Quality Control Methodology for Simulation Models of Computer Network Protocols

8th Virtual OMNeT++ Community Summit
8-10th September 2021, Zoom/Discord, Internet

MOTIVATION

- The quality of simulation results depends on the accuracy of used models
  - i.e., how precisely models reflect the behavior of the real-world system

- This paper focuses on <u>our experience with</u> the testing of developed <u>computer networking models</u> and their comparison with referential implementations

PROTOCOL DEFINITION

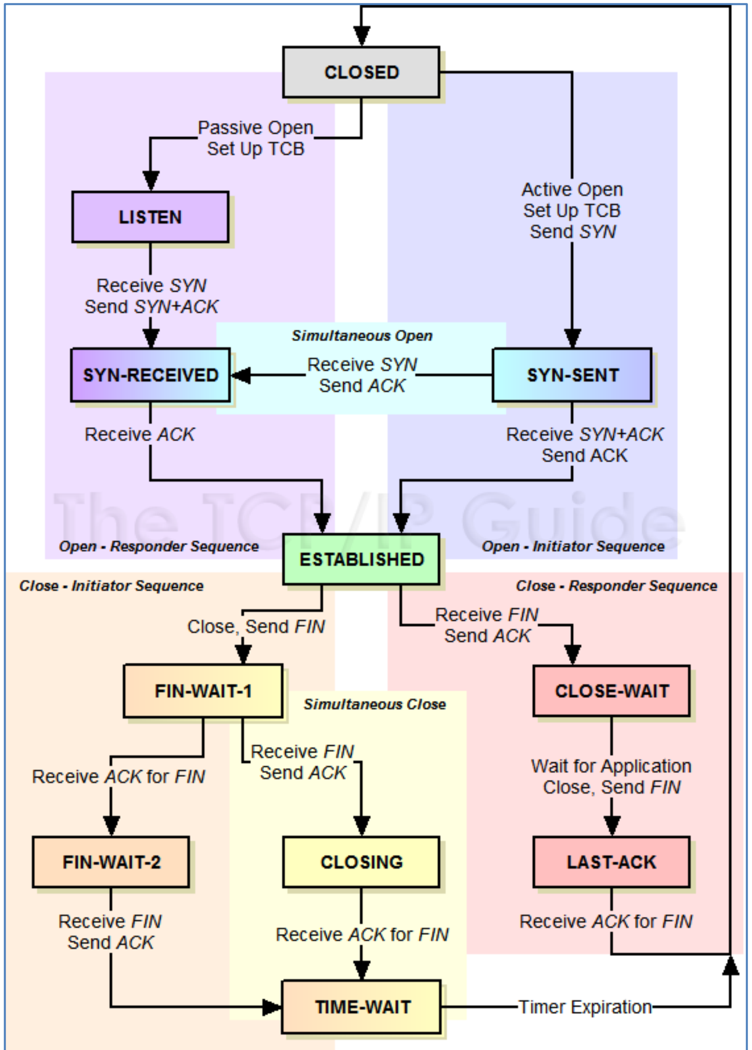
- The (computer networking) protocol
  - syntax and semantics of messages
  - rules for sharing the state

- Any protocol can be formally described using:

1) Deterministic finite-state machines (FSM)
2) Temporal logic

- FSMs are more popular
  - easier to understand
  - built-in support to create FSMs in certain tools

TCP ← FSMs → BGP



http://tcpipguide.com/free/t_TCPOperationalOverviewandtheTCPFiniteStateMachineF-2.htm

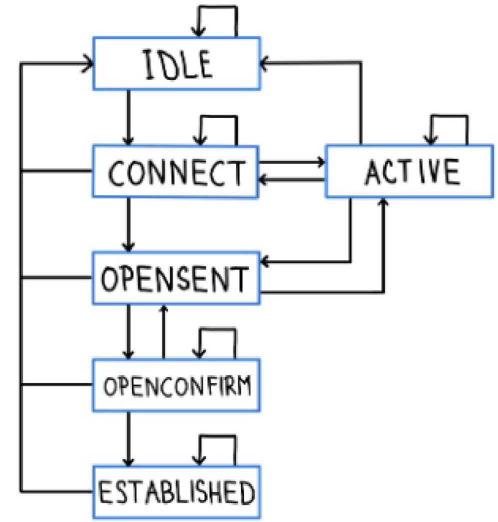
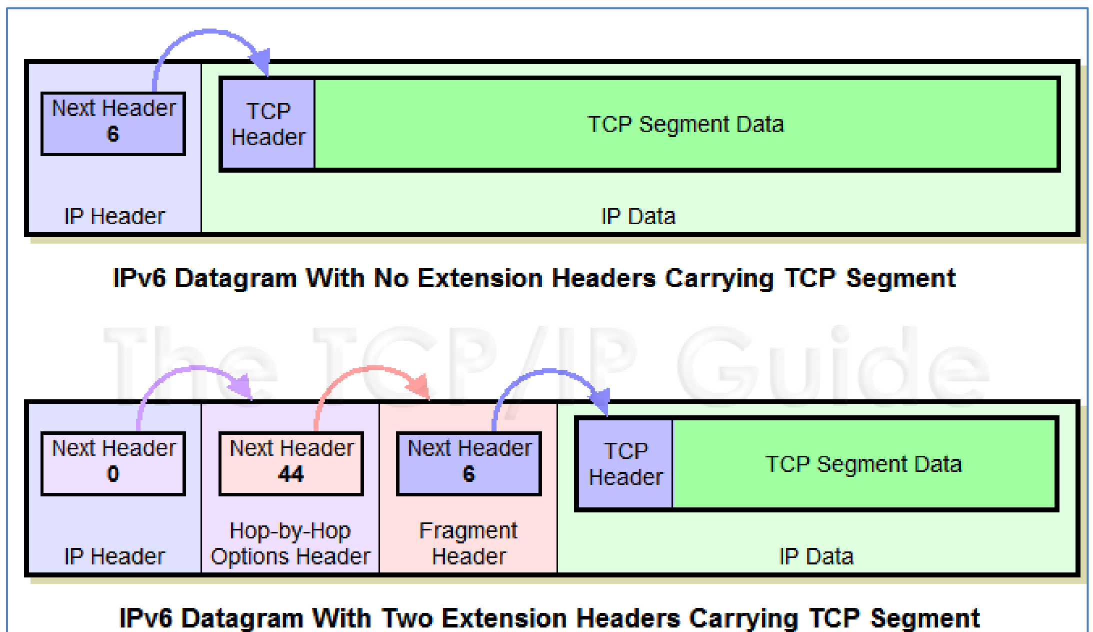


