User Study of a Network Debugger on FABRIC

Alexander Wolosewicz (IL Tech), Vinod Yegneswaran (SRI),

Ashish Gehani (SRI), Nik Sultana (IL Tech)

ILLINOIS TECH

College of Computing

Introduction

- Testbed environments are a prime resource for developing and evaluating research artifacts
- Many problems can arise
  - Misconfiguration of network elements
  - Failures of physical infrastructure or virtualization abstractions
- Common traffic monitoring tools (i.e. tcpdump) place a high burden on the experimenter for causal reasoning
- Provenance-based system ideal, captures causal relationships

Introduction

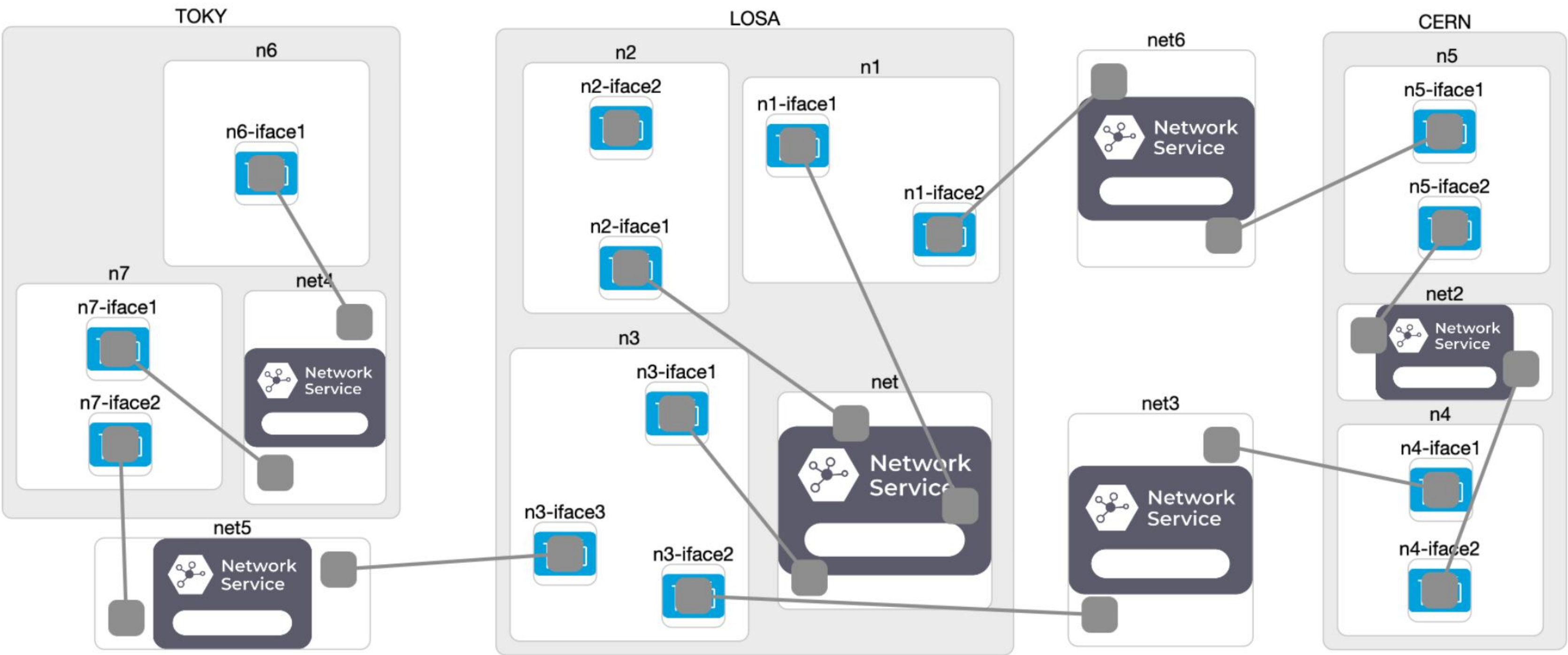
- We built a tool, Crinkle, to provide packet provenance-based debugging to FABRIC testbed users
- To explore its efficacy, we developed and carried out a user evaluation of Crinkle against a non-Crinkle baseline
- For this evaluation, we designed a telemetry system for capturing user activity and infrastructure state with low overhead
- Finally, we analyze FABRIC's suitability as a platform for controlled experiments

