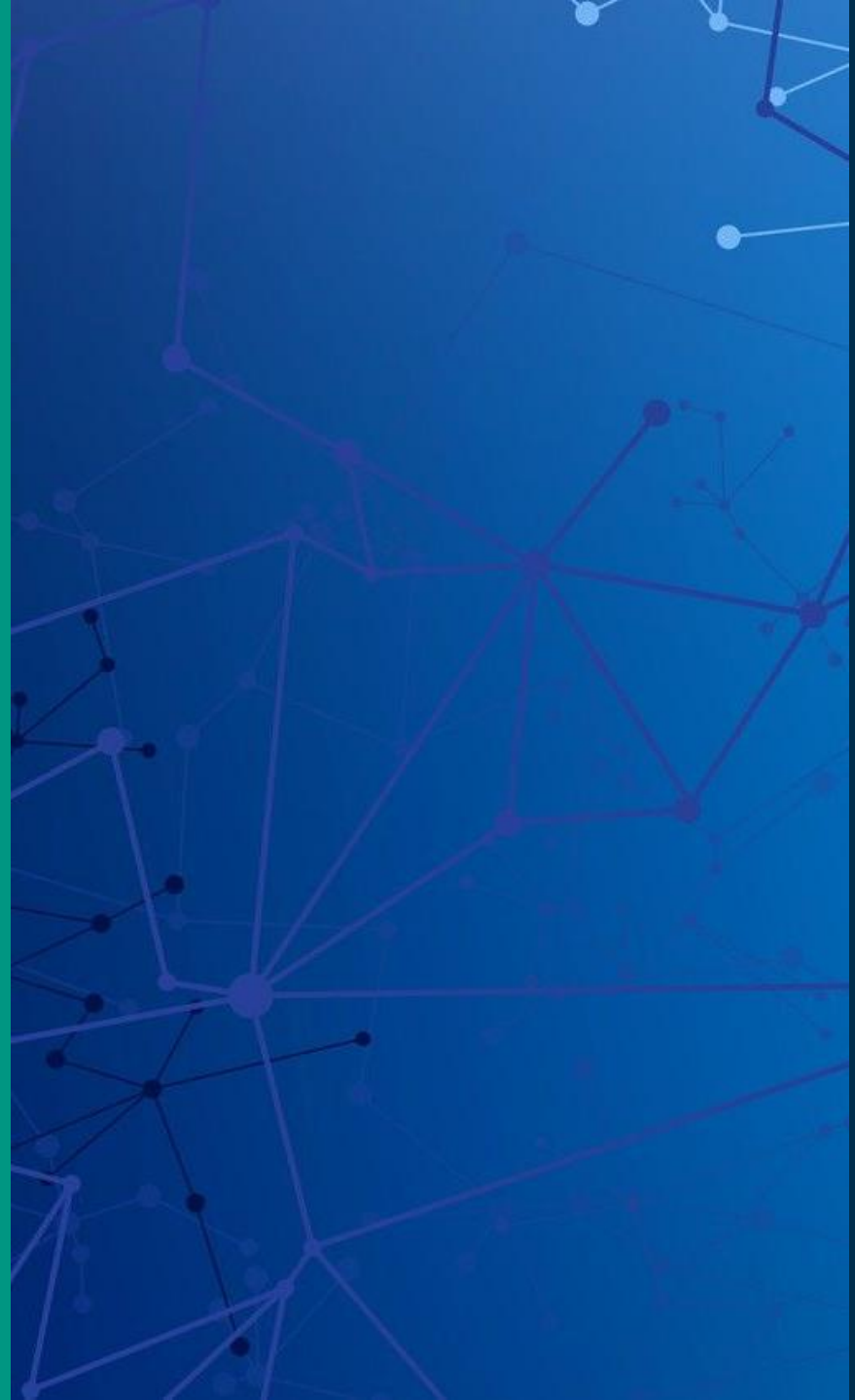
KIRA – A Scalable, Zero-Touch Routing Architecture for Resilient Control Planes



Dr. Roland Bless
Institute of Telematics
Karlsruhe Institute of Technology (KIT)
2026-07-23

Public Side Meeting – IETF Rules Apply

- **All public side meetings are subject to the IETF Note Well**
- Participants are expected to follow the usual IETF policies on personal conduct, IPR disclosure obligations, etc.
- You are contributing to IETF work
- Policies on patents and code of conduct apply
- Disclose relevant IPR or do not make contributions
- Be respectful and courteous even/especially when you disagree

Agenda

1. Introduction to KIRA (20–25min)
2. Questions & Answers
3. Discuss next steps
   - Supporting the IETF activities

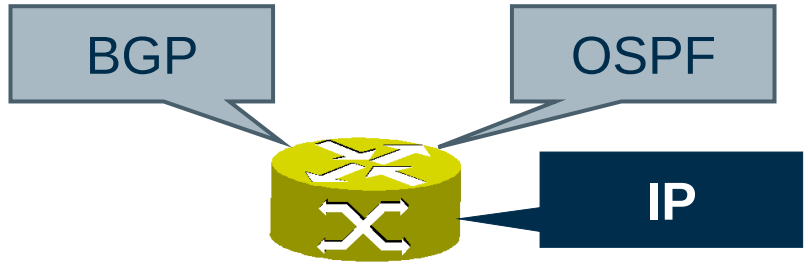
Network Infrastructures...

- are becoming more complex
  - higher interdependencies of services
- must be reliable → resilient operation
  - must be able to deal with unforeseen failures

Router

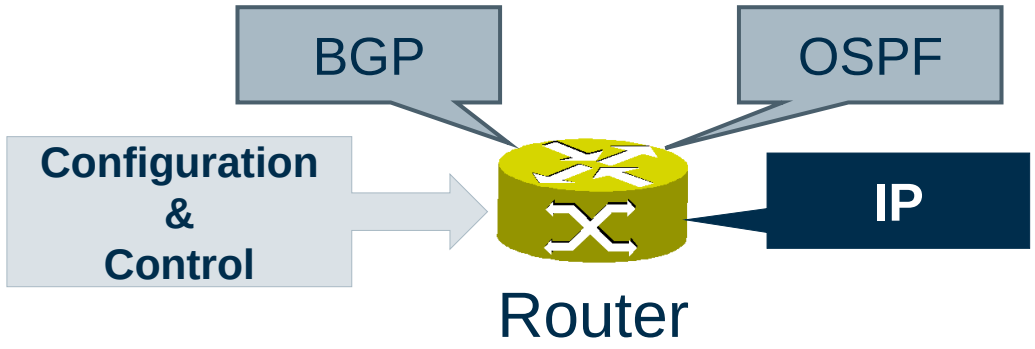
Network Infrastructures...

- are becoming more complex
  - higher interdependencies of services
- must be reliable → resilient operation
  - must be able to deal with unforeseen failures



Network Infrastructures...

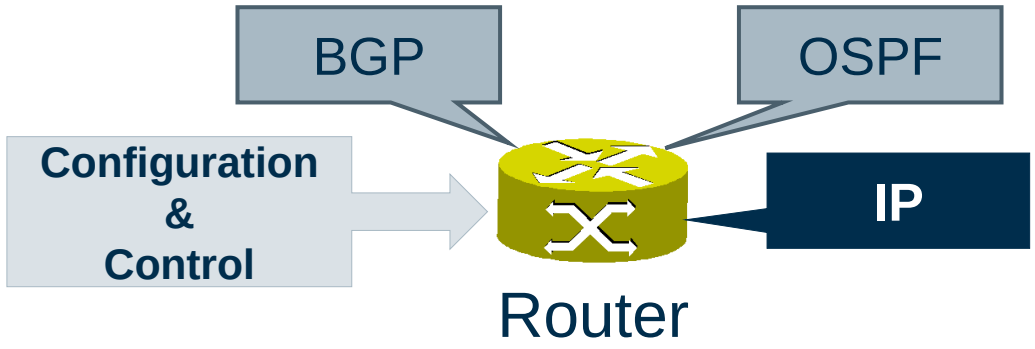
- are becoming more complex
  - higher interdependencies of services
- must be reliable → resilient operation
  - must be able to deal with unforeseen failures



requires configuration via
management/control plane network

Network Infrastructures...

- are becoming more complex
  - higher interdependencies of services
- must be reliable → resilient operation
  - must be able to deal with unforeseen failures



requires configuration via
management/control plane network

→ **Working Control Plane Network Connectivity = Foundation for Resilient Network Infrastructures**

Controllability and Control Planes

Major Service Disruptions

Meta

Oct 2021

aws

Dec 2021

Meta: “a command was issued […], which unintentionally took down all the connections in our backbone network, effectively disconnecting Facebook data centers globally. Our systems are designed to audit commands like these to prevent mistakes like this, but a **bug in that audit tool** prevented it from properly stopping the command. […] it was not possible to access our data centers through our normal means because their networks were down[…]. Our **primary and out-of-band network access was down**, so we **sent engineers onsite** to the data centers to have them debug the issue and restart the systems.
But this took time, …”

https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/

Controllability and Control Planes

Major Service Disruptions

Meta

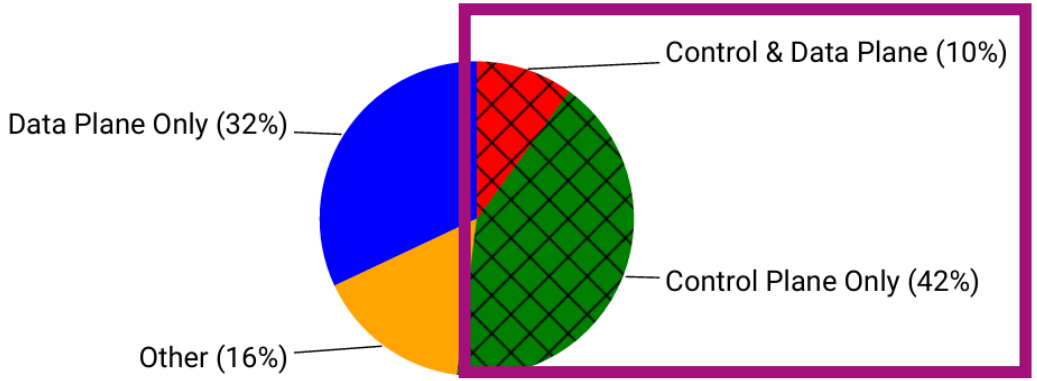
Oct 2021

aws

Dec 2021

Services depend on resilient connectivity

Control plane connectivity inherently important



Root-cause for the 41 largest outages over 4 years in Google's B4

A Decentralized SDN Architecture for the WAN

Alexander Krentsel, Google / UC Berkeley
Nitika Saran*, Cornell
Bikash Koley, Google
Subhasree Mandal, Google
Ashok Narayanan, Google
Sylvia Ratnasamy, Google / UC Berkeley
Ali Al-Shabibi, Google
Anees Shaikh, Google
Rob Shakir, Google
Ankit Singla, Google
Hakim Weatherspoon, Cornell
dsdn-sigcomm@google.com

SIGCOMM 2024

Infrastructure Complexity

A Decentralized SDN Architecture for the WAN

Alexander Krentsel, Google / UC Berkeley
Nitika Saran*, Cornell
Bikash Koley, Google
Subhasree Mandal, Google
Ashok Narayanan, Google
Sylvia Ratnasamy, Google / UC Berkeley
Ali Al-Shabibi, Google
Anees Shaikh, Google
Rob Shakir, Google
Ankit Singla, Google
Hakim Weatherspoon, Cornell
dsdn-sigcomm@google.com

SIGCOMM 2024

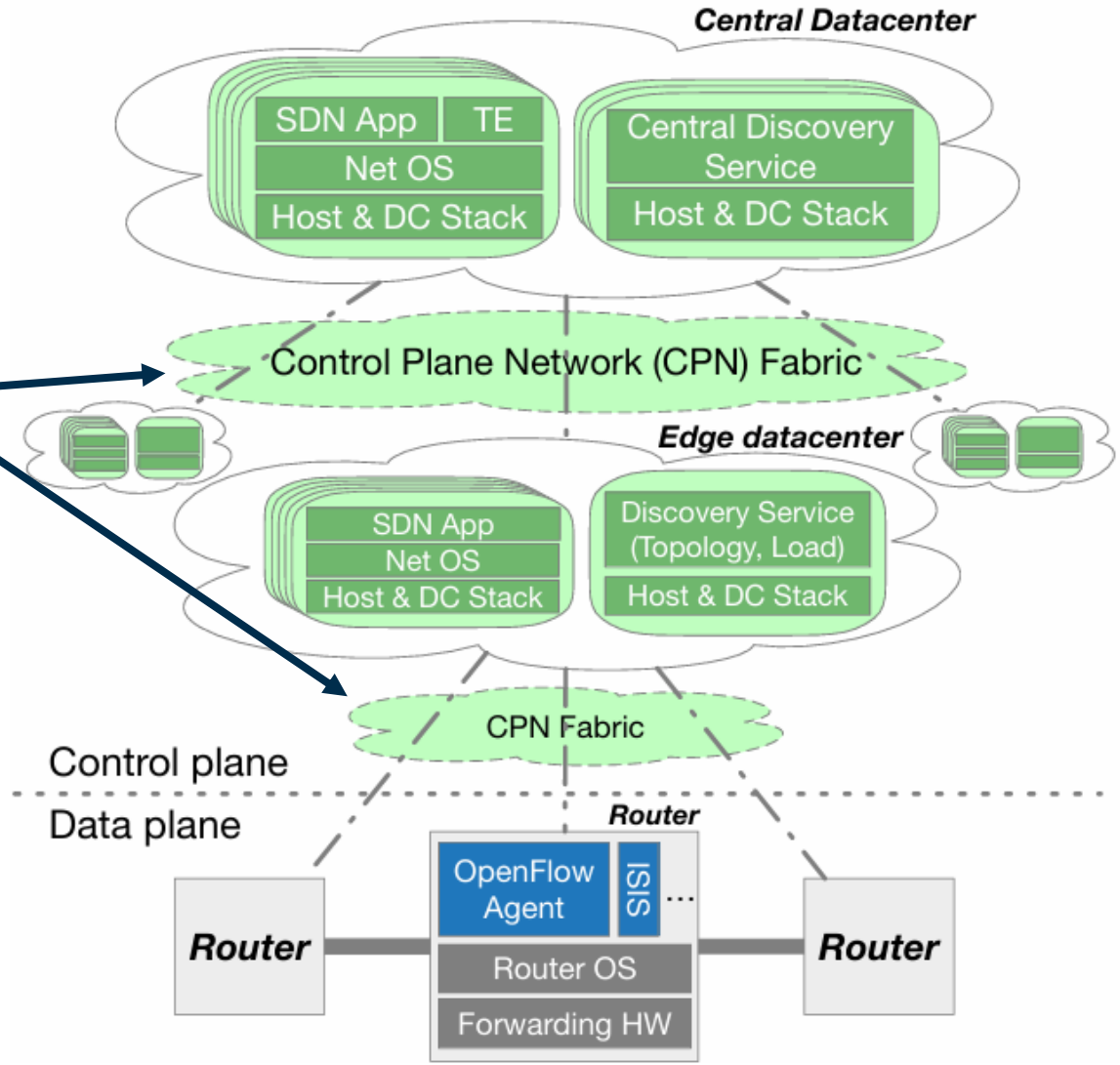
*"[Major failures] typically involve bugs or errors in multiple components […], that **interact in unanticipated** ways."*

