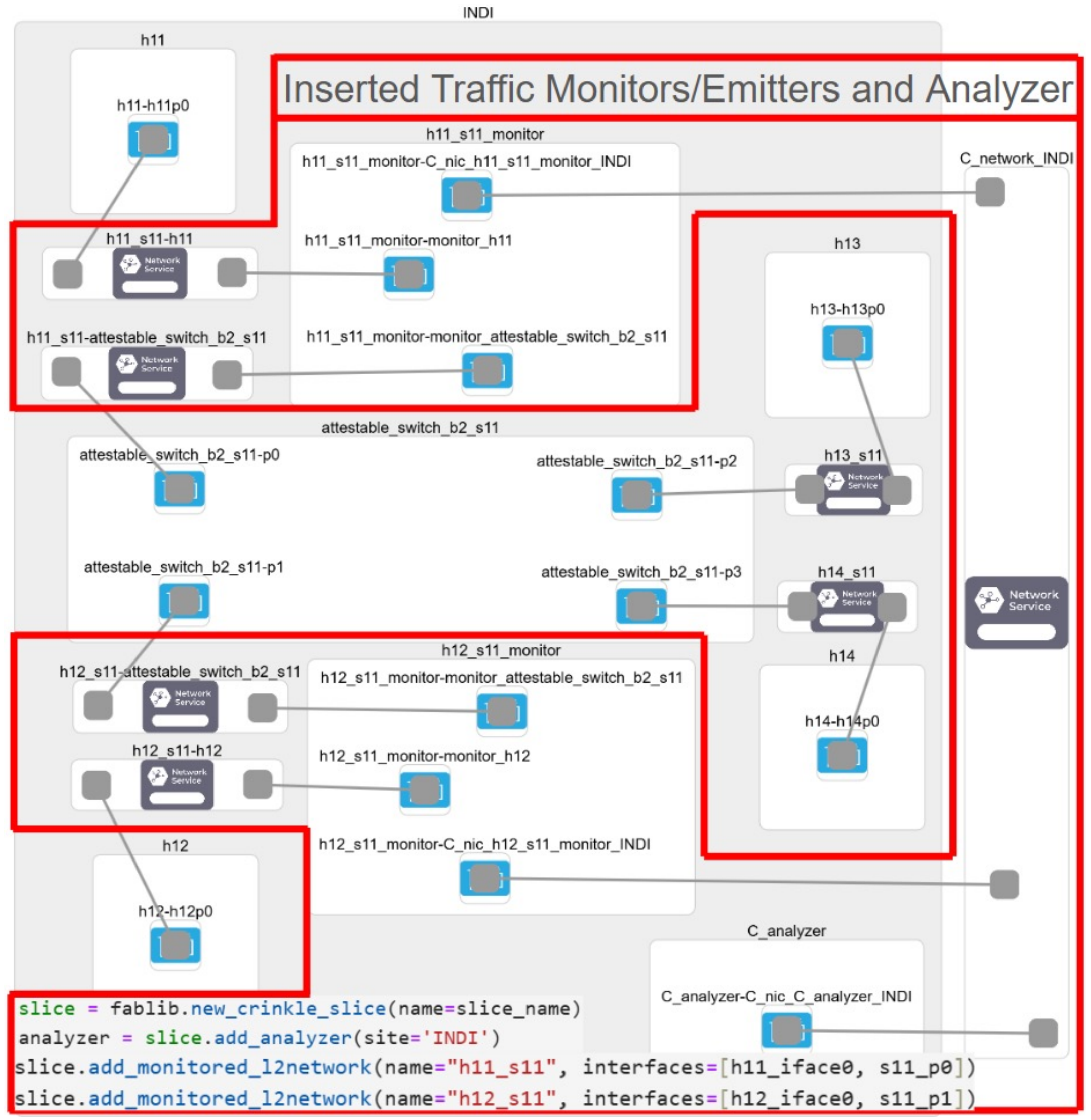
Acknowledgement

We thank Komal Thareja and Mert Cevik for technical assistance.

This work is supported by NSF CIRC 2346499.

Approach

- We extended Fablib to support adding VMs as “smart” bump-in-the-wire monitors.
- Monitors communicate with an analyzer out-of-band, and insert unique IDs to frames.
- Analyzer collects information on packet headers, ID, and location into database.
- Database contains flow-level data, position, and packet-specific ID.

