

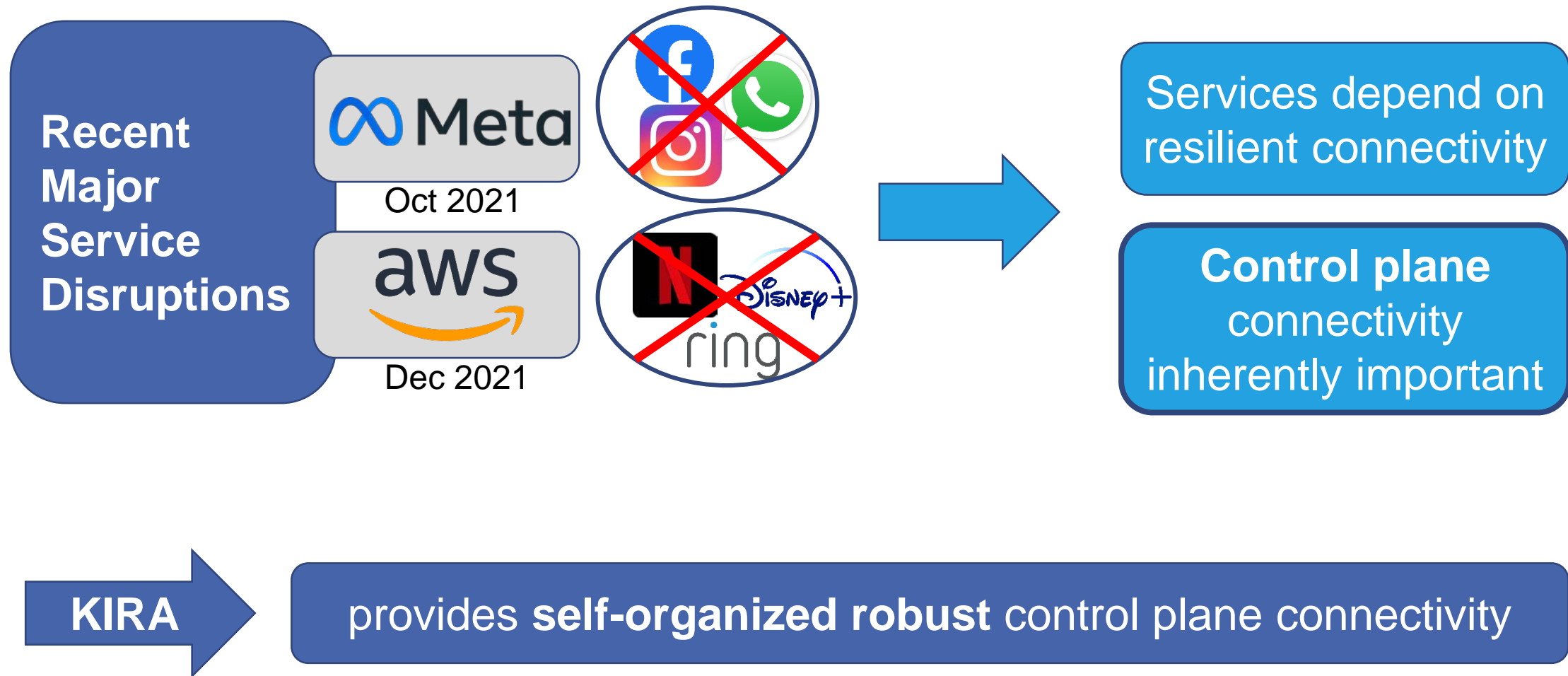
KIRA – Scalable ID-based Routing Architecture for Control Planes

Roland Bless, Martina Zitterbart
Institute of Telematics, KIT

Zoran Despotovic, Artur Hecker
Huawei Research Center, Munich

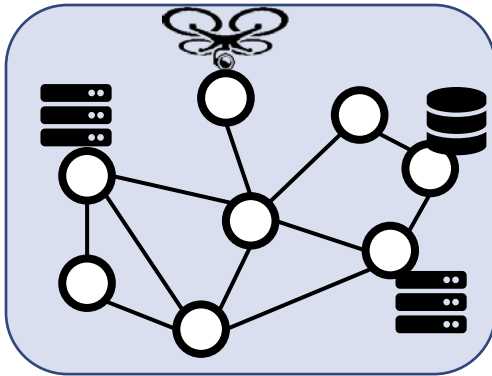
KIRA: Kademlia-directed ID-based Routing Architecture

Controllability and Control Planes



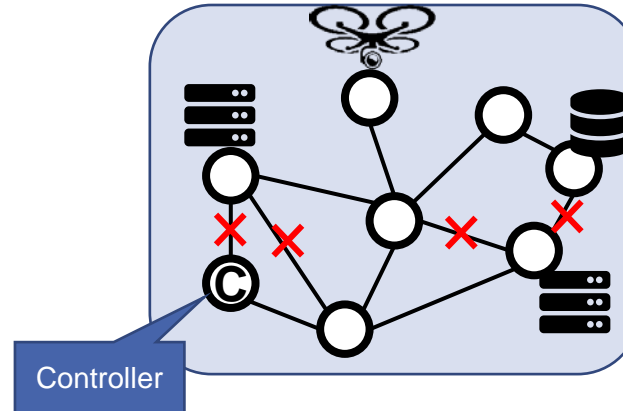
What KIRA achieves...

Interconnects a Large Pool
of Networked Resources

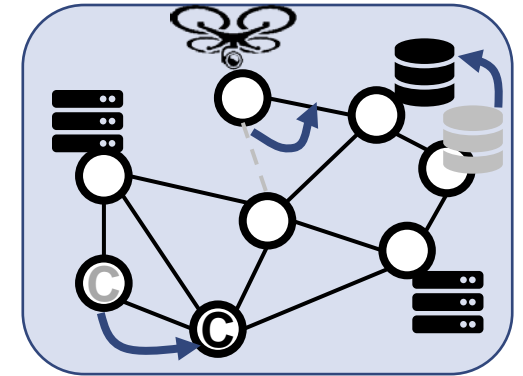


Compute, Storage, Network

Resilient Connectivity
for Control Plane

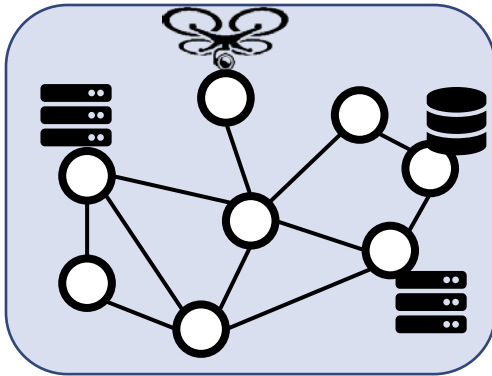


Stable Addresses
for Moving Resources



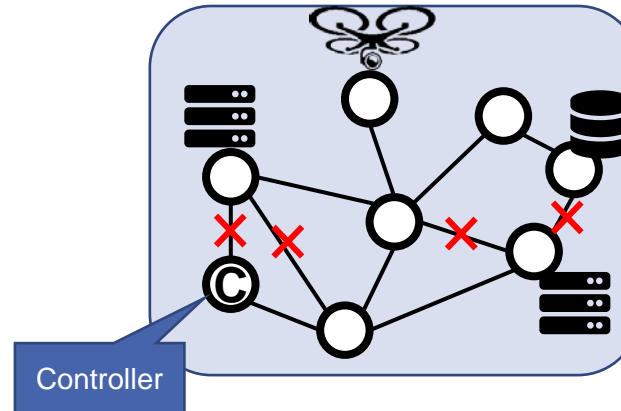
What KIRA achieves...

Interconnects a Large Pool of Networked Resources

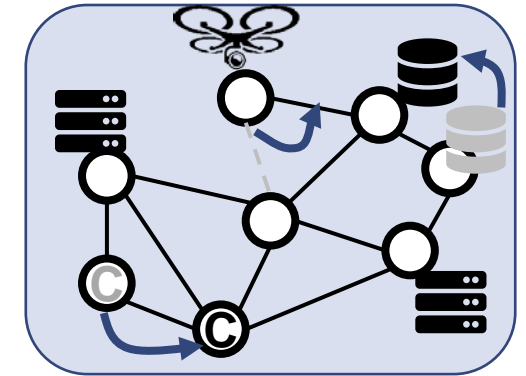


Compute, Storage, Network

Resilient Connectivity for Control Plane



Stable Addresses for Moving Resources



■ KIRA provides (all-in-one)

- Massive scalability (100,000s of nodes)
- Zero-touch (no configuration)
- Dynamics: fast convergence, loop free
- Topological versatility
- Efficient routes

■ Related Works (examples)

- UIP: lacks dynamics, efficient routes
- DISCO: lacks dynamics
- RIFT, Data Center BGP/OSPF/IS-IS: specific topologies only, not ID-based
- ...