"External infrastructure increases the surface area for bugs"

"Major outages continue to be reported by virtually every WAN operator [--]"

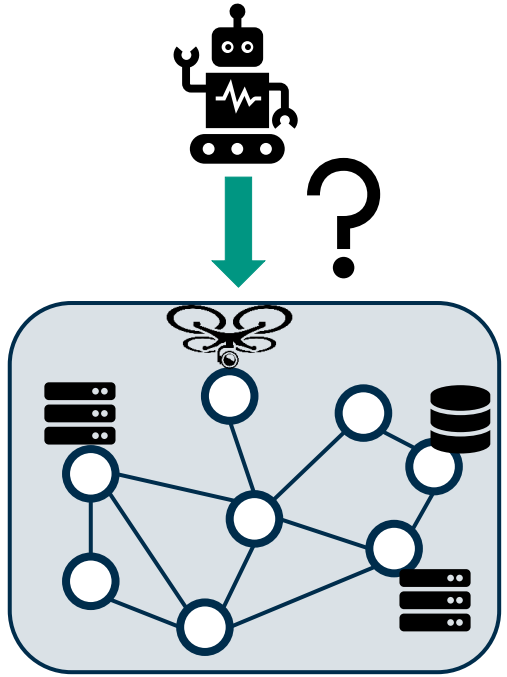
"[…] these outages persist, despite our extensive efforts to improve testing, diagnostics, change procedures, and verification"

"…address complexity directly and focus on simplifying the network."



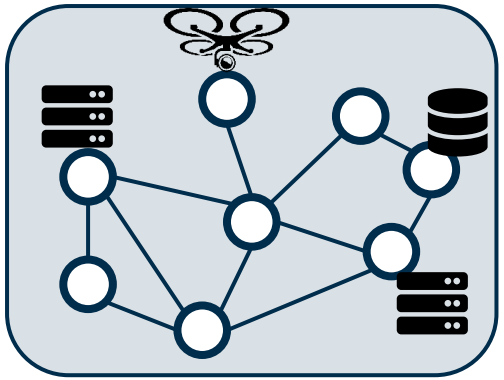
AI-based Network Management?

- AI even adds more complexity on top of it all!
- AI cannot help to get control back over a networked resource when it lost connectivity to it
- Using KIRA's connectivity you/AI can always revert to a sane system state
- Less restrictive than adding many checks and bounds



Control Planes of Future Networks Need to Support...

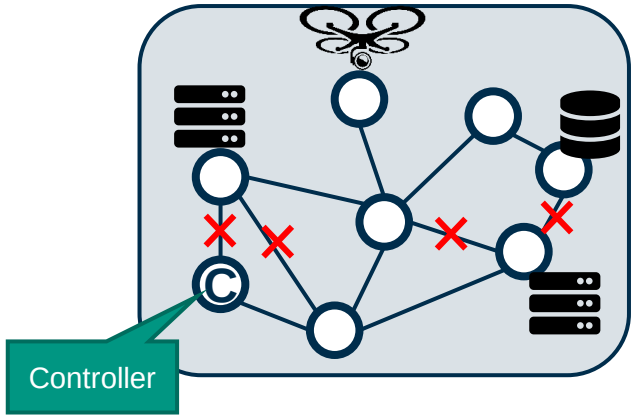
Interconnection of a Large Pool of Networked Resources



Compute, Storage, Network

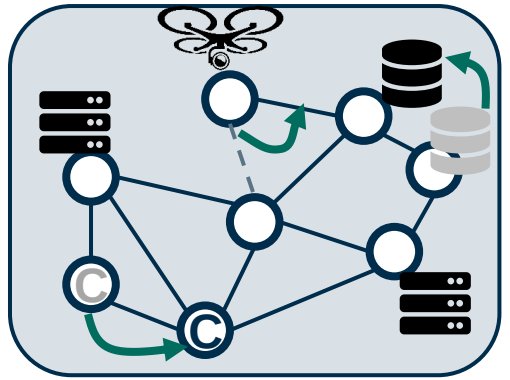
- Scalability
- Efficient routes
- In-band control
- High dynamics
- Multiple domains
- Various topologies

Resilient Connectivity for Control Plane



- Zero-touch
- Fast convergence
- Network split
- Nomadic networks

Stable Addresses for Moving Resources



- ID-based addresses

- In-band: no separate physical control network, re-uses existing data plane connections
- Out-of-band: separate dedicated network for control

The Approach: KIRA

Kademlia-directed ID-based Routing Architecture

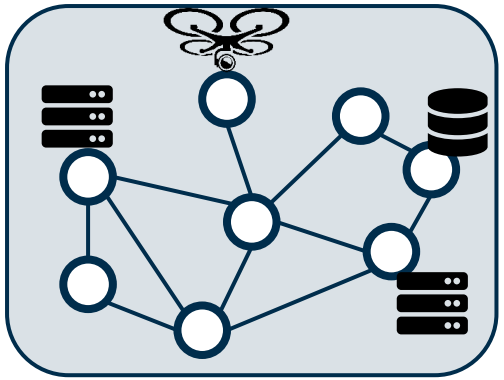
R. Bless, M. Zitterbart, Z. Despotovic and A. Hecker, “KIRA: Distributed Scalable ID-based Routing with Fast Forwarding”, 2022 IFIP Networking Conference

- Provides **zero-touch resilient** control plane connectivity (IPv6)
- No further dependencies for in-band, out-of-band or hybrid control plane connectivity
- Connectivity invariant: other network services can always be restarted/restored using KIRA’s connectivity

- In-band: no separate physical control network, re-uses existing data plane connections
- Out-of-band: separate dedicated network for control

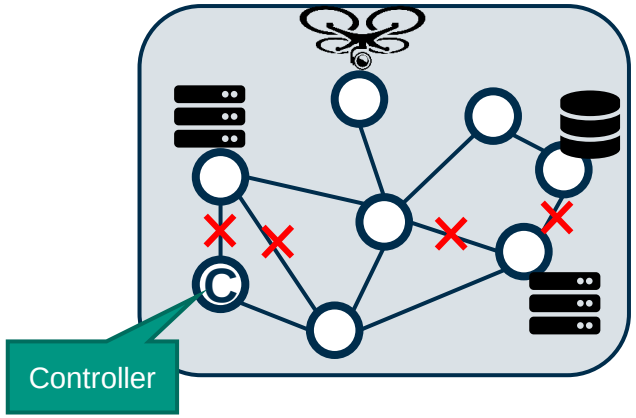
What KIRA achieves...

Interconnection of a Large Pool of Networked Resources

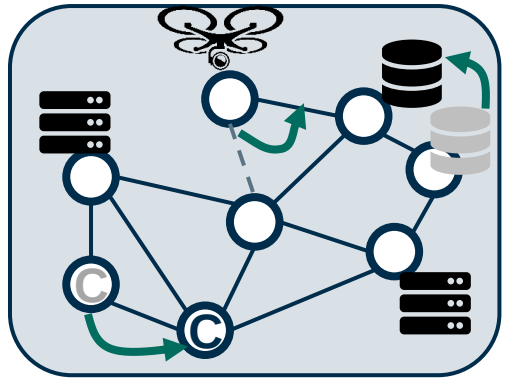


Compute, Storage, Network

Resilient Connectivity for Control Plane



Stable Addresses for Moving Resources



KIRA provides (all-in-one)

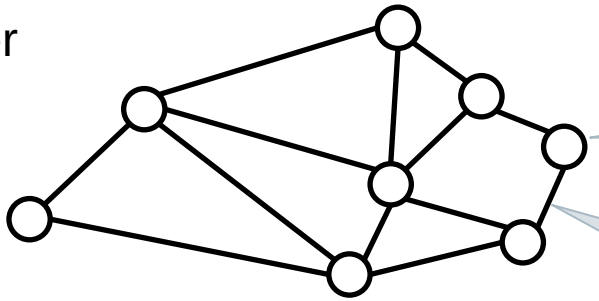
- Massive scalability (100,000s of nodes)
- Zero-touch (no configuration + adaptation)
- Dynamics: fast convergence, loop free, fast reroute
- Topological versatility
- Efficient routes

Related Works (examples)

- UIP: lacks dynamics and efficient routes
- DISCO: lacks dynamics
- RIFT, Data Center BGP/OSPF/IS-IS: specific topologies only
- RPL: traffic concentration near root, zero-touch?

What KIRA provides...

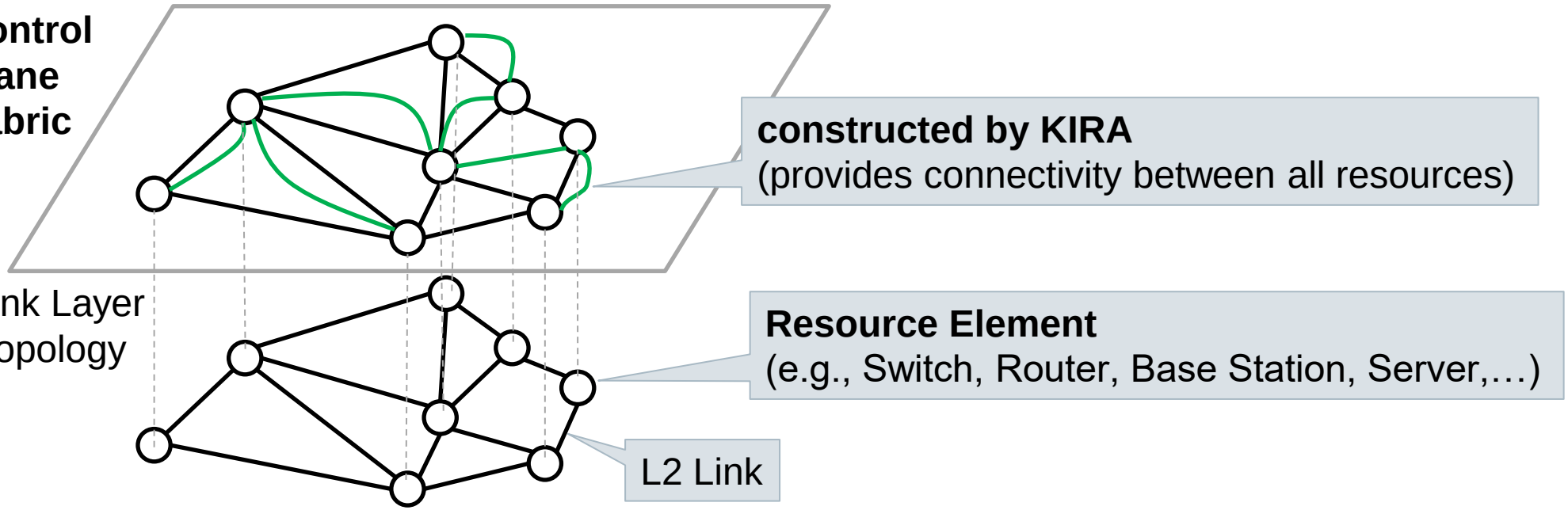
Link Layer
Topology



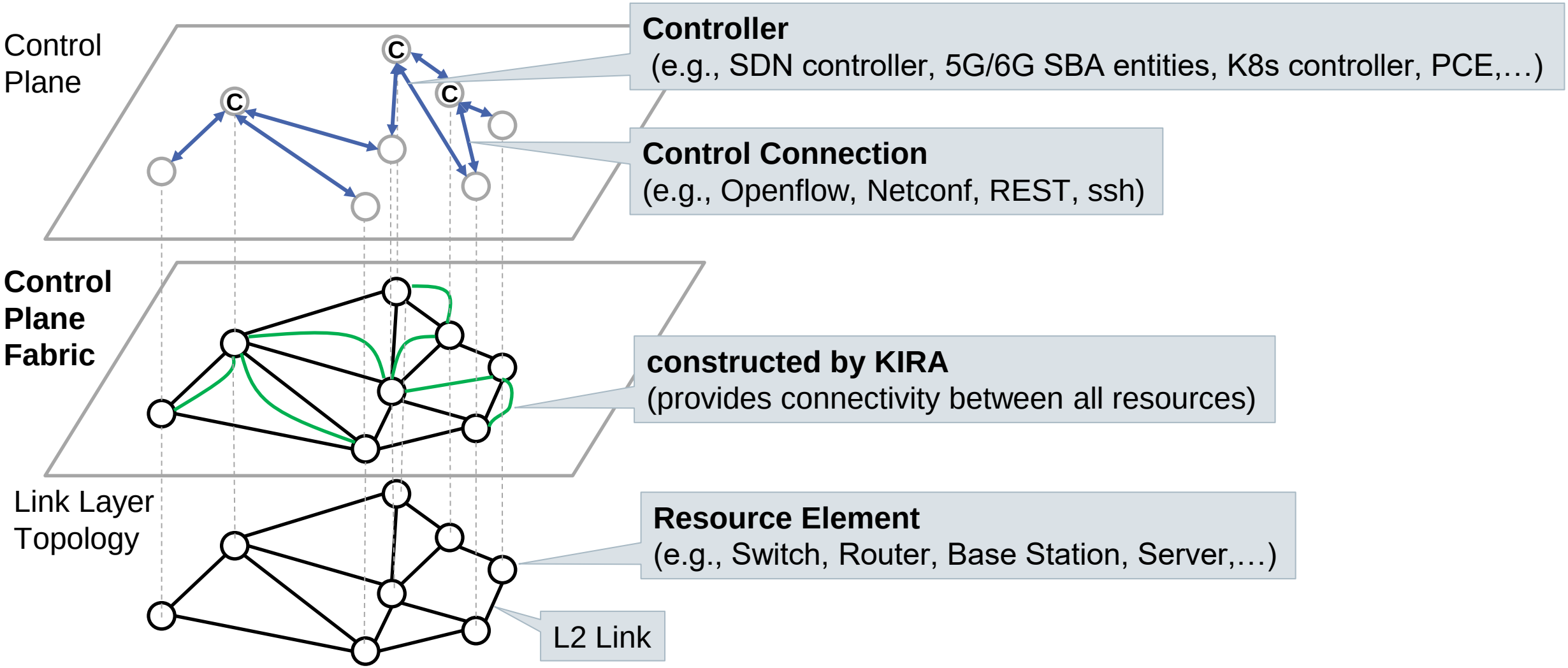
Resource Element
(e.g., Switch, Router, Base Station, Server,…)