Figure 4.4: BGP neighbor state FSM

- **CONNECT** - BGP process is waiting for TCP connection to be established. If it is successful, it sends an OPEN message, refreshes *ConnectRetry Timer* and changes state to *OPENSENT* state. If it is not successful until *ConnectRetry Timer* expires, it changes to *ACTIVE* state. In other cases falls back to *IDLE* state.
- **ACTIVE** - BGP is trying to establish a TCP connection with a neighbor. If successful, it sends an OPEN message, refreshes *ConnectRetry Timer* and changes state to *OPENSENT* state. If it is not successful, it falls back to *CONNECT* state and refreshes *ConnectRetry Timer*. Oscilation between *CONNECT* and *ACTIVE* states indicates error with TCP connection.
- **OPENSENT** - BGP is waiting for OPEN message from it's peer. If TCP connection is closed before that happens, BGP falls back to *ACTIVE* state. When OPEN message is received, it's content is checked. If this check fails, BGP sends NOTIFICATION message and falls back to *IDLE* state. If check passes BGP sends KEEPALIVE message and transfers to *OPENCONFIRM* state.
- **OPENCONFIRM** - BGP is waiting for peer's NOTIFICATION or KEEPALIVE message. If KEEPALIVE is received, BGP transfers to *ESTABLISHED* state. If HOLD TIMER expires or NOTIFICATION is received, BGP falls back to *IDLE* state.
- **ESTABLISHED** - BGP starts to exchange UPDATE messages with it's peer. HOLD TIMER is refreshed with each reception of UPDATE or KEEPALIVE message. If faulty UPDATE or NOTIFICATION message is received or HOLD TIMER expires, BGP transfers back to *IDLE* state.

PROTOCOL IMPLEMENTATION

- Standard covers the main part of protocol behavior
- Implementation of standard may add case-specific functionality
- Protocol design
  - non-flexible with hard-coded stuff (like OSPFv2 vs. OSPFv3, RIPv2 vs. RIPng)
  - extensible using type-length-value records (like IS-IS, EIGRP, BGP, Babel, TCP)



http://www.tcpipguide.com/free/t_IPv6DatagramExtensionHeaders-2.htm

Referential Implementations

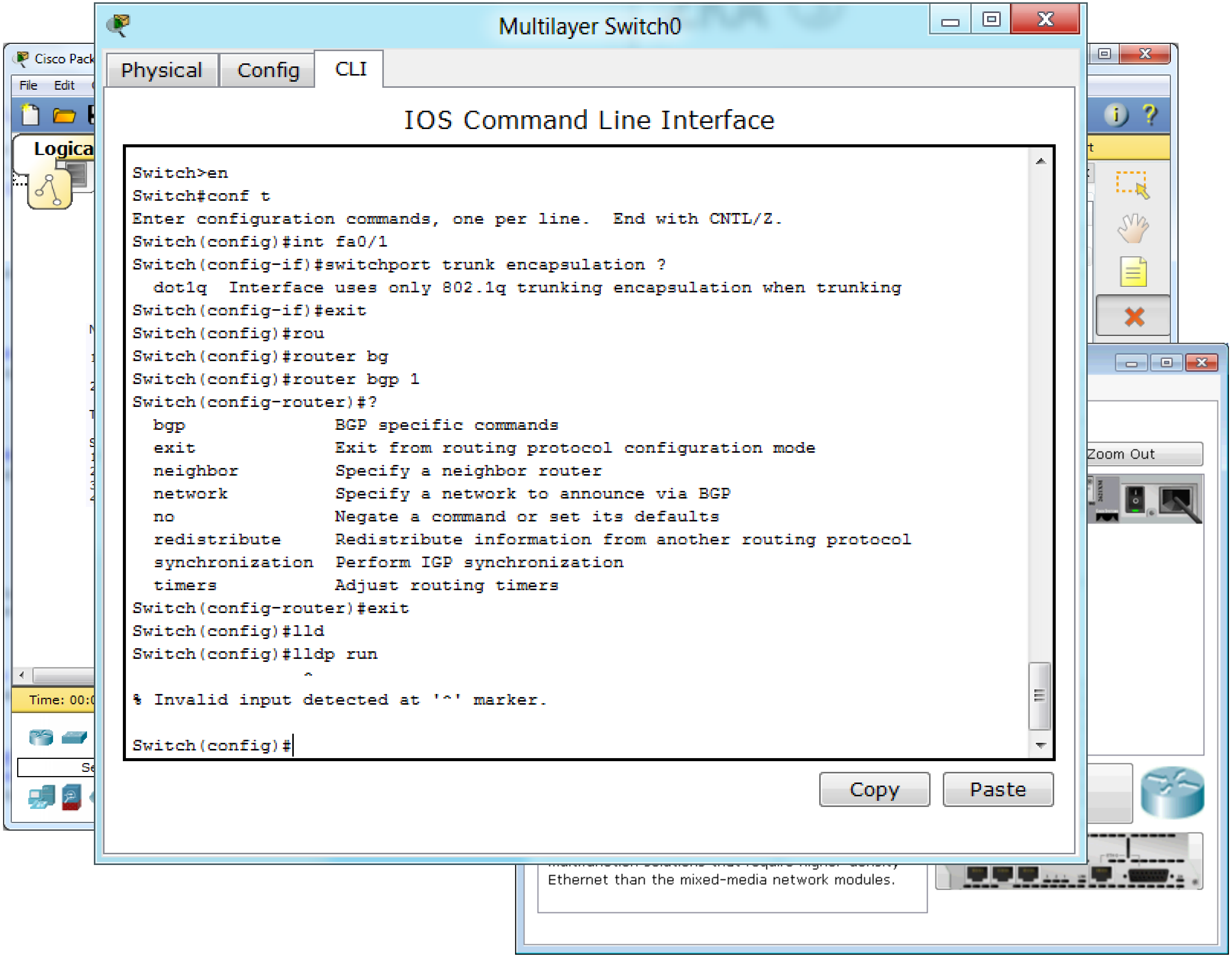
- A short non-exclusive list of referential implementations we have seen being used with respect to INET contributions:

