New IP Sandbox

Leveraging Linux for the design of applications using New IP open platform

Tutorial Overview

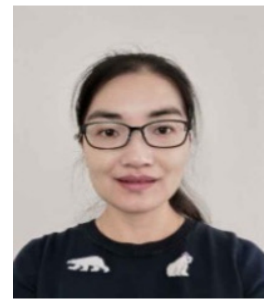
- **Introduction and Setup**
- **Part 1: Introduction to New IP (motivation and format)**
  - Rationale for New IP
  - Address Innovation
  - New IP Contracts
  - New IP Design on Linux
- **Part 2: New IP Sandbox Test Environment**
  - Introduction to Linux namespaces and NeST topology creation platform.
  - NeST Design and Features
  - Platform for New IP Sandboxing
- **Part 3: Demonstration of the use cases**
  - Forwarding based on Asymmetric addresses
  - Developing Contracts latency-aware forwarding.
  - Platform Capabilities
- **Part 4: Service extensibility on New IP platform**
  - Extension scenario of latency-aware scheduling and forwarding for multiple time constraint streams.
- **Wrap up**

Presenters

Kiran Makhijani
(Futurewei Tech, USA)

Mohit Tahiliani
(NITK, India)

Bhaskar Kataria
(NITK India)



Lijun Dong
(Futurewei Tech, USA)

Deepta and Shanshak (NITK) for support with tutorial and access to New IP VM

Setup

- Slides and VM for hands on portion of the tutorial
  - https://network2030.github.io/NewIP-Linux/
- Access Cloud Account to VM
  - ssh ubuntu@<ip_address>

  Pwd – to be provided
  - VM Download for your own use

129.154.224.36
144.24.108.91
144.24.115.88
129.154.224.207
152.67.25.130
144.24.103.180
144.24.118.49
129.154.231.126
129.154.227.93
129.154.226.88

New IP Overview

Introduction to New IP (motivation and format)

Basic Specification

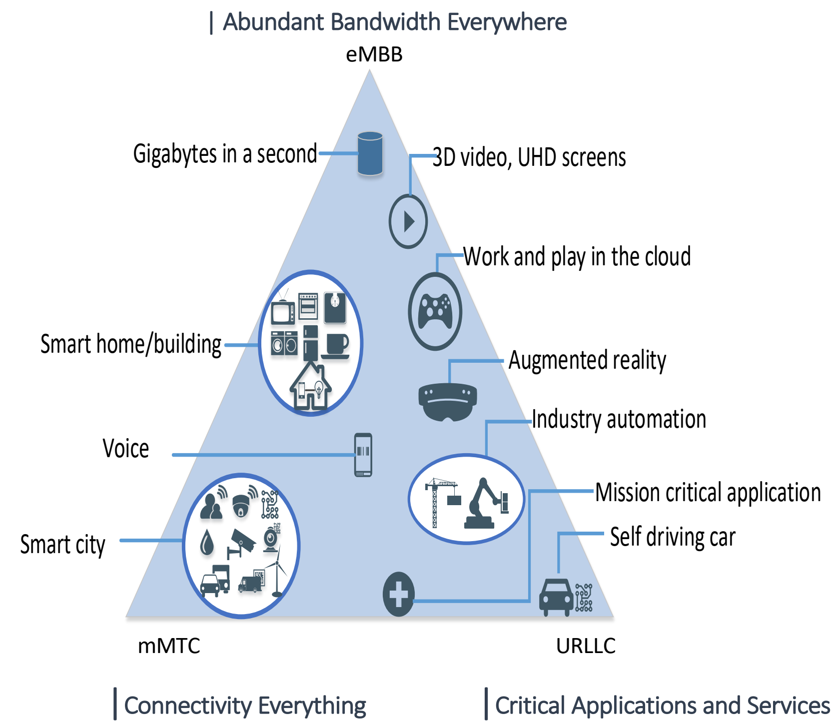
Network Requirements

- Support for several billions of devices.
- Support for a variety of connections
- Support for terabytes of high-volume data streams

Translating to Requirements on the Networks

- High reliability in machine-centric data delivery
- Fine-grained customization of traffic
- Guarantees of data delivery
  - Timeliness, quality, throughput
- Loss less flow of traffic between the endpoints
  - Reliability
- New Security aspects
  - Digital Sovereignty, Preservation of Privacy
- Digital Twins & Cyber physical systems.

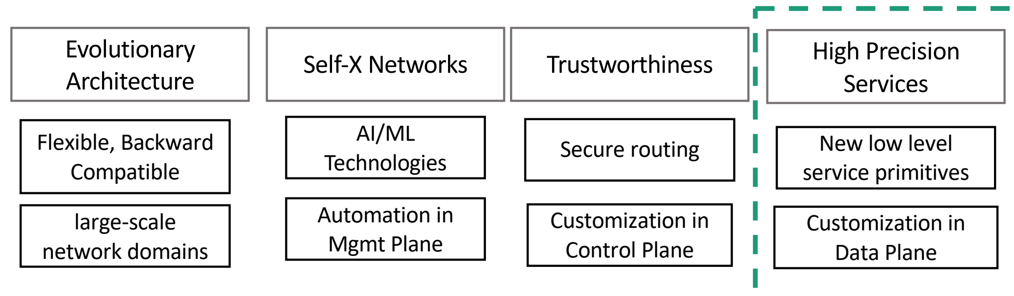
Understand impact of these requirements on the current network architecture



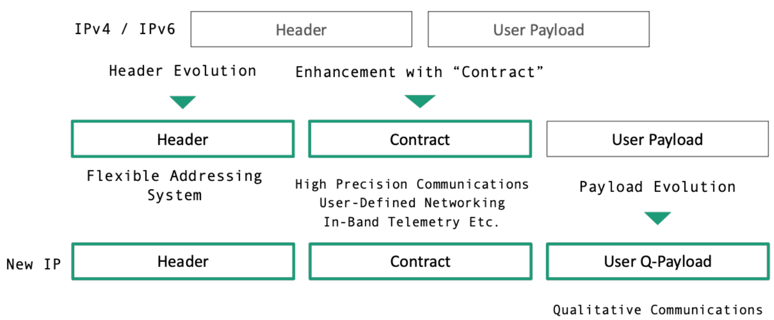
Source: ITU-T IMT-2020

What is New IP?

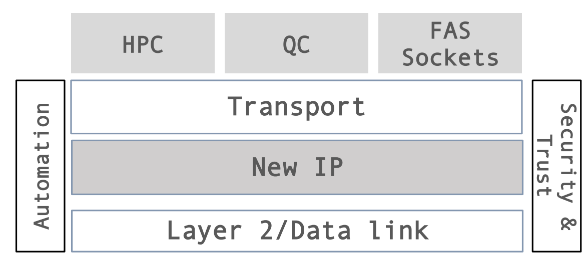
An abstraction of Network Technologies across different dimensions to enable next-gen application scenarios



Promotes Holistic approach vs Siloed approach: past few decades network innovation are driven by specific requirement within the constraints of current network architecture



An early stage of concept to support newer applications in evolutionary, flexible and programmable manner

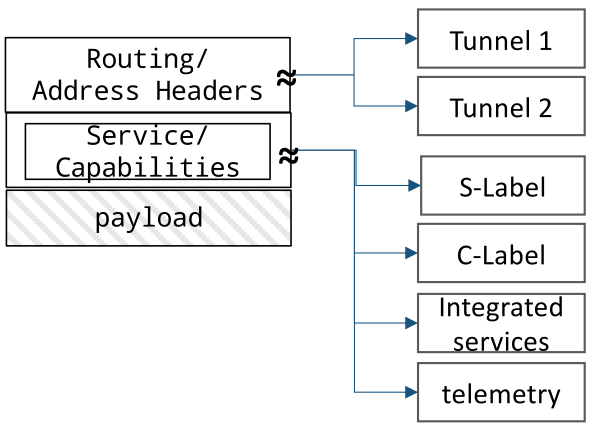


New Network Service Primitives

- Best Effort
  - Forward a traffic flow from a network node
- Differentiated services
  - Ingress - Enable classification, Police and mark traffic
  - Core – Queuing Dropping, Per-hop behavior (PHB) for minimum rate assurance, egress shaping and aggregate traffic
- Traffic Engineering
  - path selection based on constraints on bandwidth
- Limitations when dealing with newer applications
  - ✗ Fails to provide throughput guarantees
  - ✗ Fails to provide guarantees of maximum latency (in-time)
  - ✗ Fails to provide guarantees of precise latency (on-time)

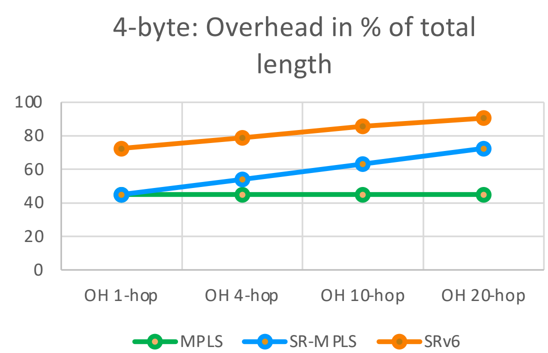
•**Technical Specification**:"New Services and Capabilities for Network 2030: Description, Technical Gap and Performance Target Analysis" (October 2019)
•**Technical Report:** "Network 2030 - Gap Analysis of Network 2030 New Services, Capabilities and Use cases" (June 2020)

Structured Headers lead to overheads



MPLS	bytes	MPLS-SR	bytes	SRV6	bytes
				IPv6 Encap	40
				SRH header	8
Transport Labels	4 to 12	Transport Labels	4 x SID count (upto 30)	transport SID	16 x SID count (upto 30)
Service Label	4	Service Label	4		
Outer IPv4 (for GTP)	20	Outer IPv4 (for GTP)	20	Service SID	16
UDP Hdr	4	UDP Hdr	4	UDP Hdr	8
GTP	12	GTP	12	GTP	12
Inner User IP	20	Inner User IP	20	Inner User IP	20
User Transport	4	User Transport	4	User Transport	4
User Payload	4 to 1200	User Payload	4 to 1200	User Payload	4 to 1200

$$transport\ OH\ \% = \frac{(Path_{oh})}{(Std_{hdr} + Path_{oh} + Pl)}$$



Example: Transport Backhaul Path overhead

- Services continuously evolve using heterogeneous sets of protocols and functions
- Different methods of tunneling, load balancing, congestion control and Quality of Service, firewalls and intrusion detection systems.

Observations

1. With Segment routing TE (MPLS or V6) overheads go up with No of hops.
2. General overhead for 4-byte packet is very high (with Std TCP/IP hdr included)
3. Segment routing TE vs MPLS has control plane complexity trade offs.

Packet Formats - Services

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Missing Information for the Networks and Routers

Throughput for the media application

Latency for the underlying application requires

Other resource requirements the application may have – level of security, path, functions

Additional Primitives: Tracking, Receipt notification, Cost & Energy

Compound Service	Criteria	Use cases	Time scales*
Qualitative Service	Conditional to network state	High throughput multimedia such as Holographic applications	~ 40 ms
Holographic Type Communications	Coordinated, time dependence and high bandwidth	High bandwidth requirements, different encoding for teleconferencing vs 3D medical imaging	~30 ms
Digital Teleportation	Coordinated, synchronized,	Digital replicated live-environment	~30ms
Tactile communications	Time dependence and reliability (zero packet-loss)	Variable encodings of haptics, optionally high bandwidth requirements, fast responses.	< 10ms

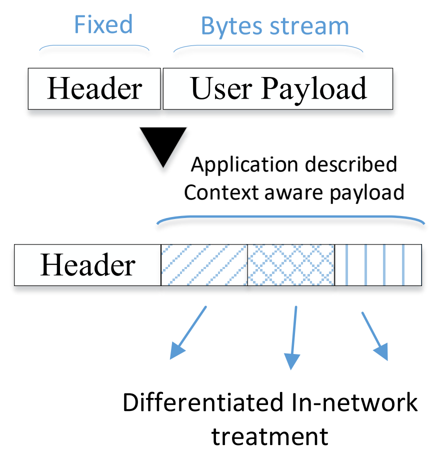
Packet Formats - Addresses

- There is no ideal fixed-length or format of an address
  - Smaller networks should not have to carry overheads of larger fixed-sizes
  - Smaller payloads + smaller headers improve energy efficiency.
  - Custom address structures protect networks from well-known vulnerabilities
  - Global Reachability ≠ Global address space
  - Symmetrical addresses – should source/destination addresses must belong to same address family?

