April 29, 2025

NSDI 2025

# MTP: Transport for In-Network Computing

**Tao Ji,** Rohan Vardekar, Balajee Vamanan, Brent E. Stephens, Aditya Akella

TEXAS The University of Texas at Austin

UNIVERSITY OF ILLINOIS CHICAGO

THE UNIVERSITY OF UTAH

# In-network computing (INC) ideas in the past

| OSI Layer | Offloaded Functions |
| --- | --- |
| L6: Presentation & L5: Session | Compression, Cryptography |
| L4: Transport | Generic Segmentation/Receive Offload<br>RDMA, TCP Offload Engine, |
| L3: Network | NAT, Firewall, Intrusion Detection/Prevention |

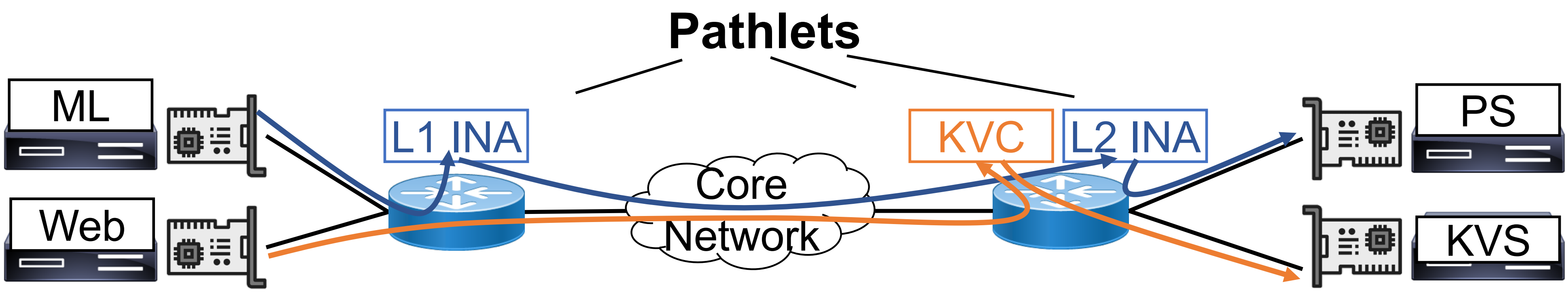
**Commonly used by most applications.**

# In-network computing (INC) ideas in the past

| OSI Layer | Offloaded Functions |
|---|---|
| L7: Application | Key-Value Cache, RPC/HTTP Load Balancer, In-Network Aggregation, Database Execution |
| L6: Presentation & L5: | |
| L4: Transport | RDMA, TCP Offload Engine, Generic Segmentation/Receive Offload |
| L3: Network | NAT, Firewall, Intrusion Detection/Prevention |

Can co-exist in the same network!