ILLINOIS TECH
College of Computing

FABRIC

- National testbed with 33 sites across the US and international partners, experimenters are virtual network tenants
- User slices consist of VMs in virtual networks using shared (SR-IOV) or dedicated (ConnectX-5/6) hardware
- Control experiments through a Python API, Fablib (and so, Jupyter)

FABRIC Slice



Example of a FABRIC slice, showing a multi-path network of:

- seven nodes (n1-n7)
- across three sites (Tokyo, LA, and CERN)
- with both intra- and inter-site links (net1-net6).

ILLINOIS TECH

College of Computing

Debugging on FABRIC

- Various network problems that can arise
- Congestion and faults in the underlay
- Correct routing: user must correctly configure all devices, as between n6 and n5 in previous figure
- Delays from congestion or unbalanced links
- Problems from increased complexity of smart NICs/FPGAs
- Run-to-run variation due to behavior of other tenants

ILLINOIS TECH
College of Computing

How People Use FABRIC

- We surveyed 22 FABRIC users on how they interact with FABRIC
- All reported using low-level techniques (tcpdump/traceroute), both in and outside of FABRIC
- Main difference for tools used on/off FABRIC was vendor-specific
- 50% of respondents ran Jupyter notebooks from other users, indicating there is active reuse of experiment artifacts
- 50% experienced infrastructure problems when running our experiment (normal testbed window) – benefit of telemetry

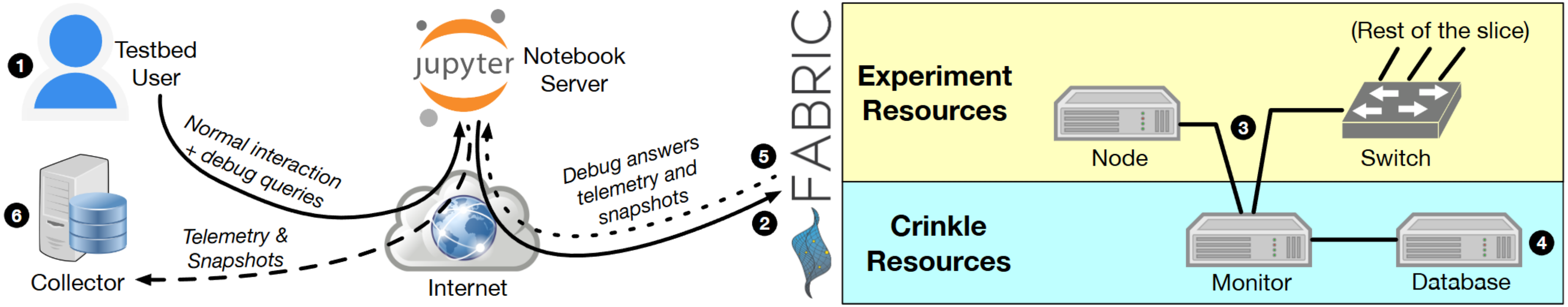
ILLINOIS TECH
College of Computing

Crinkle

- **Goal:** provide a debugging tool for users, without operator support, to debug issues with their experiment slices
- Crinkle uses only existing, user-level resources, permissions, and API
- The Crinkle API (extension of FABRIC's API) controls placing **Monitor** VMs, which bisect and provide monitoring of virtual links
- Monitoring captures per-packet provenance, stored in **Analyzer** DB
- API provides commands to quickly query provenance store and retrieve specific histories.