Diverse range of types of end hosts. E.g., IoT devices need energy efficiency, chunks of distributed media content needs descriptive addressing.

Packet Format – Qualitative Payload

- Network is blind to Semantics associated with the packets
  - Payload itself is treated as raw, uninterpreted, unchangeable.
  - The tolerance to packet losses in the volumetric media applications is extremely low, and the problem
- Means to support Qualitative Communications
  - Network perceives the semantics of the packet payload, e.g., boundary, importance variation, relationship among different parts in the packet;
  - The unit of action taken by the network does not need to be on the entire packet.
  - E.g. achieving near linear throughput, if packet losses due to congestion are eliminated.



Basic Structure of the New IP Packet

From these observations we conclude that the evolution of the network layer itself is necessary for innovations in the data plane and forwarding planes technologies.

Our proposal is very simple

1. It describes a packet format that evolves independently across several dimensions.
2. It maintains network layer as universal – accommodating all types of communications
3. Is backward compatible
4. The specification for the new packet format:

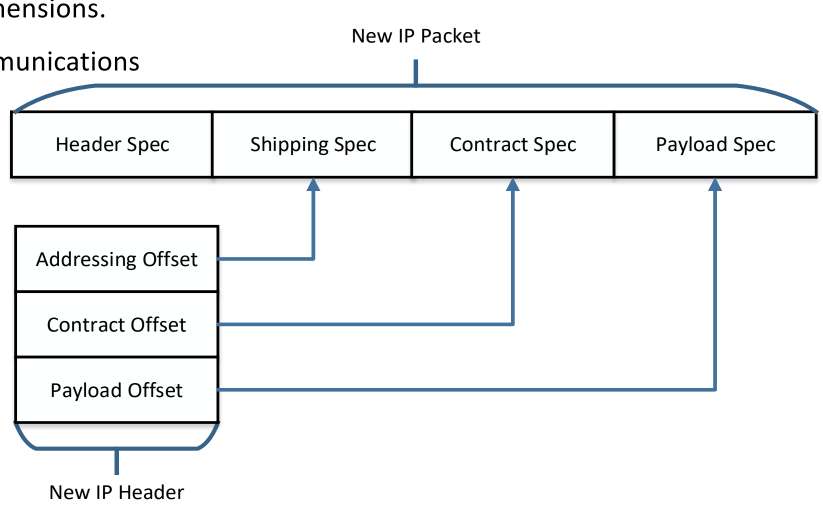
    The structure is defined using the "Header Spec"

    Shipping (Addressing and Reachability functions)

    Contract Spec (service specific goals)

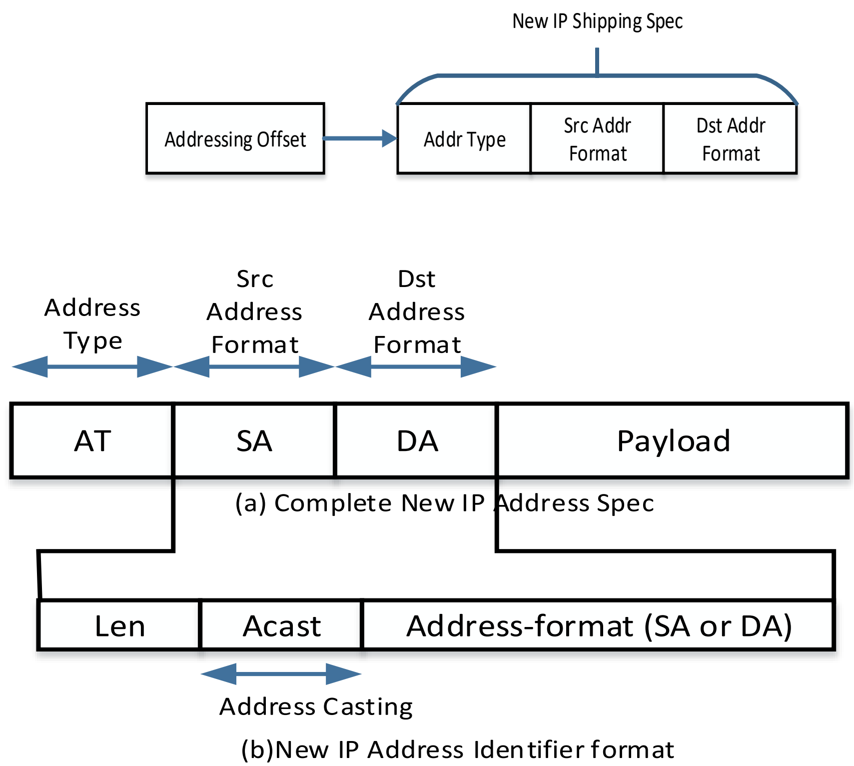
    Payload Spec (packetization of user payload)

Header Spec is a very small header with offsets to other parts of the structure



New IP Shipping Spec

- key features: (a) flexible address format scheme, (b) backward compatible, and (c) hybrid addressing.
- Hybrid addressing is supported by allowing both source and destination formats to be specified independently
- Address Type (AT) field is the first field routers or nodes look at.
  - For backward compatibility, AT will use reserved mnemonics such as IPV4, V6, MPLS etc. Then the remaining packet is the classic IP/MPLS.
  - Otherwise AT can reflect combination of source and destination address types.
- Address-cast (Acast) describes communication patterns such as unicast, groupcast, multicast, coordinated-cast etc.



New IP Shipping Spec Examples

- Backward Compatibility

  For example, IPv4 and MPLS in New IP

  - In these cases AT will use reserved mnemonics such as IPV4, V6, MPLS etc. Then the remaining packet is the classic IP/MPLS.

<MPLS>	MPLS Header	IPv4 Header	Payload

Address Type

<IP>	IPv4 Header	Payload

- Asymmetric Addresses in Shipping Spec

<IPv4,IPv6>	4	UC	IPv4 Address	16	UC	IPv6 Address	Len	Payload

New IP- Contract

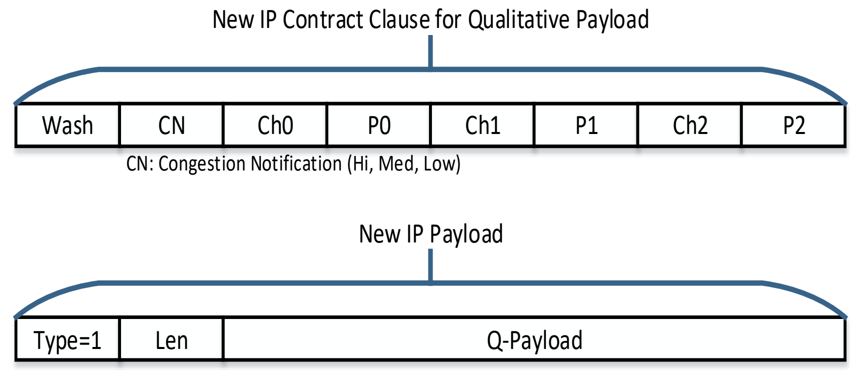
- A contract is service specification of a service associated with the packet.
- It can be optional. Then the contract offset will be set as NA.
- Event and Conditions specify when Action take place. – tremendous potential for the research and development of data plane technologies
- It can be composed of one or more clauses; Each clause will be a high-level data plane action. For example, in order to support Ultra-Reliable Low Latency (uRLLC) in 5G, two contracts C1 and C2 can be used.
  - C1 contract clause indicates the BoundedLatency action, and clause
  - C2 has action NoPktLoss i.e., the low latency and reliability are to be met.

```
<Contract> := <Contract clause>
        | <Contract clause> AND <Contract>

<Contract clause> := <Contract ECA>
        | <Contract ECA> OR <Contract clause>

<Contract ECA> := <Event, Condition, Action>
        | <Metadata>
        | <Event, Condition, Action> <Metadata>
        | <Action>
        | <Action><Metadata>
```

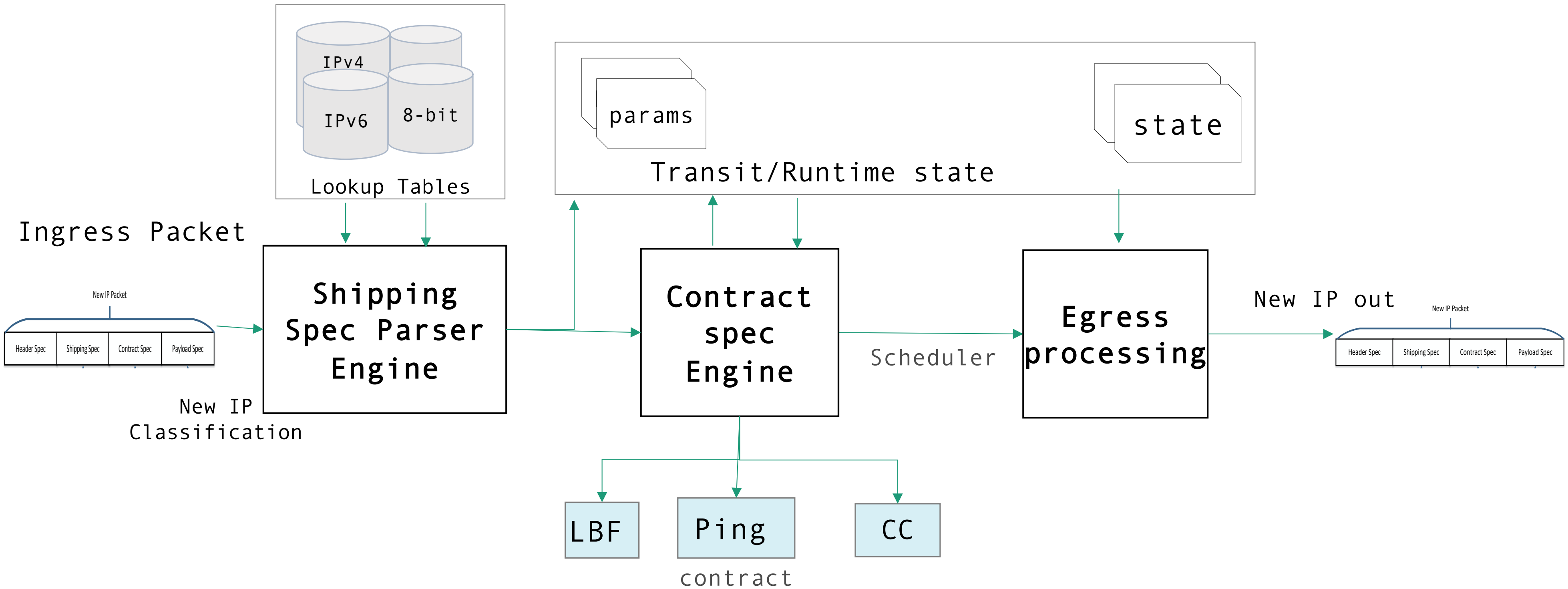
```
<Action> := <BoundedLatency>
       | <OnTimeDelivery> | <Coordinate>
       | <NoPktLoss> | <PktMonitor>
       | <ReportInsuringParty>
       | […]
```