L2 Link

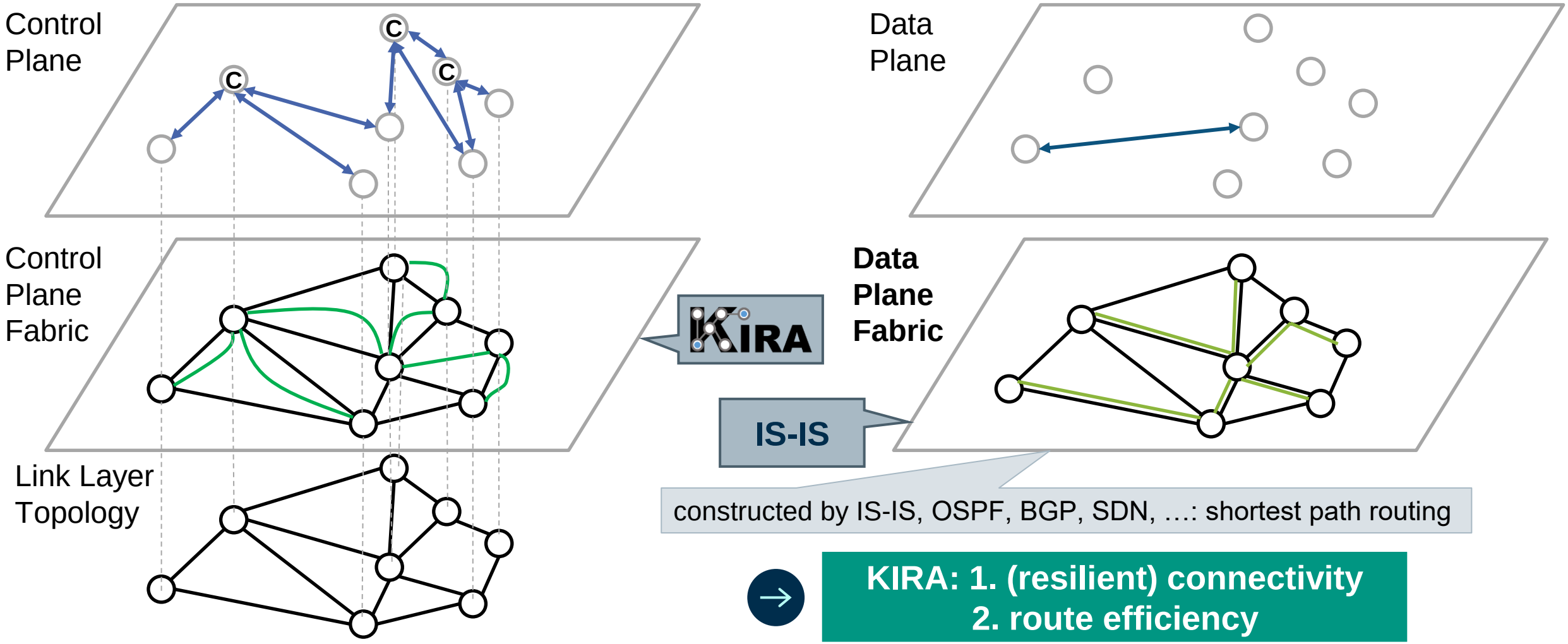
What KIRA provides...



What KIRA provides...

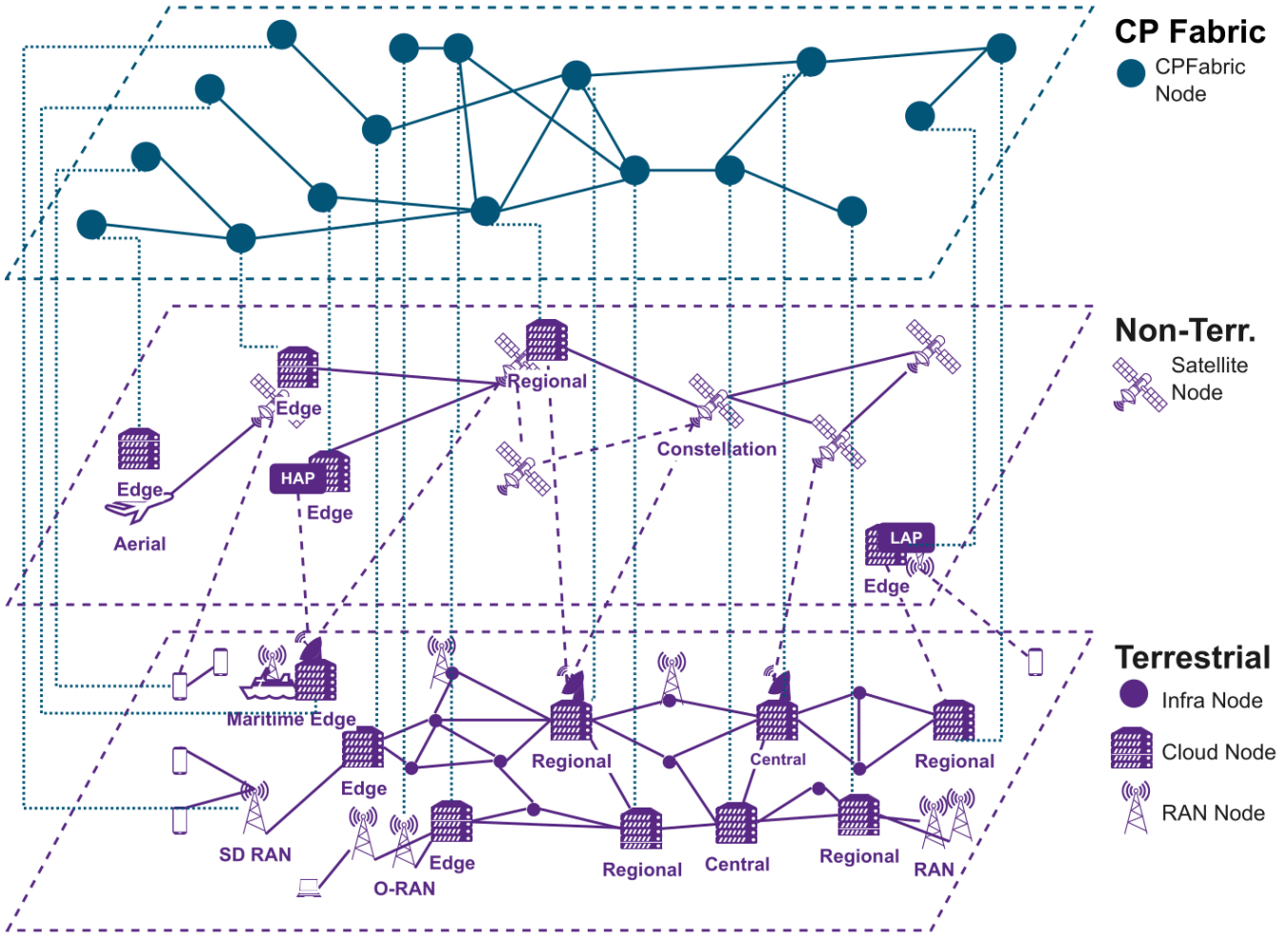


What KIRA provides...



Use Case – 6G Control Plane

- Non-terrestrial Networks (Drones, Satellites)
  → dynamic and mobile
- Nomadic Networks
  → autonomous,
  self-organizing control plane
- 10^6 of base stations in China
  in a single provider network
  → scalability
- Single routing solution
  that interconnects everything
  for the Control Plane