- **Cisco Packet Tracer**
  - simulator supporting teaching activities within Cisco NetAcad
  - limited functionality, non-conformant messages and behavior

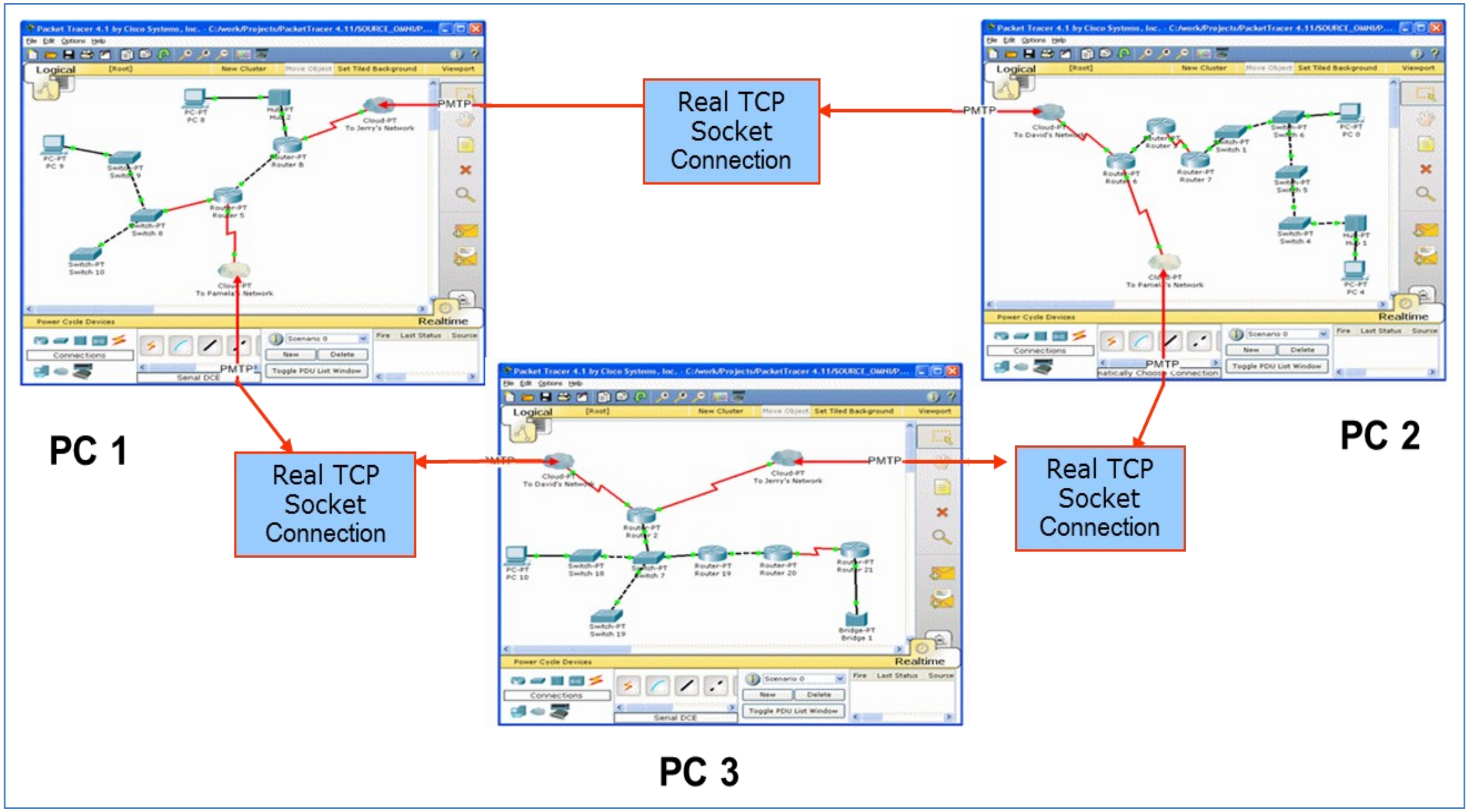
- **Physical device**
  - vendor specific functionality
  - expensive for results reproduction (potentially large set of exactly same hardware devices running exactly the same software)

- **GNS3/EVE-ng**
  - emulator / virtualization of active network devices
  - capable to run even selected proprietary systems (e.g., Cisco IOS)

CISCO PACKET TRACER (1)