New IP – Q-Payload

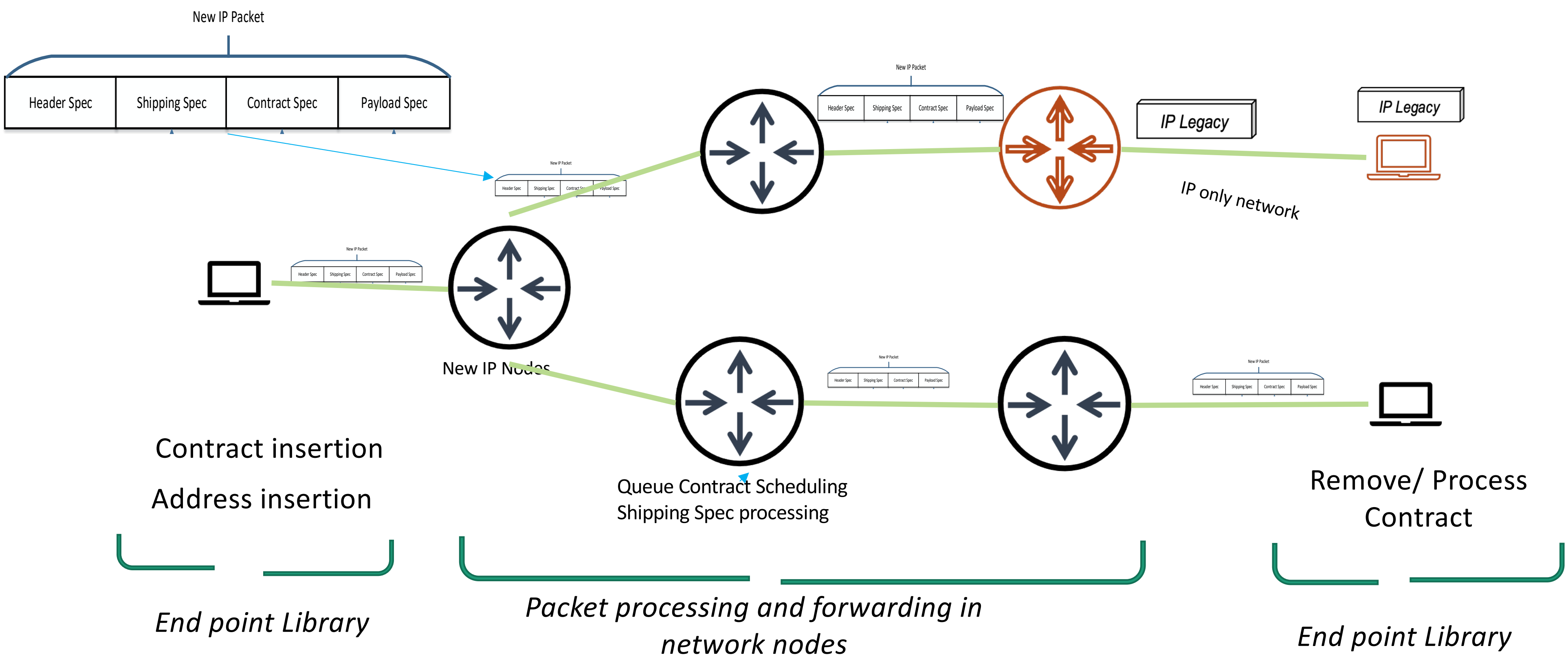
- The Payload specification is used to support qualitative payload.
- The Type field indicates whether it is a traditional or Q-payload.
- Q-payload itself needs support from Contract Specification to describe actions on Q-payload.
- For example
  - Wash: a generic operation to arbitrarily or selectively remove the bits/bytes inside the packet payload. For example, remove every 8th bit, remove every fifth byte, etc.
  - Repair and Recover: can salvage lost portions from the residual payload based on context present along with the action.
  - Enrich: The re-insert lost portions with locally cached chunks when the network condition improve.



New IP Conceptual Packet Processing Pipeline



Developing New IP Forwarding Platform



Design Principles

- Since New IP Packet format is new and carries a lot of intelligence, it was important to understand those processing overheads from
  - Design and over all implementation framework perspective
  - From performance and ease of development
- More Focus on packet processing/ not hardware
- Emphasis on Fast and flexible test and verification environment
- Our Platform choice – Linux with network name spaces
  - Reuse not reinvent
  - Linux gave us a very stable environment and a plethora of tools and features.
  - Very short learning curve
- NeST provides meets all the above design considerations

References

- Technical Specification: "[New Services and Capabilities for Network 2030: Description, Technical Gap and Performance Target Analysis]" (October 2019)
- Technical Report: "[Network 2030 - Gap Analysis of Network 2030 New Services, Capabilities and Use cases]" (June 2020)
- R. Li, K. Makhijani and L. Dong, "New IP: A Data Packet Framework to Evolve the Internet : Invited Paper," 2020 IEEE 21st International Conference on High Performance Switching and Routing (HPSR), 2020, pp. 1-8, doi: 10.1109/HPSR48589.2020.9098996.
- K. Makhijani and L. Dong, "Asymmetric Addressing Structures in Limited Domain Networks," 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), 2021, pp. 1-7, doi: 10.1109/HPSR52026.2021.9481811.
- [IEEE GLOBECOM 2019: Richard Li, "IP: GOING BEYOND THE LIMITS OF THE INTERNET"]
- 6G RESEARCH VISIONS, NO. 6
- White paper on 6G networking, Led by Tarik Taleb. https://www.6gchannel.com/items/6g-white-paper-networking/

Network Stack Tester (NeST)

New IP Sandbox Test Environment

What is NeST?

- A Python package to emulate networks
- Uses network namespaces to simplify the process of setting up networks
- Provides intuitive APIs to
    - Build a virtual network topology
    - Run experiments on the virtual network topology
    - Collect statistics
    - Plot results

Why NeST?

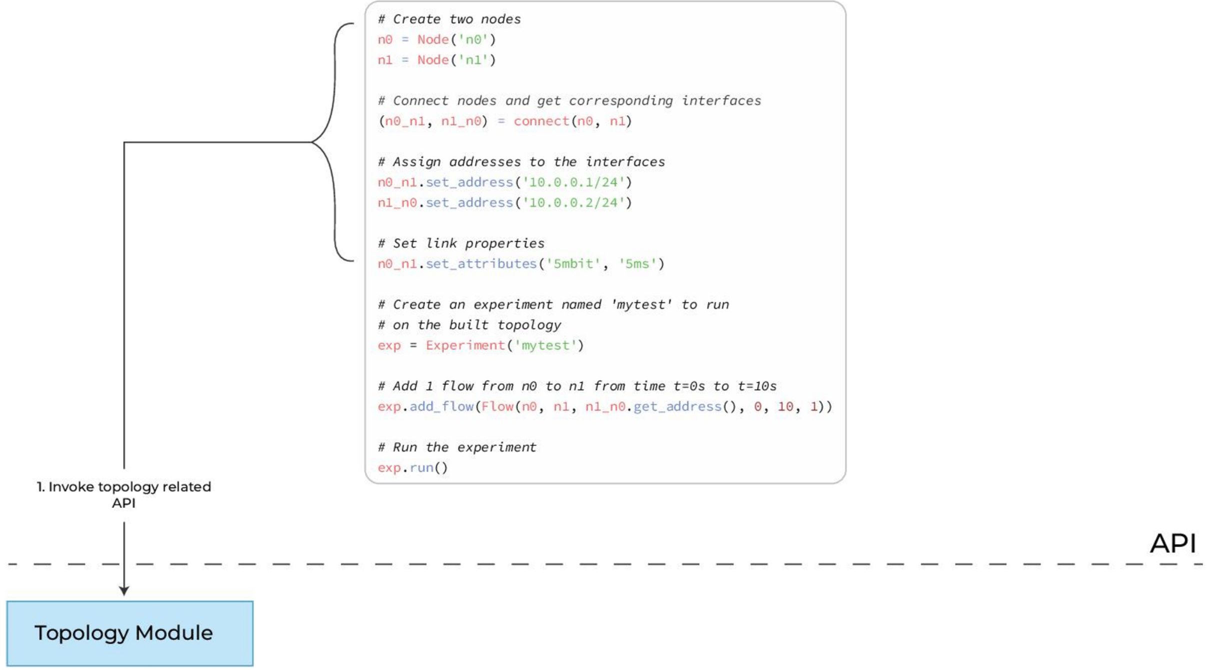
- Simplifies the process to reproduce network experiments
- Less physical resources, less error prone and less prerequisites
- Multiple instances of the same network topology can co-exist, and different experiments can be run in parallel on every instance
  - NeST assigns random names to the network namespaces
  - Cleans up the environment (deletes network namespaces) on termination
- Open source tool released under GPLv2 License