# INC-enabled datacenter network model

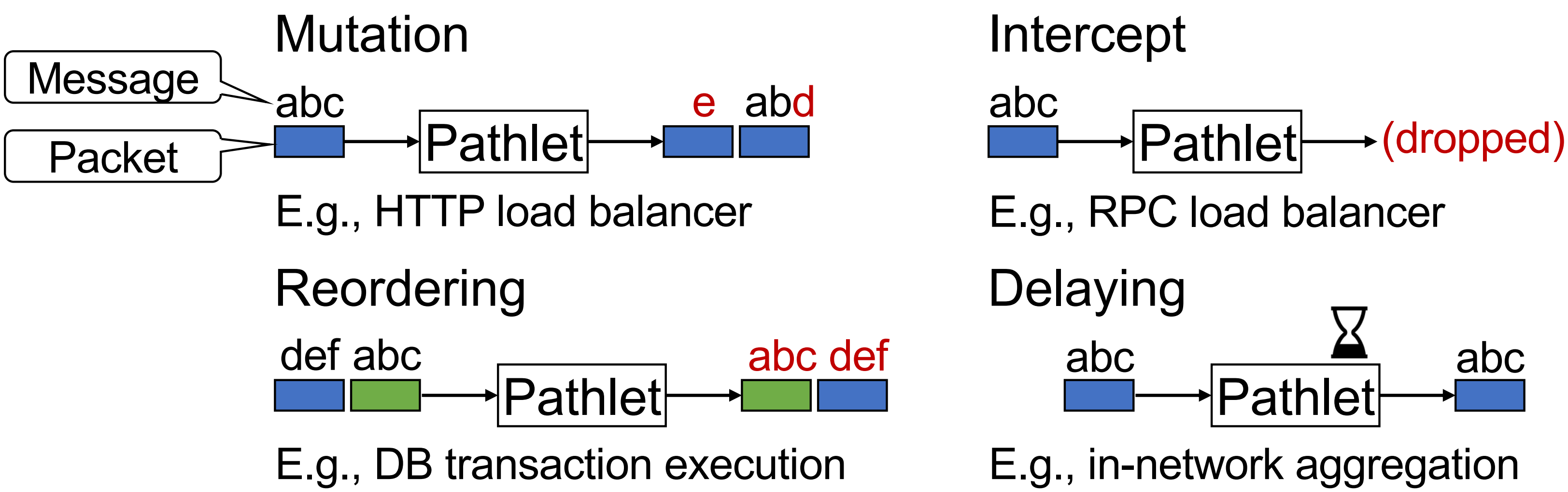


# In-network message operations

Pathlets process messages that can span one or more packets

- E.g., RPCs, HTTP requests/responses

Message operations




**These operations happen between communicating application processes on different servers: transport's responsibility**

# Today's transports are incompatible

Mutation/intercept

- More/fewer packets/bytes delivered than sent
- Breaks correctness of ACK-retransmit mechanism

Reordering

- Hole in packet/byte sequence seen as signal of loss
- Causes spurious retransmission

Delaying

- False positive timeout and spurious retransmission
- Inaccurate congestion control

# Alternative approaches are not generalizable

Termination (i.e., running transport endpoint processing at offloads) is not feasible with all INC hardware

- Architectures such as RMT cannot run complex logic
- Transport stacks can run on cores but need many for line rate

Existing workarounds are not widely applicable:

- Relying on insight of a specific pathlet; or
- Not supporting all message operations

# MTP to the rescue

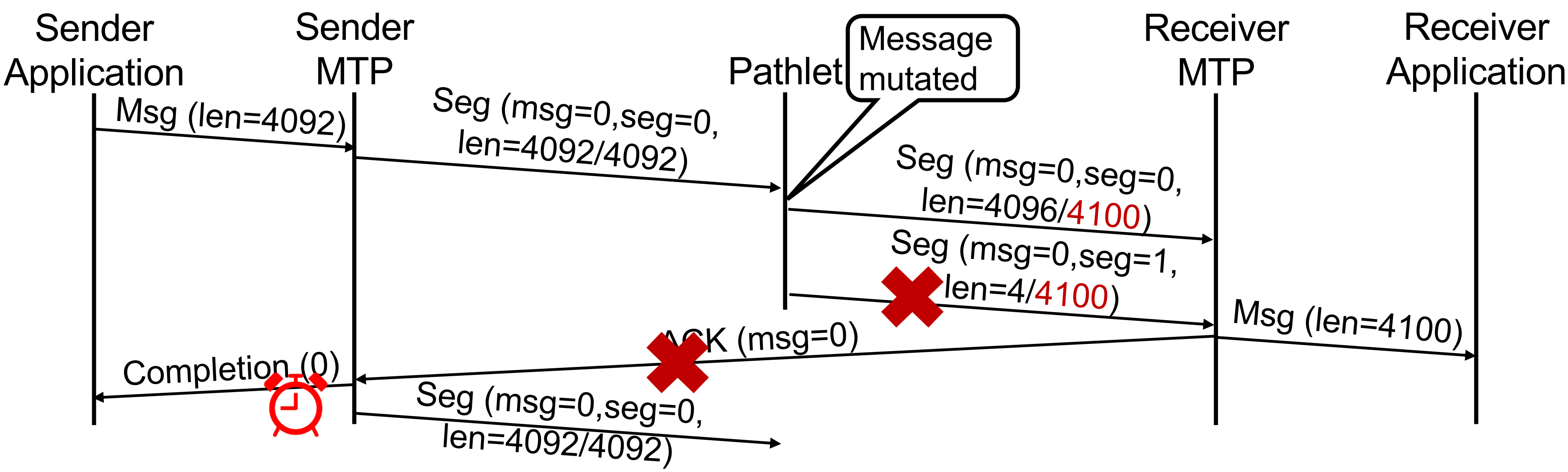
MTP (Message Transport Protocol): first transport to natively support in-network message operations while providing essential services:

