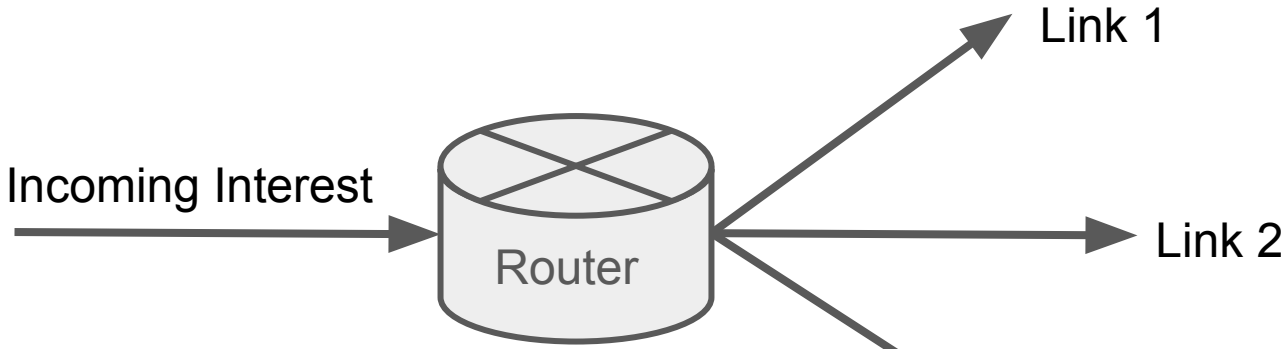
Architecting A Scalable Routing System for Named Data Networking

Tianyuan Yu, Mark Theeranantachai

Background

- NDN routing scalability is a concern from day one



Router FIB

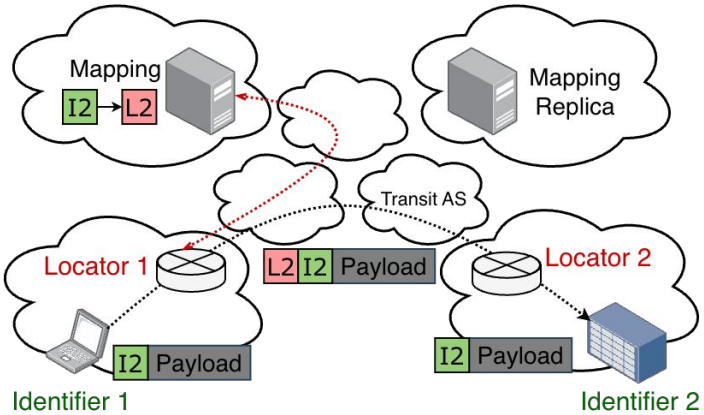
DNS names

Extensive Multicast

/edu/ucla/repo	**1**
/ownly.ndn/comm2026/adam	**2**
/ownly.ndn/comm2026/sync	**2, 3**

Map-and-Encap

- Even in IP, routing scalability is a long-lived concern
  - IP has long history of Identity/Locator conflict
- Map-and-encap as recurring architectural answer
  - For both unicast and multicast
- This work offers a map-and-encap implementation to NDN



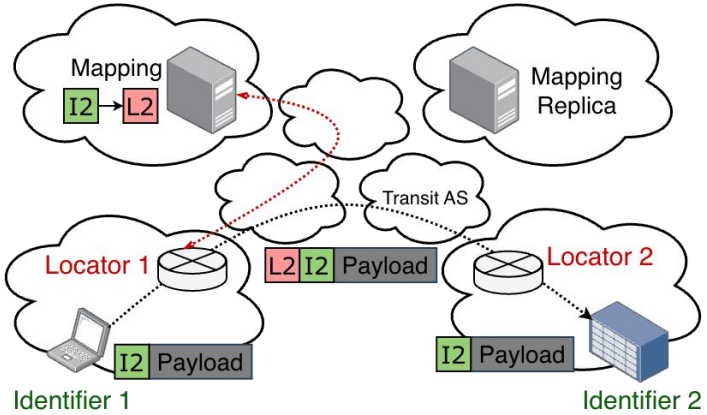
Leveraging NDN for secured and efficient mapping replication