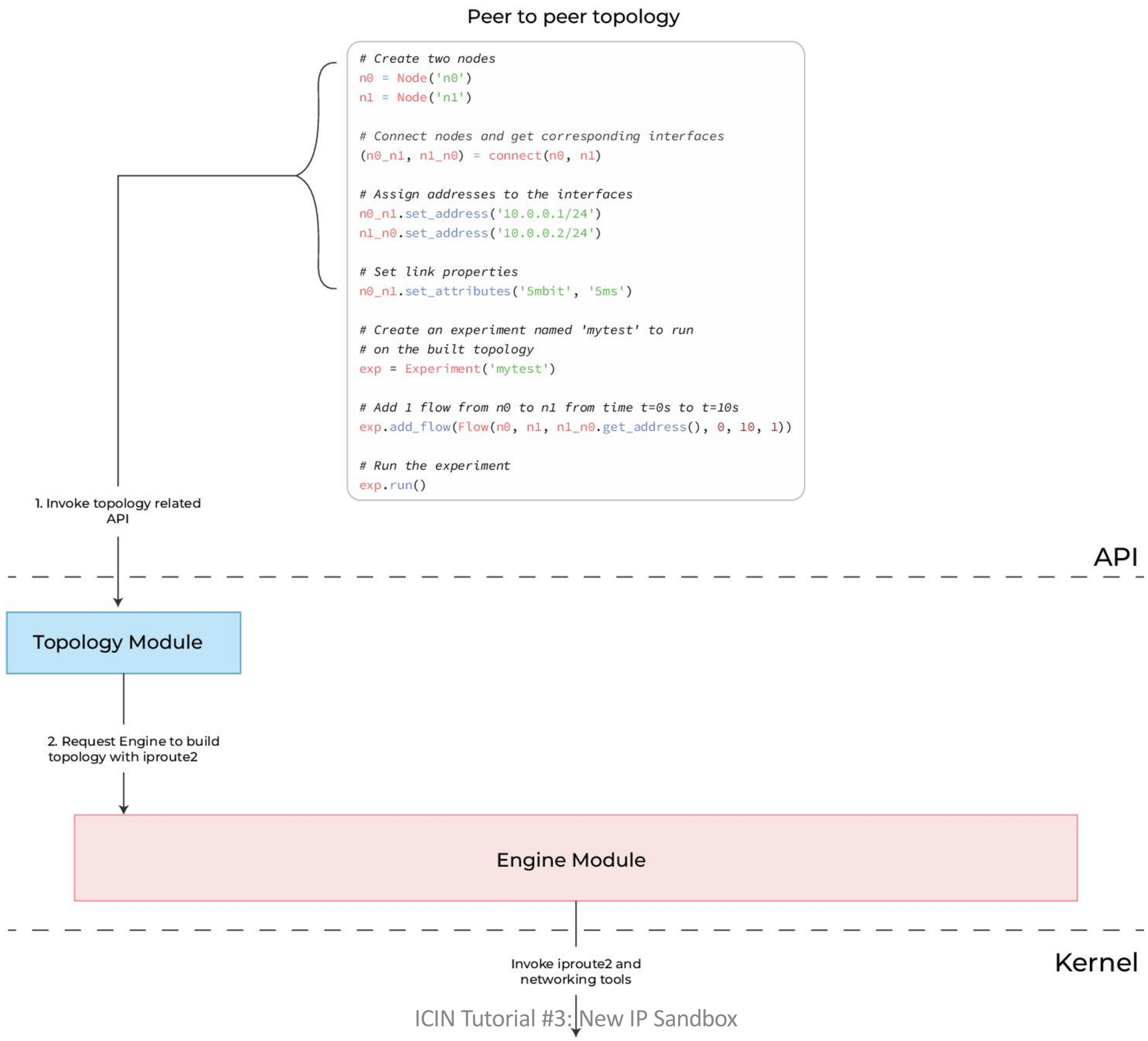
Peer to peer topology

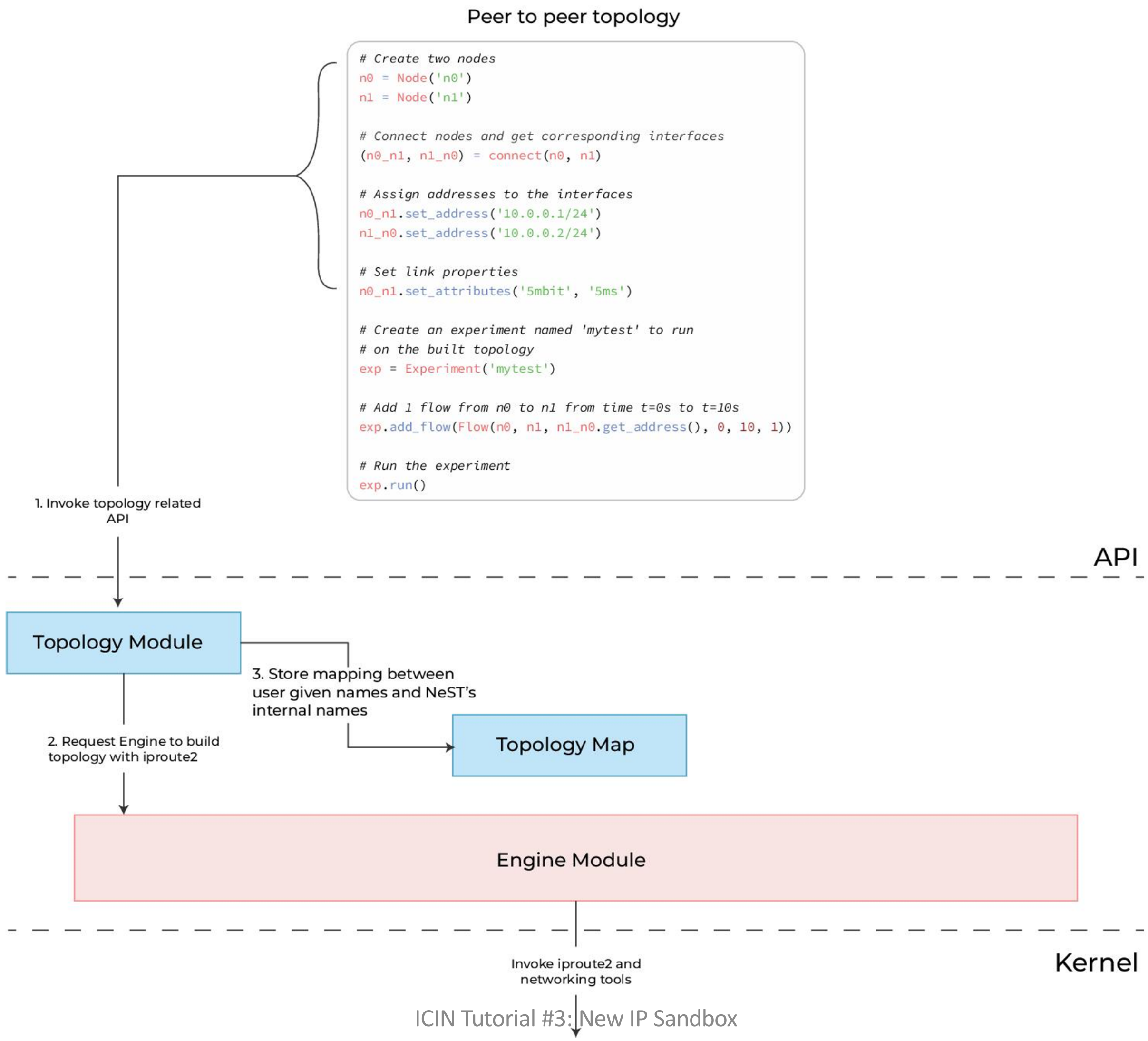
```
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')

# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)

# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')

# Set link properties
n0_n1.set_attributes('5mbit', '5ms')

# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')

# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))

# Run the experiment
exp.run()
```

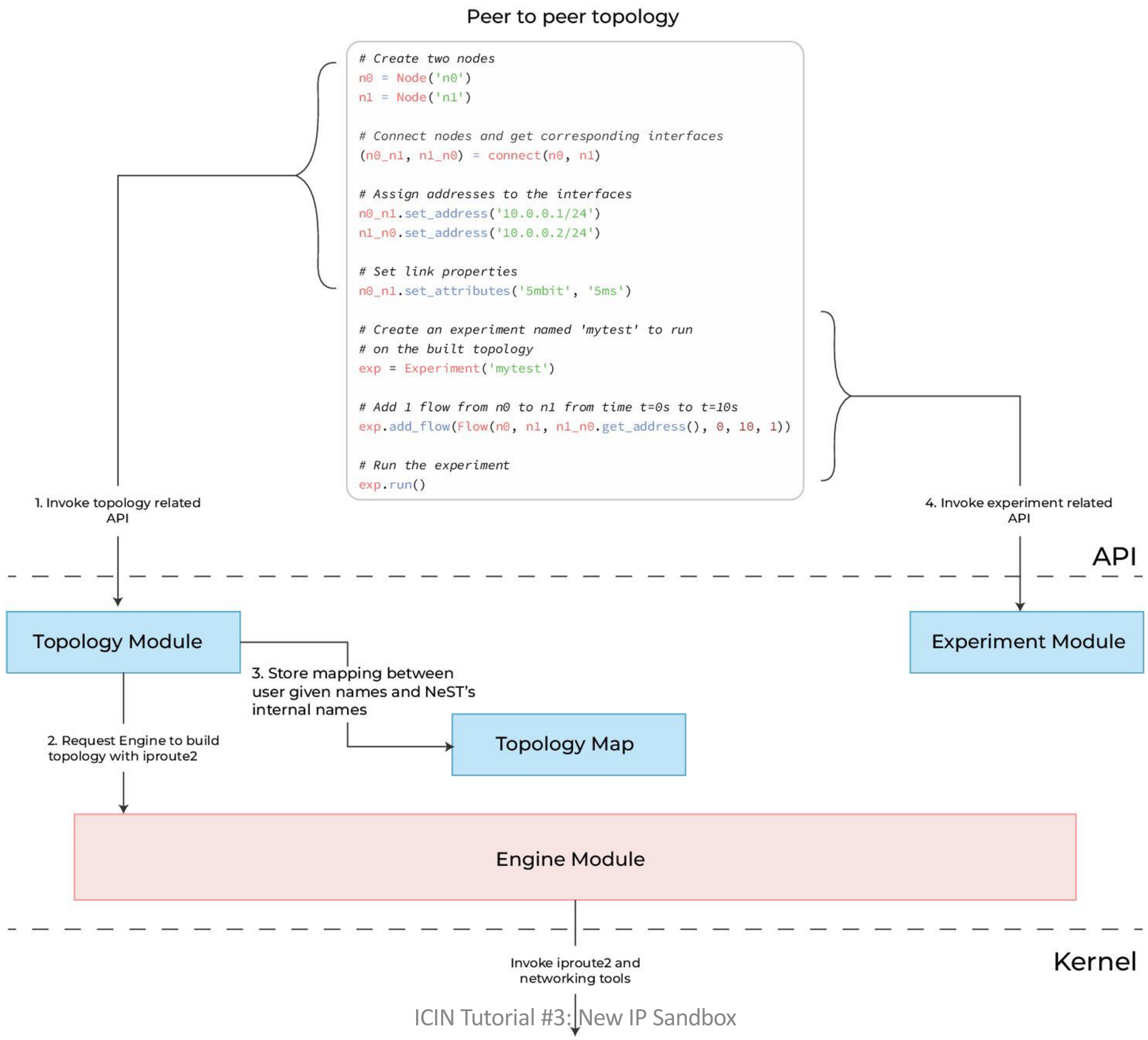
Peer to peer topology

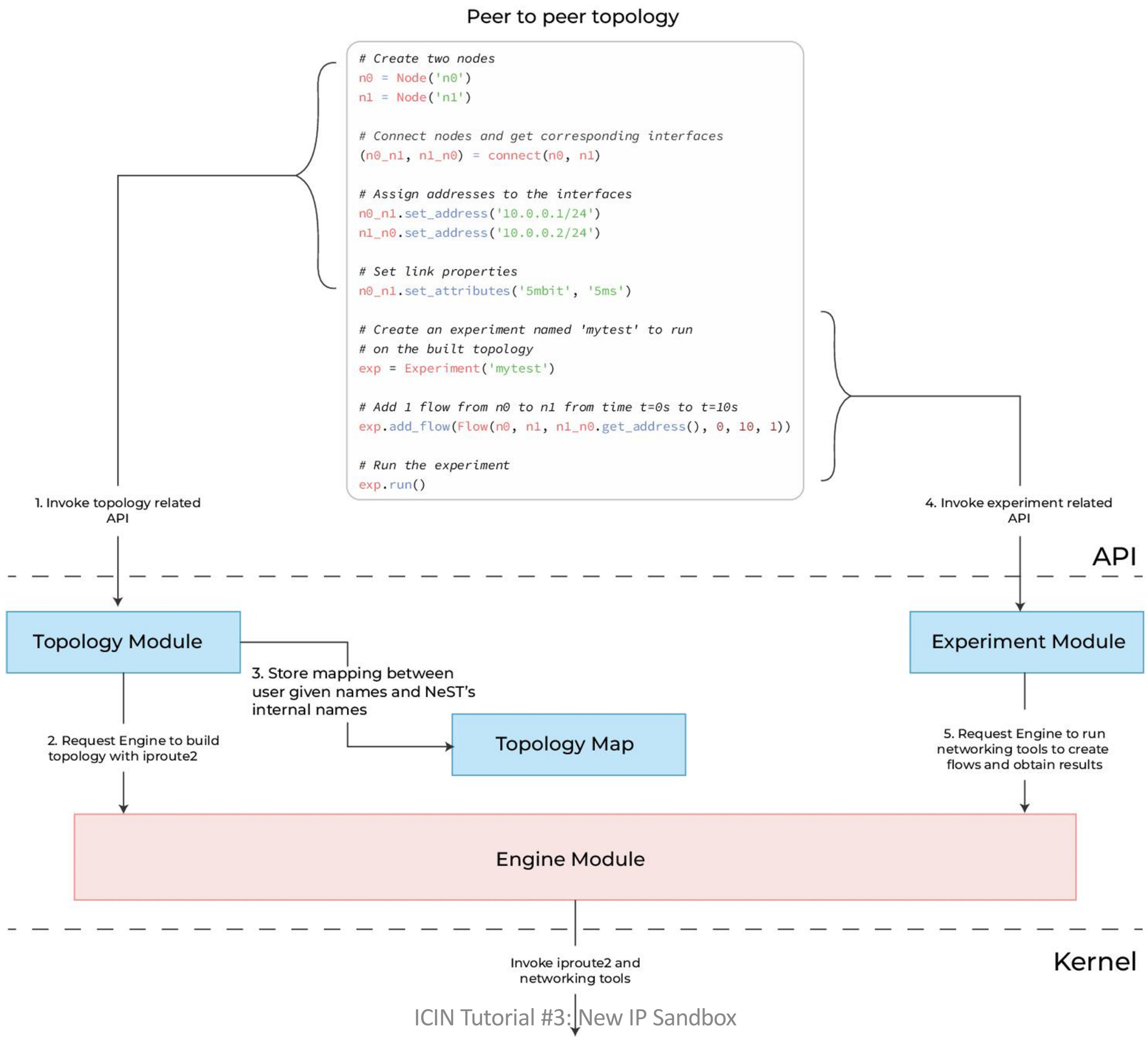
```python
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')

# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)

# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')

# Set link properties
n0_n1.set_attributes('5mbit', '5ms')

# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')

# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))

# Run the experiment
exp.run()
```