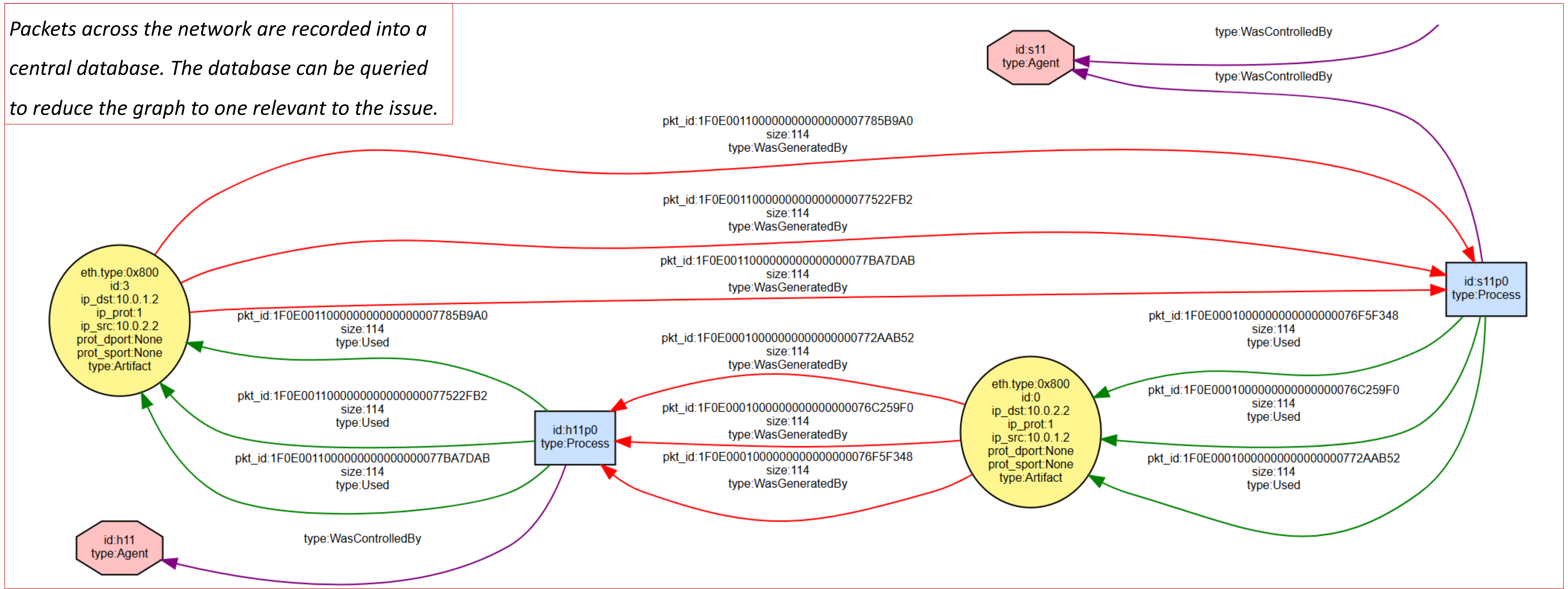

INDI
h11
h11-h11p0
Inserted Traffic Monitors/Emitters and Analyzer
h11_s11_monitor
h11_s11_monitor-C_nic_h11_s11_monitor_INDI
C_network_INDI
h11_s11-h11
Network Service
h11_s11_monitor-monitor_h11
h13
h13-h13p0
h11_s11-attestable_switch_b2_s11
h11_s11_monitor-monitor_attestable_switch_b2_s11
attestable_switch_b2_s11
attestable_switch_b2_s11-p0
attestable_switch_b2_s11-p2
h13_s11
attestable_switch_b2_s11-p1
attestable_switch_b2_s11-p3
h14_s11
h12_s11_monitor
h14
h12_s11-attestable_switch_b2_s11
h12_s11_monitor-monitor_attestable_switch_b2_s11
h14-h14p0
h12_s11-h12
h12_s11_monitor-monitor_h12
h12
h12_s11_monitor-C_nic_h12_s11_monitor_INDI
h12-h12p0
C_analyzer
C_analyzer-C_nic_C_analyzer_INDI

```
slice = fablib.new_crinkle_slice(name=slice_name)
analyzer = slice.add_analyzer(site='INDI')
slice.add_monitored_l2network(name="h11_s11", interfaces=[h11_iface0, s11_p0])
slice.add_monitored_l2network(name="h12_s11", interfaces=[h12_iface0, s11_p1])
```

Results

- Packet tracing can be done from a central point or the Jupyter instance, with significantly fewer commands than spawning and analyzing tcpdump jobs.
- Devices which do not support capturing their interfaces can be monitored as easily as any node.
- Lays a foundation for more active debugging by enabling anywhere probing and on-the-fly editing of header field values.

Packets across the network are recorded into a central database. The database can be queried to reduce the graph to one relevant to the issue.


id:s11
type:Agent
type:WasControlledBy
type:WasControlledBy
pkt_id:1F0E0011000000000000000007785B9A0
size:114
type:WasGeneratedBy
pkt_id:1F0E0011000000000000000077522FB2
size:114
type:WasGeneratedBy
pkt_id:1F0E00110000000000000000077BA7DAB
size:114
type:WasGeneratedBy
eth.type:0x800
id:3
ip_dst:10.0.1.2
ip_prot:1
ip_src:10.0.2.2
prot_dport:None
prot_sport:None
type:Artifact
id:s11p0
type:Process
pkt_id:1F0E0011000000000000000007785B9A0
size:114
type:Used
pkt_id:1F0E0001000000000000000000772AAB52
size:114
type:WasGeneratedBy
pkt_id:1F0E00010000000000000000076F5F348
size:114
type:Used
pkt_id:1F0E0011000000000000000077522FB2
size:114
type:Used
id:h11p0
type:Process
pkt_id:1F0E00010000000000000000076C259F0
size:114
type:WasGeneratedBy
eth.type:0x800
id:0
ip_dst:10.0.2.2
ip_prot:1
ip_src:10.0.1.2
prot_dport:None
prot_sport:None
type:Artifact
pkt_id:1F0E00010000000000000000076C259F0
size:114
type:Used
pkt_id:1F0E00110000000000000000077BA7DAB
size:114
type:Used
pkt_id:1F0E00010000000000000000076F5F348
size:114
type:WasGeneratedBy
pkt_id:1F0E0001000000000000000000772AAB52
size:114
type:Used
id:h11
type:Agent
type:WasControlledBy