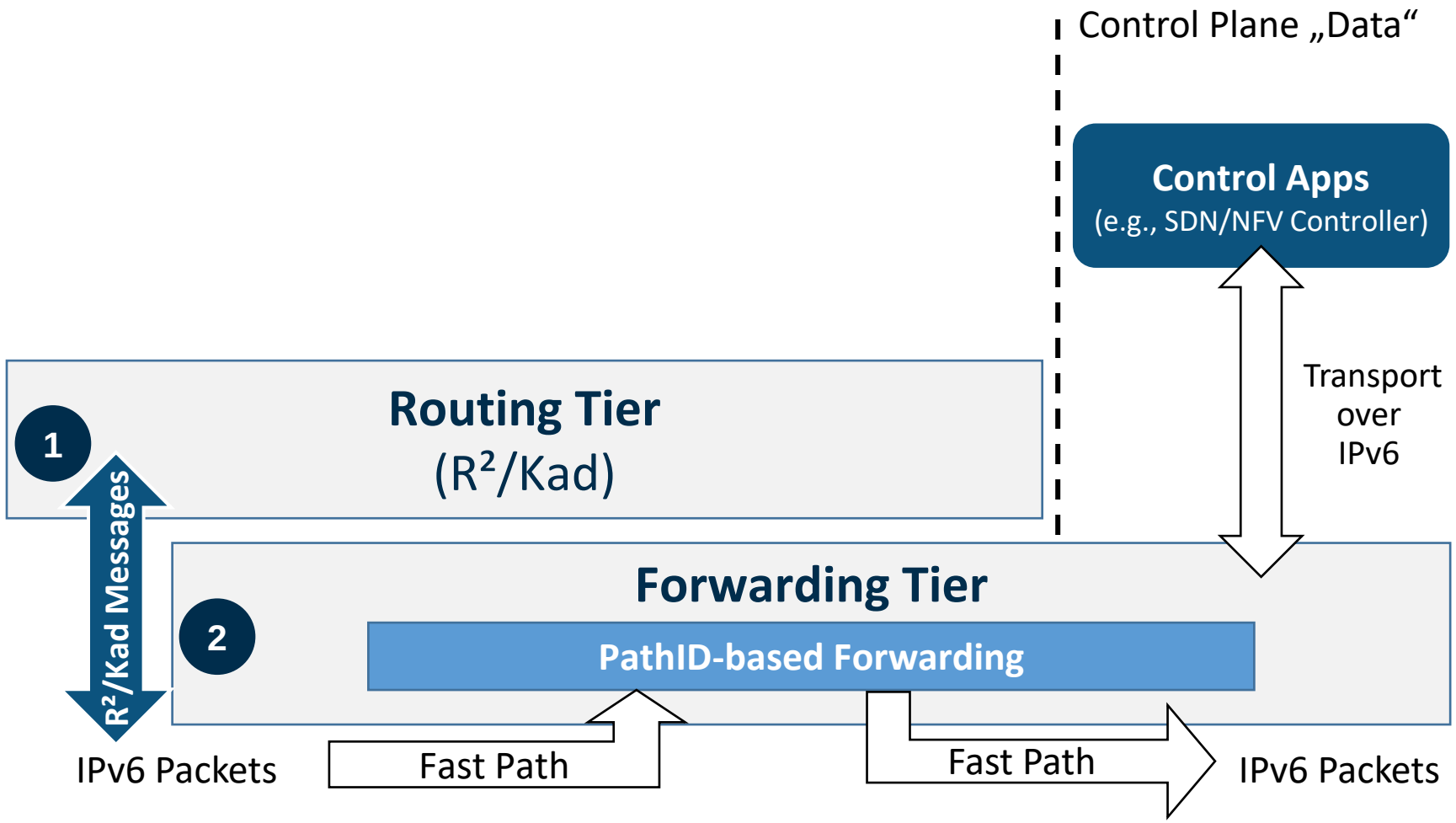


https://ieeexplore.ieee.org/document/10175535

Features

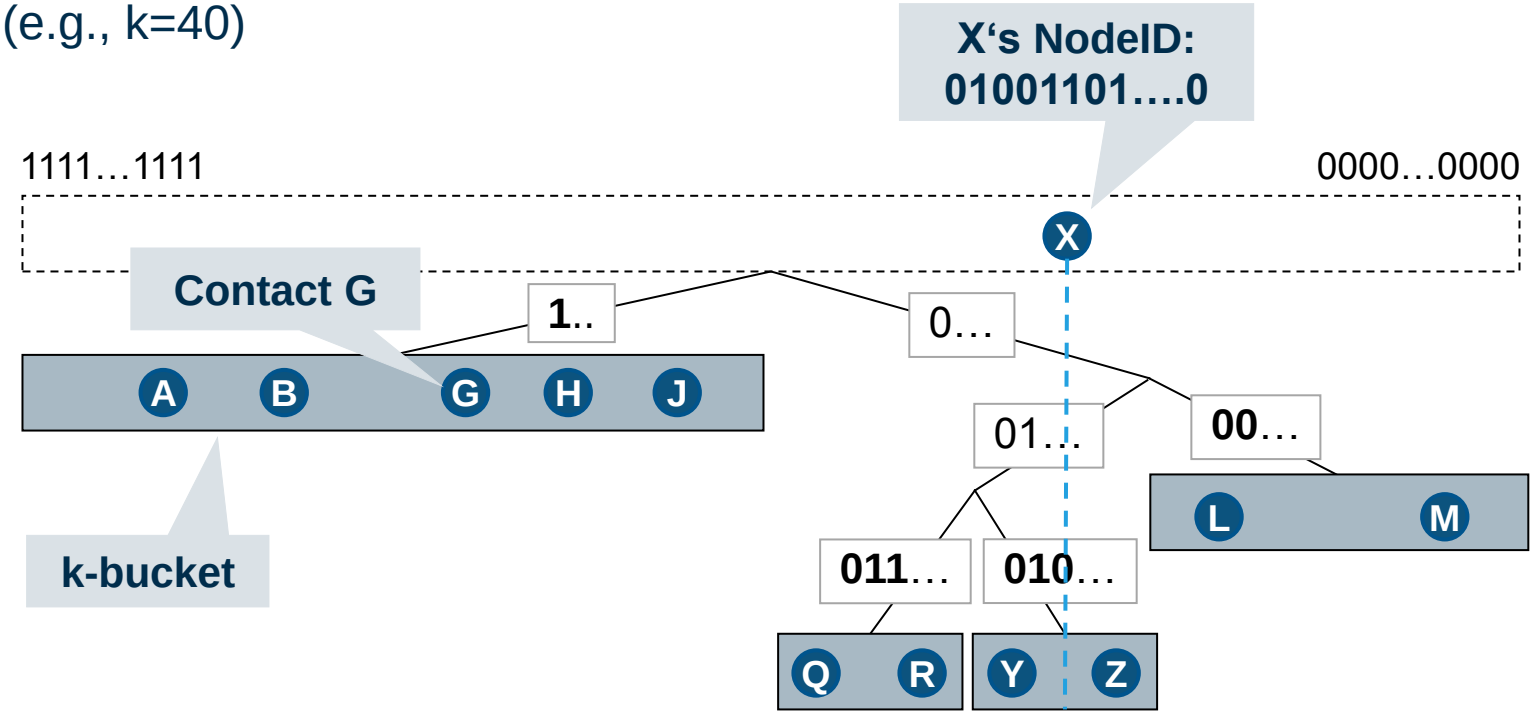
- Kademlia as ID-based Overlay (XOR metric)
  - 112bit NodeID as address → IPv6 address w/ 16bit prefix
  - Small routing tables – $O(\log n)$ n: number of nodes
    → Route stretch as trade-off
  - Supports multi-homing and mobility
  - No Locators, genuine ID-based routing (no mapping system)
- Route efficiency by Proximity Neighbor Selection and Proximity Routing
- Path Discovery (First Path, Later Path) + Path Rediscovery (Dynamics)
- Loop-free even during convergence
- Source routing for routing messages + path validation
  - path-based approaches: expressive, consensus-free
- PathID-based forwarding for control plane data
  - no special forwarding hardware required (IPv6 support, ideally SRv6)

Architecture



Routing Tier – Routing Table

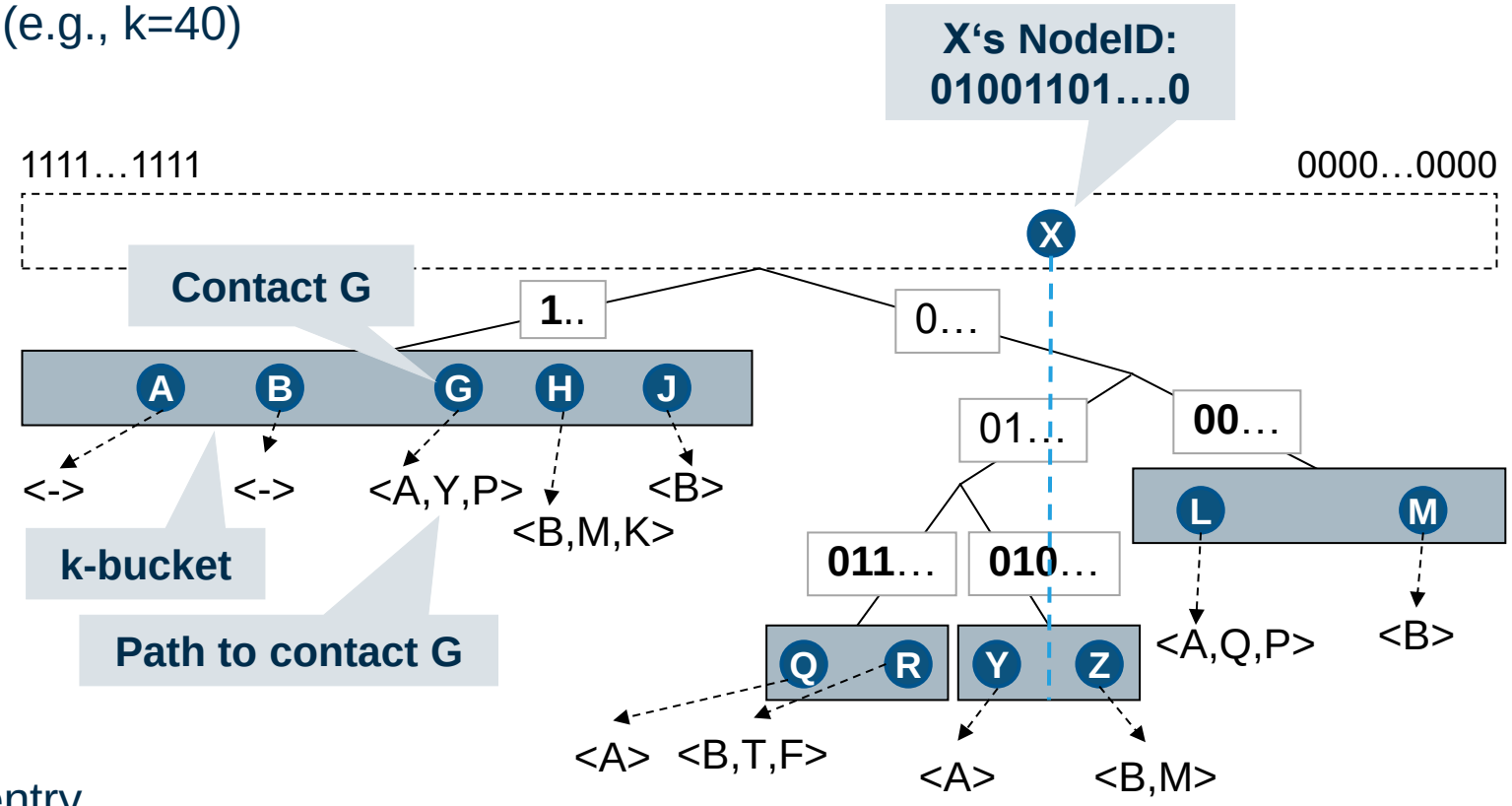
- Tree of buckets holding up to k contacts (e.g., k=40)
  - Arranged by XOR distance
  - Bucket split if it contains own NodeID



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

Routing Tier – Routing Table

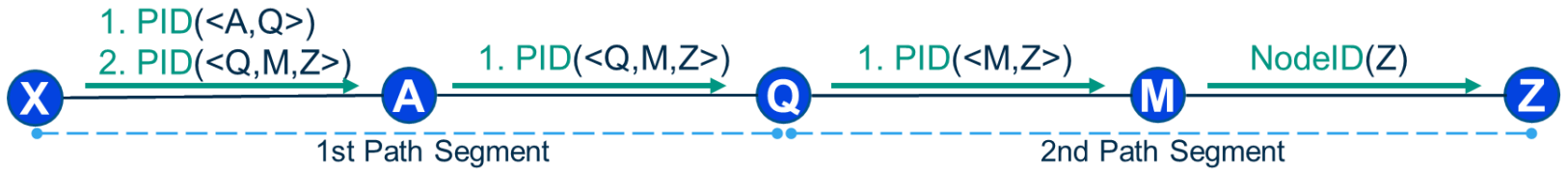
- Tree of buckets holding up to k contacts (e.g., k=40)
  - Arranged by XOR distance
  - Bucket split if it contains own NodeID
  - ID-wise neighbors in deepest bucket
- Path vectors are discovered and stored for each contact
- Efficient routes
  - Shorter routes preferred
- Size k of k-buckets can be set per node
  → flexible memory / stretch trade-off
- Each node routes to XOR-wise closest entry



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

Forwarding Tier (Forwards Control Plane Data)

- Approach: replace source routes with PathIDs
  - PathID(<A,Q,M,Z>)= Hash(A | Q | M | Z)
- Need PathID in addition to NodeIDs → encapsulation (e.g., SRv6, GRE)
- Use PathID as unique label for path segment → “Label Switching”
- Avoid path setup: precalculate PathIDs for 2-hop (underlay) vicinity
- Use two path segments at most per source route (better aggregation)
  - Explicit path setup for paths ≥ 6 hops

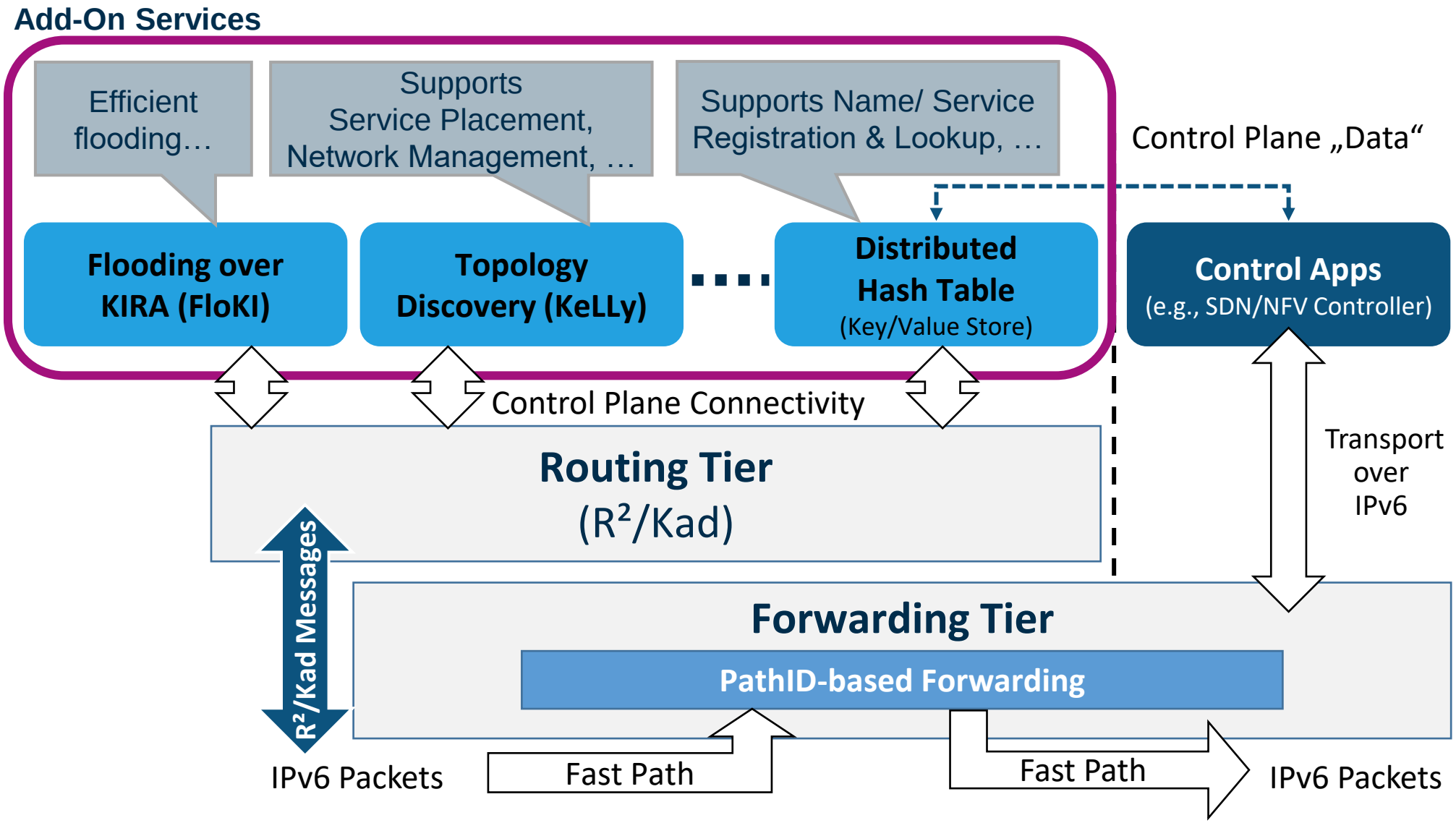


- Forwarding State – Cost?
  - 100,000 nodes powerlaw topology: 99th percentile fwd table ~ 2500 entries
- Uses existing forwarding plane mechanisms
  - Works with Longest Matching Prefix, nftables, SDN, eBPF etc.