KIRA – Main Components

■ Routing Tier → connectivity

R²/Kad

- ID-based addresses
- Source routing
- On top of link layer

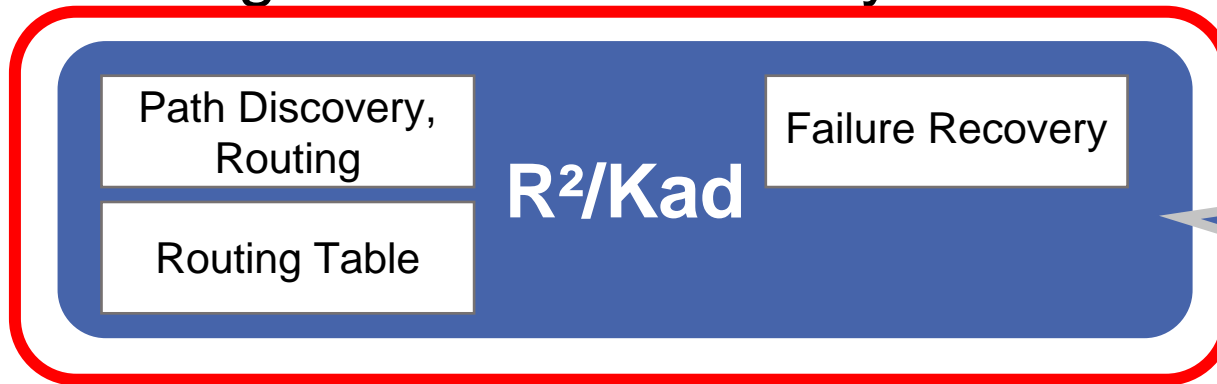
■ Forwarding Tier → optimization

PathID-based Forwarding

- Eliminates source routing
- Label switching approach
- Reduces overhead

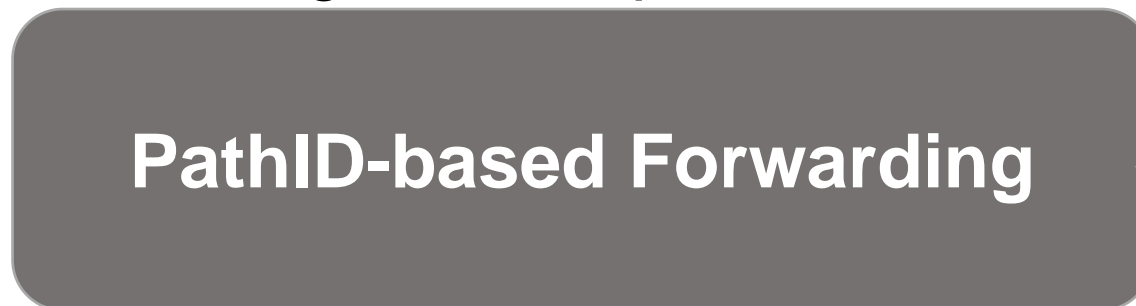
KIRA – Main Components

■ Routing Tier → connectivity



- ID-based addresses
- Source routing
- On top of link layer

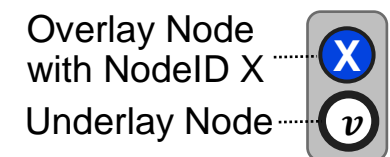
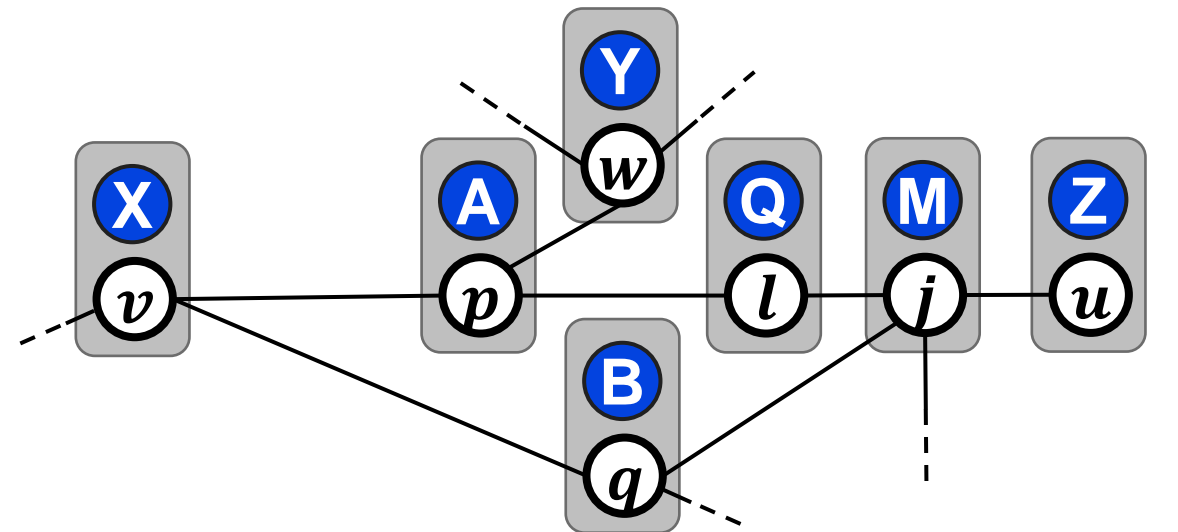
■ Forwarding Tier → optimization



- Eliminates source routing
- Label switching approach
- Reduces overhead

R²/Kad – Path Discovery

- Each node
 - randomly chooses its **NodeID** (Overlay)
 - learns its **2-hop vicinity** (Underlay)
 - X learns contacts A, Y, B, M, ...



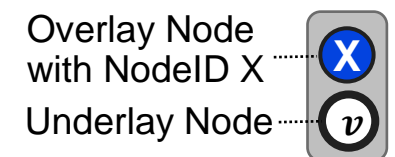
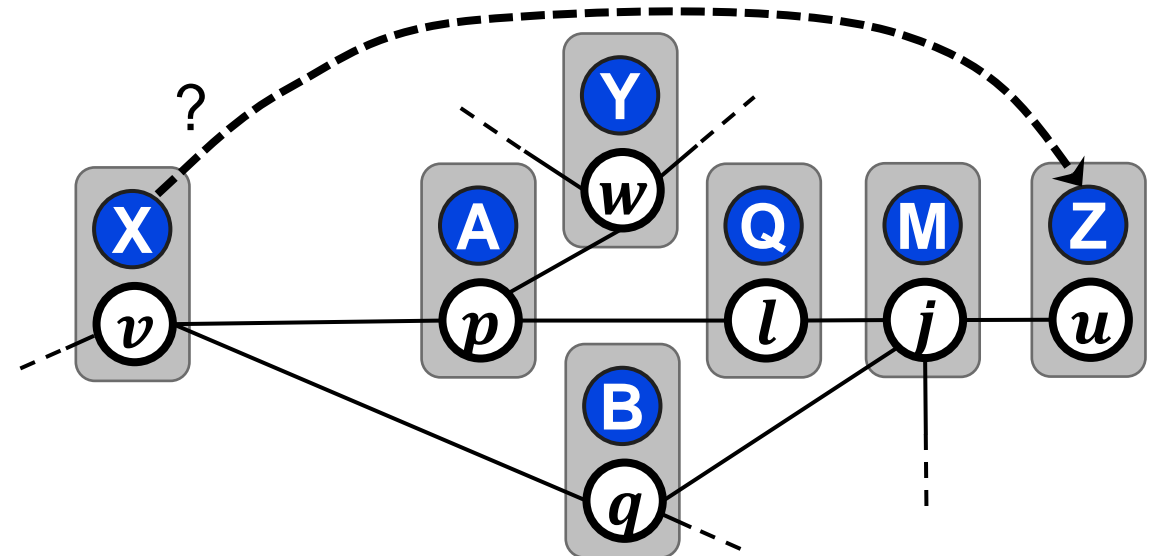
R²/Kad – Path Discovery

- Each node
 - randomly chooses its **NodeID** (Overlay)
 - learns its **2-hop vicinity** (Underlay)
 - X learns contacts A, Y, B, M, ...

X: path to Z?

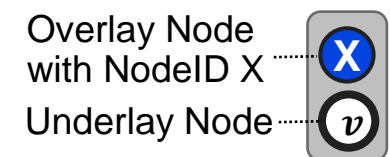
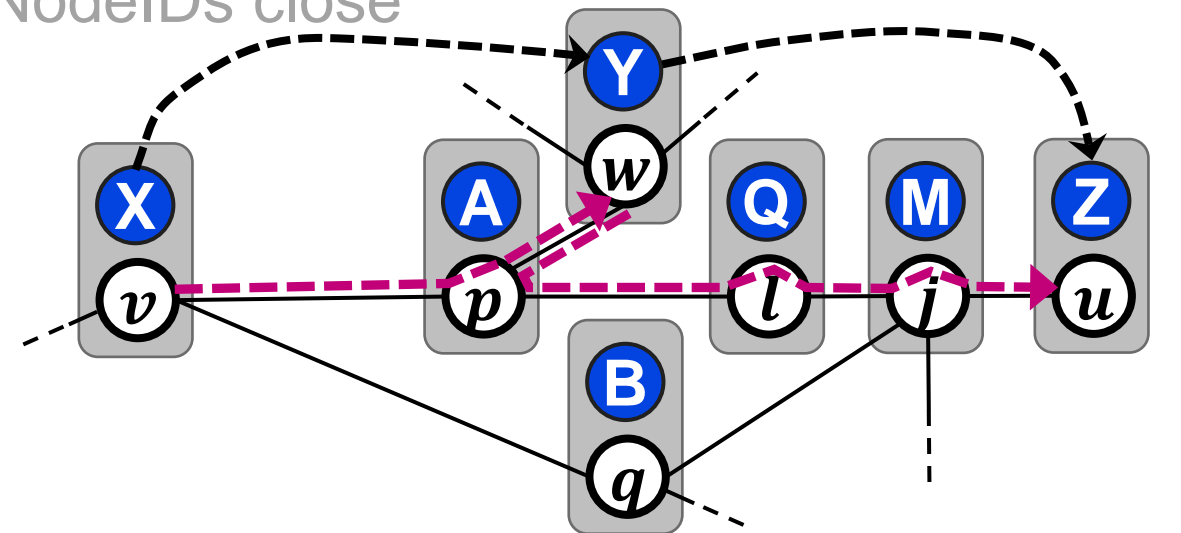
Approach:
 construct underlay routes
 by using the **NodeID-based overlay**

- Source route to contact that is closest to destination NodeID
- Distance of NodeIDs: **XOR metric** $d(X, Y) = X \oplus Y$
 - Longer shared prefix \rightarrow closer