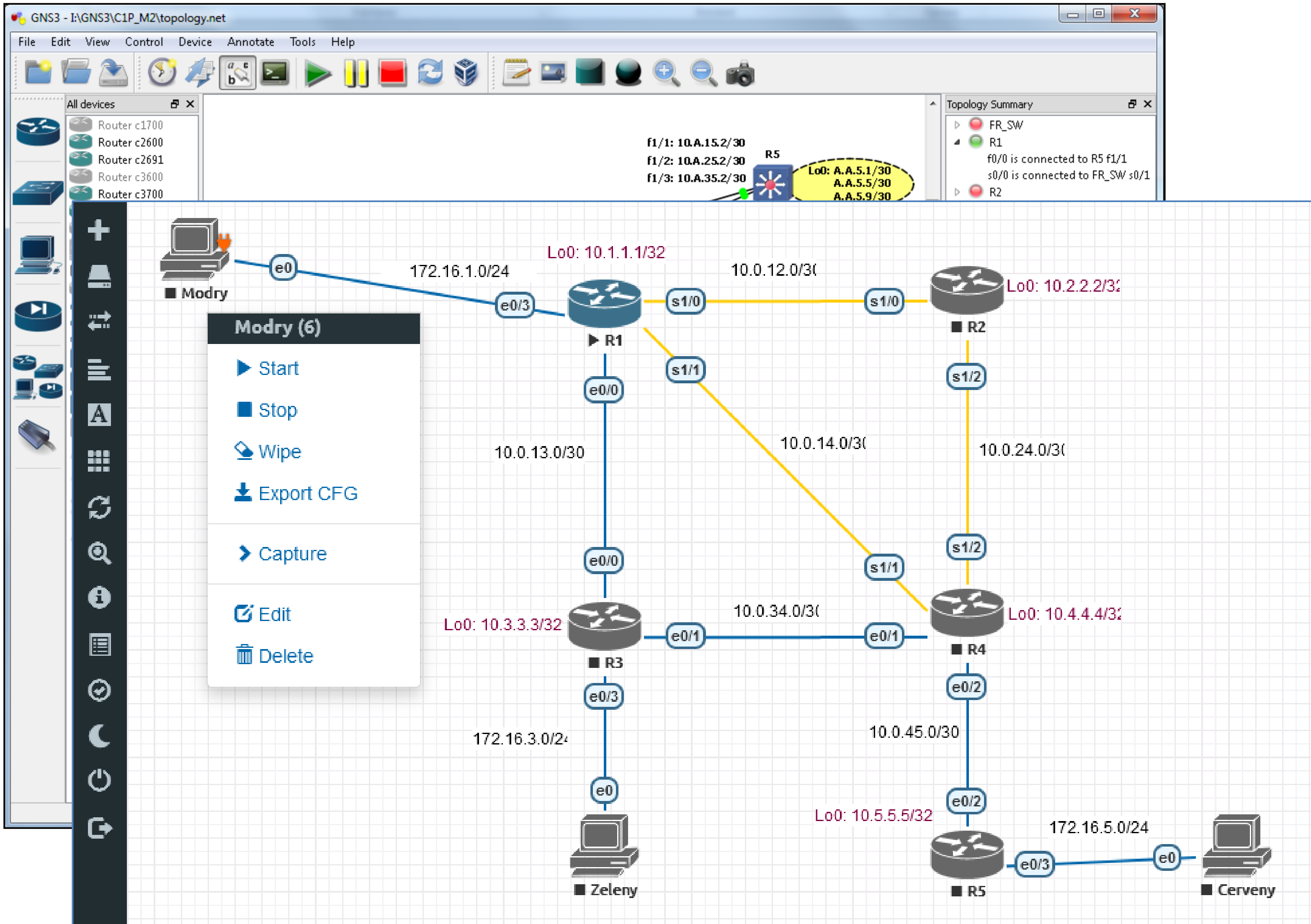


CISCO PACKET TRACER (2)



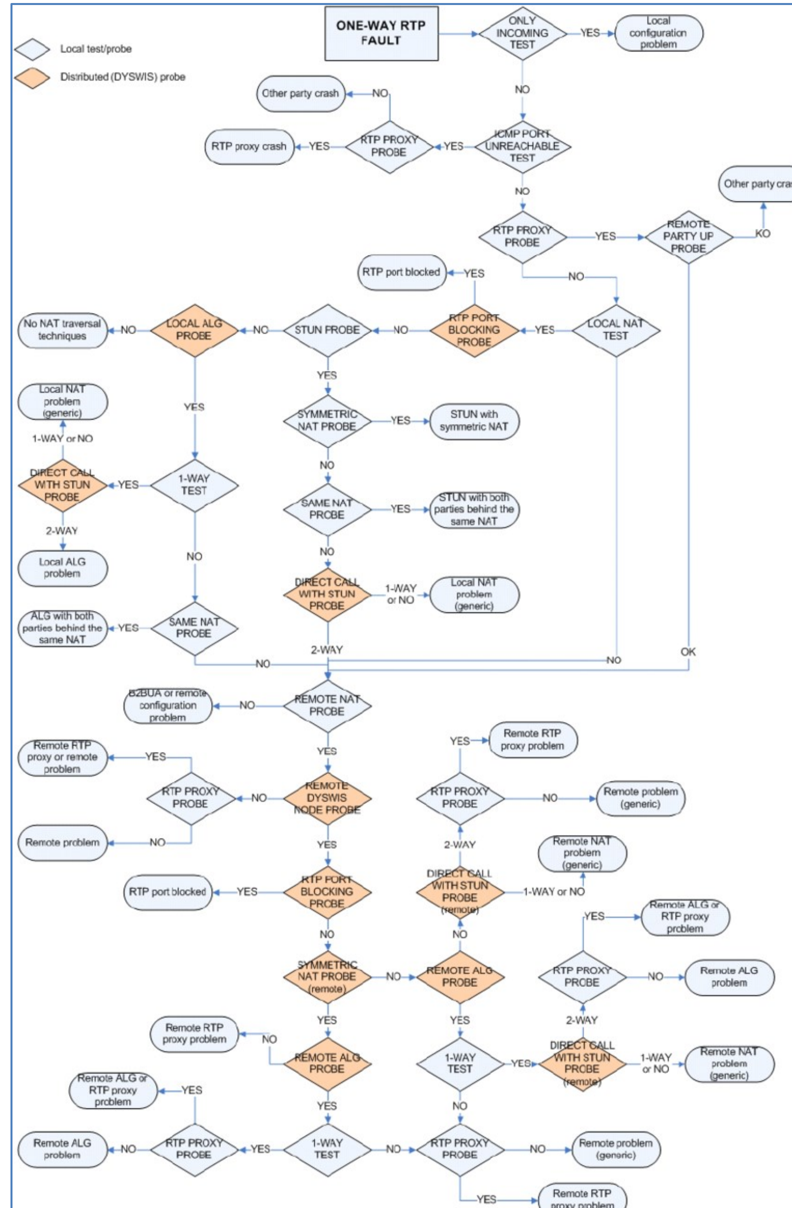
GNS3 / EVE-NG

- Dynamips / QEM



CHALLENGES: LEVEL OF ACCURACY

- A complete conversion of all protocol rules may lead to extremely complicated FSM (with many states/transitions and complex message variants)
  - even more tricky with protocols that offload signalization or data transfer onto other protocols
- *What should we omit? (and make protocol less accurate)*



https://www.researchgate.net/figure/Flow-diagram-representing-the-whole-diagnosis-process_fig3_221418915