- Reliable delivery: recovering from packet losses in the network correctly and efficiently
- Congestion control: preventing congestion drops and achieving high utilization at bottleneck pathlets

Not requiring transport-specific state in network

- All kinds of INC hardware can easily participate in MTP

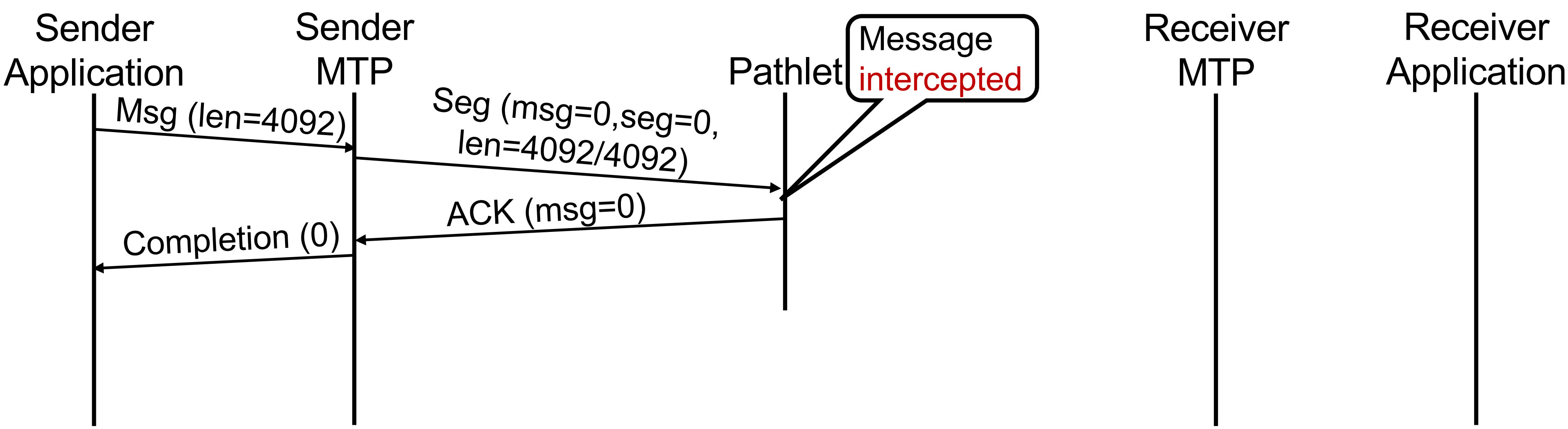
# Basic reliability protocol: workflow with mutation