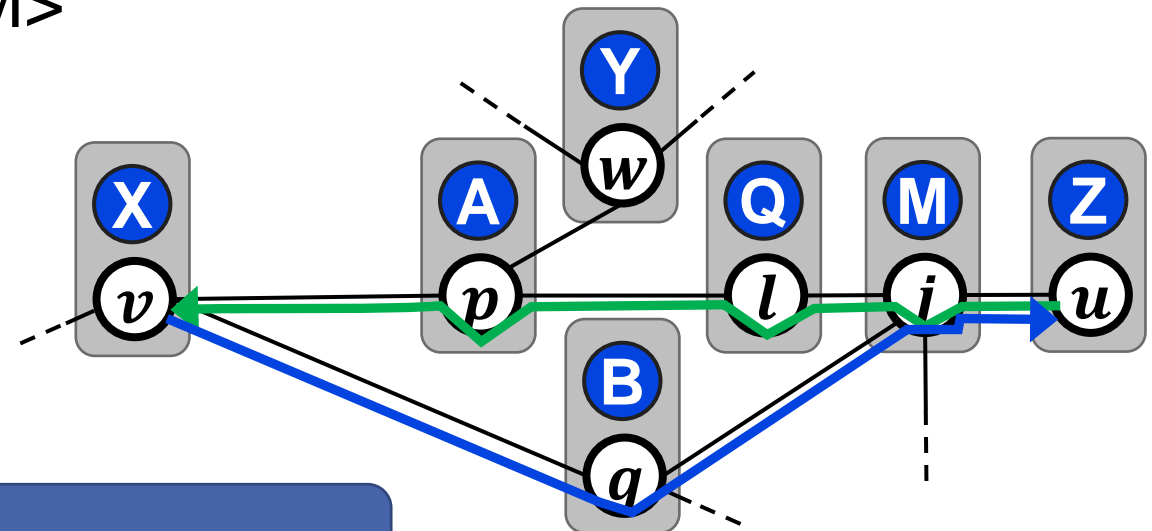
R²/Kad – Path Discovery Example

- X sends FindNodeReq to contact **closest** to NodeID Z
 - Example: letters close in alphabet ↔ NodeIDs close
 - Next (overlay) hop: Y
- X → Y via **source route** <A>
- Assume Y knows Z already
- Y → Z via **source route** <A,Q,M>
- FindNodeReq records complete route <X,A,Y,A,Q,M>
 - incurs path stretch: $\frac{|\text{selected path}|}{|\text{shortest path}|}$



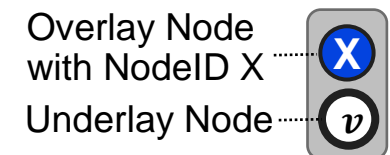
R²/Kad – Path Discovery Example

- Shortened recorded route $\langle A, Q, M \rangle$ is returned to X in FindNodeRsp
- Later packets use shorter route $\langle B, M \rangle$
 - if X already knows M via $\langle B \rangle$



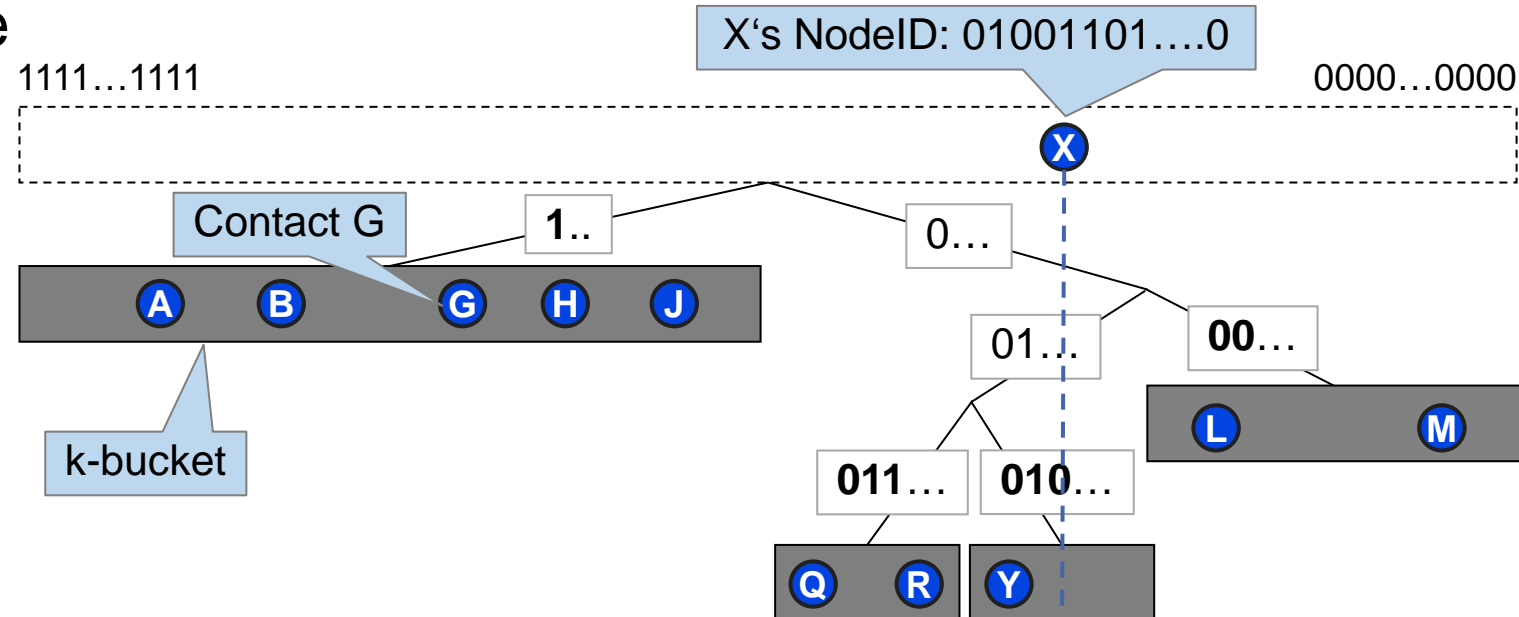
Initial stretch can be reduced for later packets

- R²/Kad offers flexible memory/stretch trade-off...



R²/Kad – Routing Table

- Tree of **buckets** holding up to **k contacts** (e.g., k=20)
 - Arranged by XOR distance
 - Bucket split if already full and contains own NodeID



R²/Kad – Routing Table

- Tree of buckets holding up to k contacts (e.g., $k=20$)

- Arranged by XOR distance
- Bucket split if already full and contains own NodeID

- Path vectors are stored for each contact

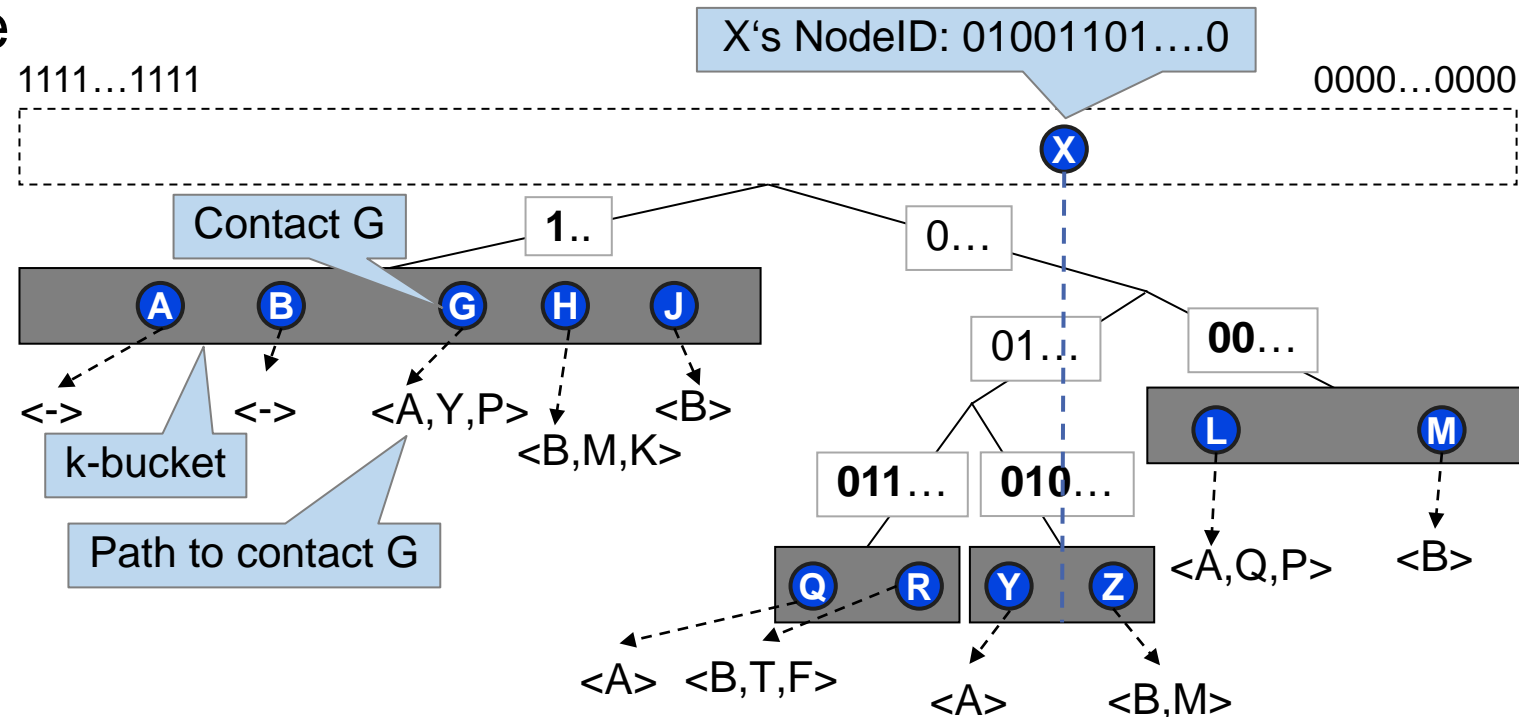
- Efficient routes

- Shorter routes preferred

- Size k of k -buckets can be set per node

→ flexible memory / stretch trade-off

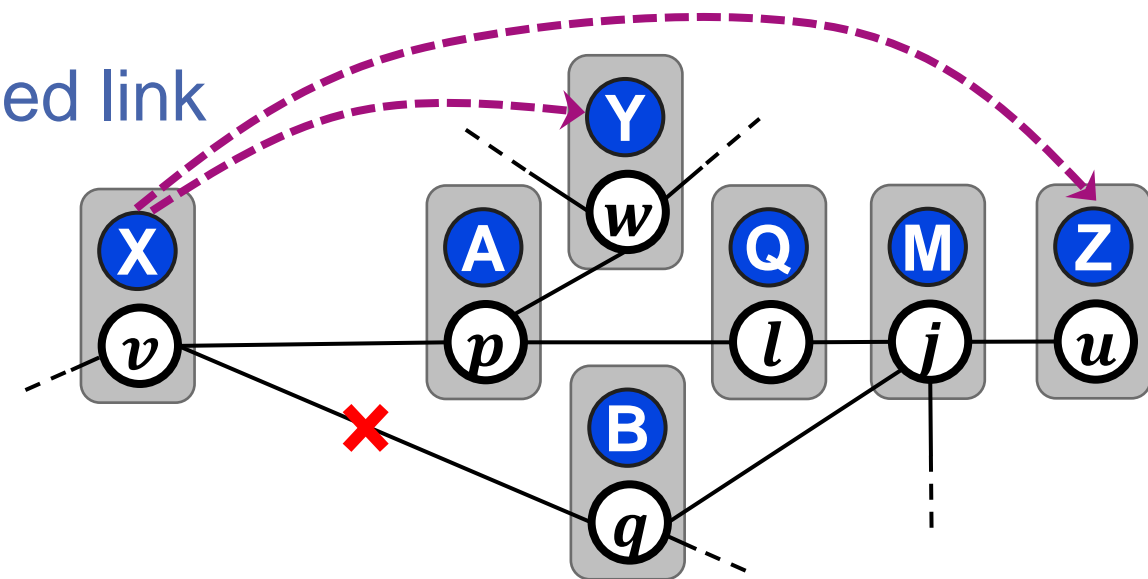
- Routing table size: $O(l_G \log n)$, l_G average path length