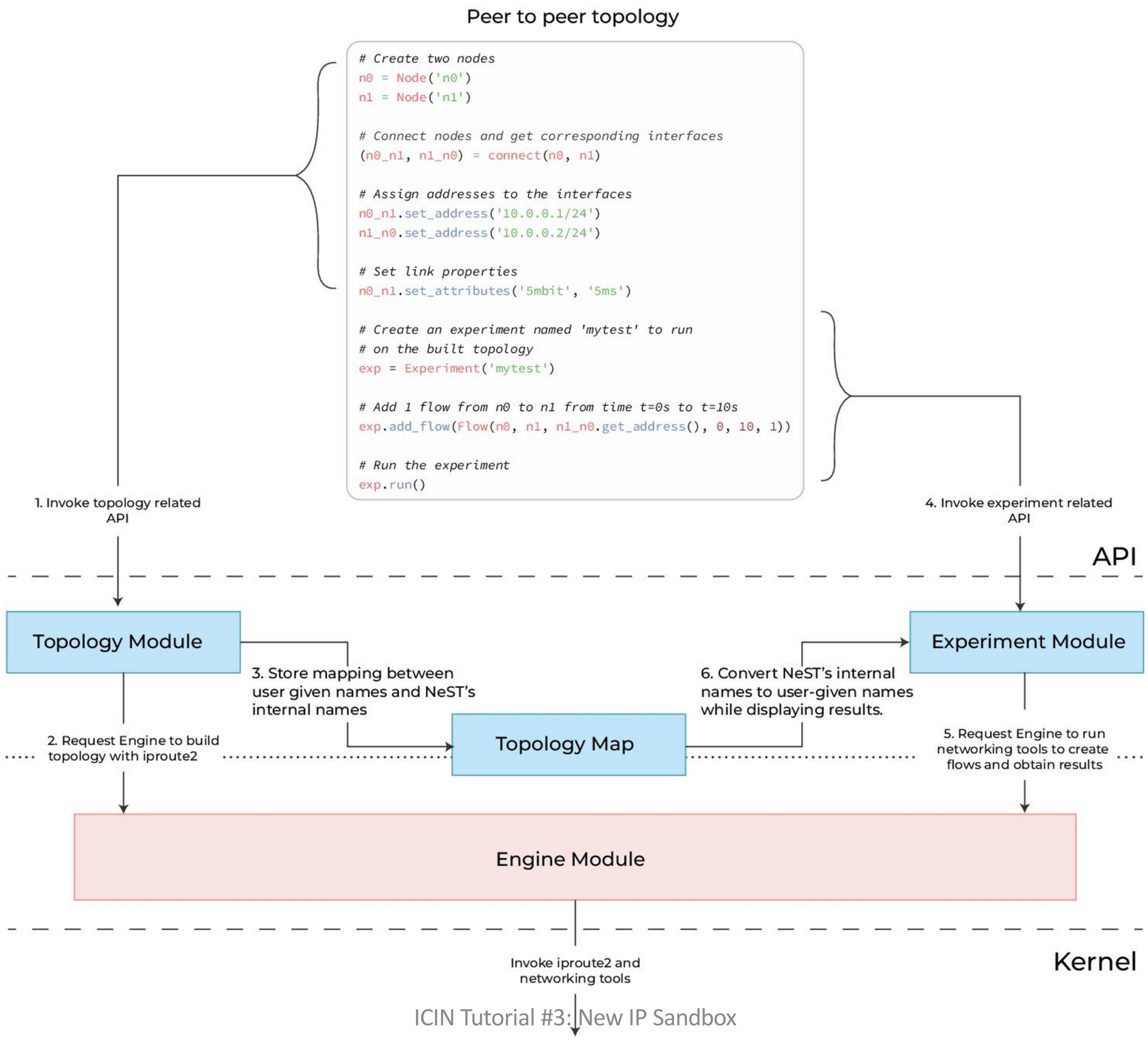



Peer to peer topology
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')
# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)
# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')
# Set link properties
n0_n1.set_attributes('5mbit', '5ms')
# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')
# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))
# Run the experiment
exp.run()
1. Invoke topology related API
API
Topology Module
2. Request Engine to build topology with iproute2
Engine Module
Kernel
Invoke iproute2 and networking tools


Peer to peer topology
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')
# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)
# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')
# Set link properties
n0_n1.set_attributes('5mbit', '5ms')
# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')
# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))
# Run the experiment
exp.run()
1. Invoke topology related API
API
Topology Module
3. Store mapping between user given names and NeST's internal names
Topology Map
2. Request Engine to build topology with iproute2
Engine Module
Kernel
Invoke iproute2 and networking tools


Peer to peer topology
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')
# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)
# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')
# Set link properties
n0_n1.set_attributes('5mbit', '5ms')
# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')
# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))
# Run the experiment
exp.run()
1. Invoke topology related API
4. Invoke experiment related API
API
Topology Module
Experiment Module
3. Store mapping between user given names and NeST's internal names
Topology Map
2. Request Engine to build topology with iproute2
Engine Module
Kernel
Invoke iproute2 and networking tools


Peer to peer topology
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')
# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)
# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')
# Set link properties
n0_n1.set_attributes('5mbit', '5ms')
# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')
# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))
# Run the experiment
exp.run()
1. Invoke topology related API
4. Invoke experiment related API
API
Topology Module
Experiment Module
3. Store mapping between user given names and NeST's internal names
Topology Map
2. Request Engine to build topology with iproute2
5. Request Engine to run networking tools to create flows and obtain results
Engine Module
Invoke iproute2 and networking tools
Kernel


Peer to peer topology
# Create two nodes
n0 = Node('n0')
n1 = Node('n1')
# Connect nodes and get corresponding interfaces
(n0_n1, n1_n0) = connect(n0, n1)
# Assign addresses to the interfaces
n0_n1.set_address('10.0.0.1/24')
n1_n0.set_address('10.0.0.2/24')
# Set link properties
n0_n1.set_attributes('5mbit', '5ms')
# Create an experiment named 'mytest' to run
# on the built topology
exp = Experiment('mytest')
# Add 1 flow from n0 to n1 from time t=0s to t=10s
exp.add_flow(Flow(n0, n1, n1_n0.get_address(), 0, 10, 1))
# Run the experiment
exp.run()
1. Invoke topology related API
4. Invoke experiment related API
API
Topology Module
Experiment Module
3. Store mapping between user given names and NeST's internal names
6. Convert NeST's internal names to user-given names while displaying results.
Topology Map
2. Request Engine to build topology with iproute2
5. Request Engine to run networking tools to create flows and obtain results
Engine Module
Invoke iproute2 and networking tools
Kernel

PoC of New IP

Demonstration of the use cases

Bhaskar Kataria
Rohit M. P
Leslie Monis
Mohit P. Tahiliani
NITK Surathkal

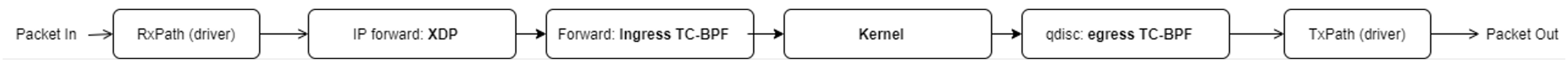
Kiran Makhijani
Futurewei, USA

Goals

Implement a programmable data plane for New IP packet processing

- **Address customization**: applications and routers should be able to forward packets between hosts with different address formats.
- **Design an end-to-end model to meet service delivery guarantees**: routers should be able to process various in-network New IP contracts as described by the applications.
- **Rapid experimentation** of the New IP components should be possible.

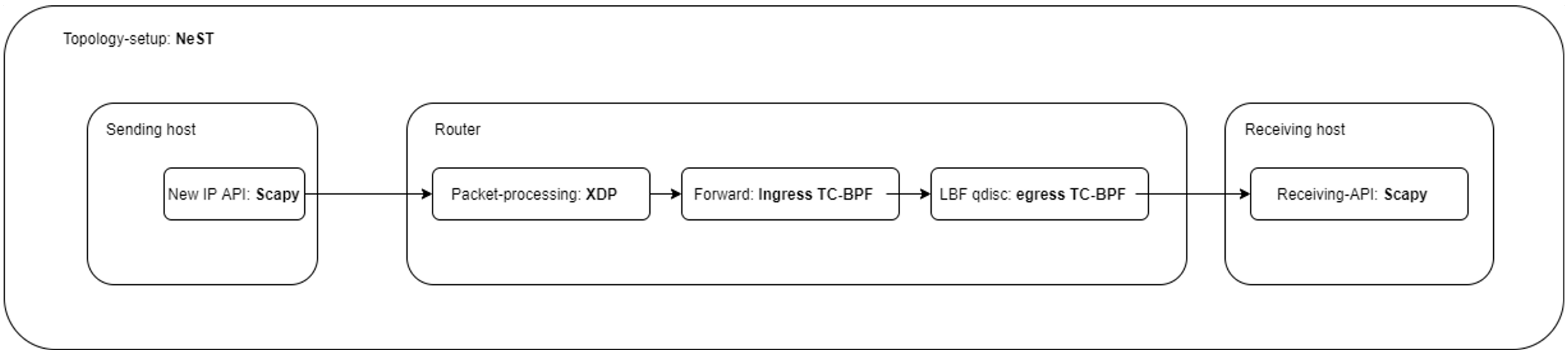
IP packet flow in Linux



Main Components

1. Topology setup - Network Stack Tester (NeST)
2. Scapy - Host side API
3. eXpress Data Path XDP - shipping and contract processing
4. Traffic Control-BPF (TC) - LBF queue discipline

New IP packet flow



Topology setup: NeST

- NeST is a python3 package designed for emulating real-world networks
- Uses Network namespaces to create the topology
- Makes testing easier

For this project, NeST is used to:

- Assign IP addresses to the nodes
- Populate the routing tables
- Form the packet by providing necessary information like MAC addresses

Host side API: Scapy

- Python library for powerful interactive packet manipulation
- Helps forming custom packet formats
- Uses Raw sockets internally

For this project, Scapy is used to:

- Craft and send the packets from the virtual interface setup by NeST
- Sniff and decode the packet at the receiver

XDP

- XDP is an eBPF based high performance data path
- Merged in the Linux kernel
- Provides packet processing at the lowest point in the software stack

For this project, XDP is used to:

- Update the Ethernet header in the packet
- Identify the interface from which the packet is to be sent using kernel routing tables or BPF maps
- Store the exit interface index (ifindex) in metadata of the packet
- Pass the packet on to the TC BPF hook

TC-BPF

- BPF programs attached to the traffic control (tc) ingress and egress hook
- Runs after the networking stack has done the initial processing of the packet
- Has access to the metadata of the packet

For this project, TC-BPF is used to:

- Add our queueing discipline (we cannot have queue discipline with just XDP)
- Read the ifindex from metadata at ingress hook
- Redirect the packet to the egress hook of the interface associated with ifindex
- Run our queue discipline on the egress hook

Latency Based Forwarding (LBF)

- Designed by Toerless Eckert (toerless.eckert@futurewei.com), Alexander Clemm (alex@futurewei.com)
- Delivers packets within bounded latency (both minimum and maximum)
- Uses number of hops to determine how much time the packet should wait at a node
- Experienced delay stored in the packet so no need of clock synchronization
- LBF contract packet includes Minimum delay, Maximum delay, experienced delay, fib_todelay, fib_tohops
- Users provide Minimum and Maximum delay
- Implemented as Linux Kernel module
- Can be loaded using TC

Principles

- Ideally, the packet should spend number of hops/minimum delay (fib_tohops/min_delay) at each hop
- Time to be spent at each hop = (minimum delay - delay experienced)/number of hops left
- If experienced delay greater than maximum delay then drop the packet
- Packet is dropped if the number of hops exceed the expected number of hops

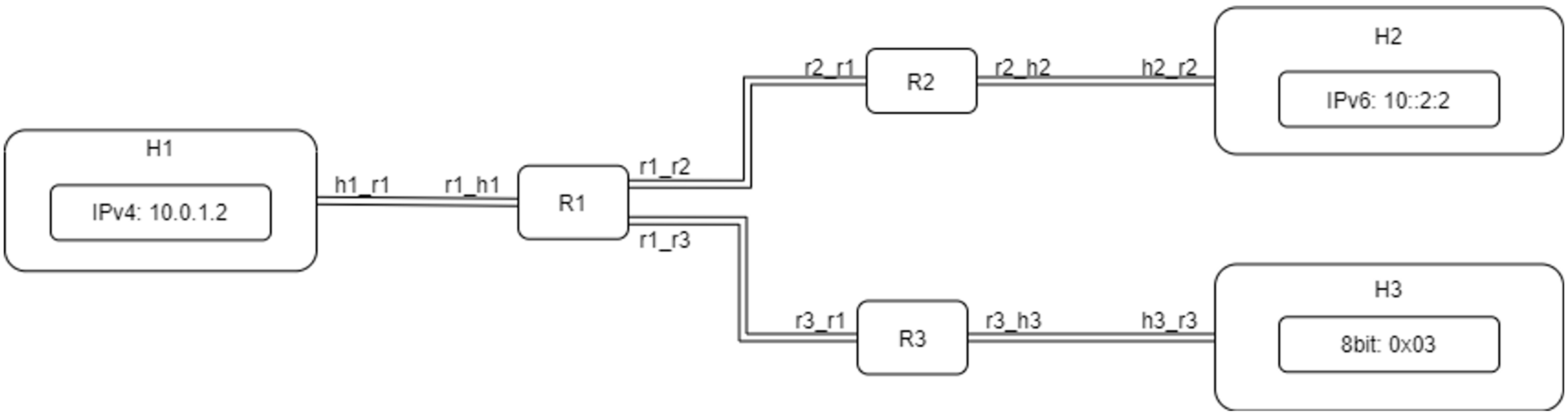
Example 1

- 4 nodes (0, 1, 2, 3)
- number of hops = 3
- minimum delay = 300ms
- no queueing delay
- At node 0: packet delayed by LBF = (minimum delay - experienced delay)/number of hops = (300 - 0) / 3 = 100ms
- At node 1: (300 - 100) / 2 = 100ms
- At node 2: (300 - 200) / 1 = 100ms