Further Features (Continued)

- Fast Failover Mechanisms
  - Fast Next Hop Detour (Forwarding Tier)
  - Fast Vicinity Alternatives (Routing Tier)
- Multi-path capable
  - due to small routing tables and source routing
- Supports different routing metrics
- Built-in route flapping prevention
- Path validation of indirectly learned paths
- Special end-system mode (non-routing nodes)
- Supports Domains (topological, organizational)
- Can provide additional services like
  - Key-Value store, name registration and resolution service (DHT)
  - KeLLy – scalable and efficient topology discovery
  - FloKI – scalable and efficient flooding

Architecture

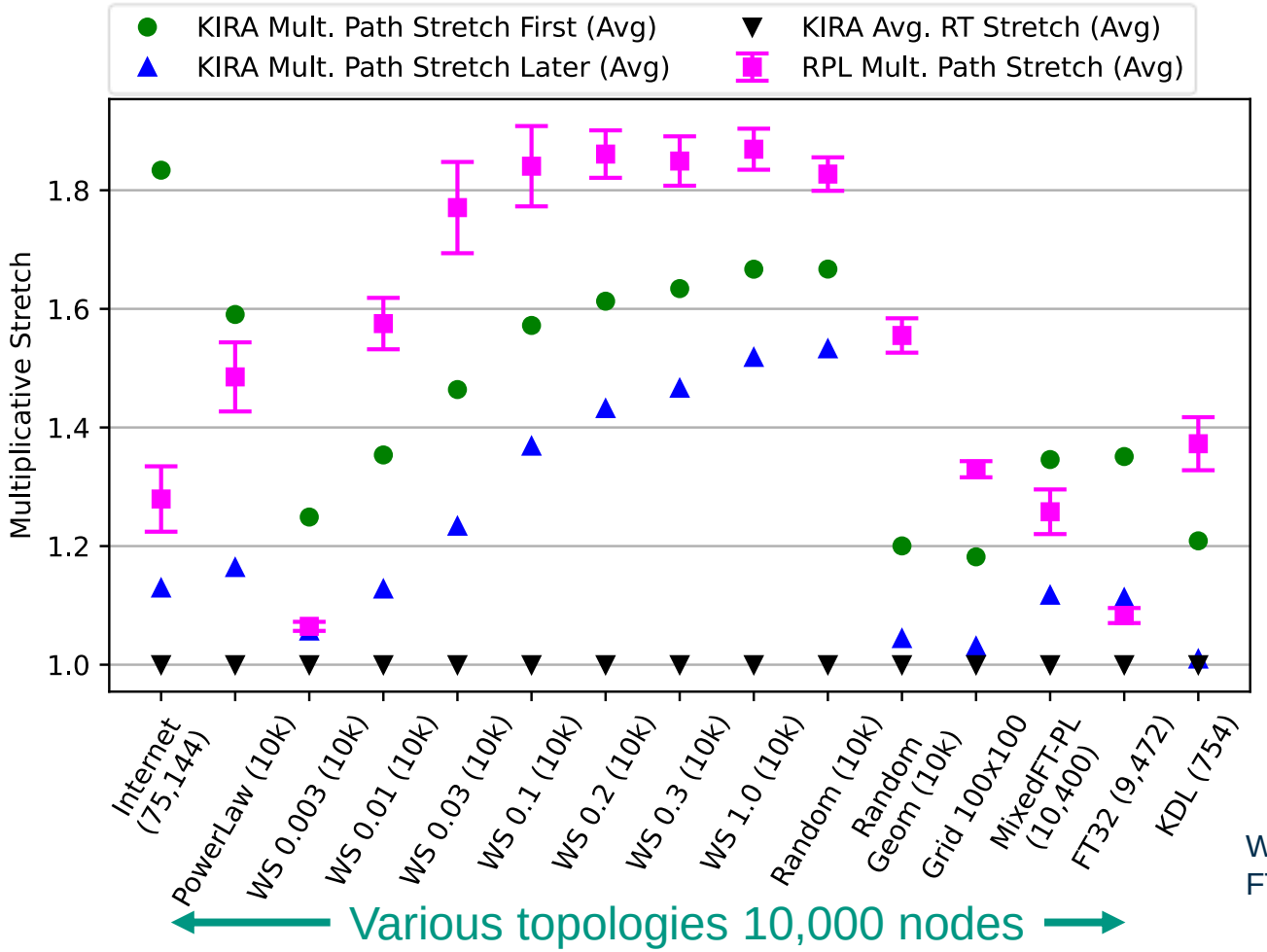


Stretch in Different Fixed Network Topologies

Multiplicative Stretch:
1.2 → path 20% longer than shortest path

RPL-ACP:

- Storing-mode
- Single DODAG
- Single DODAG version



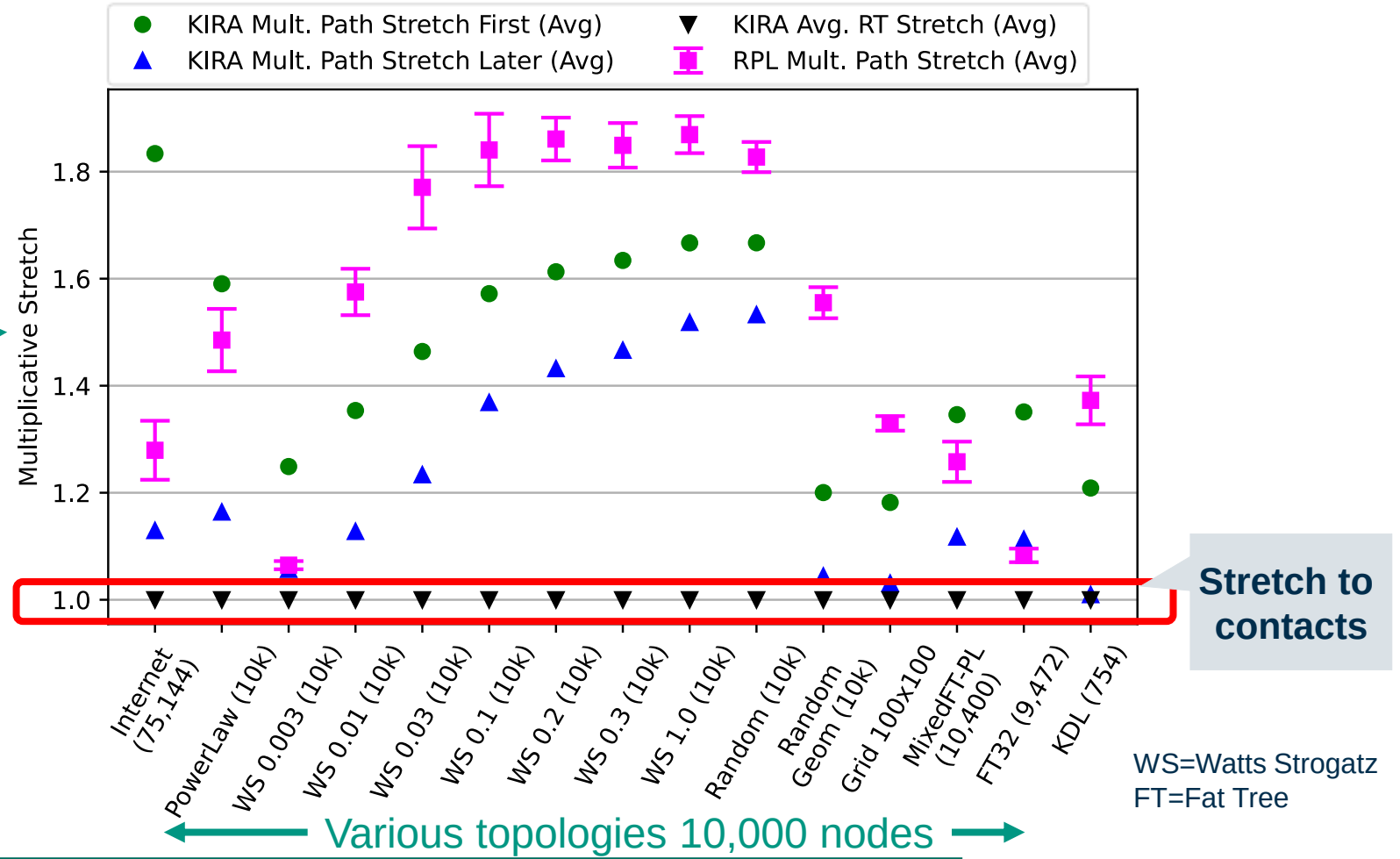
WS=Watts Strogatz
FT=Fat Tree

Stretch in Different Fixed Network Topologies

Multiplicative Stretch:
1.2 → path 20% longer than shortest path

RPL-ACP:

- Storing-mode
- Single DODAG
- Single DODAG version

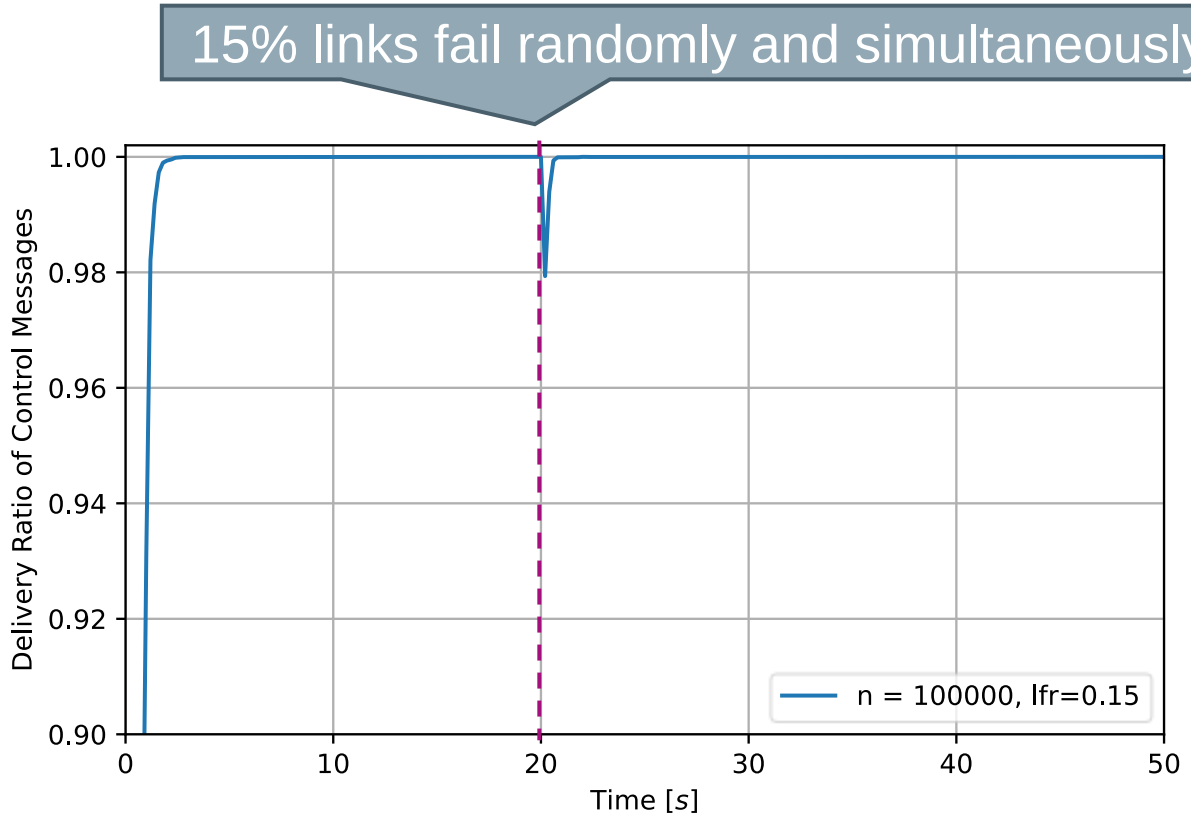


WS=Watts Strogatz
FT=Fat Tree

→ **Low stretch across various topologies + Shortest paths to contacts**

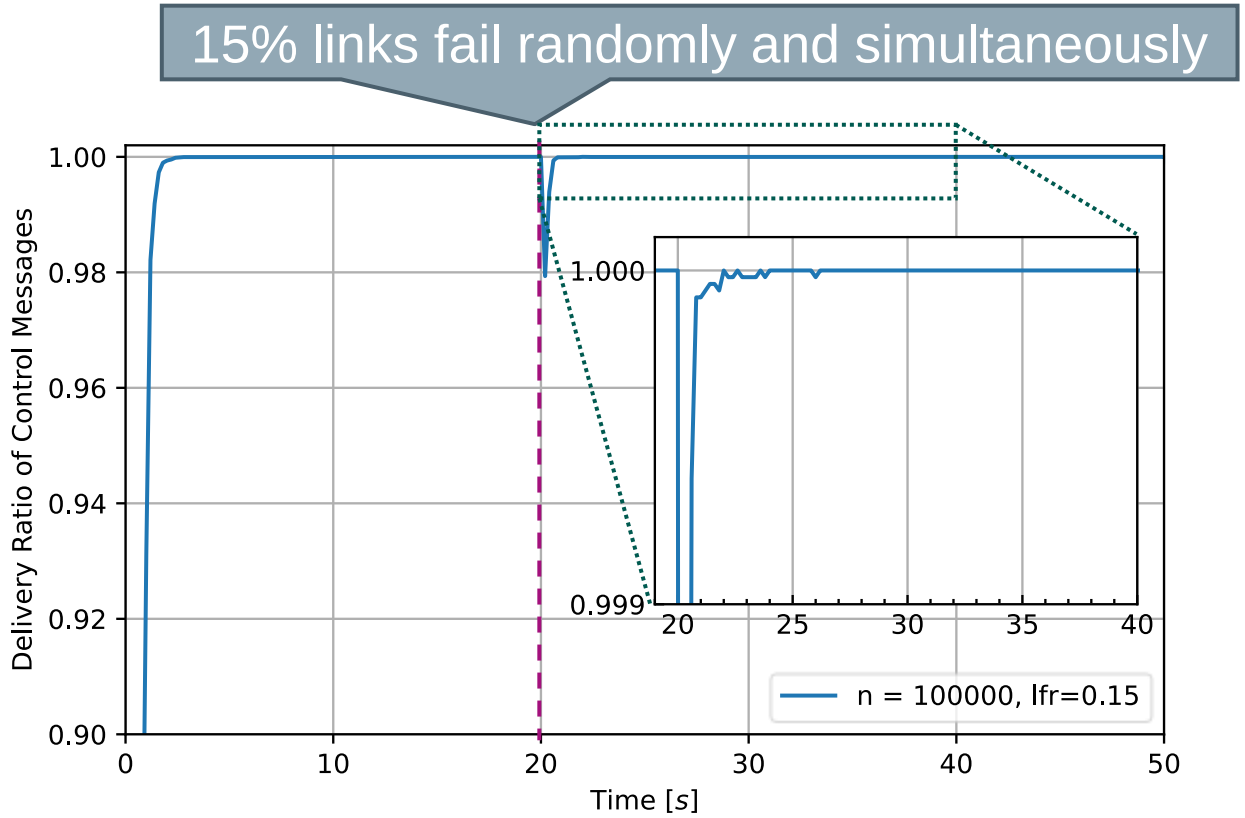
Dynamics – Handling of Link Failures

PowerLaw topology with 100,000 nodes



Dynamics – Handling of Link Failures

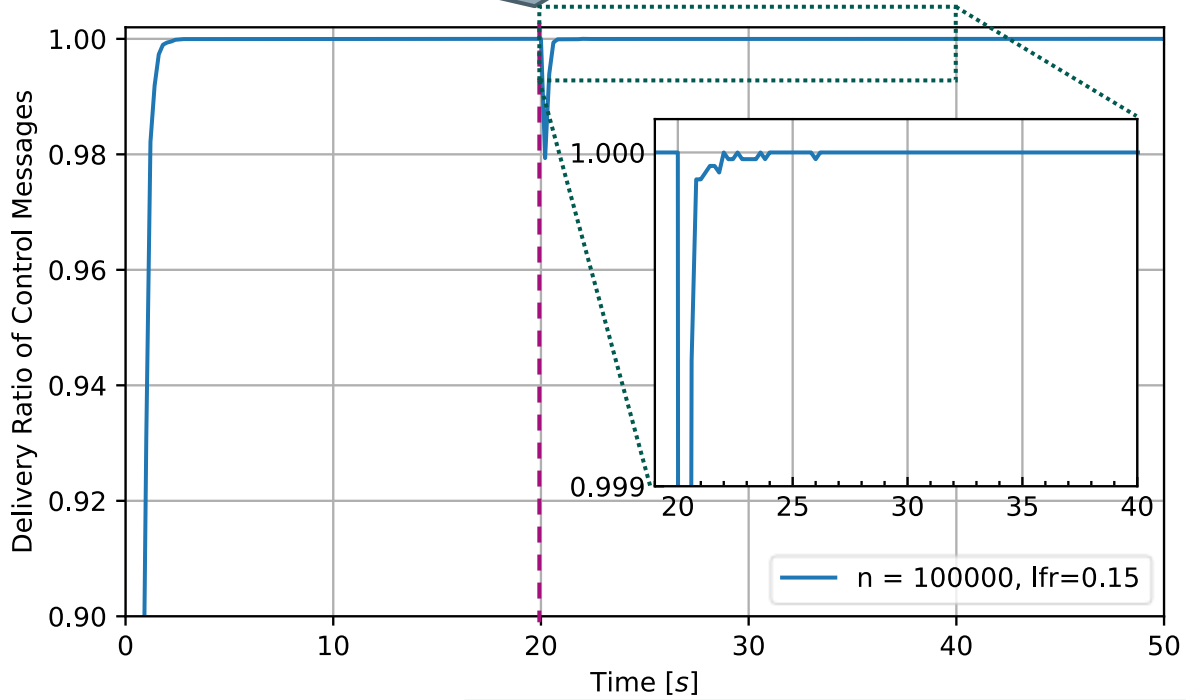
PowerLaw topology with 100,000 nodes

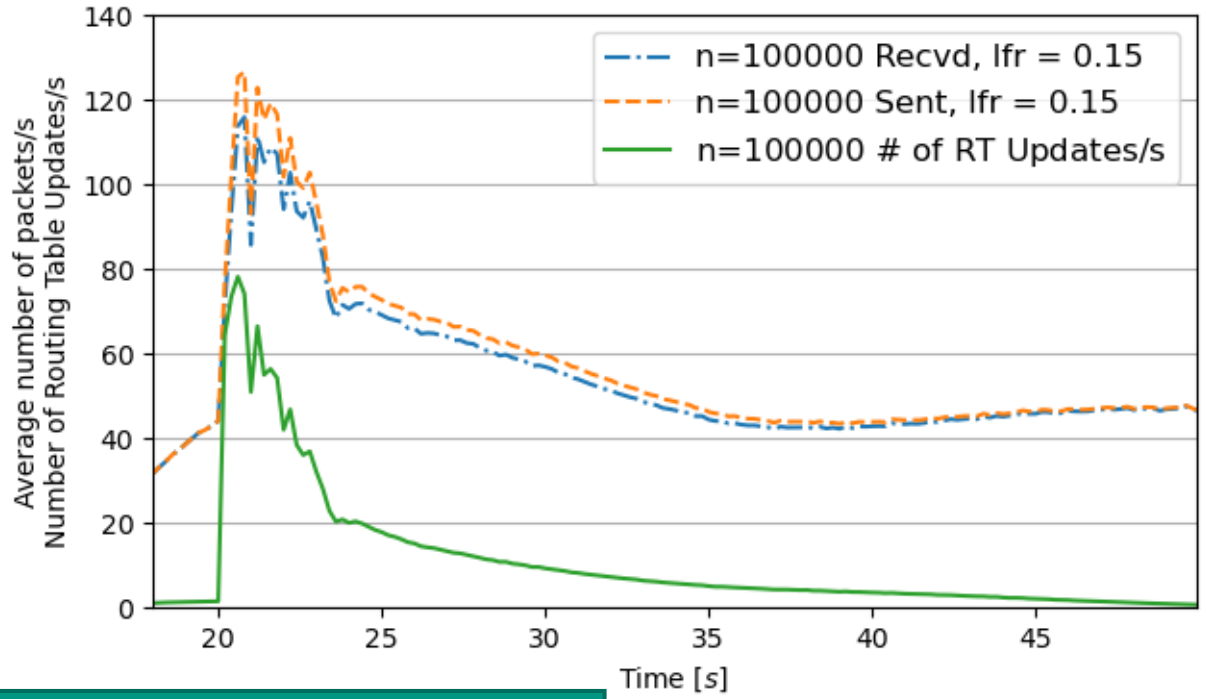


Dynamics – Handling of Link Failures

PowerLaw topology with 100,000 nodes

15% links fail randomly and simultaneously

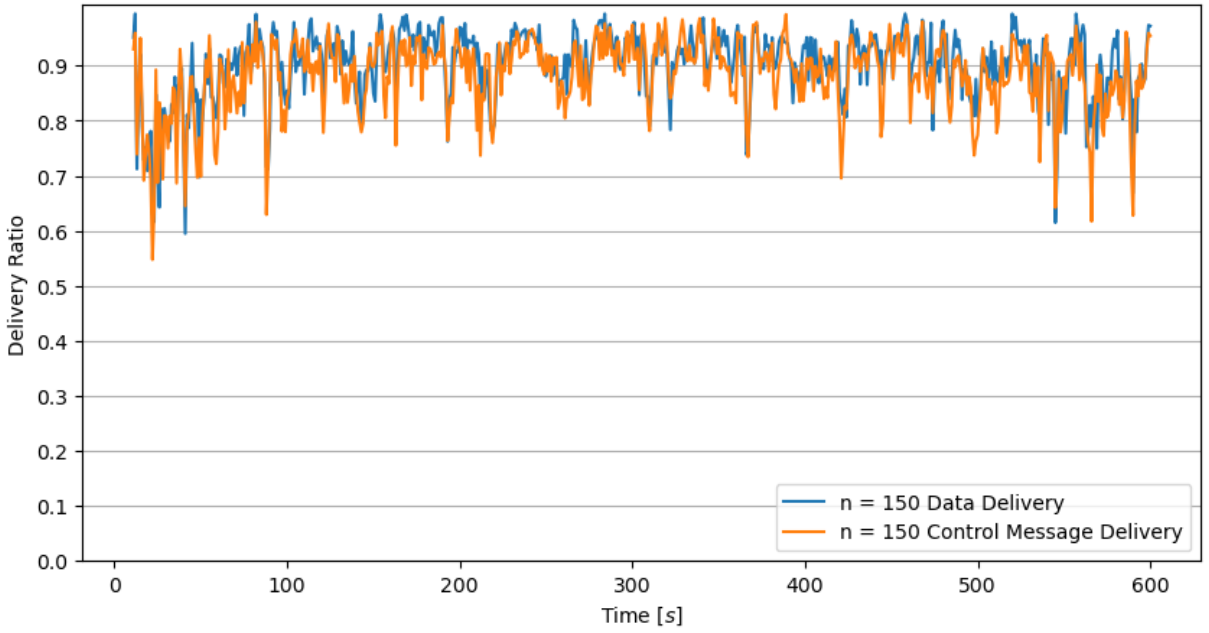




→ **Fast Convergence + Scalable Overhead**

Mobile Ad-Hoc Networks and Satellite Networks

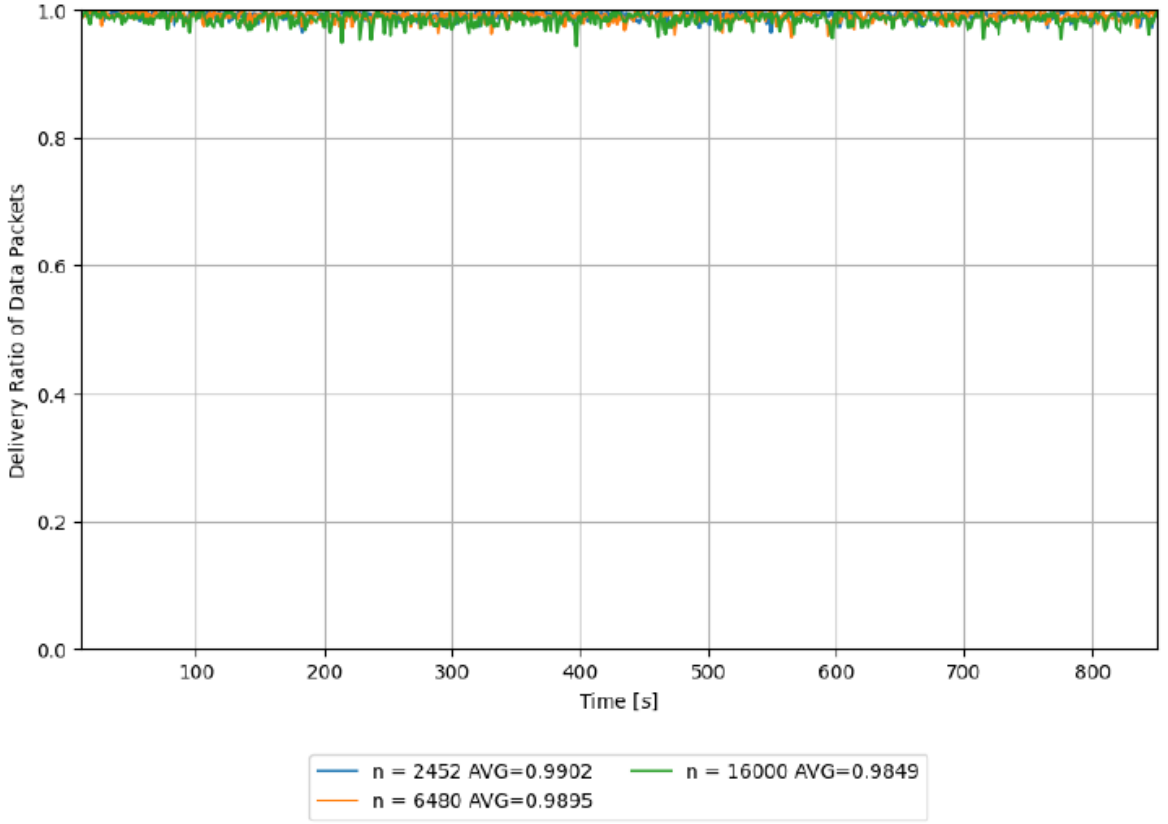
Delivery Ratio Mobile Ad-Hoc Network



RandomWaypoint 150 nodes, 1000x1000m², 100m Wi-Fi range, v=[1,10]m/s

Delivery Ratio: Data 90.18%, Routing Msgs 87.44%

Delivery Ratio LEO Satellite Constellations



Delivery Ratio Data 98.49% (16000 nodes)

Conclusions

- Resilient control plane connectivity is essential to never lose control over your network infrastructure!
- KIRA is a proposal for a routing architecture – tailored to control planes
  - Scalable and zero-touch
  - Large scale simulations up to 500,000 nodes
  - Provides useful add-ons for self-organizing services
- KIRA shows promising results in simulations of mobile cellular networks, mobile ad-hoc networks, and LEO satellite constellations
- Need support to standardize this in the IETF
  - Internet-Draft available https://datatracker.ietf.org/doc/draft-bless-rtgwg-kira/
- Running code available (https://github.com/kit-tm/KIRA-Rust)
  - Native Routing Daemon Linux (Rust) → Zero-touch IPv6 Connectivity

KIRA

https://s.kit.edu/KIRA

Backup Slides

KIRA – Main Components

Routing Tier → connectivity

R²/Kad

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

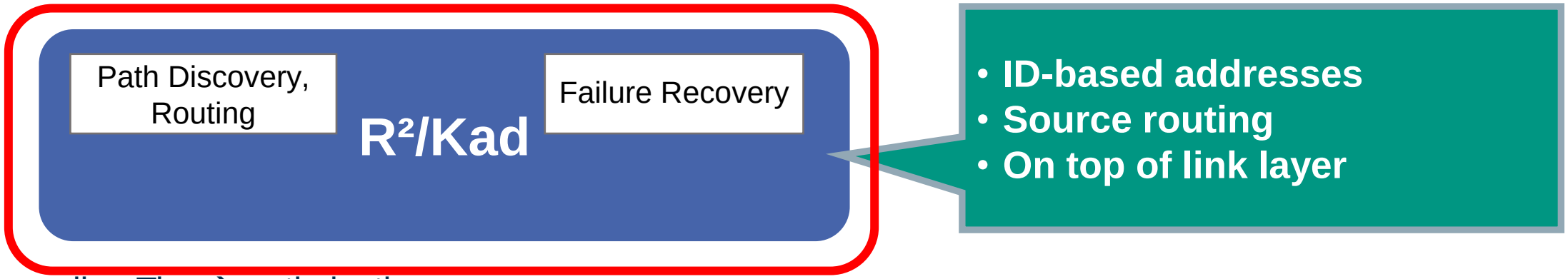
Forwarding Tier → optimization

PathID-based Forwarding

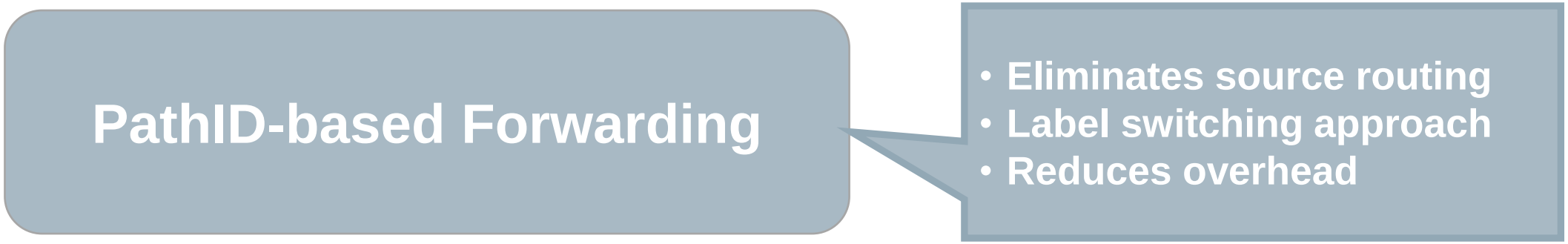
- Eliminates source routing
- Label switching approach
- Reduces overhead