R²/Kad – Dynamics: Rediscovery Procedure



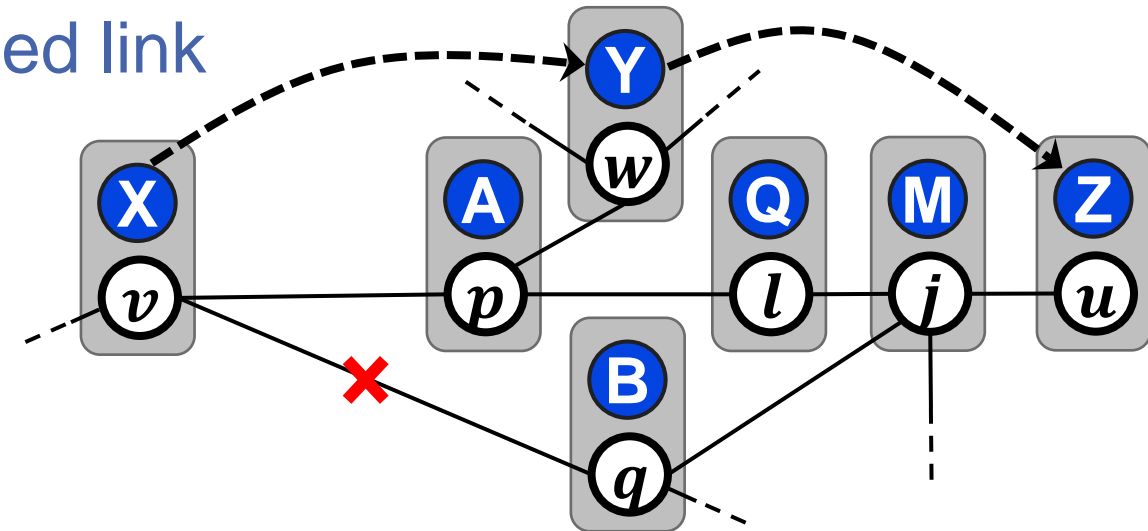
- Detection of node/link failure in the underlay
- Two step strategy
 - 1.) inform ID-wise neighbors about failed link
 - 2.) ...



R²/Kad – Dynamics: Rediscovery Procedure



- Detection of node/link failure in the underlay
- Two step strategy
 - 1.) **inform** ID-wise neighbors about **failed link**
 - 2.) **rediscover** alternative paths via overlay routes (includes “**Not Via**” information)
- Periodically
 - probe contacts for broken paths
 - lookup own NodeID
- Validity
 - State sequence numbers
 - Path information **age**



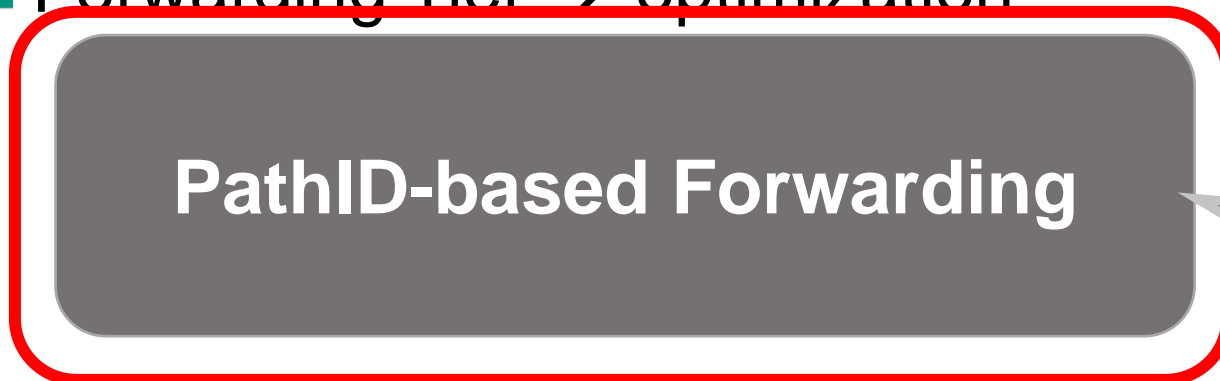
KIRA – Main Components

■ Routing Tier → connectivity



- ID-based addresses
- Source routing
- On top of link layer

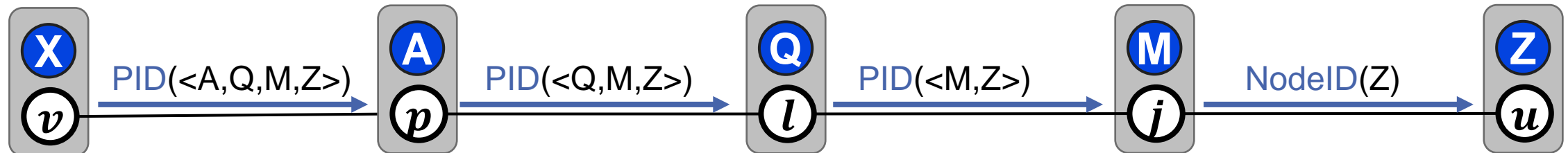
■ Forwarding Tier → optimization



- Label Switching Approach
- Eliminates Source Routing
- Reduces Overhead

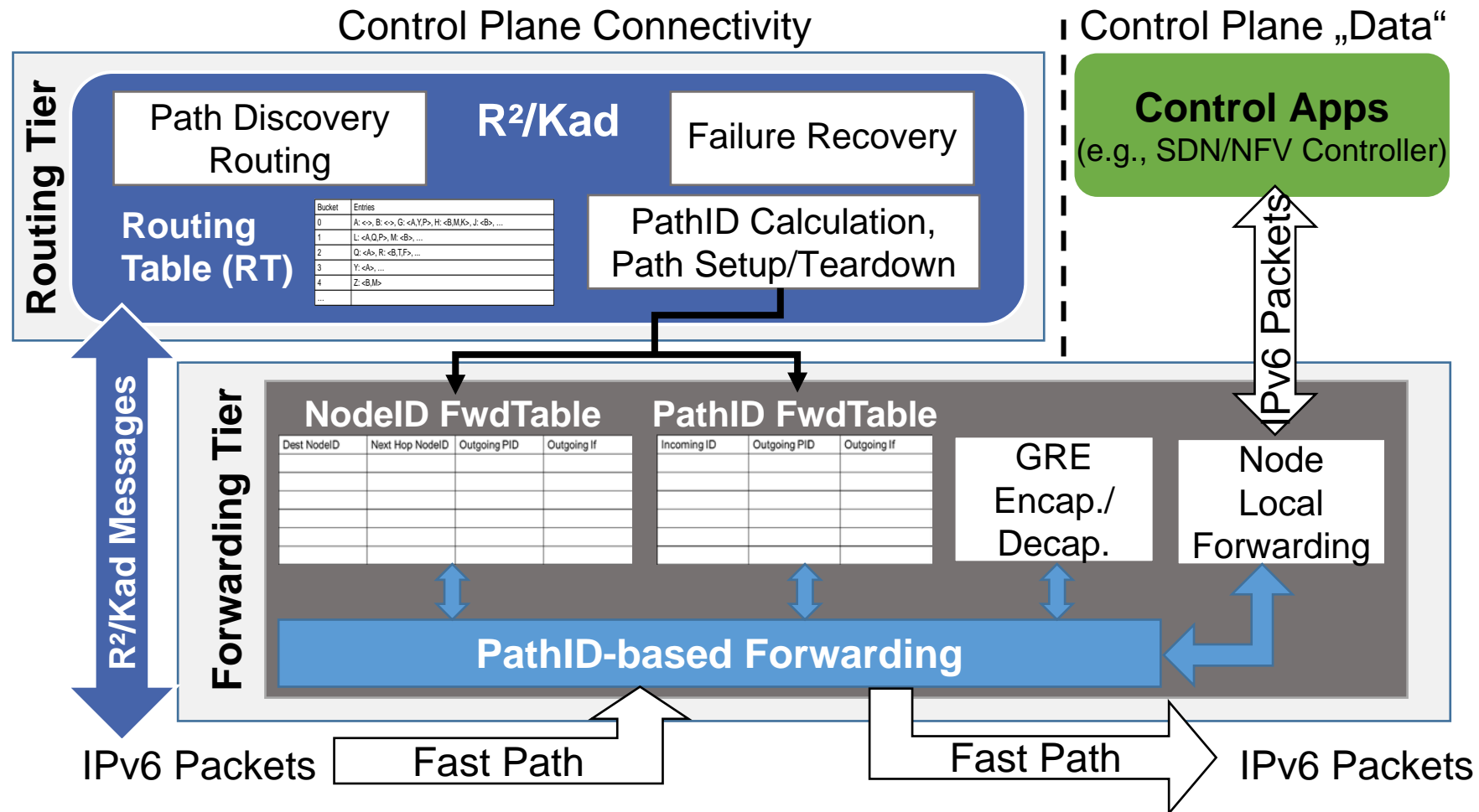
Forwarding Tier – Fast Forwarding

- Get rid of source routes for control plane packets
 - Reduce per packet overhead
- Approach: **replace source routes with PathIDs**
 - $\text{PathID}(\langle A, Q, M, Z \rangle) = \text{Hash}(A \mid Q \mid M \mid Z)$
- Use PathID as label for source route → Label Switching