CHALLENGES: APPLICATION OF CRYPTOGRAPHY

- Guarantees confidentiality, integrity and authentication
- Pros
  1) the conformance of the message generated by the simulator with the referential simulation; thus
  2) it is the only way how to support hardware in the loop (HIL) simulation
- Cons
  1) it is a known fact before running the simulation whether confidentiality/integrity/authenticity is guaranteed or not between involved parties;
  2) the boilerplate of the simulation model source code tends to increase dramatically by adding external libraries handling cryptography (such as OpenSSL); which leads to
  3) application of cryptography poses an overhead on resources (mainly CPU time and memory) when running the simulation (we need to wait longer for results or we could be even unable to simulate complex topologies)
- *Shall we include of exclude cryptography?*

Challenges: Timing

- Referential implementation of the protocol runs in real-time, while the simulation is governed by a discrete event scheduler
  - Due to the lack of global clocks, it is hard to measure durations, trigger actions, and control events between devices in real-time
- It is mandatory to employ time synchronization protocols
  - **NTP** (RFC 5905)
  - **PTP** (IEEE 1588-2019)
- Scheduling of events
  - **ScenarioManager** for OMNeT++
  - **Embedded Event Manager** and **TCL** for Cisco IOS
  - **Expect**, **Ansible** and other remote management scription tools
- *What toolbox are we going to use to guarantee timing?*

Challenges: Control Plane Randomness

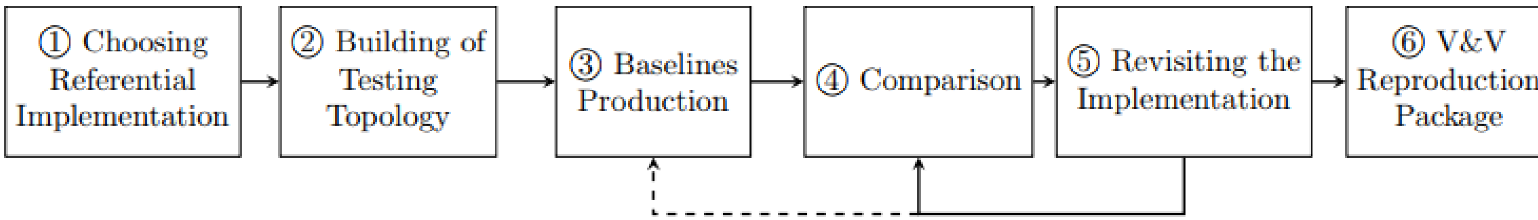
- The control plane of the actual device runs and dynamically switches between processes based on resource schedulers
  - This context switching introduces a degree of randomness, which impacts the reproducibility and baselines' readability
- Following symptoms relate to this challenge:
  - Stochastic delays are observed in the functionality of referential implementation when the control plane is preoccupied with another process
  - Consecutive protocol messages have non-standard gaps between each other due to the packet pacing. This jitter between messages is purposely introduced by the control plane either to avoid potential racing conditions between protocol instances or to guarantee stable bandwidth consumption
- *Any comparison with baseline produced by a real control plane should consider this randomness…*

GOAL

- This paper aims to define a structured V&V process that any programmer may use as a cookbook for quality control of simulation models
- Various challenges which may be encountered during the development and testing phases
- Each step of methodology based on all previously mentioned observations in this article

METHODOLOGY (1)

- The proposed methodology consists of six consecutive phases depicted in the following diagram and described below:



1) Choosing Referential Implementation

- either physical devices from a trusted vendor or a network emulator with a corresponding firmware image

2) Building of Testing Topology

- conduct V&V on the smallest possible topology, which would offer a good testing ground to assess normal behavior and treatment of edge cases
- It is important to keep parameters (e.g., interface speeds, IP subnetting) constant across real and simulated topology to maintain integrity

METHODOLOGY (2)

3) Baseline Production

- following three types of baselines
  - Syslog messages
  - outputs of `show/debug` commands and monitoring dashboards
  - PCAP files with computer traffic dumps
- all above type of baselines should be equipped with (up to nanosecond level) timestamps

4) Comparison

- two levels of comparison
  - **protocol level** (where we are focusing on the generated messages and their integrity – both syntactical and semantical)
  - **abstract data structure level** (which focuses on states of abstract data structures used by the protocol, such as the routing table, interface table, CAM table, topology table for EIGRP, link-state database for OSPF, etc.)
- conduct testing repeatedly on different scenarios with various configs

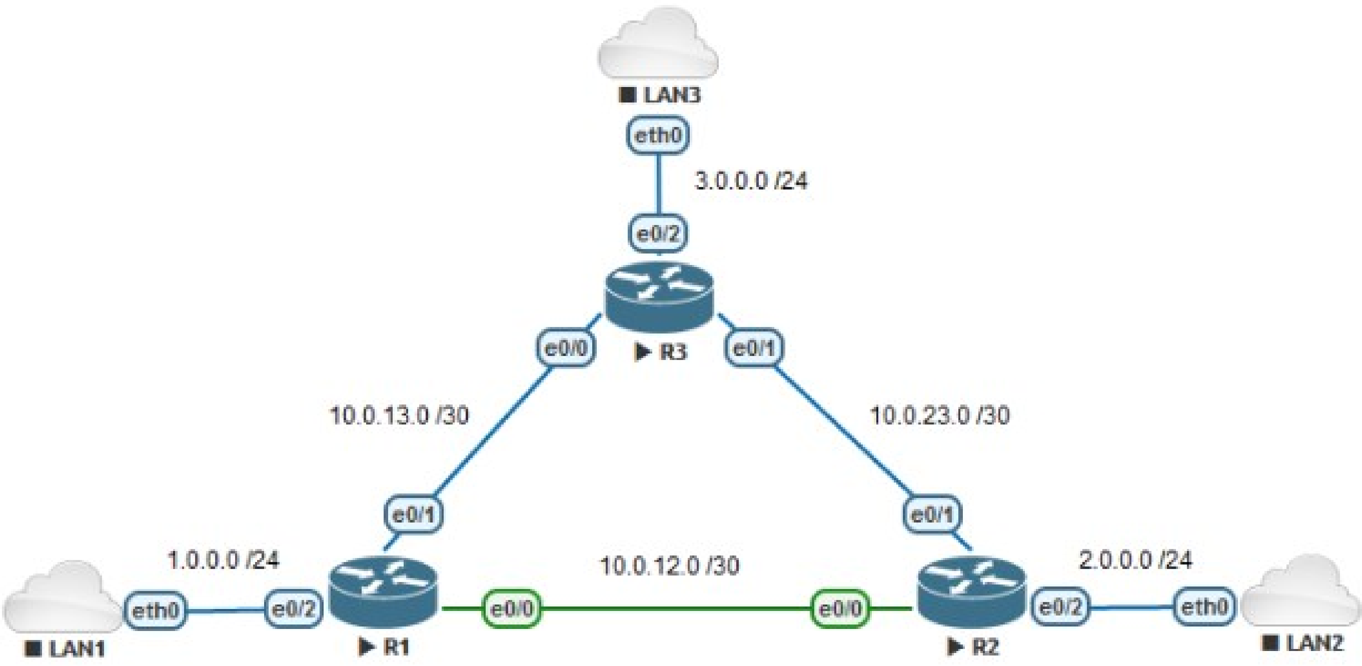
Methodology (3)