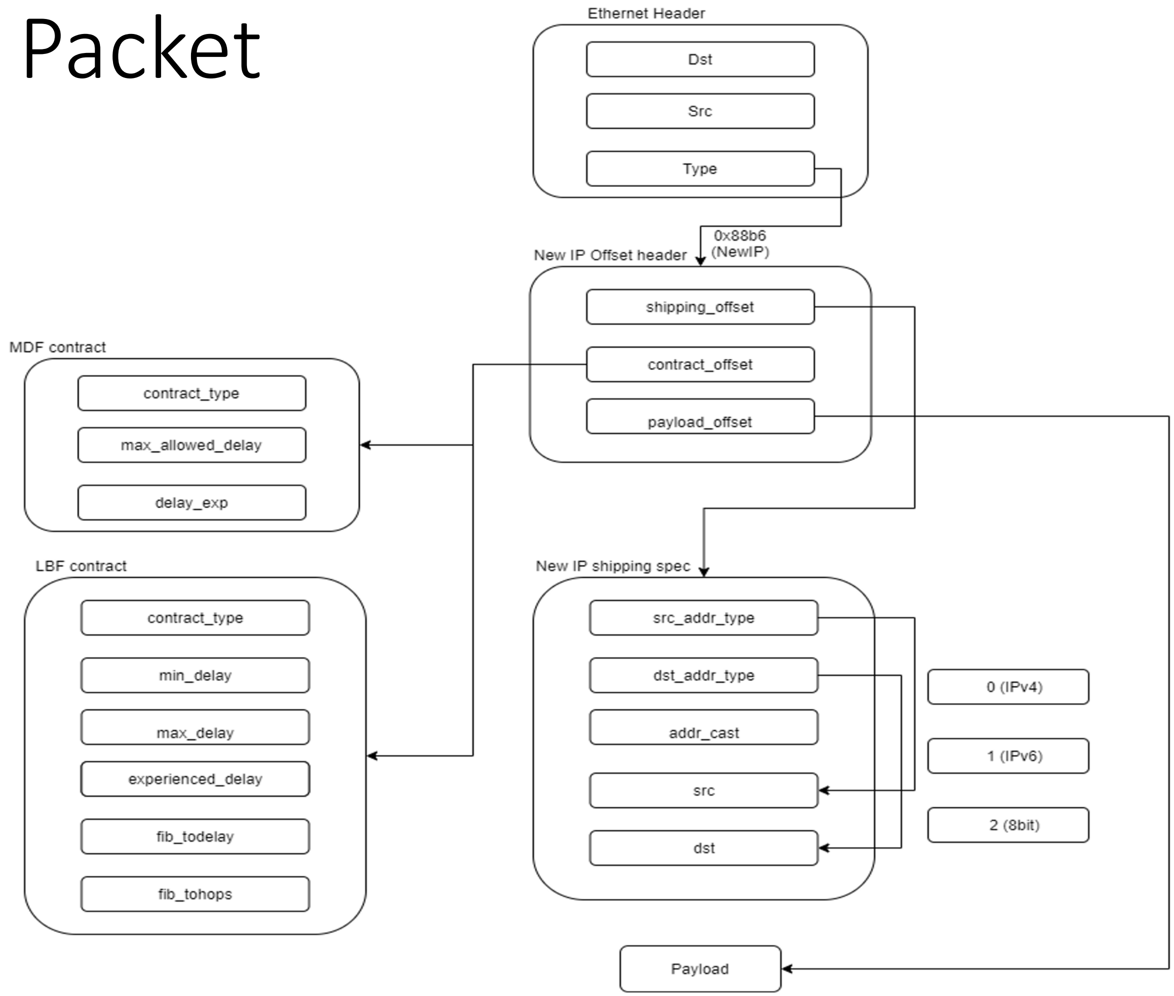
Example 2

- 4 nodes (0, 1, 2, 3)
- number of hops = 3
- minimum delay = 300ms
- Queueing delay = 50ms at every queue
- At node 0: packet delayed by LBF = (minimum delay - experienced delay)/number of hops = (300 - 50) / 3 = 83ms
- At node 1: (300 - (83 + 50 +50)) / 2 = 58.5ms
- At node 2: (300 - (83.5 + 50 + 50 + 50 + 58.5) / 1 = 8ms

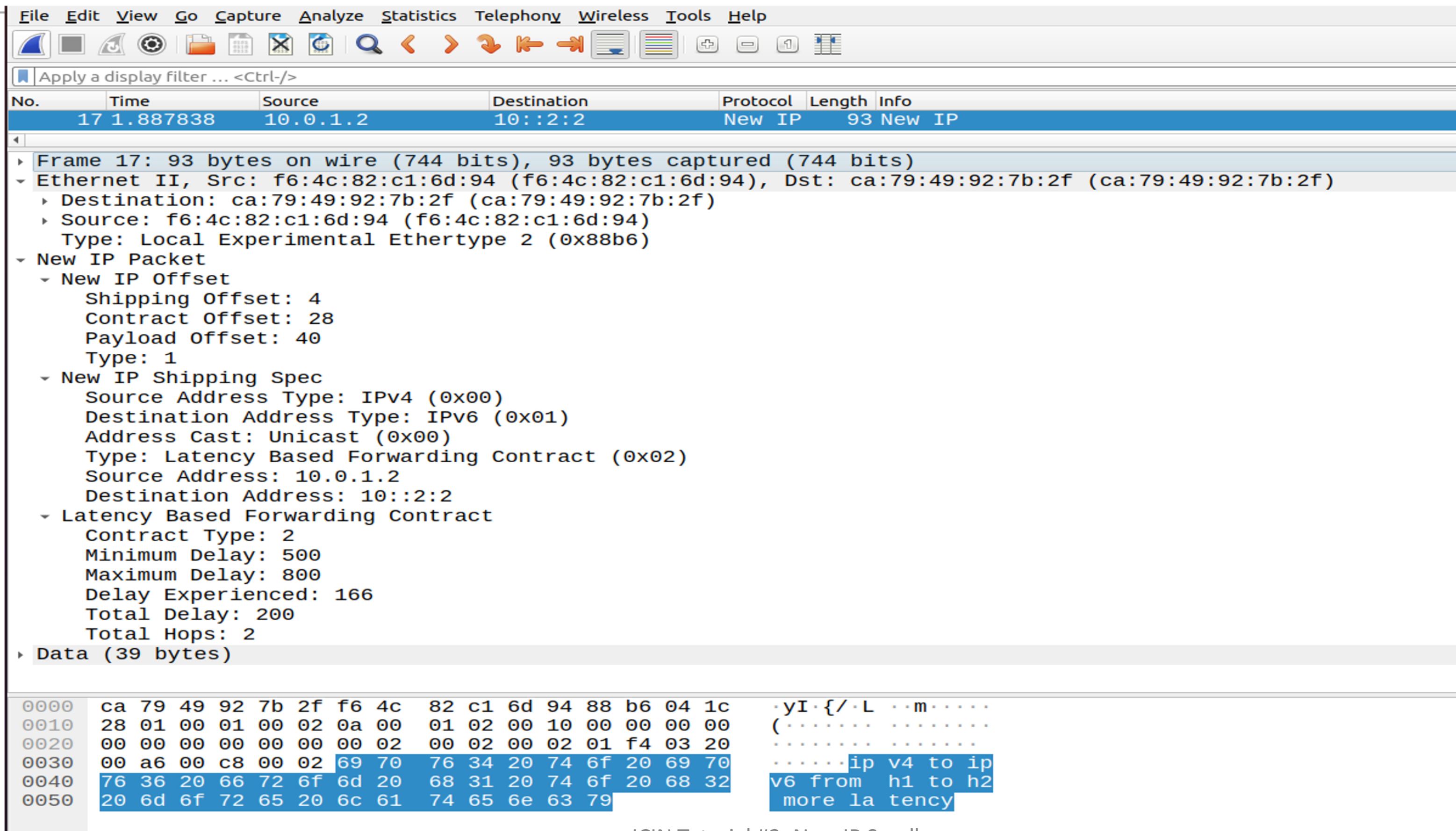
Topology Setup - NeST



New IP Packet



New IP Packet in Wireshark



API for Crafting a New IP packet

```
sender_obj = Sender()

sender_obj.make_packet(src_addr_type="ipv4",
src_addr="10.0.1.2", dst_addr_type="ipv6",
dst_addr="10::2:2",content="ipv4 to ipv6 from h1 to h2 more
latency")

lbf_contract = LatencyBasedForwarding(min_delay = 500, max_delay
= 800, fib_todelay = 0, fib_tohops = 3)

sender_obj.set_contract ([lbf_contract])
sender_obj.send_packet(iface='h1_r1', show_pkt=True)
```

References

XDP
- https://www.iovisor.org/technology/xdp
- https://github.com/xdp-project/xdp-tutorial

NeST
- https://nest.nitk.ac.in/docs/

Scapy
- https://scapy.readthedocs.io/en/latest/usage.html

LBF
- High-Precision Latency Forwarding over Packet-Programmable Networks : https://ieeexplore.ieee.org/document/9110431

New IP Use cases

Service extensibility on New IP platform

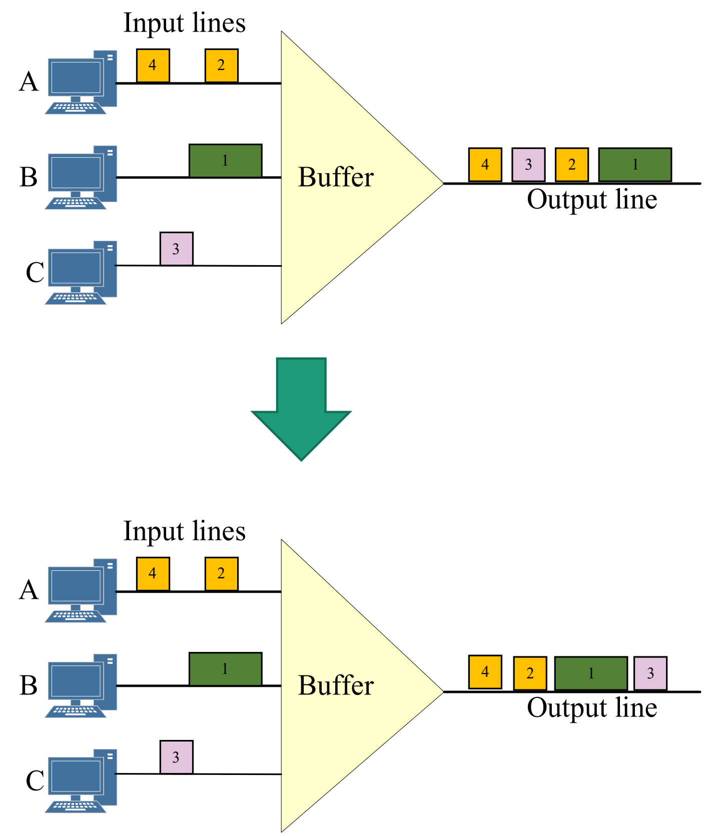
Part II: Some Examples of New IP Use Cases

- Optimal Latency Guarantee for Multiple Concurrent Packets
- Qualitative Communication
- Information Exchange Oriented Vehicle Clustering
- And many more

Statistical Multiplexing to Computational Multiplexing

- Statistical multiplexing does not work at a packet level. They do not distinguish among packets on whether they are latency-sensitive or not, thus do not treat the latency-sensitive packets differently.

- Compared to statistical multiplexing in the current Internet, a computational multiplexing concept is proposed in New IP.
  - For a latency-sensitive packet, the precise location in the outgoing queue based on its time constraint can be computed, such that the scheduling order of all latency-sensitive packets can be determined.
  - Computational scheduling takes into consideration of per-packet timing data at a finer granularity as compared to flow-level granularity.
  - With computational multiplexing, simultaneously arriving packets on same output ports are scheduled deliberately in order to satisfy their deadlines if they are specified in the packets.