- Precalculate PathIDs for 2-hop (physical) vicinity
- Explicit **path setup** for paths ≥ 4 hops

KIRA – Routing Architecture



R²/Kad Messages

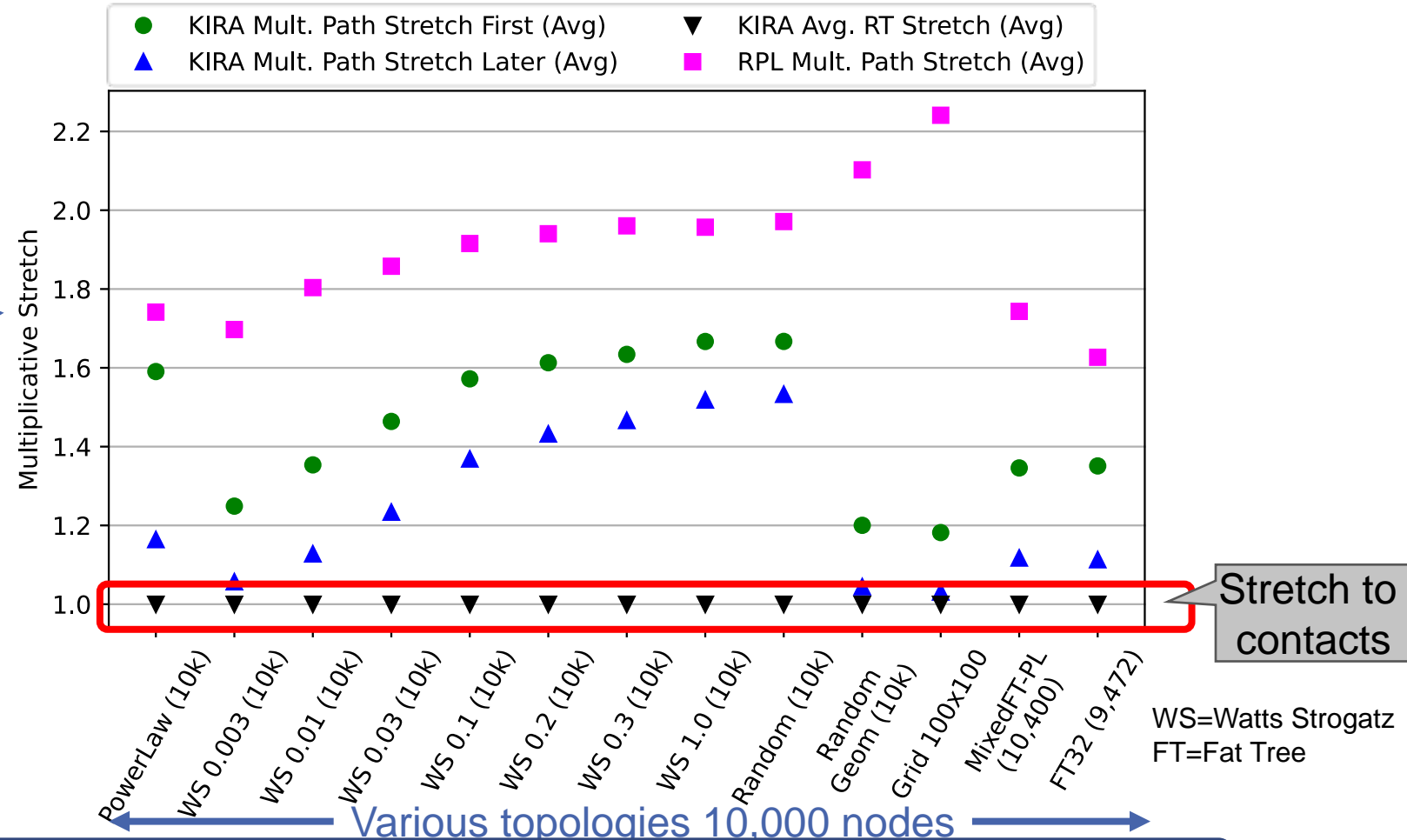
IPv6 Packets

Evaluation – Simulation Setup

- Simulations using **RoutingSim** → **Dynamics** (node/link failures)
- OMNeT++ 5.7
- 10 repetitions with different seeds
- Random processing time per node uniformly drawn from $[0 \dots 500] \mu\text{s}$
- **Various topologies** of different sizes up to **200,000** nodes:
 - Small World: Power-Law, Watts Strogatz, Internet-AS level
 - Regular: Grid, Fat Tree, Mixed Fat Tree/Power Law
 - Random: Random, Random Geometric
 - Real: Topology Zoo

Evaluation – Topological Versatility

- Multiplicative Stretch
- Bucket size k=40
- RPL: Storing-mode, Single DODAG, Single DODAG version

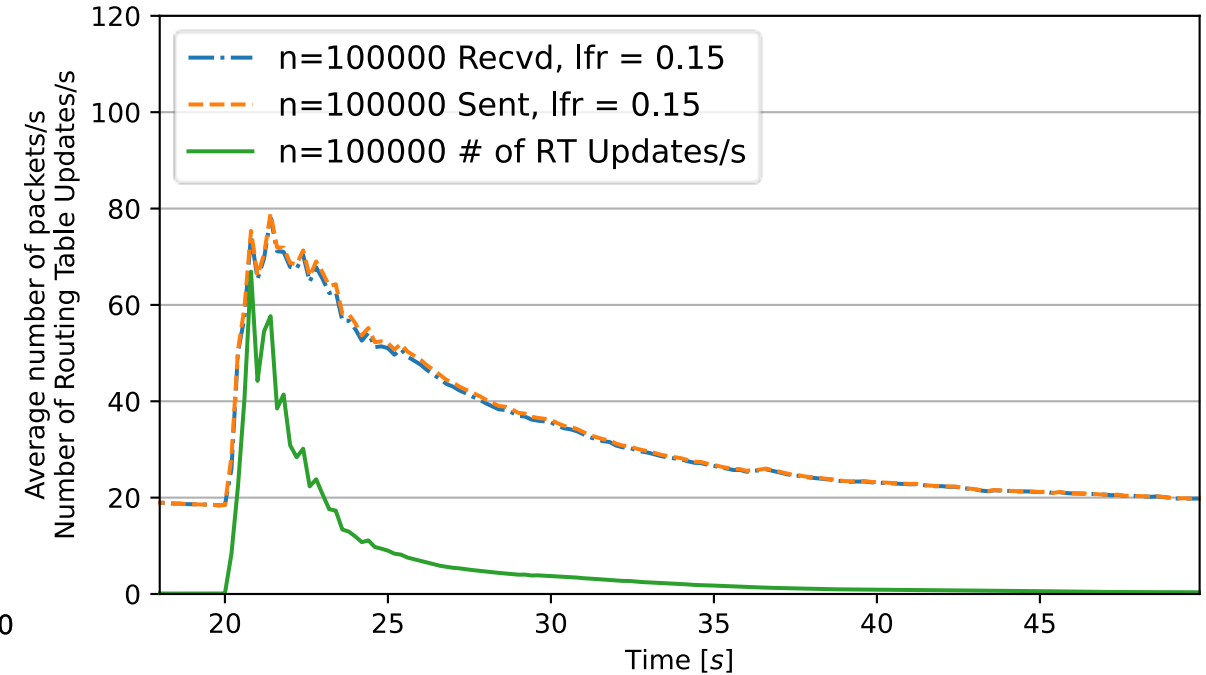
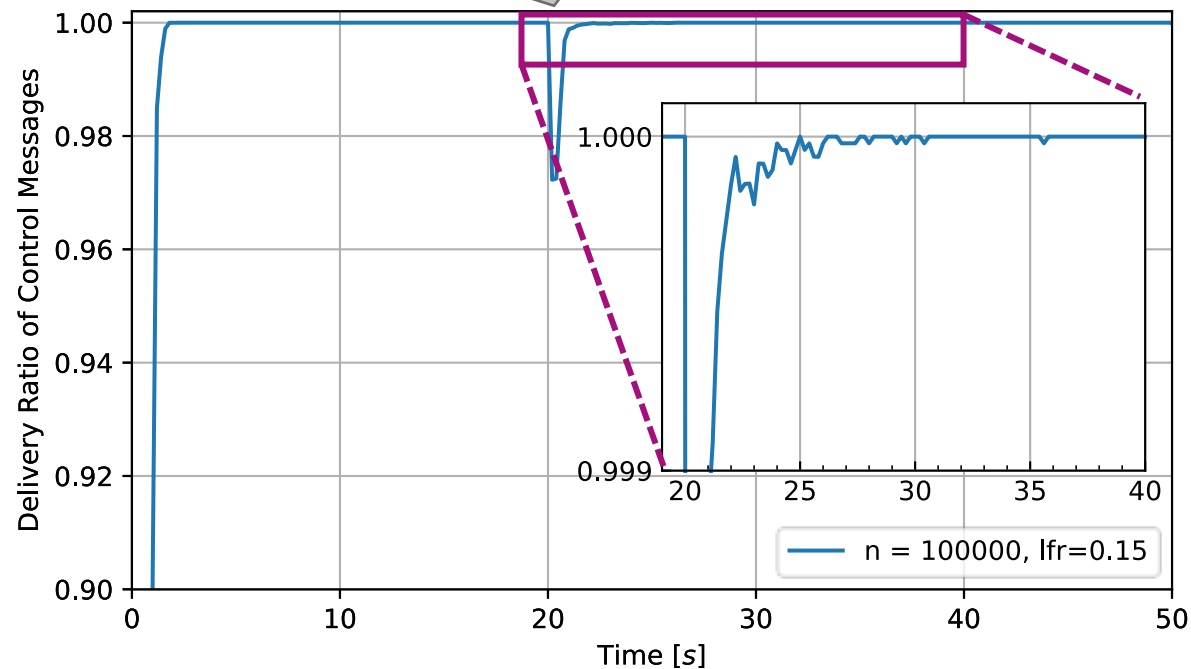