ILLINOIS TECH
College of Computing

Crinkle



1. User interacts with FABRIC across the internet
2. Users indicate which parts of their network they want to monitor
3. Crinkle handles bisecting virtual links and instantiating monitors
4. Captured provenance form monitors is transmitted/stored out-of-band
5. Debugging query responses and experiment telemetry relayed back to Jupyter
6. Telemetry is streamed to an outside collector for analysis

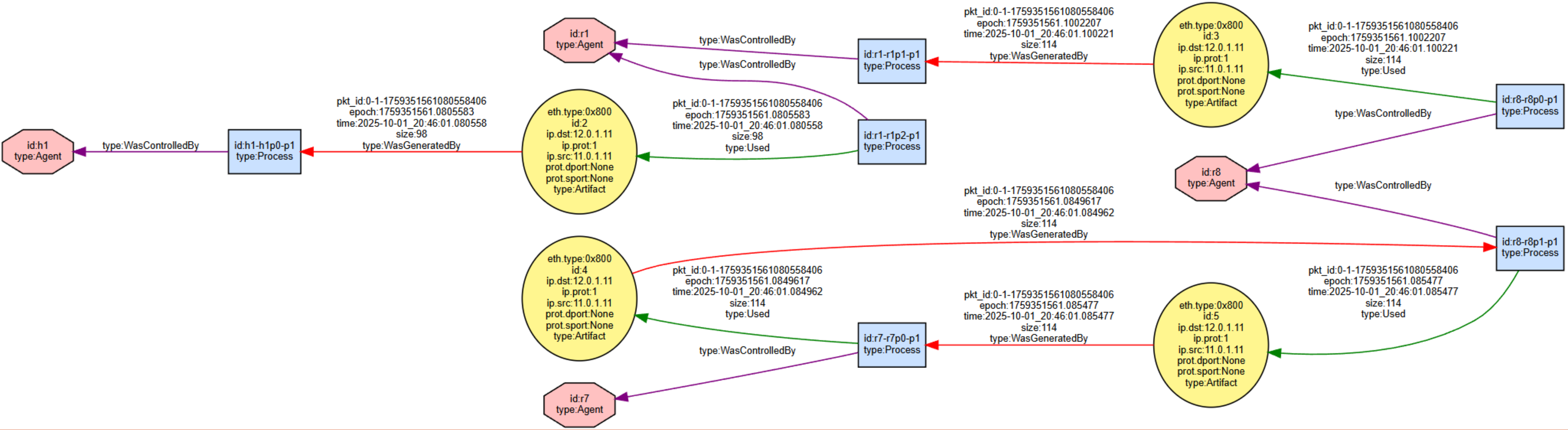
ILLINOIS TECH
College of Computing

Crinkle

- **Monitors:** Ubuntu 22.04 with a DPDK (C library) application
  - Each crossing frame is given a unique identifier (if lacking one)
  - Reports to **Analyzer** that a given ID crossed (with time, contents, direction)
- **Analyzer:** Ubuntu 22.04 running the SPADE provenance recorder
  - SPADE runs an SQL database back-end
  - User interacts with API that transforms common expressions (pcap syntax, timestamps, frame IDs) into SPADE QuickGrail prompts for the provenance
  - Provenance is returned as a provenance graph

Provenance Graph

- r7 receives a packet, reflects back to r8 where it drops.