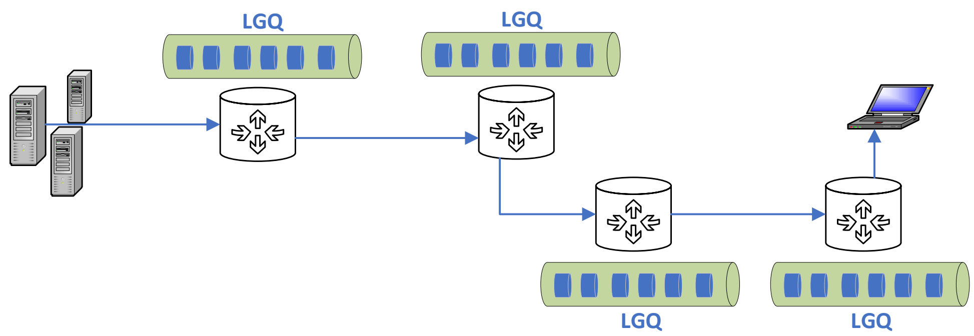


New IP Contract Design

- We take advantage of the Contract Spec that is defined in New IP to inform the intermediate network nodes that a packet forwarding requires end-to-end in-time guarantee. We design the corresponding Contract Clause ($InTimeGuarantee$) in the Contract Spec as follows to achieve the latency guarantee of packet delivery at finest granularity:
  - Action: deadline-aware scheduling, with the optimal algorithm proposed in the following of the paper, the packets specified with $InTimeGuarantee$ are able to meet their corresponding end-to-end deadlines.
  - Metadata includes: (1) budget: it denotes the residual budget before the packet deadline runs out and is considered unsuccessful. (2) hop: it denotes the residual number of hops towards destination. Given the routing path is configured, and the total number of hops between the source and destination is fixed.

Dedicated LGQ for Latency Guaranteed Packets

- At each router, there is a dedicated queue for latency guaranteed packets, called latency guarantee queue (LGQ)
- All packets forwarded by the router with $InTimeGuarantee$ contract clause are put in LGQ. The packets in LGQ have the highest priority to be scheduled compared to other packets without deadline constraints.



Motivation

- From a router’s perspective, it may have more than one packet (multiple packets) that require latency guarantee. The scheduling of those packets matters.
  - If FIFO is adopted, some packets in LGQ may miss the deadline.
  - The time that a packet spent at a router before it is forwarded to the next hop (called as **dwell time**)includes
    - Processing delay: likely fixed (we do not consider in the document)
    - Queueing delay: affected by packet scheduling in LGQ.
    - Transmission delay: affected by the size of the packet, proportional to the packet size.

- The **average dwell time determines averagely how long a packet stays in a router** before its last bit gets transmitted completely.

- Although previously proposed scheduling policy may be able to satisfy the deadline requirements of the packets, they did not consider how transmission delay affects the scheduling.

- A packet scheduling algorithm that adopts the computational multiplexing enabled by New IP could be designed, which considers both the packet transmission delay and deadline requirement, achieve the minimal average dwell time.

Step by Step to Illustrate TDMS

Packets	Deadline	Transmission Time
p1	10	5
p2	14	2
p3	15	1
p4	6	3

Line 2: sort packets by decremental order of transmission time

Packets	Deadline	Transmission Time
p1	10	5
p4	6	3
p2	14	2
p3	15	1

Outer loop 1

TDMS	Deadline	Transmission Time
p2	14	2

Total transmission time of set J_1 is 5+3+2+1=11, the last packet in the TDMS schedule must be the packet with the largest transmission time that meets its deadline, which is p2.

Outer loop 2

TDMS	Deadline	Transmission Time
p1	10	5
p2	14	2

Total transmission time of set J_2 is 5+3+1=9, the last packet in the TDMS schedule must be the packet with the largest transmission time that meets its deadline, which is p1.

Outer loop 3

TDMS	Deadline	Transmission Time
p4	6	3
p1	10	5
p2	14	2

Total transmission time of set J_3 is 2+1=3, the last packet in the TDMS schedule must be the packet with the largest transmission time that meets its deadline, which is p4.

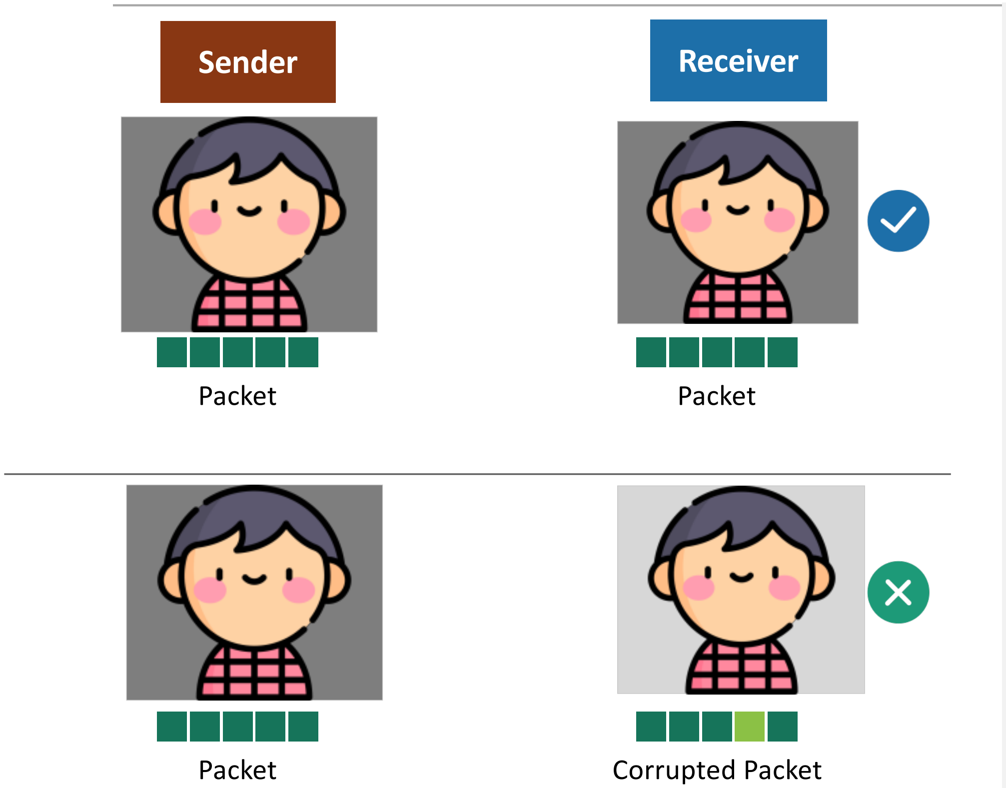
Outer loop 4

TDMS	Deadline	Transmission Time
p3	15	1
p4	6	3
p1	10	5
p2	14	2

Total transmission time of set J_4 is 1. The only left packet is p3, which is put in the very front of the queue. Return TDMS.

TDMS achieves minimal average dwell time, which is (1+4+9+11)/4=25/4 =6.25

Integrity of Packet



Syntax

What is received = What is sent

Every bit and byte has the same significance to routers/switches

Good for

- File/Document Transfer
- Banking, Shopping

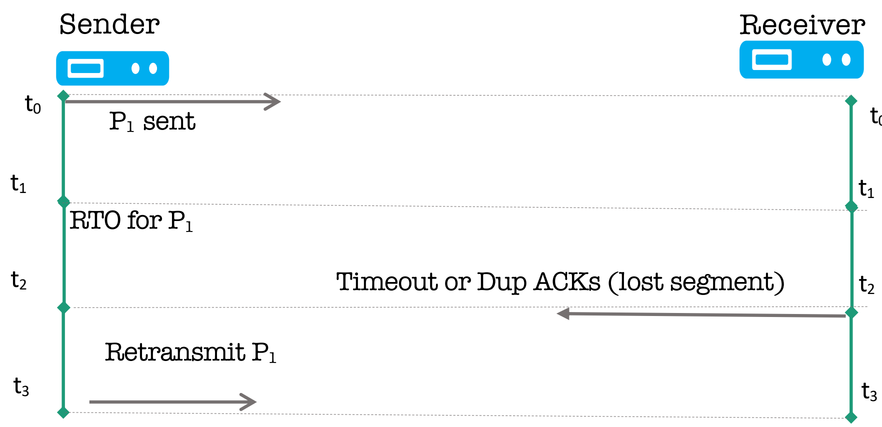
Overkill for some applications

- Video
- Holograms

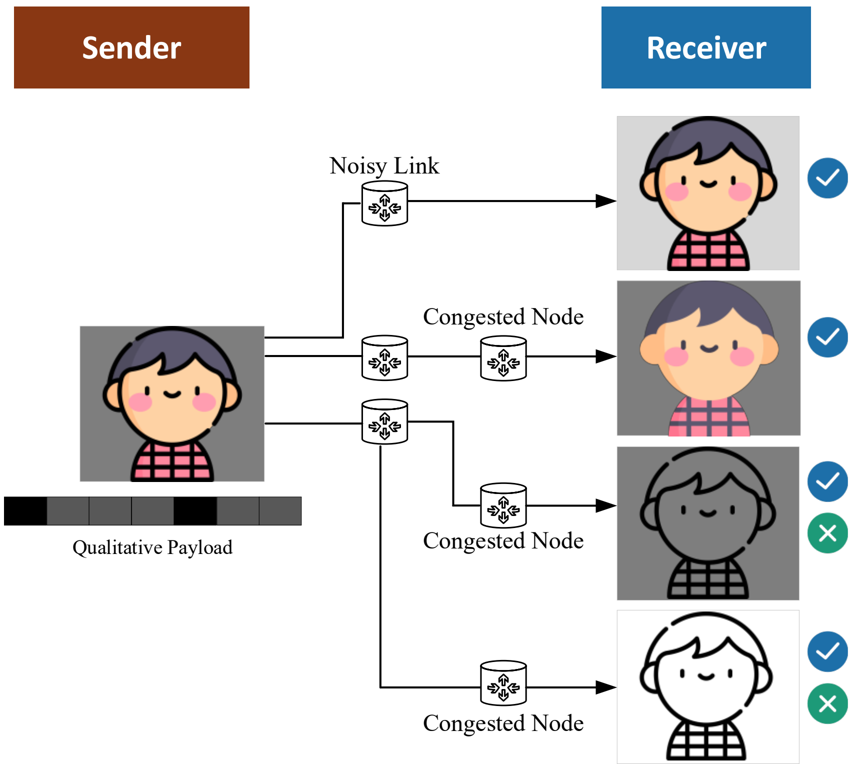
Cost of retransmissions due to packet dropping

- When reliable transport layer protocol is used, packet drops result in the retransmission of the packet.

- Cost of re-transmissions
  - Wastes network resources
  - Reduces the overall throughput,
  - Unpredictable longer delays.



Qualitative Communication: A structure of bits and bytes



Semantics

What is received = What is *maximally meant*

In payload, bits and bytes are not equally significant. Instead, they are different in their entropies

Less significant bits and bytes may be dropped

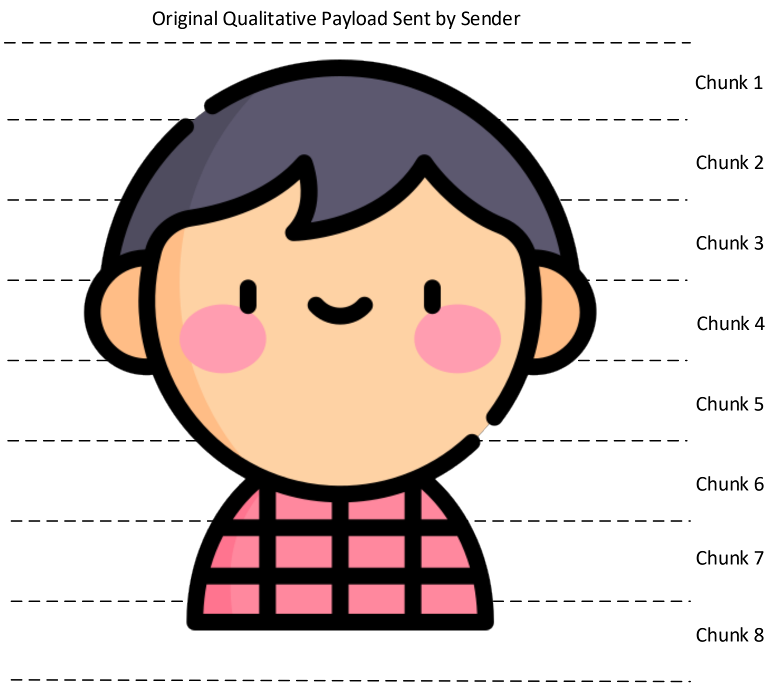
Partial or degraded, yet useful, packets may be repaired and recovered before being rendered

Good for

- Large volume of image-like data
- Holographic type communications
- Media with digital senses
- Disaster Environment

Qualitative Communication: an example for illustration only

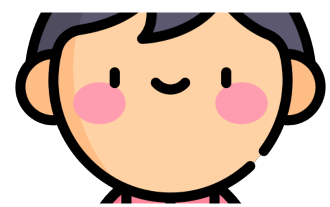
User Data Payload: Divided into 8 chunks.