Low stretch across various topologies + Shortest paths to contacts

Evaluation – Dynamics

100.000 nodes Power-Law

15% links fail randomly and simultaneously



Fast Convergence + Scalable Overhead

Conclusions

- KIRA= R²/Kad + PathID-based forwarding

Side meeting **today!**

Nov 10th, 19.00 Mezzanine 12

provides **self-organized robust** control plane connectivity

- Not a replacement for OSPF/IS-IS/BGP
- Designed for large provider domains (e.g., 5G, 6G) or even across multiple providers (Domain concept under development)
- Security may be added
- Special **end-systems** mode → reduces overhead even more
- Supports **multi-path routing** and **forwarding**
- KIRA may integrate a **DHT** for simple name resolution/service discovery
- Supports scalable and efficient **topology discovery**

Backup Slides

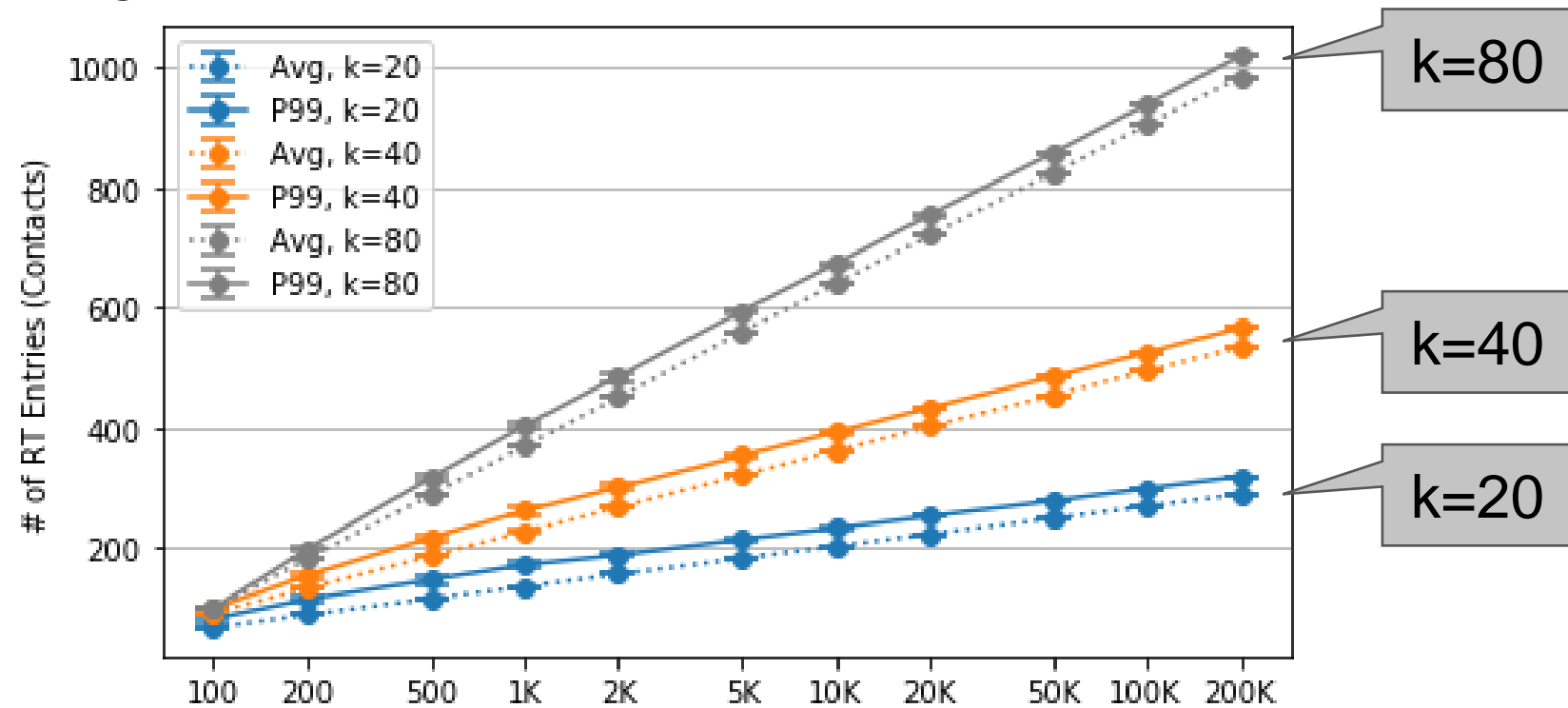
Further/Ongoing Work

- End-system mode variant
 - Saves a lot overhead
- Multi-path routing extension
- Efficient and lightweight topology discovery

- Multicast routing and forwarding
- Theoretical analysis
 - Guarantees and bounds
- Investigations w.r.t. mobility, ad-hoc/mesh networks
- Implementation in Rust

Evaluation – Scalability

Power-Law topologies

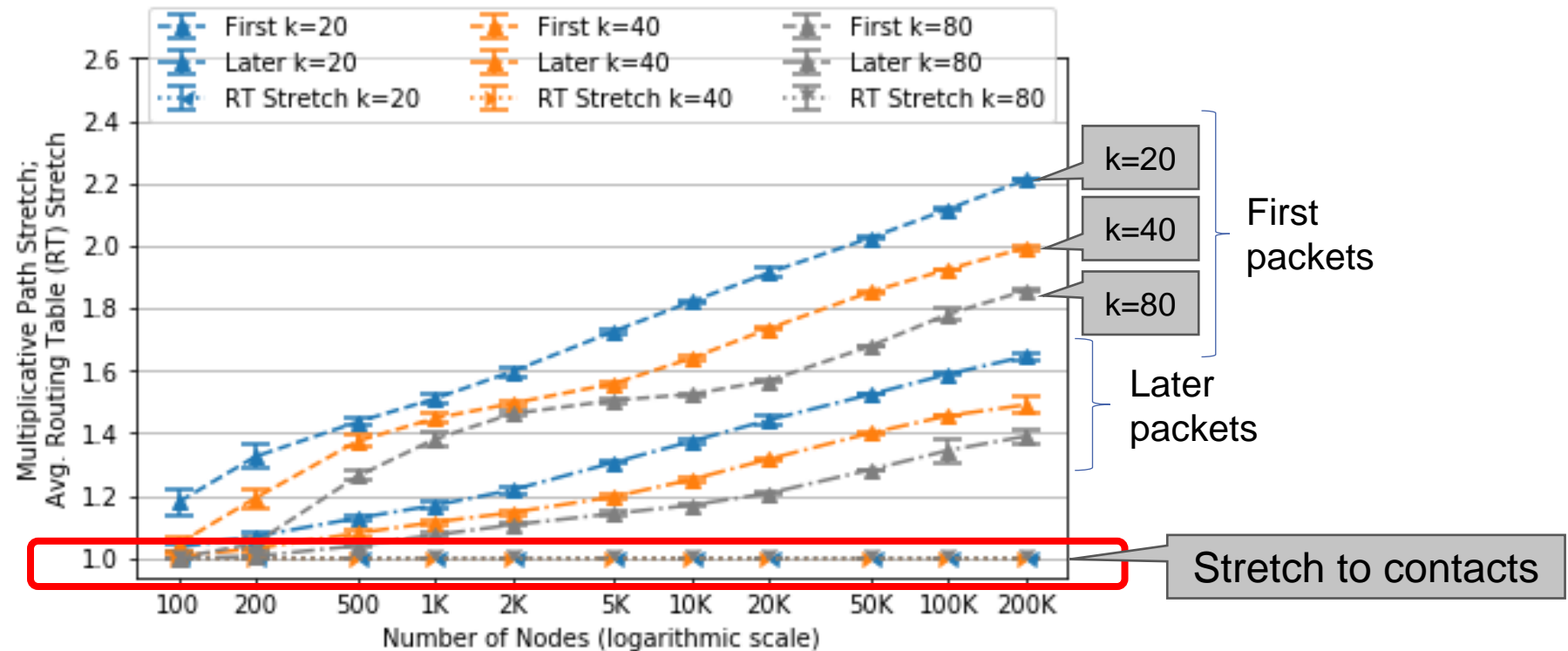


High Scalability: # of Routing Table Entries $\sim O(\log n)$

Evaluation – Stretch

■ Multiplicative path stretch

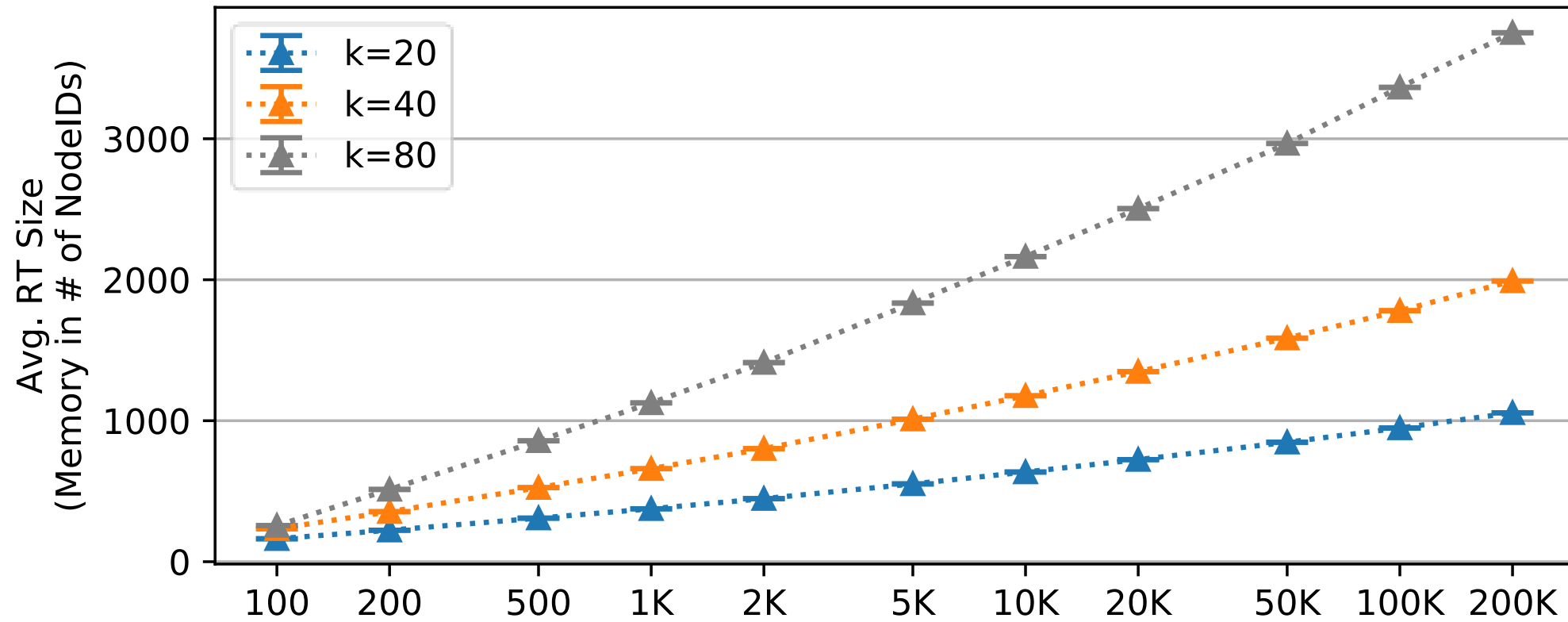
$$= \frac{|\text{selected path}|}{|\text{shortest path}|}$$