5) **Revisiting the Implementation**
   - the simulation model can be modified, updated, or even completely redesigned depending on findings from the previous steps 3) and 4)
   - this process is repeated until the quality of the simulation model is sufficient *(hopefully, the quality would even exceed original expectations)*

6) **V&V Reproduction Package**
   - any contributed simulation model should be accompanied by materials (e.g., referential implementation version, baselines including PCAP/Syslog dumps, simulation trace files) that allow reproduction of resulting behavior as proclaimed by the author
   - additional testing and V&V done by the community have a chance to find new errors or unhandled cases that may further improve the quality of resulting simulation models

Demonstration: Phase ① and ②

- INET 4.3 running in OMNeT++ 6.0 pre10
- Cisco IOS 15.7(3)M2



- Scenario I: Initial route discovery
- Scenario II: Topology change propagation

DEMO: PHASE ③ (BASELINE PRODUCTION)

No.	Time	Source	Destination	Protocol	Length	Info
1	3.729698	10.0.12.1	224.0.0.10	EIGRP	84	Hello
2	3.738674	10.0.12.2	224.0.0.10	EIGRP	84	Hello
3	3.739372	10.0.12.2	10.0.12.1	EIGRP	60	Update
4	5.745241	10.0.12.2	10.0.12.1	EIGRP	60	Update
5	5.745326	10.0.12.1	10.0.12.2	EIGRP	60	Update
6	5.753815	10.0.12.2	224.0.0.10	EIGRP	276	Update
7	5.753882	10.0.12.2	10.0.12.1	EIGRP	60	Hello (Ack)
8	5.766658	10.0.12.1	224.0.0.10	EIGRP	276	Update
9	5.766981	10.0.12.2	10.0.12.1	EIGRP	60	Hello (Ack)
10	8.758402	10.0.12.2	10.0.12.1	EIGRP	187	Update
11	8.767811	10.0.12.1	10.0.12.2	EIGRP	60	Hello (Ack)
12	8.776812	10.0.12.2	224.0.0.10	EIGRP	143	Update
13	8.777383	10.0.12.1	224.0.0.10	EIGRP	143	Update
14	8.777768	10.0.12.2	10.0.12.1	EIGRP	60	Hello (Ack)
15	8.778087	10.0.12.1	10.0.12.2	EIGRP	60	Hello (Ack)

Figure 4: Scenario I - Captured EIGRP traffic between R1 and R2 displayed with Wireshark

#	Time	Relevant Hops	Name	ID / Source	Destination	Type	Length
1	54.500'072	R1 --> R2	EIGRP_HELLO_MSG	10.0.12.1	224.0.0.10	EIGRP	78 B
2	54.500'072	R2 --> R1	EIGRP_HELLO_MSG	10.0.12.2	224.0.0.10	EIGRP	78 B
3	54.500'144	R1 --> R2	EIGRP_UPDATE_MSG	10.0.12.1	10.0.12.2	EIGRP	72 B
4	54.500'144	R2 --> R1	EIGRP_UPDATE_MSG	10.0.12.2	10.0.12.1	EIGRP	72 B
5	54.500'211'200	R1 --> R2	EIGRP_ACK_MSG	10.0.12.1	10.0.12.2	EIGRP	72 B
6	54.500'211'200	R2 --> R1	EIGRP_ACK_MSG	10.0.12.2	10.0.12.1	EIGRP	72 B
7	54.500'278'400	R1 --> R2	EIGRP_UPDATE_MSG	10.0.12.1	10.0.12.2	EIGRP	286 B
8	54.500'278'400	R2 --> R1	EIGRP_UPDATE_MSG	10.0.12.2	10.0.12.1	EIGRP	286 B
9	54.500'516'800	R1 --> R2	EIGRP_ACK_MSG	10.0.12.1	10.0.12.2	EIGRP	72 B
10	54.500'516'800	R2 --> R1	EIGRP_ACK_MSG	10.0.12.2	10.0.12.1	EIGRP	72 B
11	54.500'584	R1 --> R2	EIGRP_UPDATE_MSG	10.0.12.1	224.0.0.10	EIGRP	154 B
12	54.500'584	R2 --> R1	EIGRP_UPDATE_MSG	10.0.12.2	224.0.0.10	EIGRP	154 B
13	54.500'716'800	R1 --> R2	EIGRP_ACK_MSG	10.0.12.1	10.0.12.2	EIGRP	72 B
14	54.500'716'800	R2 --> R1	EIGRP_ACK_MSG	10.0.12.2	10.0.12.1	EIGRP	72 B

Figure 5: Scenario I - Captured EIGRP traffic between R1 and R2 diplayed in OMNeT++

Demo: Phase ④ (Protocol Comparison)