Passive receiver: only transmits ACK upon receiving whole message

- Sender times out upon loss and retransmit the whole message

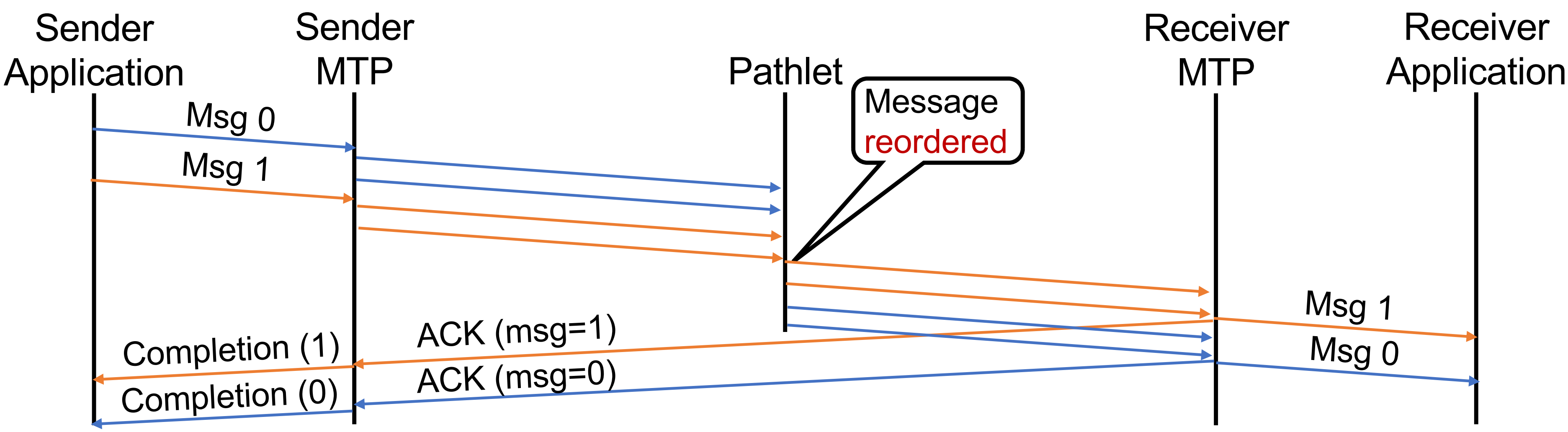
# Basic reliability protocol: intercept




Pathlet sends back ACK for message to intercept

- Sender proceeds as if the message was delivered

# Basic reliability protocol: reordering




Message reordering naturally supported:

- Sender and receiver handle segments/ACK for each message independently

# Challenge: message delaying

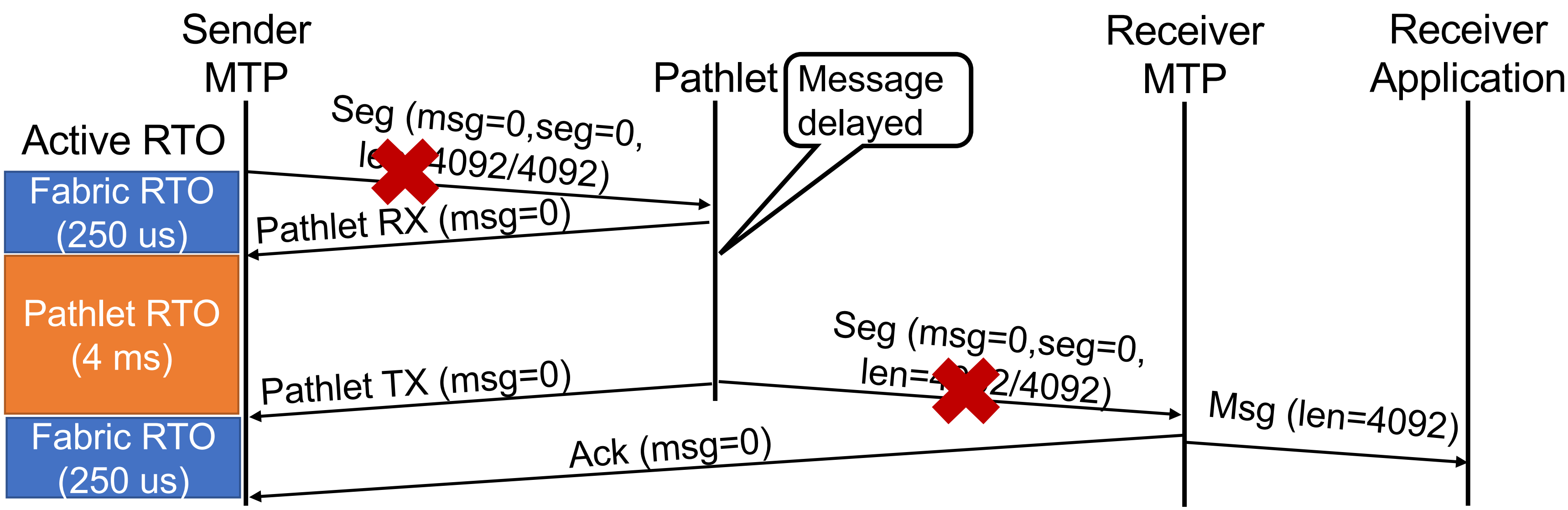
Sender-based retransmission timeout (RTO) without other loss signals