Telemetry

- As users hit certain checkpoints running the Jupyter notebook, tuples are delivered to our endpoint via HTT P POST requests
- Tuples include user identity, experiment stage, FABRIC site, raised errors, and other troubleshooting details
- Telemetry enabled us to proactively reach out to struggling users and provide assistance for overcoming technical issues
  - Common case: slice setup failure, from site issue or SSH key issues

ILLINOIS TECH
College of Computing

Experiment Design

- **Goal:** Determine if Crinkle is an improvement over existing techniques in ease
- **Measures:** Time spent and attempts taken to localize a fault
- Participants were existing network professionals
  - **Pros:** more experienced than students, more likely to give genuine effort
  - **Cons:** more time constrained
- Design must balance time (and difficulty) with depth

ILLINOIS TECH

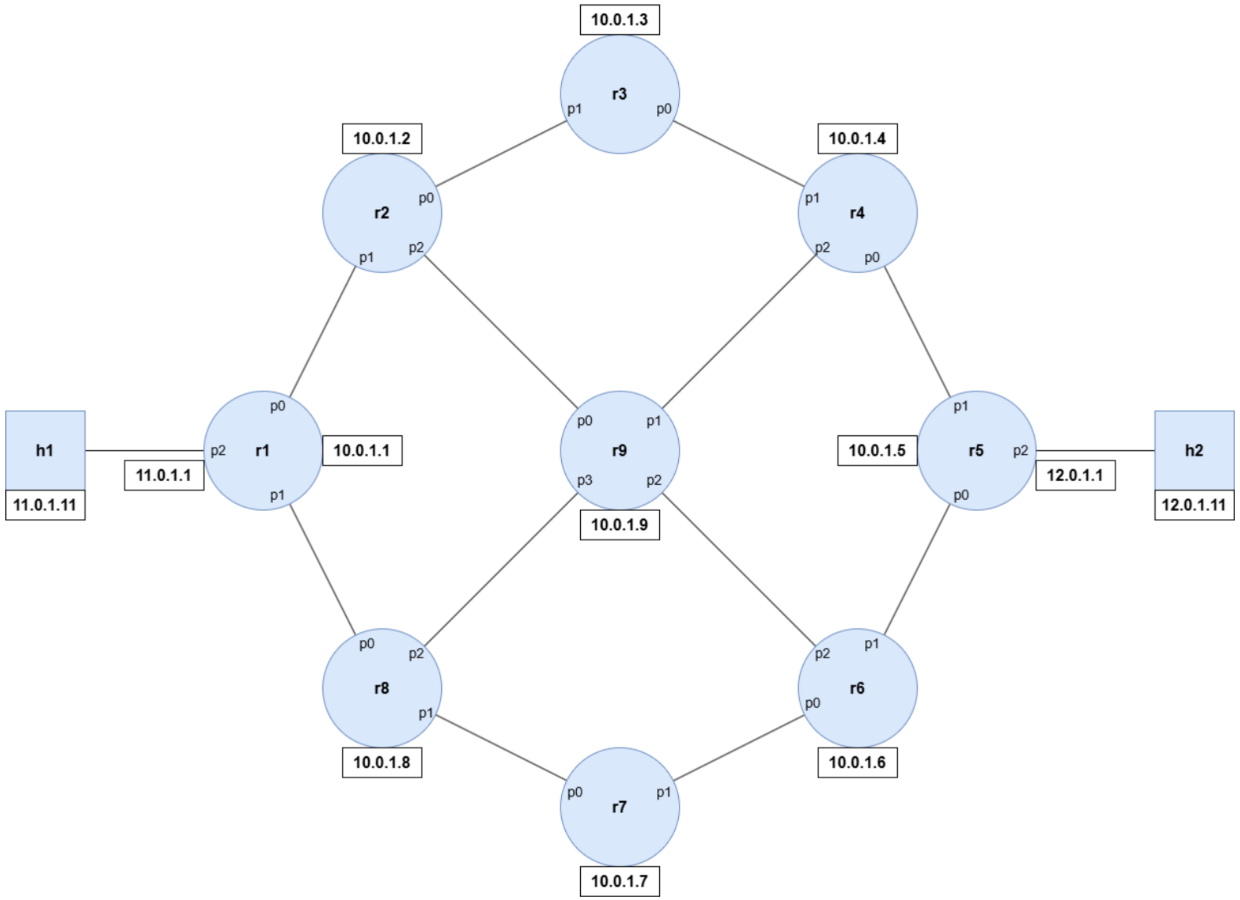
College of Computing

Routers

- Running the P4 programmable reference software switch BMv2 in Ubuntu 24.04
  - P4: language for programmable network hardware (ex: Tofino switches)
- Bare-bones IP forwarding program, mildly complicated to alternate outgoing links
  - **Does not respond to ICMP**
- One would be broken, user's goal is to find the broken one