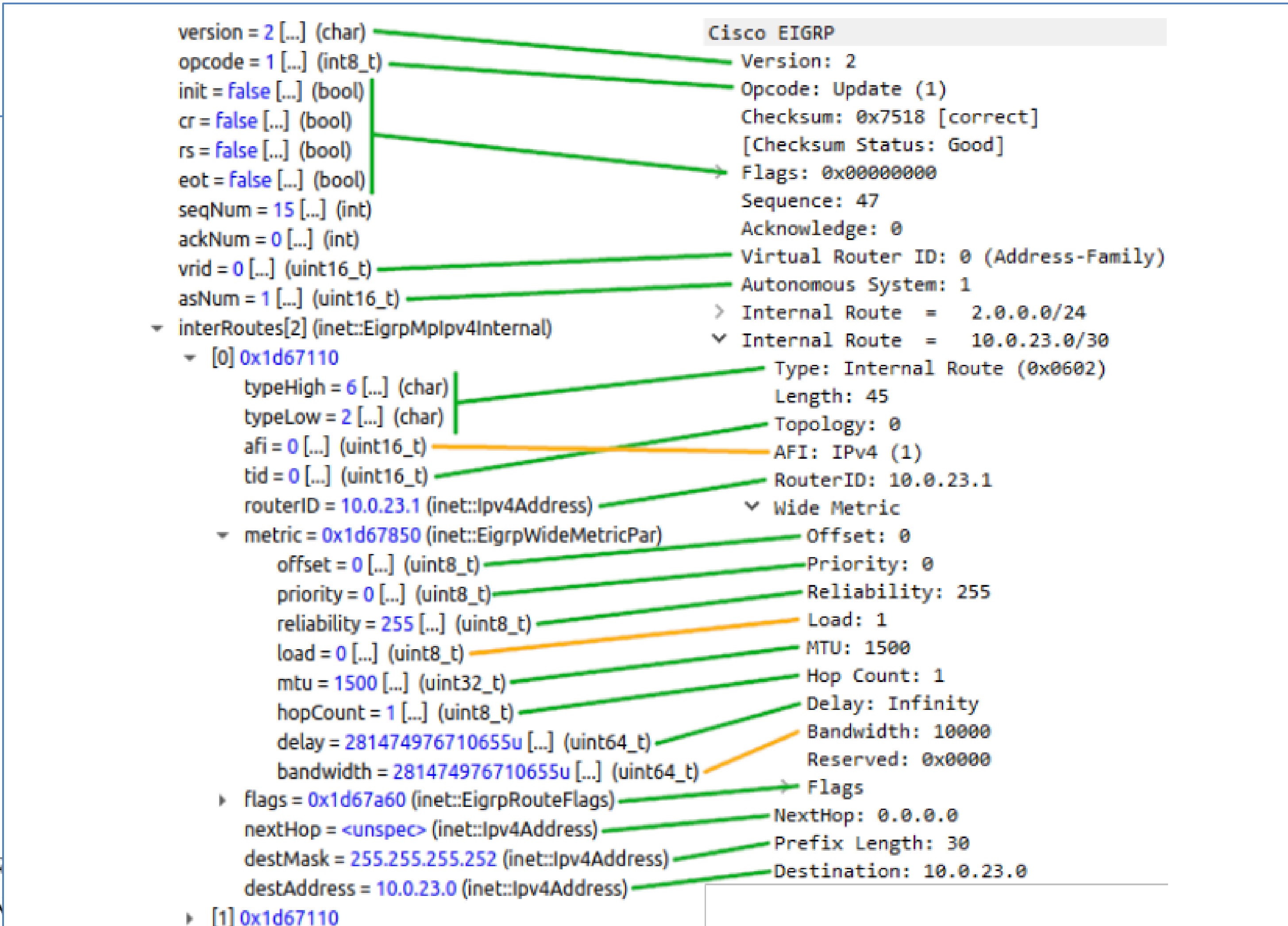


Figure 7: Scenario I - Comparison of Update message **13** from referential topology and Update message **11** from the OMNeT++ simulation.

Demo: Phase ④ (Traffic Comparison)

Cisco	OMNeT++	Description
1, 2	1, 2	Exchange of Hello packets. When a router receives a Hello message from a new neighbor, it creates a new entry for this specific neighbor and sets its status to `pending`. The content and format of these messages are shown in Figure 6.
3, 4, 5	3, 4	Exchange of Update packets with `INIT` flag. These do not contain any routing information. On the referential topology router `R1` did not acknowledge message **3** in time, so router `R2` re-sent the Update as message **4**, message **5** contains piggybacked acknowledgement for this message.
6	-	This Update message contains advertised routes from router `R2` and only appears on the referential topology. This message is sent because the neighbor status from `R2`'s point of view went from `pending` to `up`. This causes the message to be ignored and not acknowledged by router `R1` because from its point of view, `R2`'s neighbor status is still pending as `R1` did not receive an acknowledgment for its initial update message, message **5**, yet.
7	5, 6	Acknowledgments for initial Update messages. There is only one acknowledgment on the referential topology because it was piggybacked into the Update message as previously mentioned.
8, 10	7, 8	Exchange of Update messages containing all advertised routes by both routers. On the referential topology, one Update is sent as unicast because it is a retransmission of message **6**. It is also smaller because the router applied the *split-horizon* rule which prohibits an advertisement of a route towards its next hop. Another Update on the referential topology is sent as multicast. This is in contrast to the simulation model which uses unicast for the Update messages during the initial synchronization.
9, 11	9, 10	Acknowledgements for Update messages.
12, 13	11, 12	Exchange of Update messages advertising networks which have a successor on this interface as unreachable, i.e, `2.0.0.0/24` and `10.0.23.0/30` by `R1` and `1.0.0.0/24` and `10.0.13.0/30` by `R2`. This is according to the *poison reverse* rule. The content and format of these messages is shown in Figure 7.
14, 15	13, 14	Acknowledgments for Update messages.

Table 1: Scenario I - Analysis of the traffic between routers `R1` and `R2`.