- Adopted by prior art based on the assumption that delays in non-INC datacenter are bounded.
- Broken by message delaying (e.g. INA waiting for straggler)
  - Easily longer than the network delay, causing false positives

Idea: separating fabric delay and pathlet processing delay

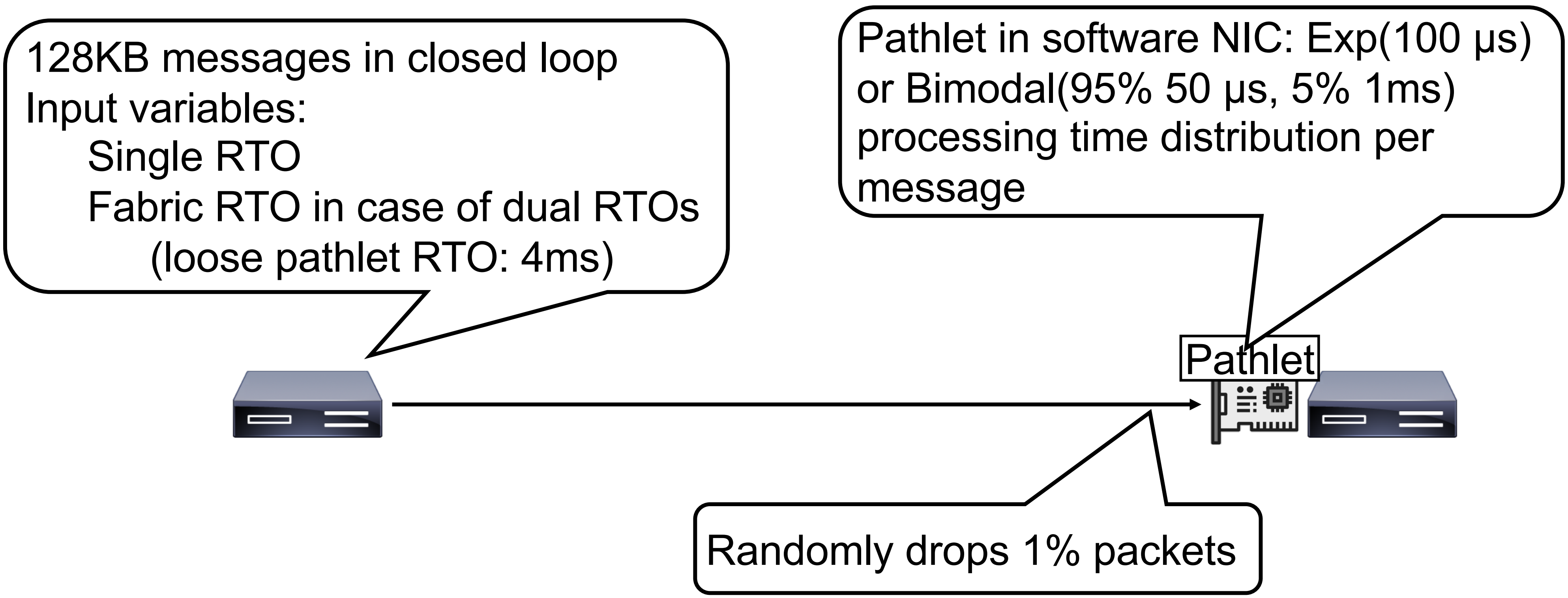
- Using separate RTO lengths to account for the two delays.
- Pathlet transmits special ACKs to indicate the message has entered/left the pathlet.