Flexible memory / stretch trade-off + Shortest paths to contacts

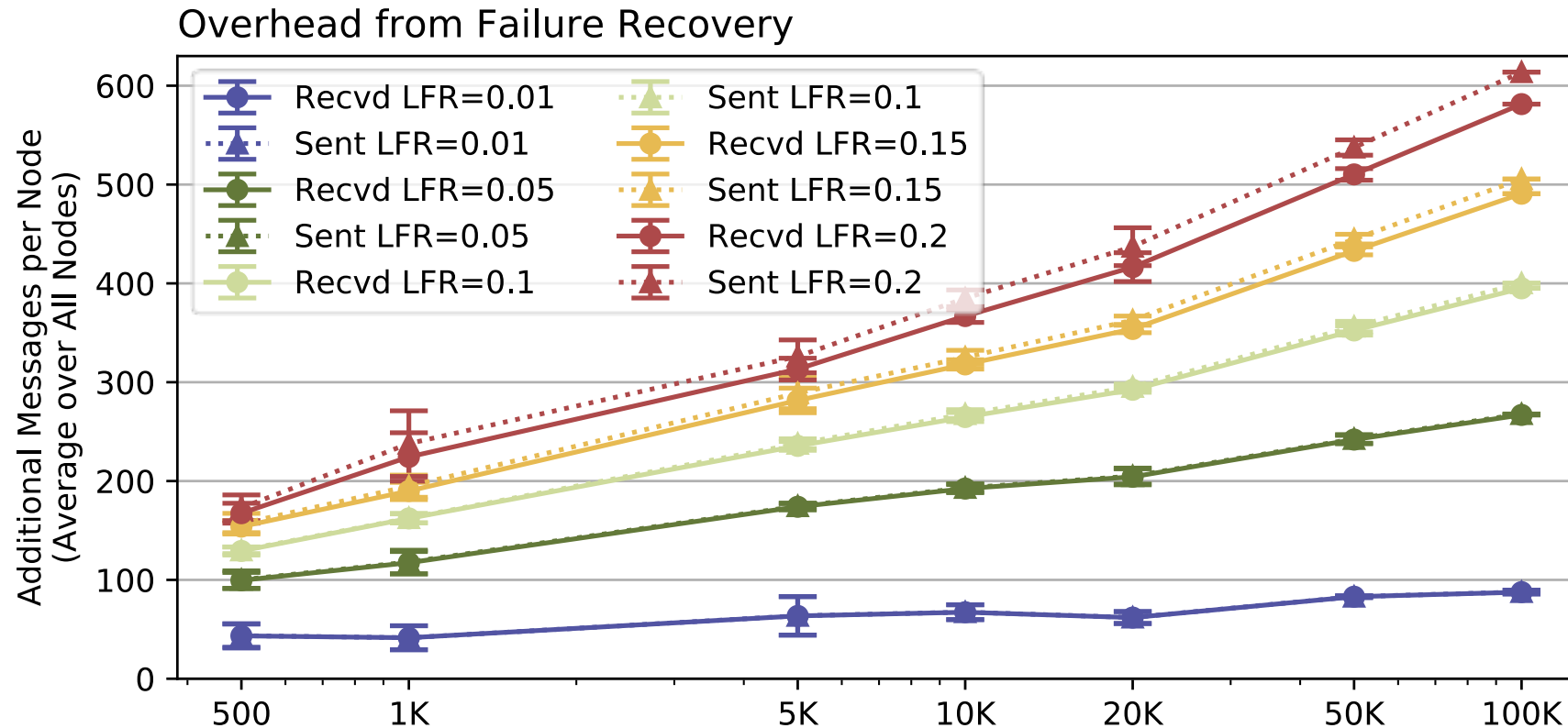
Memory Size in Number of IDs

■ Contacts + Path Vectors



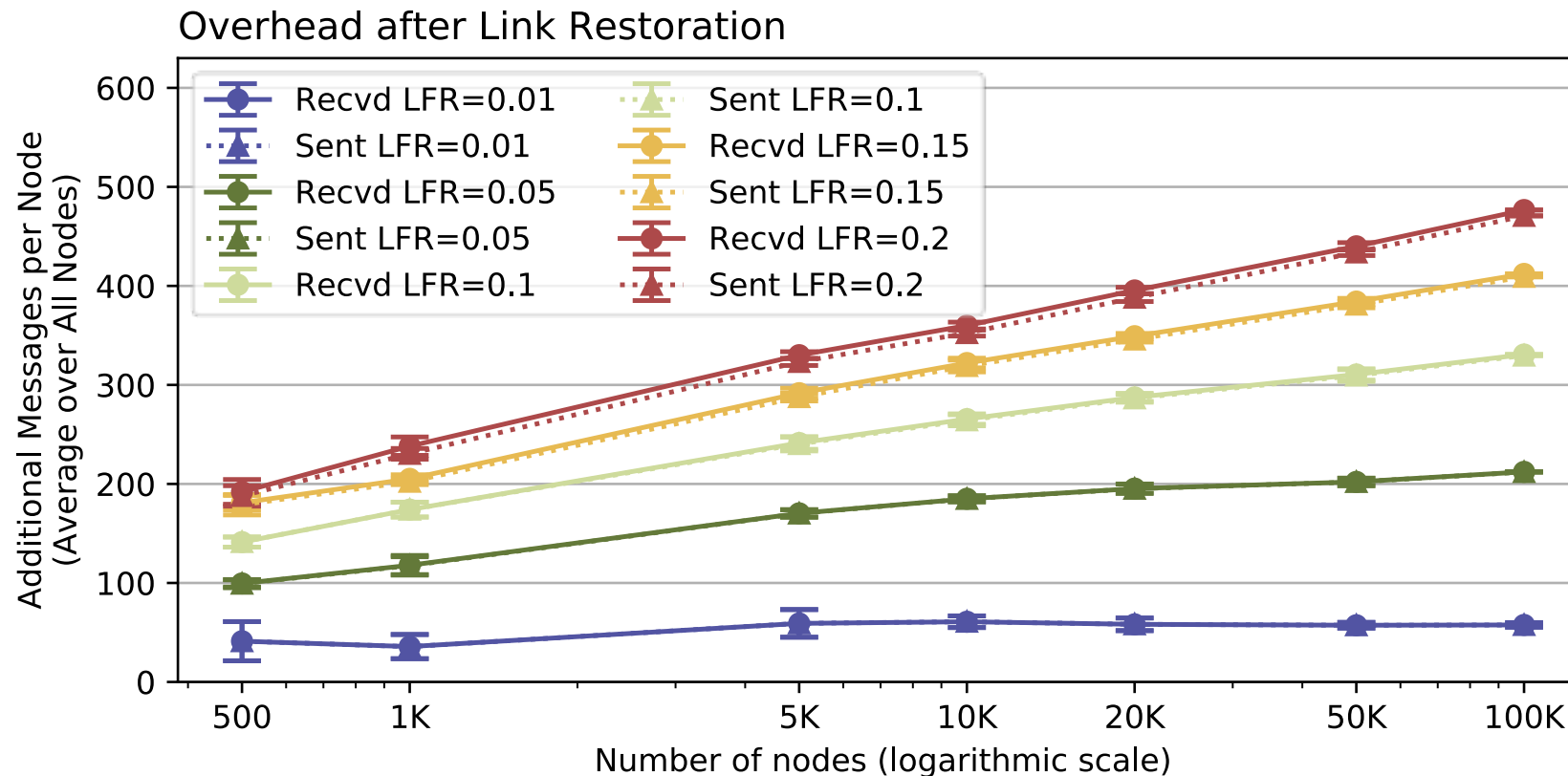
Dynamics – Control Message Overhead (1)