Intro

SotA

QC

Demo

Outro

DEMO: PHASE ④ (ADT COMPARISON)

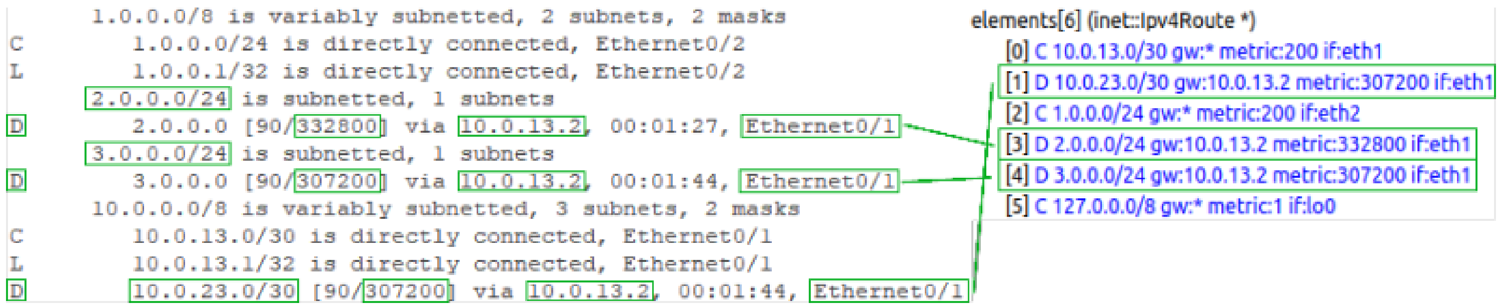


Figure 8: Scenario I - Comparison of router R1's routing table in its initial state.

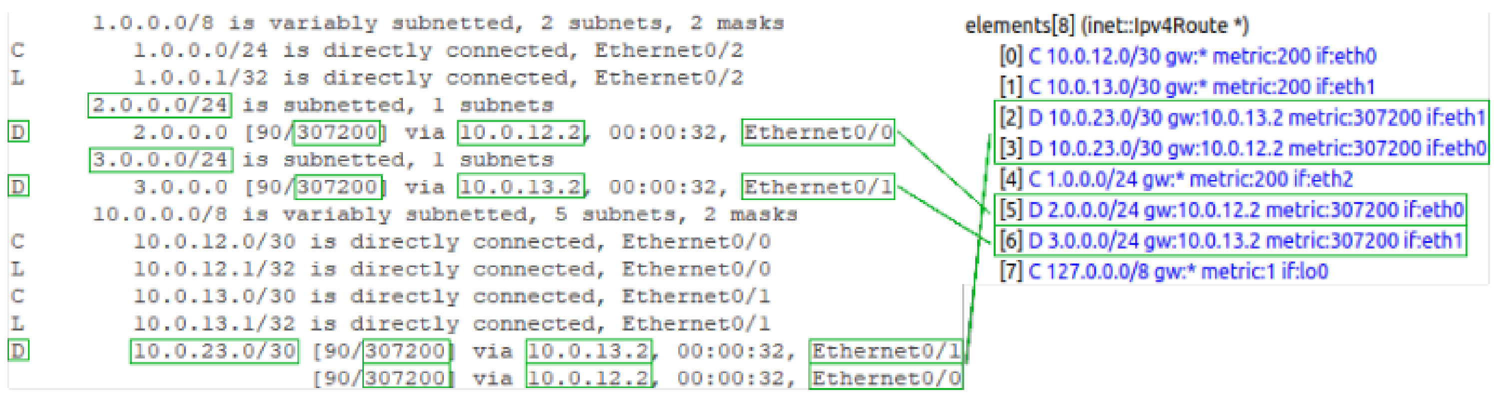


Figure 9: Scenario I - Comparison of router R1's routing table after the topology reached convergence.

Final Remarks

- **<u>There is a very thin line between making the objective comparison</u> of ground truth baseline and simulated behavior, <u>and subjectively choosing matching</u> simulation <u>results</u> onto the corresponding baseline ☺**

- *We hope this paper will stimulate discussion within the OMNeT++ community (and hopefully beyond it), which would help find a common agreement on the verification and validation process for any contributions!*

CONTRIBUTIONS

- We are preparing pull request towards INET with our BGP improvements
- https://github.com/ANSA/results-reproduction/tree/master/bgp-multi-address-family

Intro
SotA
QC
Demo
Outro

Demo: Phase ⑥ (Reproduction package)

Projects | Wiki | Security | Insights | Settings

OMNeT Community Summit 2021

AwziNihilist edited this page 1 minute ago · 4 revisions

Edit | New Page

This Wiki page contains a demonstration of our V&V methodology proposed in Section 3.2 of our V&V paper written for OMNeT++ Community Summit 2021. All files which are referred to in this article can be found here.

V&V Demonstration

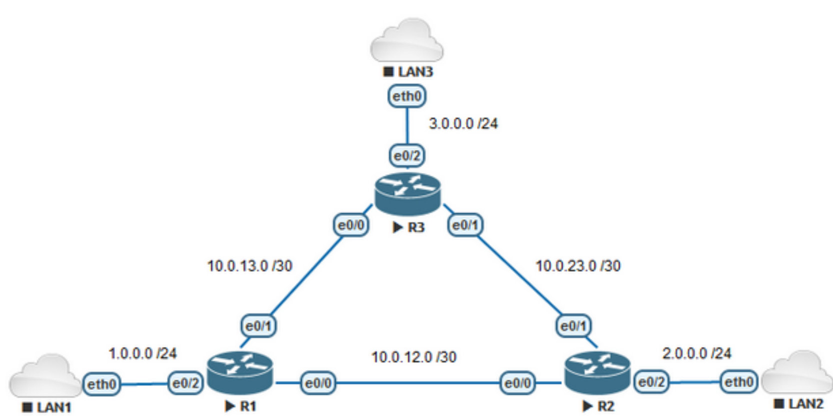
We decided to demonstrate this process on INET's EIGRP simulation model as available in INET 4.3 running in OMNeT++ 6.0 pre10. The simulation model is configured via a combination of Ipv4NetworkConfigurator for assigning IP addresses and EIGRP specific .xml file which has a deliberately similar structure to Cisco configuration.

Choosing Referential Implementation

Currently, the only viable choice for referential implementation of EIGRP is Cisco. Firstly, Cisco Systems is the author of this protocol. As such, there should be the smallest amount of discrepancies between the RFC and actual implementation caused by genuine errors and mistakes in the implementation. Secondly, there are very few other implementations that could be used as a reference. However, some chapters (e.g. stub routing) are still completely missing in RFC 7868 even though they are fully operational on Cisco devices. All that being said, we are using network emulator EVE-ng with IOS version 15.7(3)M2 to build our referential topology.

Building of Testing Topology

We decided to use the topology shown in the following figure:



+ Add a custom sidebar

Clone this wiki locally

https://github.com/ANSA/results-

Intro
SotA
QC
Demo
Outro