# Reliability with dual RTOs

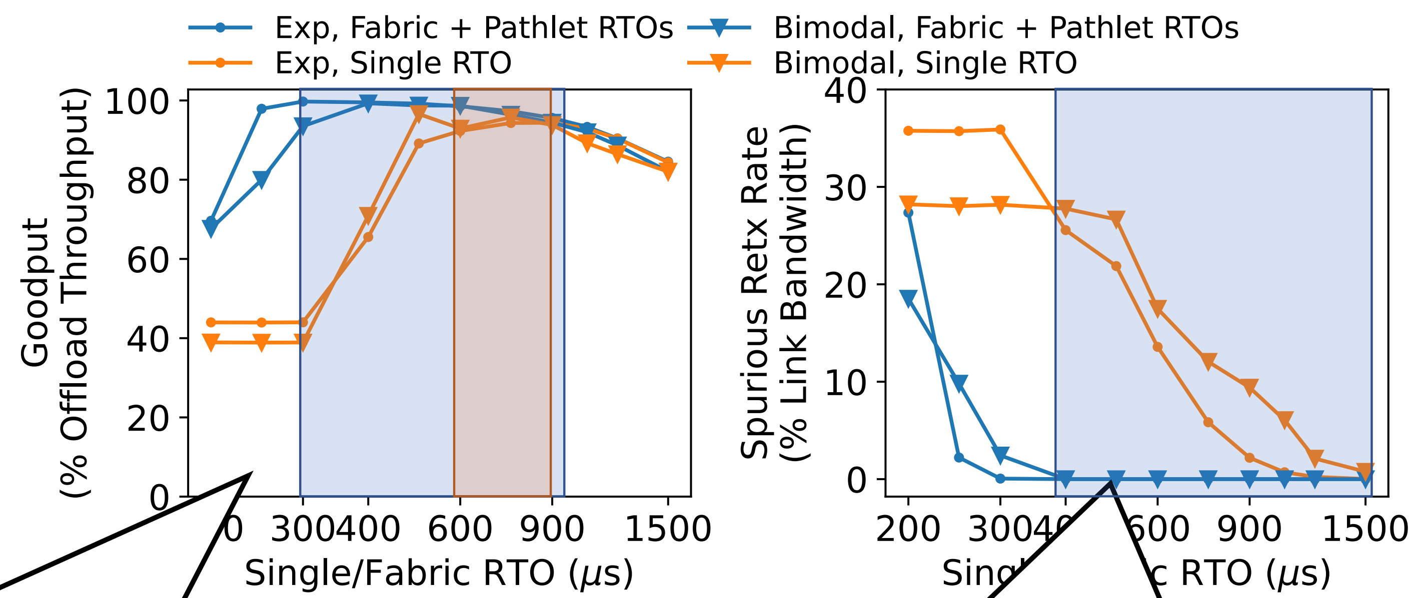



**Sender reacts to fabric drop quickly with fabric (short) RTO**

# Loss recovery efficiency with dual RTOs



# Dual RTOs make configuration easier




Dual RTOs allows much larger RTO ranges achieving >90% and 100% goodput for both distributions.

Dual RTOs prevents spurious retransmissions for all RTOs above 400 μs.