KIRA – Main Components

Routing Tier → connectivity



Forwarding Tier → optimization

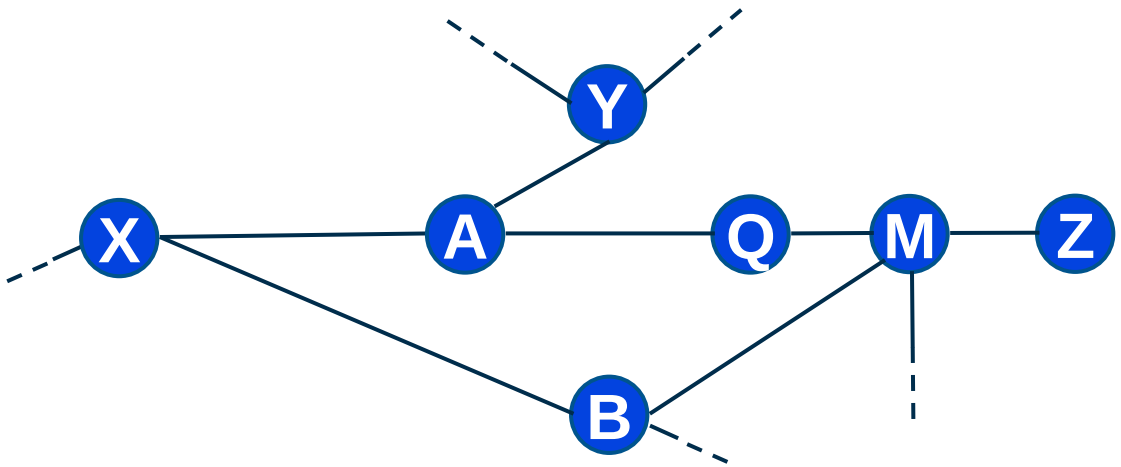


R²/Kad – Path Discovery

Each node

- randomly chooses its NodeID (Overlay)
- explores its 2-hop vicinity (Underlay)

1. X learns contacts A, Y, Q, B, M, …



R²/Kad – Path Discovery

Each node

- randomly chooses its NodeID (Overlay)
- explores its 2-hop vicinity (Underlay)

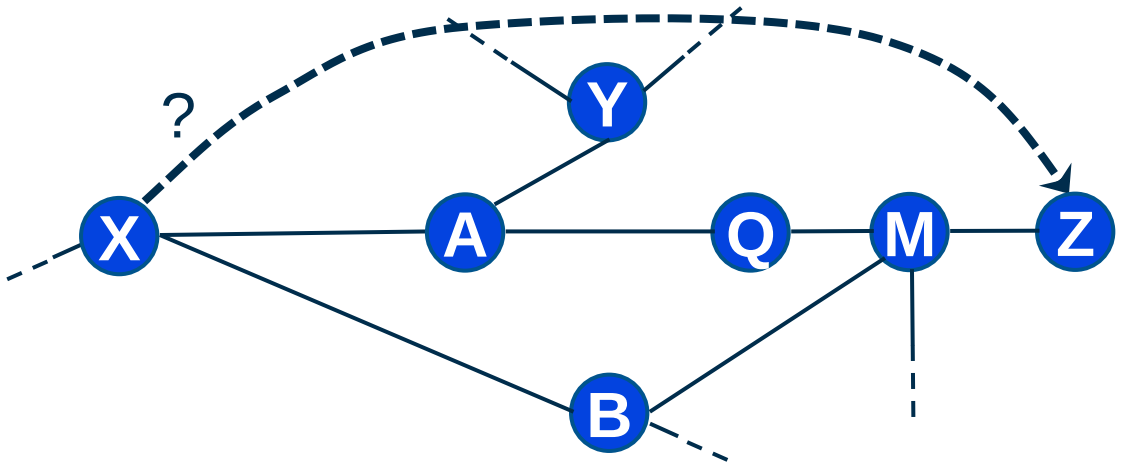
1. X learns contacts A, Y, Q, B, M, …

X: path to Z?

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

- Source route to contact that is ID-wise closest to destination NodeID (→ recursively)
- Distance of NodeIDs: XOR metric $d(X,Y) = X \oplus Y$

1. Longer shared prefix → 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

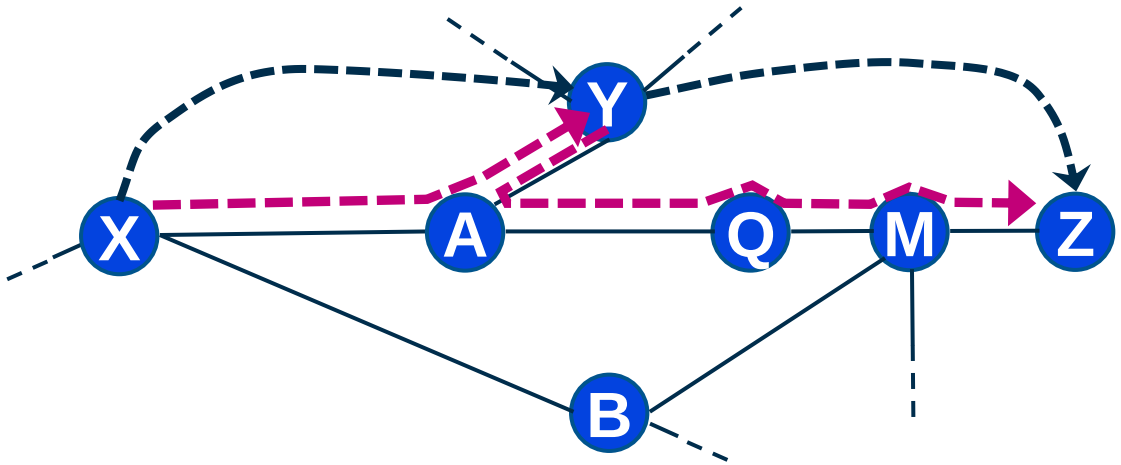
X → Y via source route <A>

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

Path Discovery, Routing | R²/Kad | Failure Recovery

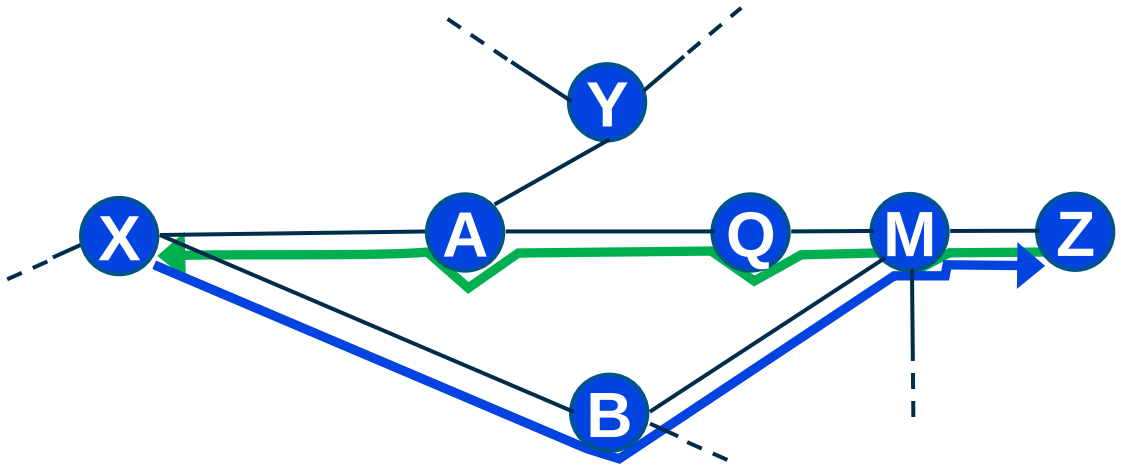
Shortened recorded route <A,Q,M> is returned to X in FindNodeRsp

Later packets use shorter route <B,M>

- if X already knows M via <B>

Initial stretch can be reduced for later packets!

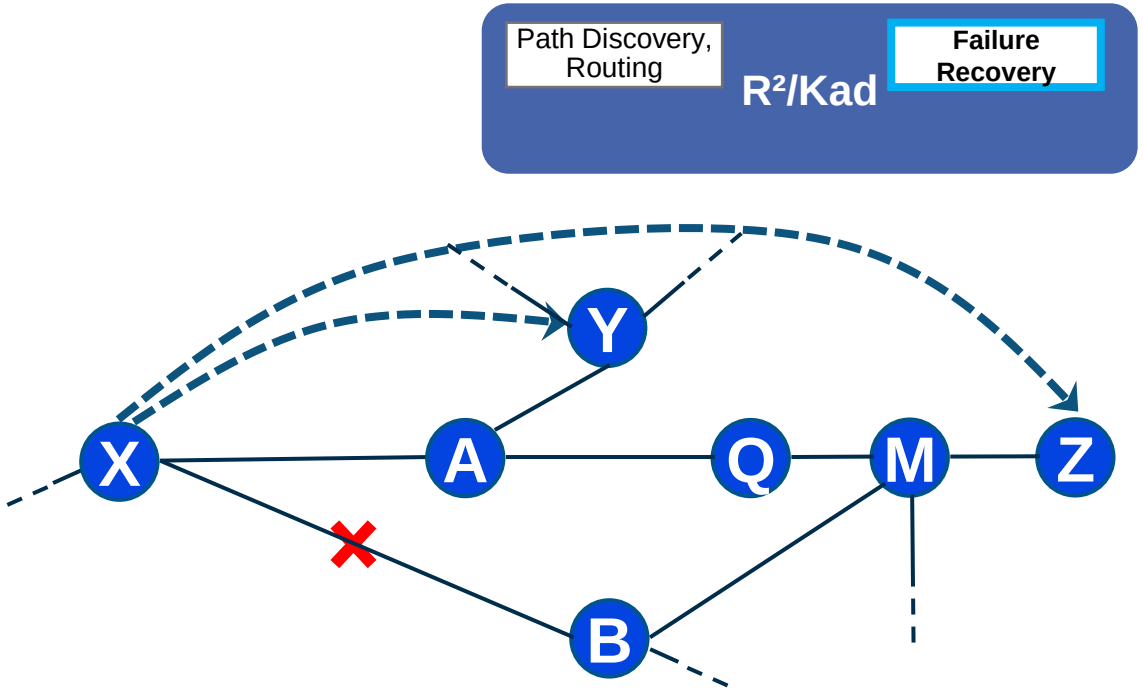
R²/Kad offers flexible memory/stretch trade-off…



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

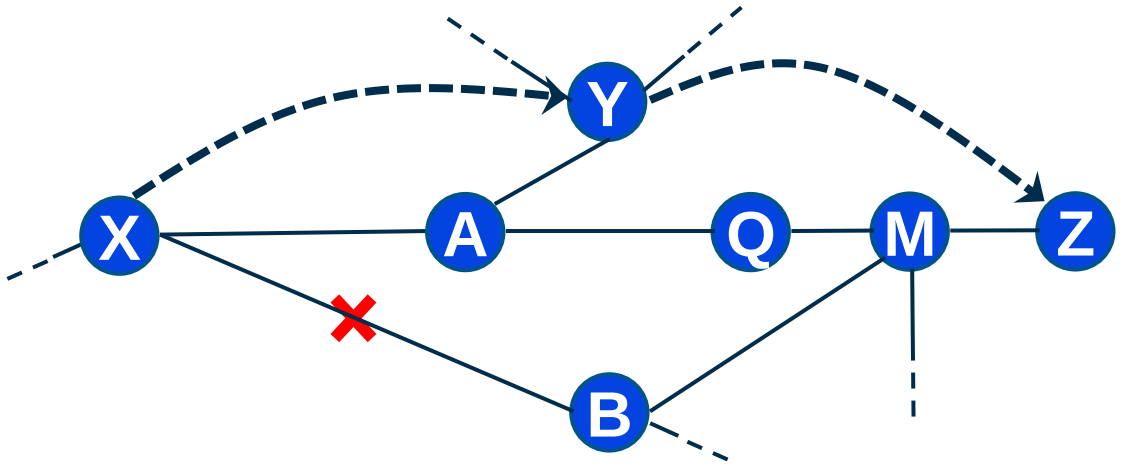
Two step strategy
1.) inform ID-wise neighbors about failed link
2.) rediscover alternative paths via
overlay routes (includes “Not Via” information)

Validity

- State sequence numbers
- Path information age

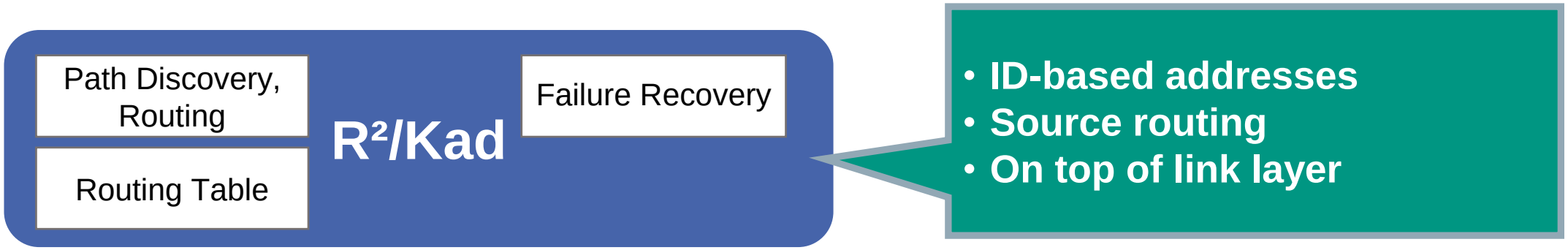
Periodically

- probe contacts for broken paths
- lookup own NodeID



KIRA – Main Components

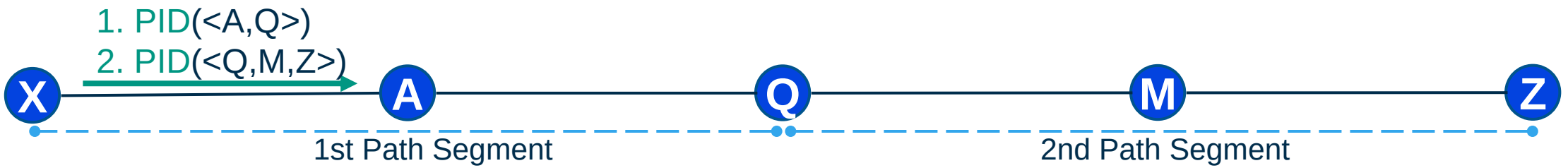
Routing Tier → connectivity



Forwarding Tier → optimization

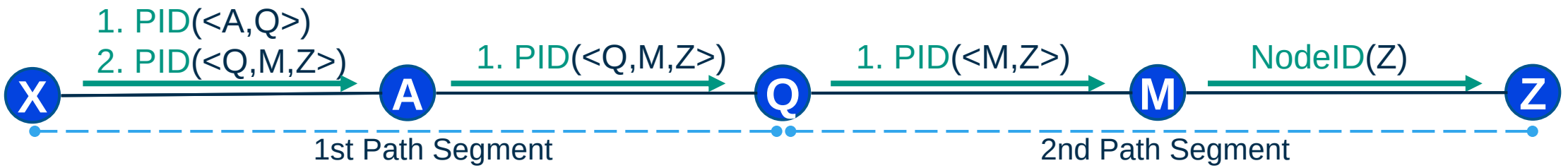
Forwarding Tier – Fast Forwarding

- Get rid of source routes for control plane traffic
  - Reduce per packet overhead
- Approach: replace source routes with PathIDs
  - PathID(<A,Q,M,Z>)= Hash(A | Q | M | Z)
- Use PathID as unique label for path segment → Label Switching