Contract:

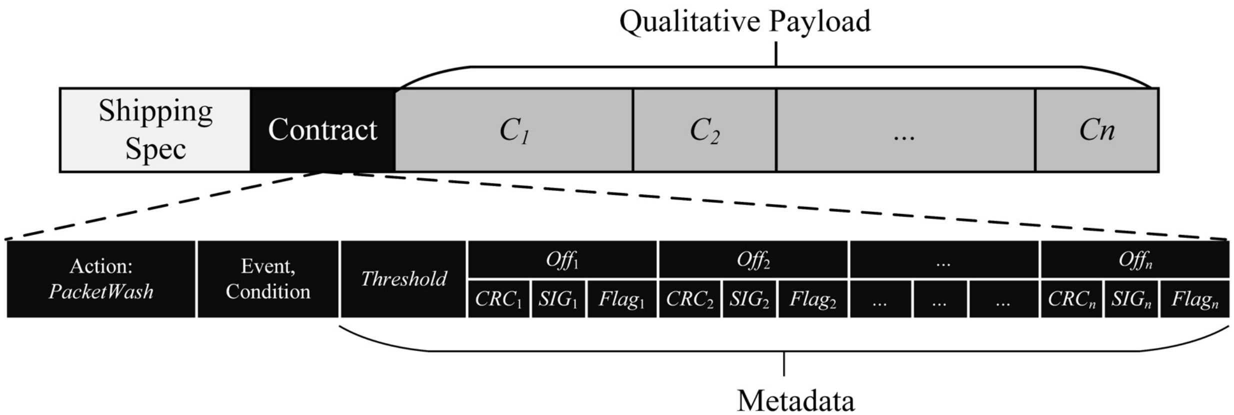
Event: Congested **OR** Radio Unstable
Action: Packet Wash
Meta-Data: Chunk 1, Chunk 2, chunk 6, Chunk 7, Chunk 8

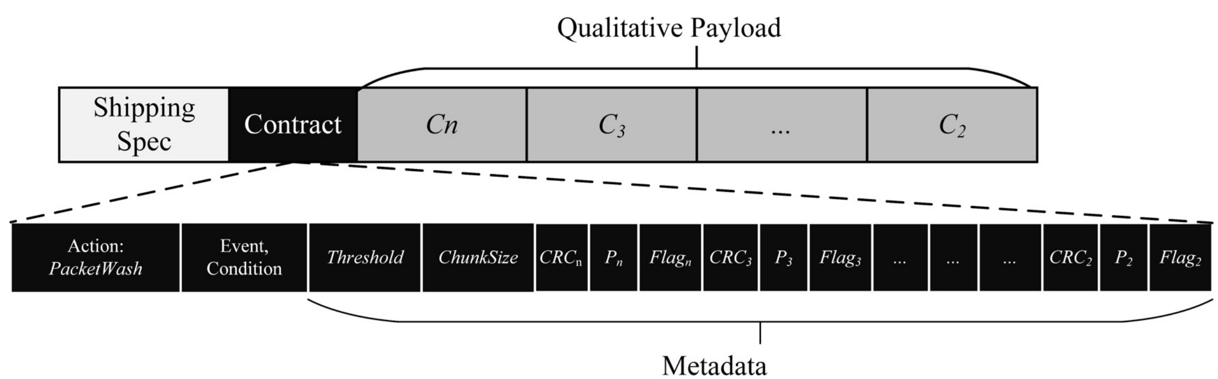
On congested node: after cut



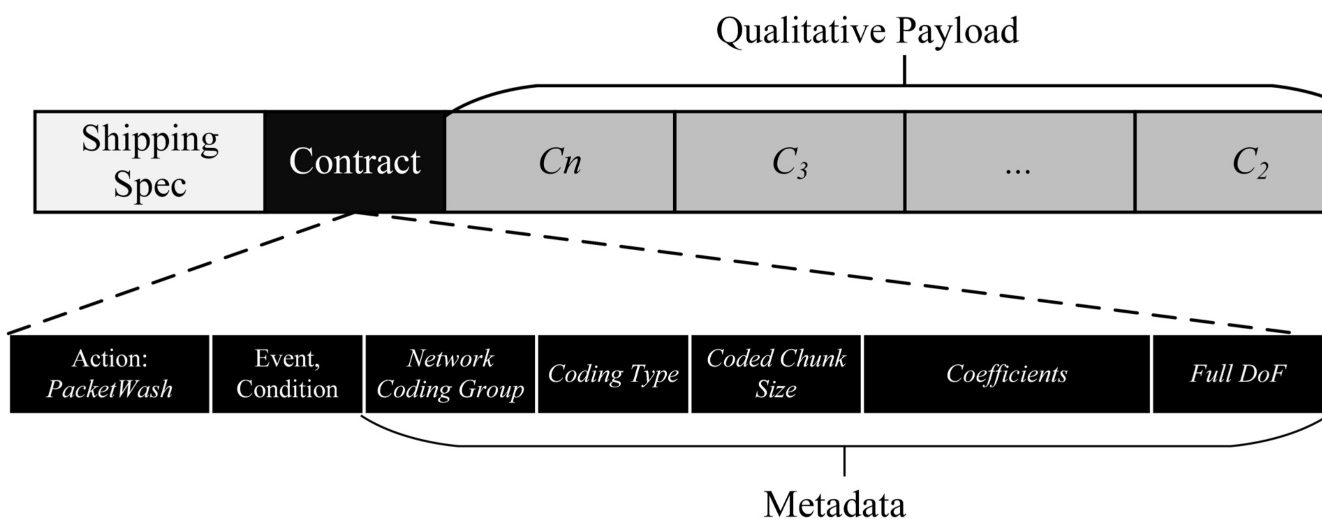
Packet Wash is preferred to drop-and-retransmit

Significance based packetization

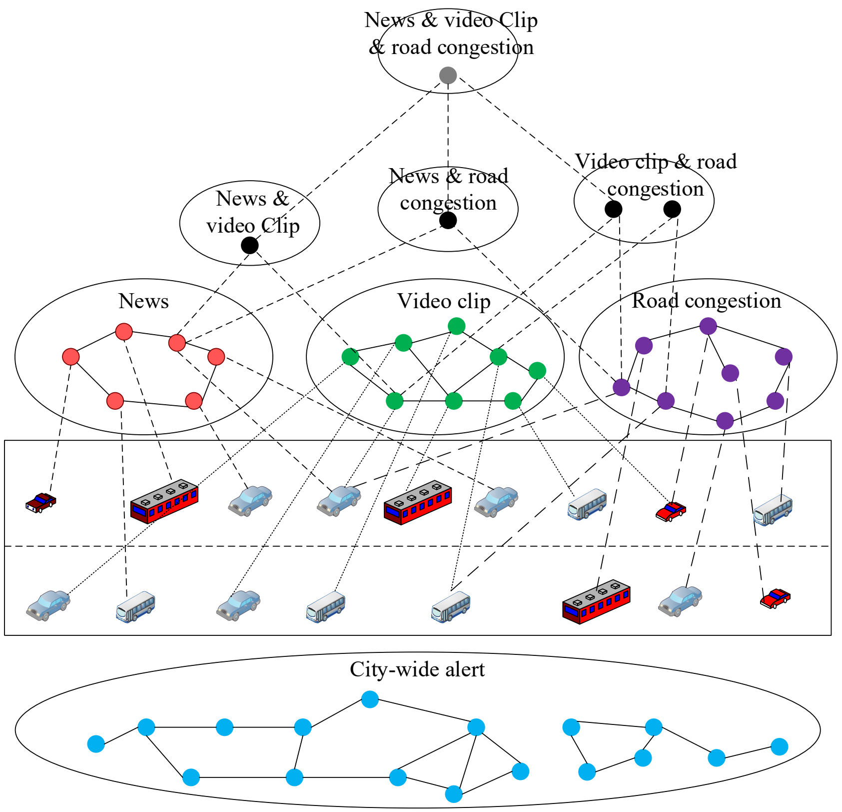




Random Linear Network Coding based packetization



New IP Enables Information Exchange Oriented Vehicle Clustering



- Some vehicle may not have access to the infrastructure and needs other vehicle with Wifi or cellular interface to relay its data request to the infrastructure nodes and the data sent from the infrastructure nodes.
- An information topic based clustering builds a collaborative vehicular system for information exchange being delegated to those vehicles that have infrastructure access and are willing to provide the service to other neighboring vehicles.

Action: forward to cluster head
Metadata: content identifier, topic name, cluster tag, aggregation permitted
Source: requesting vehicle IP address
Destination: content host IP address (could be set to cluster head IP address)

New IP packet for information request

Metadata: content identifiers, aggregation performed
Source: cluster head IP address
Destination: content host IP address

New IP packet for aggregated requests

Metadata: content identifier, offset in payload for corresponding data.
Source: requesting vehicle IP address
Destination: content host IP address (could be set to cluster head IP address)

New IP packet for concatenated data

References for Future Reading

- Lijun Dong, Richard Li, Collaborative Computation in the Network for Remote Driving, IEEE SSCI, 2021.
- Lijun Dong, Richard Li, "Distributed Knowledge Inference and Reasoning in the Network Based on New IP," International Symposium on Networks, Computers and Communications (ISNCC), 2021.
- Lijun Dong, Lin Han, "New IP Enabled In-Band Signaling for Accurate Latency Guarantee Service," IEEE WCNC, 2021.
- Lijun Dong, Alex Clemm, "High-Precision End-to-End Latency Guarantees Using Packet Wash," IFIP/IEEE International Symposium on Integrated Network Management (IM), 2021.
- Lijun Dong, Richard Li, "Optimal Chunk Caching in Network Coding Based Qualitative Communication," Journal of Digital Communications and Networks, available online, June 2021.
- Richard Li, Uma S. Chunduri, Alexander Clemm and Lijun Dong, "New IP: Enabling the Next Wave of Networking Innovation," book chapter in "Design Innovation and Network Architecture for the Future Internet", pp 1-42, DOI:10.4018/978-1-7998-7646-5.ch001, Jan 2021.
- Lin Han, Lijun Dong, Richard Li, "A Study of In-Vehicle-Network by New IP," The Thirteenth International Conference on Evolving Internet, 2021.
- Lijun Dong, Kiran Makhijani, Richard Li, "Qualitative Communication Via Network Coding and New IP," IEEE HPSR 2020.
- Lijun Dong, Richard Li, "Scheduling of Complex Computation Offloading Tasks in Mobile Edge Cloud Network with New IP Framework," IEEE World Forum on Internet of Things 2020.
- Lijun Dong, Richard Li, "Packet Level In-Time Guarantee: Algorithm and Theorems," IEEE Globecom 2020.
- Lijun Dong, Richard Li, "Enhance Knowledge Inference and Reasoning by New IP," IEEE iThings 2020.
- Lijun Dong, Richard Li, "Semantics and Deviation Aware Content Request," IEEE ISCC 2019.
- Lijun Dong, Richard Li, "Information Exchange Oriented Clustering for Collaborative Vehicular System," IEEE WOCC 2018

Please contact presenters for questions, feedback, and comments

Mohit P. Tahiliani tahiliani@nitk.edu.in
Lijun Dong lijun.dong@futurewei.com
Bhaskar Kataria bhaskar.181co213@nitk.edu.in
Deepta Devkota deeptadevkota.191cs117@nitk.edu.in
Shashank D shashankd.181co248@nitk.edu.in
Kiran M kiranm@futurewei.com
Acknowledgements: @ameyanrd, @rohit-mp, @lesliemonis

Thank You