# Congestion control framework for pathlets

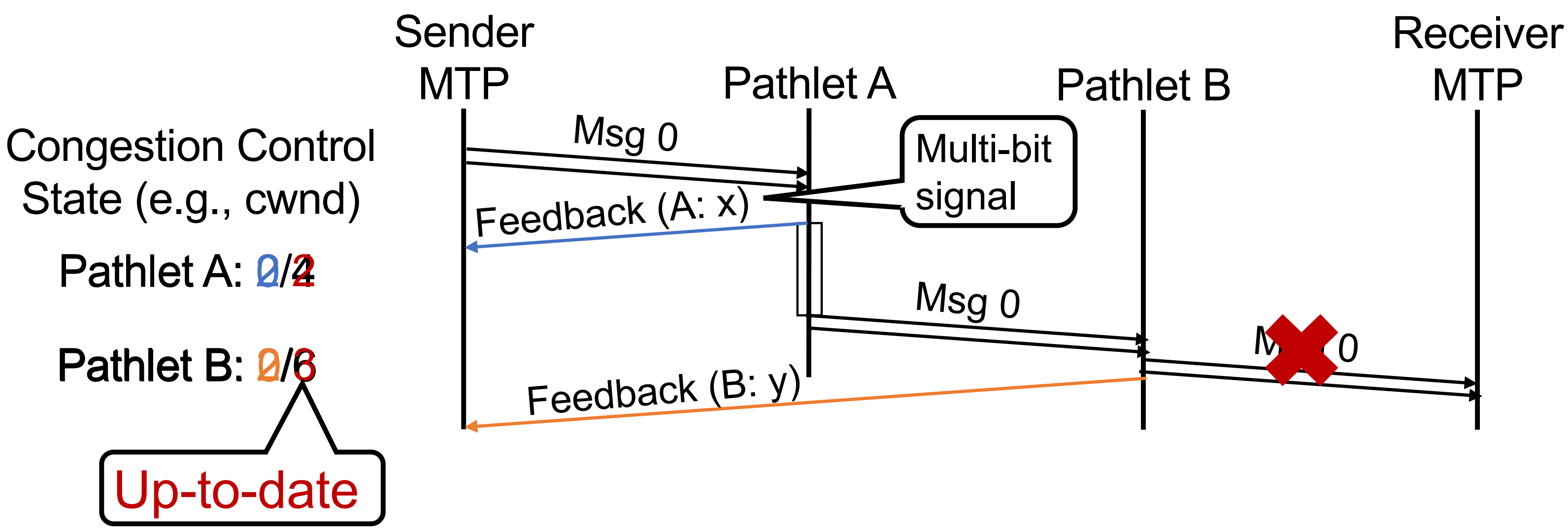
Pathlets can be network bottlenecks due to complex processing or limited hardware capacities.

Challenge: heterogeneous performance characteristics

- Operator cannot empirically tune congestion signals (e.g., ECN threshold or target RTT) and apply network-wide
- End-to-end congestion signals can be delayed or dropped along with message that carries them

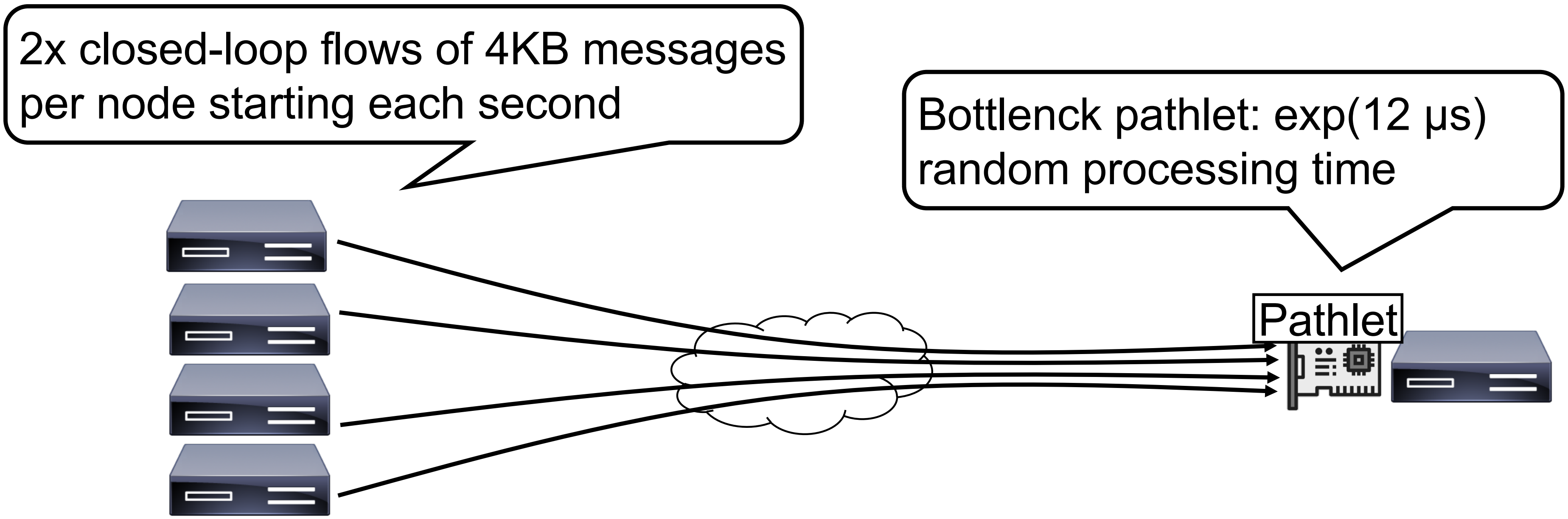
Our approach: **pathlet-specific** and **early** congestion feedbacks