- Additional messages sent to recover from link failures

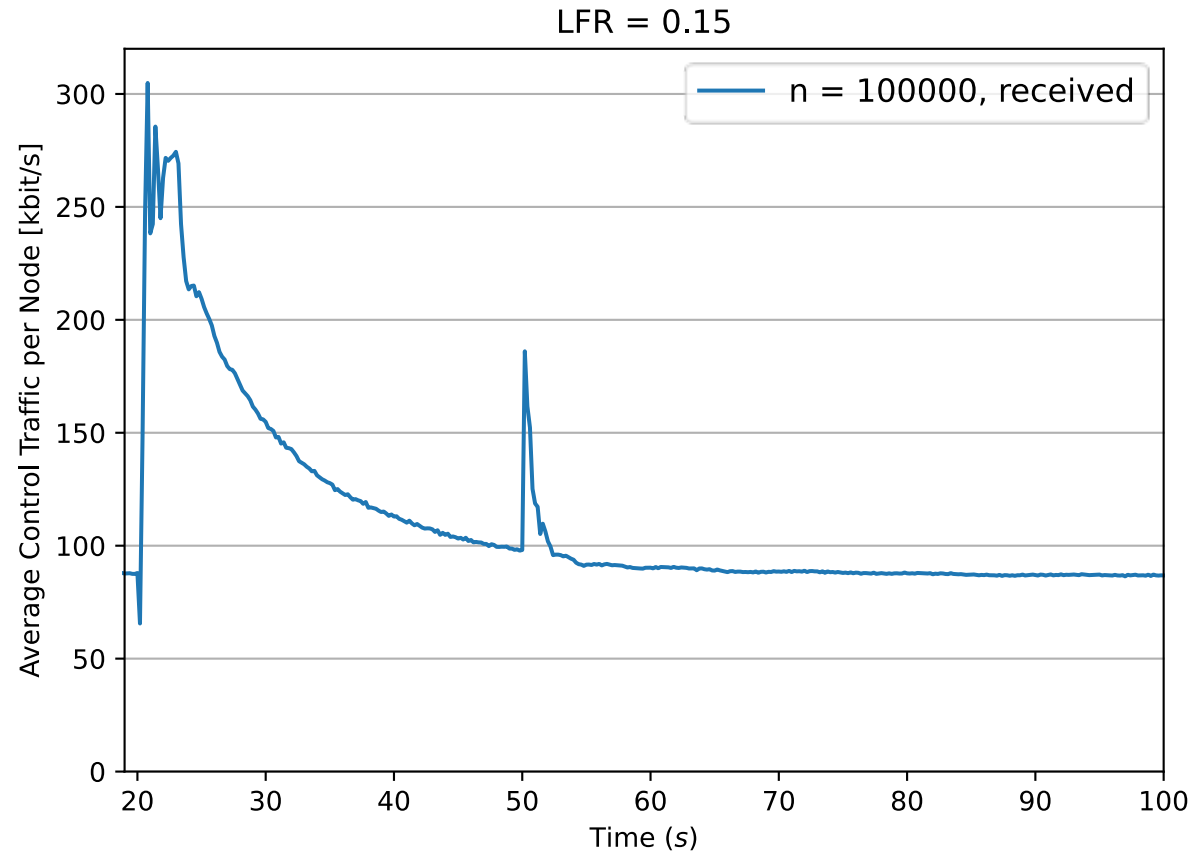


Dynamics – Control Message Overhead (2)

- Additional messages sent after failed links have been restored

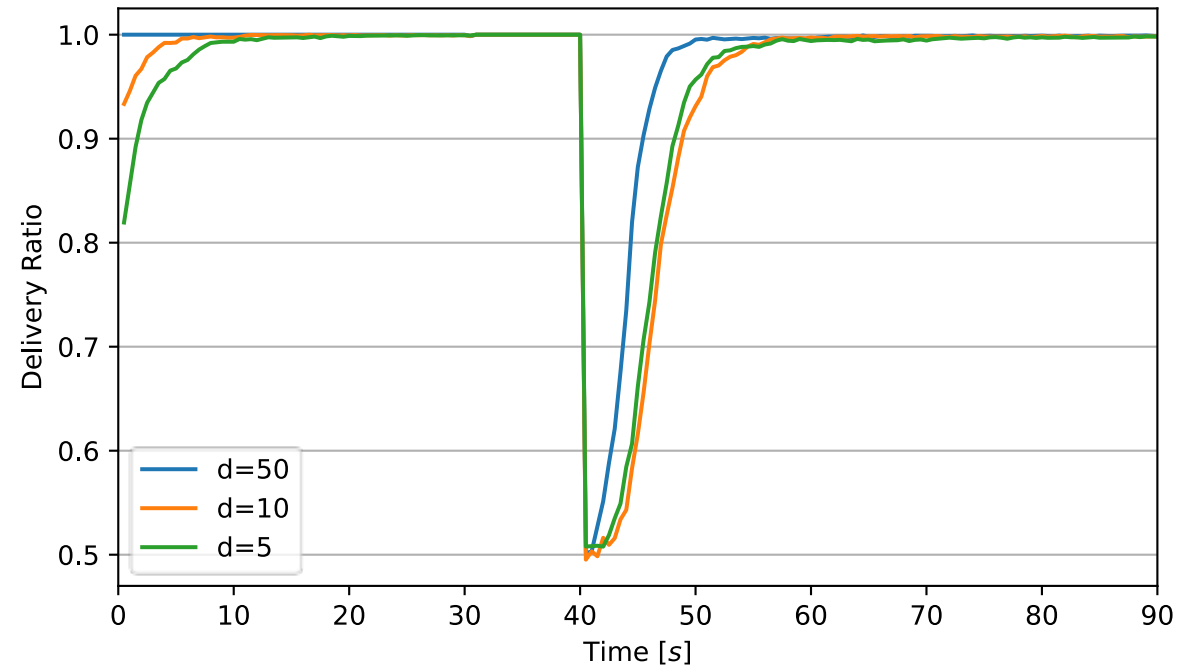


Control Traffic Data Rates



Network Partitioning

- Network partition 2 x 5000 nodes connected by d links
- $d \in \{5, 10, 50\}$
- D links break at 30s simultaneously, restored at 40s



State Sequence Numbers

- Per Node: **State Sequence Number**

- reflects changes in node's physical neighbor set
- Link down
- Link up (also detecting a new node)

- 32-bit

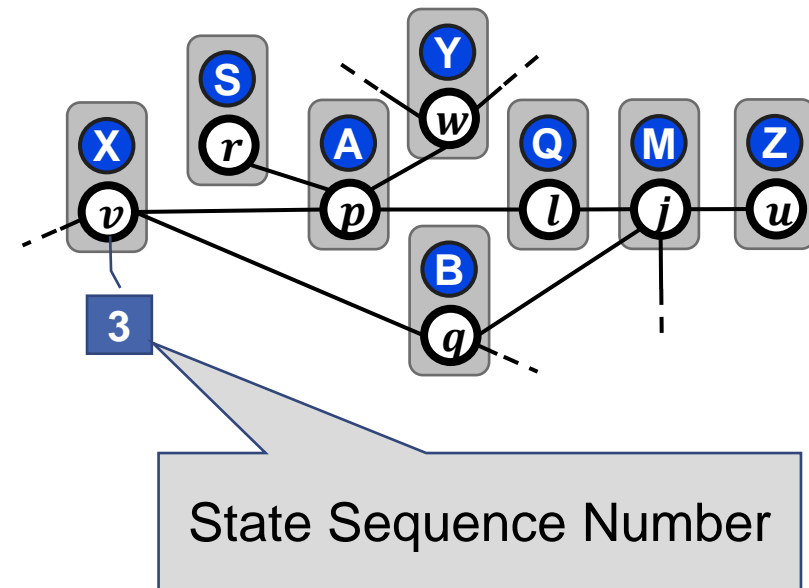
- Wrap around and special comparison:

- $s < s' \bmod 2^{32}$ if $0 < (s' - s) \bmod 2^{32} < 2^{31}$

- Get periodically synchronized

- Node crashes

- Node either uses new NodeID after restart
 - or, node stores NodeID and State Sequence Number across restart



State Sequence Numbers

- Per Node: **State Sequence Number**

- reflects changes in node's physical neighbor set
- Link down
- Link up (also detecting a new node)

- 32-bit

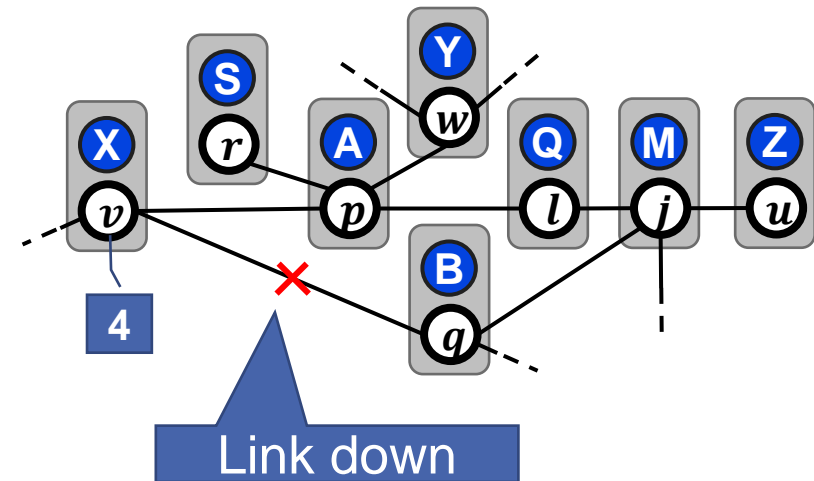
- Wrap around and special comparison:

- $s < s' \bmod 2^{32}$ if $0 < (s' - s) \bmod 2^{32} < 2^{31}$

- Get periodically synchronized

- Node crashes

- Node either uses new NodeID after restart
 - or, node stores NodeID and State Sequence Number across restart



State Sequence Numbers

- Per Node: **State Sequence Number**

- reflects changes in node's physical neighbor set
- Link down
- Link up (also detecting a new node)

- 32-bit

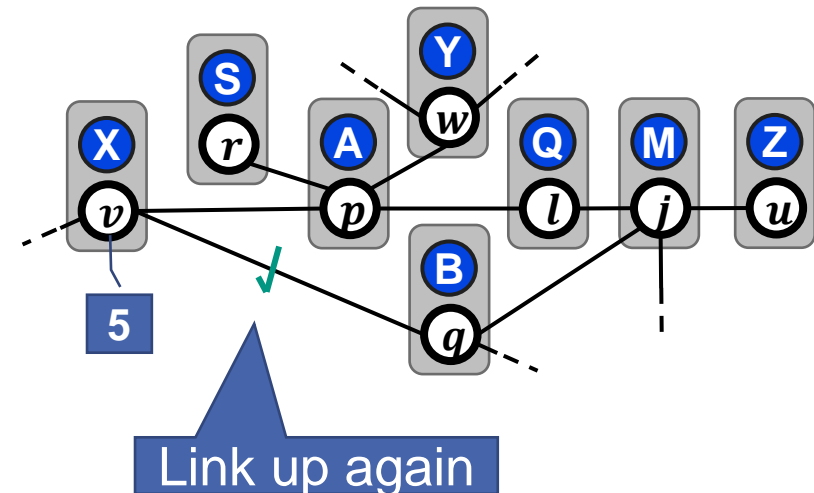
- Wrap around and special comparison:

- $s < s' \bmod 2^{32}$ if $0 < (s' - s) \bmod 2^{32} < 2^{31}$

- Get periodically synchronized

- Node crashes

- Node either uses new NodeID after restart
 - or, node stores NodeID and State Sequence Number across restart



End-system Mode

- End-systems do not route, but may be multi-homed and mobile
- Reduce overhead by not transmitting routing updates to/from end-systems
- Routers are responsible to keep information on end-system reachability