ILLINOIS TECH
College of Computing

Topology



- **Minimize size** to minimize setup time
  - N: 11 nodes, 14 network services, 28 interfaces
  - C: 26 nodes, 29 network services, 71 interfaces
- **Maximize points of failure** to accurately assess tools in complex situation
  - 12 total paths, 9 failure points

ILLINOIS TECH
College of Computing

Problems

- **Problem 1:** Router r7 given a single bad table rule so r6-bound traffic goes to r8 (so h1-h2 pings that go through r7 go back to r8 and are dropped)
- **Problem 2:** Router r9 is faulty and black-holes traffic
- Users submit which router they think is malfunctioning (telemetry transmitted), and receive feedback on if answer is correct/incorrect
- Faults are revealed to user by running 10 pings per problem
  - **Run a baseline** post-startup to avoid outside faults

ILLINOIS TECH
College of Computing

Experiment Population

- **22 users** (of 33 signed up), with a range of experience in both using FABRIC and debugging networks
- **Crinkle group (C):** Given 15-minute demo video, required to use Crinkle
- **Non-Crinkle (N):** Allowed to use any tools they wanted

Attribute	**Non-CRINKLE**	**CRINKLE**
Network Debugging Experience		
Beginner	3	4
Medium	3	4
Advanced	5	3
Months of FABRIC Experience		
0-11	3	3
12-23	0	2
24-35	1	2
36-48	4	2
48+	3	2
Has P4 Experience		
Yes	3	7
No	6	3
No Answer	2	1

ILLINOIS TECH
College of Computing

Results – Telemetry

- Compared received telemetry against participants’ logs collected at the end of the experiment
- Telemetry system delivered all records successfully, with negligible overhead
  - 364 records in total
  - 85 exercise attempts
  - 3 issues not caught by telemetry, but could be in the future
  - Other issues detected and enabled proactive help