Forwarding Tier – Fast Forwarding

- Get rid of source routes for control plane traffic
  - Reduce per packet overhead
- Approach: replace source routes with PathIDs
  - PathID(<A,Q,M,Z>)= Hash(A | Q | M | Z)
- Use PathID as unique label for path segment → Label Switching



- Precalculate PathIDs for 2-hop (physical) vicinity
- Explicit path setup for paths ≥ 6 hops

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)

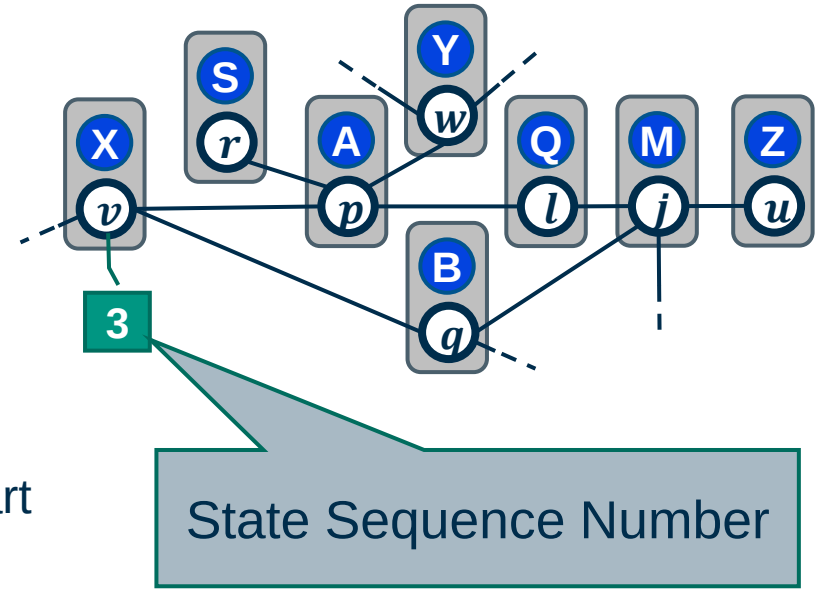
32-bit

- Monotonically increasing, with reset

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)

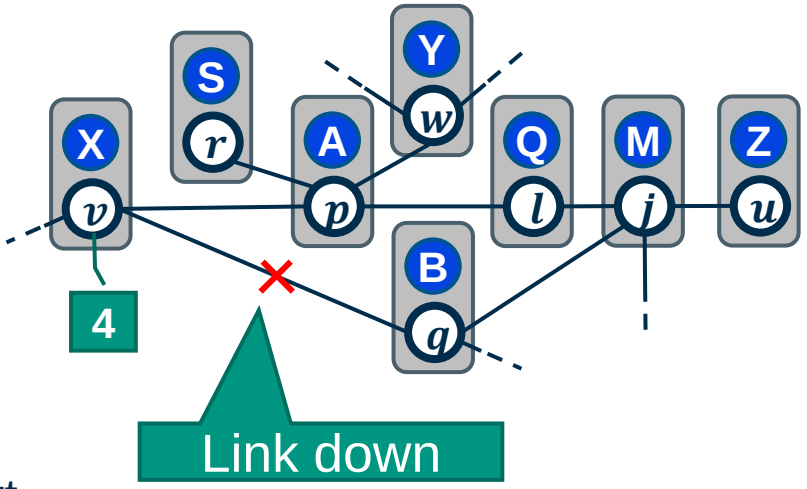
32-bit

- Monotonically increasing, with reset

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)

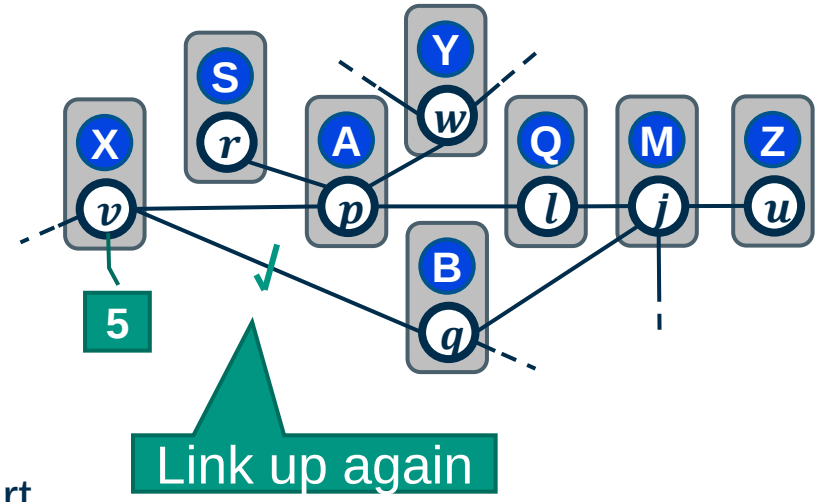
32-bit

- Monotonically increasing, with reset

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

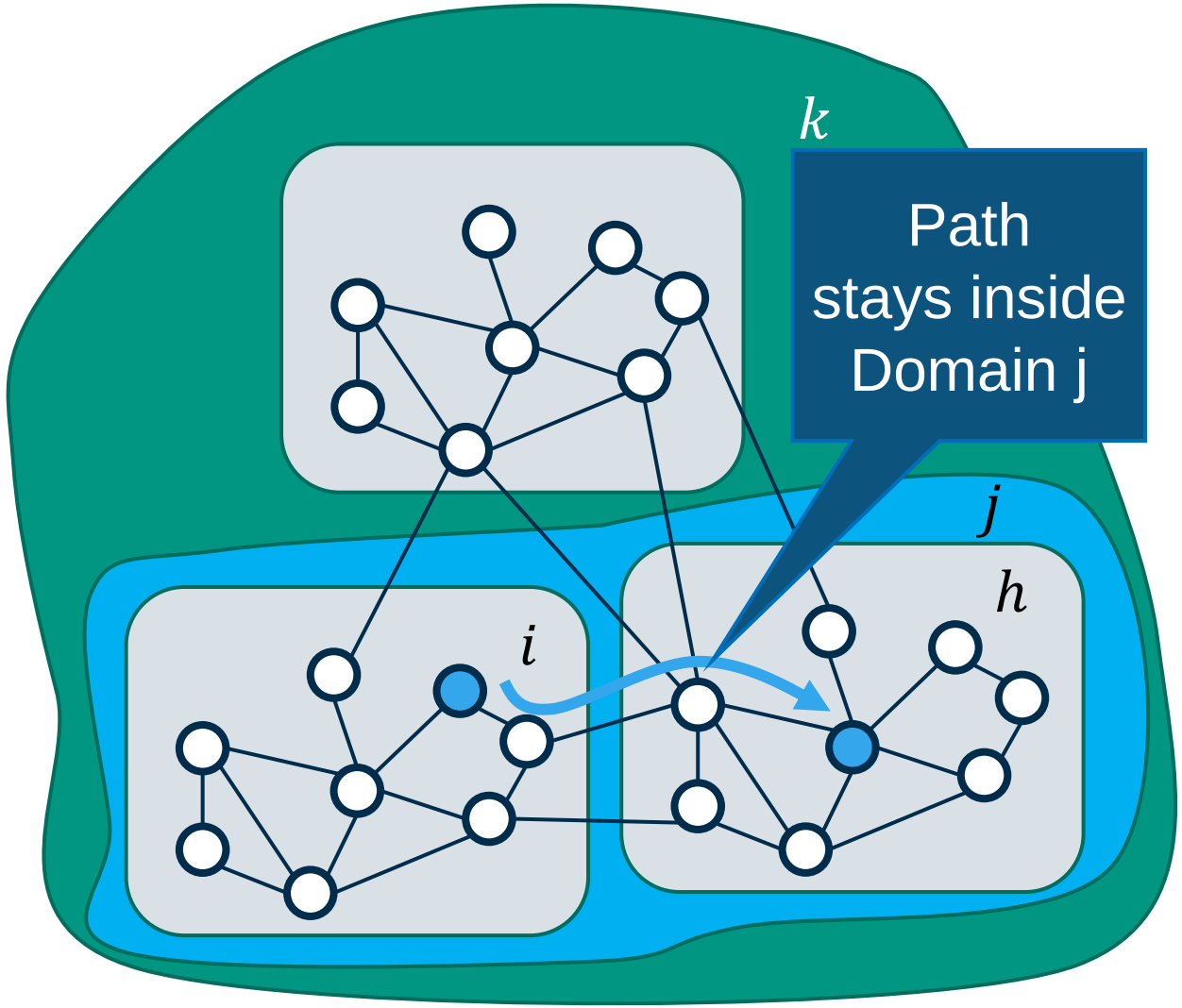
KIRA – Domain Scopes

Domain Scopes

- Global, Organizational (e.g., ASes), Topological
- KIRA nodes keep their NodeID across domains!

Need Domain-ID (64 bit)
in data packet for routing table selection

- SRH TLV?



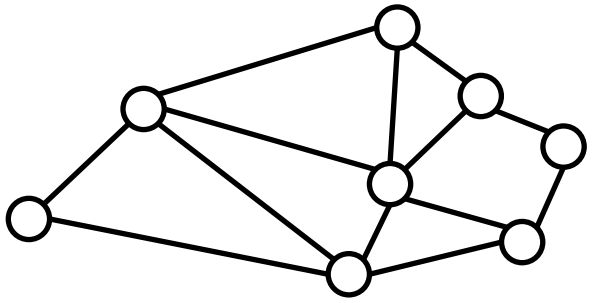
NodeID Collision

- For NodeIDs of 112 bit the collision probability is very small:
  - for $n = 10^{11} \approx 9.63 \cdot 10^{-13}$ (100 Billion nodes)
  - for $n = 10^{12} \approx 9.63 \cdot 10^{-11}$ (1 Trillion nodes)
- For NodeIDs of 64 bit the collision probability is already high (≈ 0.5) for 4 Billion nodes
  - so depending on how large your domain is, smaller NodeID space may also work
- Collisions can be detected during the join process
  - detecting nodes can then generate a new NodeID

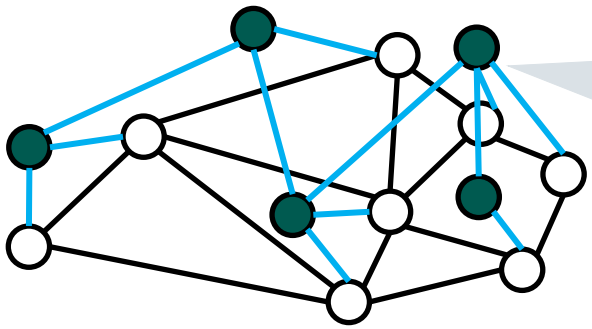
Security

- NodeIDs can be self-certifying (e.g., hash of public key)
  - onboarding process possible if enrollment entities present
  - untrusted nodes are treated as end-systems (do not route)
- Path information from other nodes is validated before actual use
- Open for discussion:
  - Hop-by-hop secure transport (e.g., DTLS)
  - Proof-of-transit
  - How to change PathID hash algorithm

Out-of-band Control Plane Network



Data Network

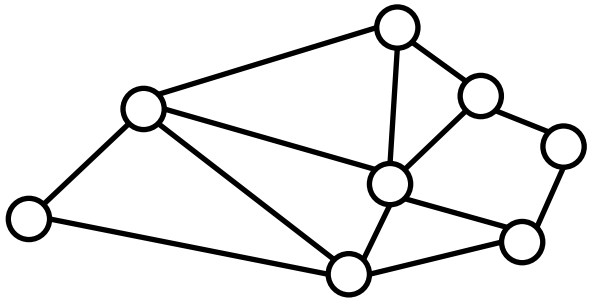


Data Network

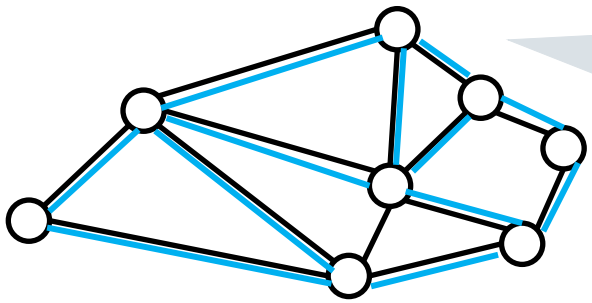
out-of-band control plane network fabric

Separate Control Plane Network for control and management: requires also setup, configuration, routing, etc.

In-band Control Plane Network



Data Network



Data Network

in-band control plane network fabric

In-band Control Plane Network uses existing links for control and management (prioritization required)

Example – Networking Control and Management

- KIRA bootstraps the control plane fabric
- Resources can register themselves to be found (e.g., run ANIMA on top)
- Topologically dependent routing for data plane can be built on top
  - e.g., divide the network into areas, assign and distribute prefixes, configure OSPF routers, use SDN, etc.
- Distributed controllers may claim control over some resource subset (with auto scaling)
- In case things go (horribly) wrong
  → restart from scratch, revert to last working configuration