# Pathlet congestion control framework




**Example usage: 8-bit quantized queue size feedback + Swift**

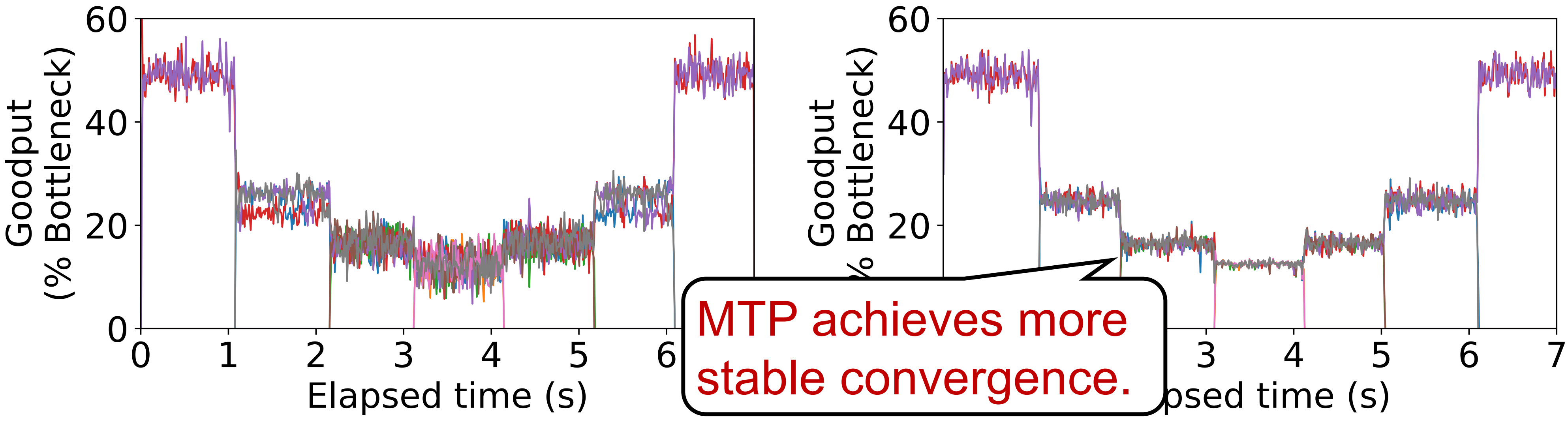
# Convergence under unpredictable pathlet service times



# Convergence under unpredictable pathlet service times

Legacy congestion control:
Single-bit end-to-end ECN + DCTCP

MTP congestion control (example):
Pathlet multi-bit early feedback + Swift



# Other considerations

Exactly-once delivery guarantee

- Strawmen require unbounded state size or causes HoL blocking
- Virtual channel: **reordering-friendly constant-sized** deduplication

Mitigating hotspots

- Multiple same-type pathlets can be available
- MTP can **dynamically switch to an alternative pathlet**

More evaluation

- End-to-end benchmarks with NetCache
- Cost of integration analysis

# Summary

- Abstracted L7 offloads in an INC-enabled network as **pathlets** along overlay paths.
- Identified essential pathlet message operations, i.e., **mutation, intercept, reordering and delaying**, arguing that today's transports are incompatible.
- Described MTP which natively supports in-network message operations from pathlets, providing effective and efficient **reliable delivery and congestion control**.

**Thank you!**