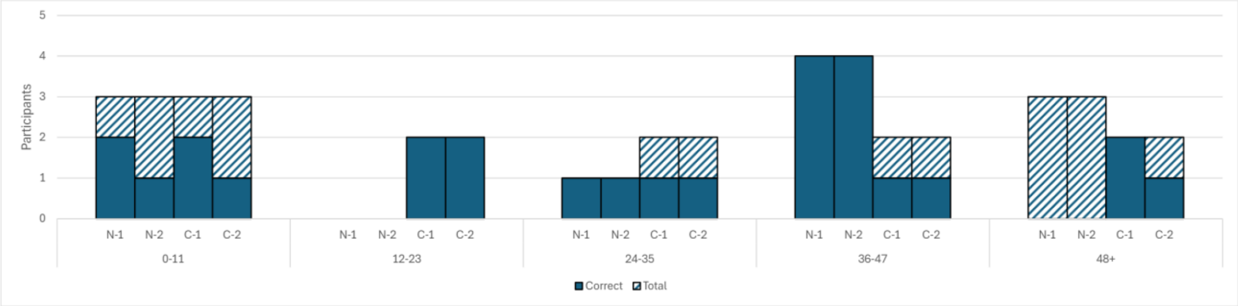
ILLINOIS TECH
College of Computing

Results – FABRIC

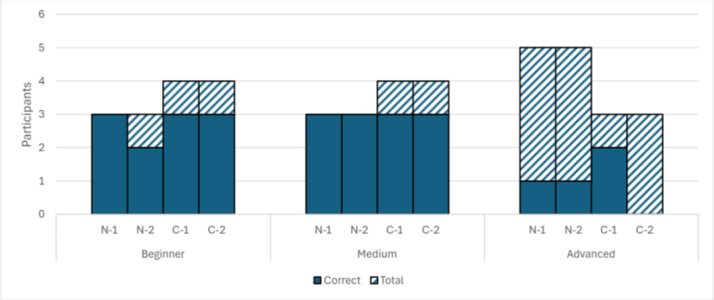
- 38 slices created across 24 sites
- Telemetry-revealed faults:
  - 1 participant with missing permissions, 2 without a FABRIC project
  - 1 participant missing a Jupyter environment configuration
  - 4 participants with missing or expired SSH keys
  - 5 participants had a slice fail baseline, then succeed on next slice (3N, 2C)
  - 1 participant encountered a reservation issue due to a lack of resources
- Of 22 users, only 5 had non-persistent infrastructure issues

ILLINOIS TECH
College of Computing

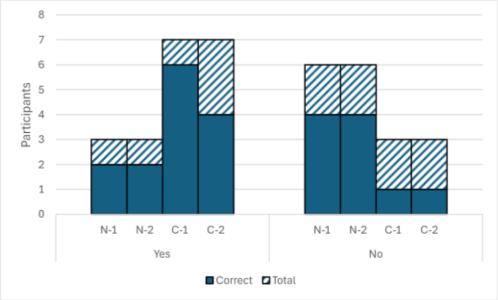
Results – Crinkle



(a) Users who provided correct answers to an exercise, grouped by months of FABRIC experience.



(b) Users who provided a correct answer, grouped by network debugging experience.



(c) Users who provided a correct answer, grouped by having P4 experience.

- **Users with medium experience debugged up to 37.5% faster**
- Advanced users: time constrained, lack of familiar tools
- Participants reported difficulty understanding provenance graphs

ILLINOIS TECH
College of Computing

Findings – Using FABRIC

- **Overall, FABRIC is a viable platform for carrying out a distributed, asynchronous user study**
- Primary downside: participants are not in controlled environment
  - One reported having to make "a lot of context switches" doing other work
  - Conversely, we had a wider set of participants
- Simplifies providing participants with equal environments
- Other issues:
  - Inconsistency: started slices that fail baseline test
  - Long startup time: N avg. 27.23 σ 22.36 minutes, C 46.68 σ 5.58 minutes

ILLINOIS TECH
College of Computing

Recommendations

- For testbeds:
  - **Slice startup callback**, so slice setup can proceed in the background and the user be notified when it completes
  - **Ability to start slices for other users**, without juggling SSH keys, so participants can immediately start
- For experimenters:
  - **Post-startup baseline test** catches many subtle issues from provisioning, ensure data integrity, and with telemetry can allow for quickly catching issues

ILLINOIS TECH
College of Computing

Conclusion

- To our knowledge, first controlled user study on FABRIC
- FABRIC, with telemetry collection, has several upsides for running distributed user evaluations
- **Future work:** adapting these techniques so participants can evaluate and provide feedback on cybersecurity research prototypes
  - Ideally, provides a system that other FABRIC researchers can reuse
- More on Crinkle, part of the CREASE project: https://crease.cs.iit.edu/

ILLINOIS TECH
College of Computing

Artifacts

- Updated demo: https://crease.cs.iit.edu/knit11/large-artifact
- Original Non-crinkle (possesses telemetry):
  - Instructions: https://crease.cs.iit.edu/userexperiment/instructions_n.pdf
  - Notebook: https://crease.cs.iit.edu/userexperiment/crease_user_experiment_n.tgz
- Original Crinkle (possesses telemetry):
  - Instructions: https://crease.cs.iit.edu/userexperiment/instructions_c.pdf
  - Notebook: https://crease.cs.iit.edu/userexperiment/crease_user_experiment_c.tgz

ILLINOIS TECH
College of Computing