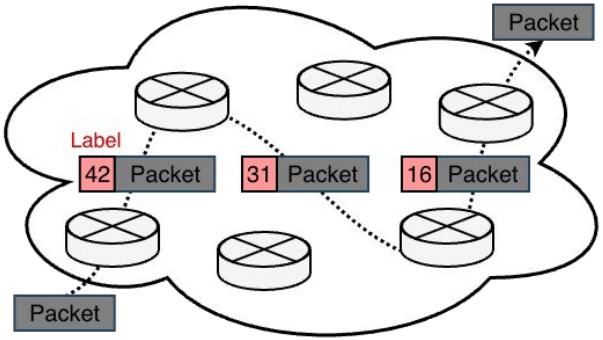
With the mapping, supporting stateless multicast via BIER

Map-and-Encap in Unicast

- Started from ILNP/ LISP
  - The Identifier/Locator Split
  - ***Encapsulating*** packets to a topological endpoint
  - A ***mapping*** system to resolve a logical identifier into a topological one
  - Difficulties in deploying a *new* global system, when incremental mitigations like MPLS works
    - Lesson: mapping have to be design in

LISP: global coordinations required



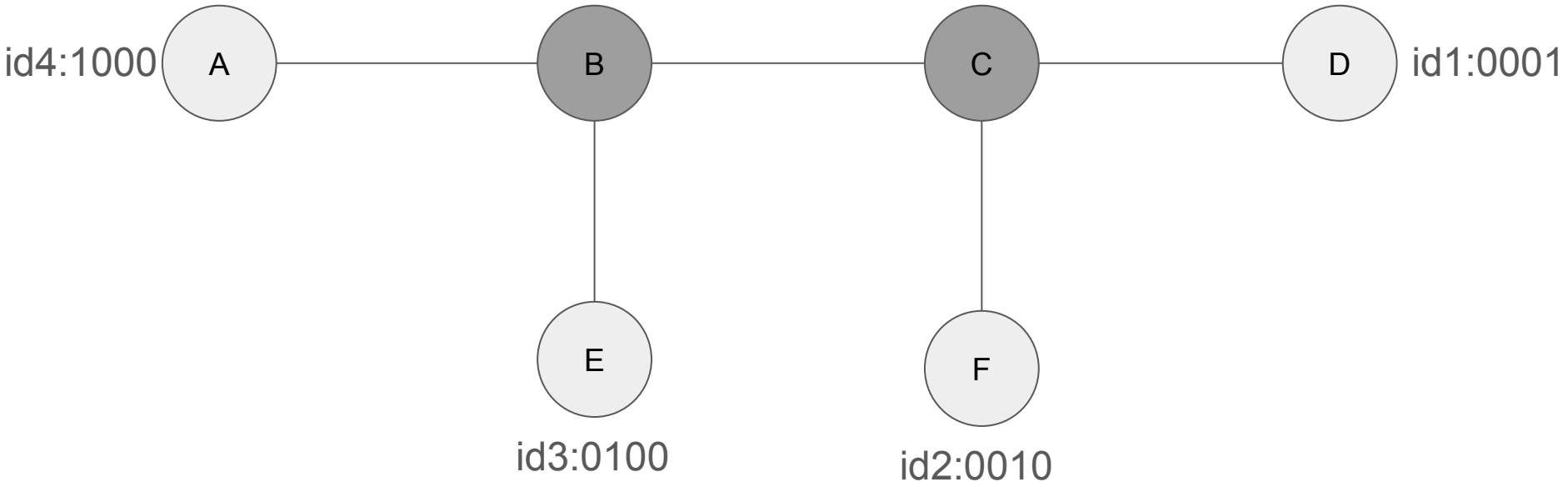


MPLS: incremental and working

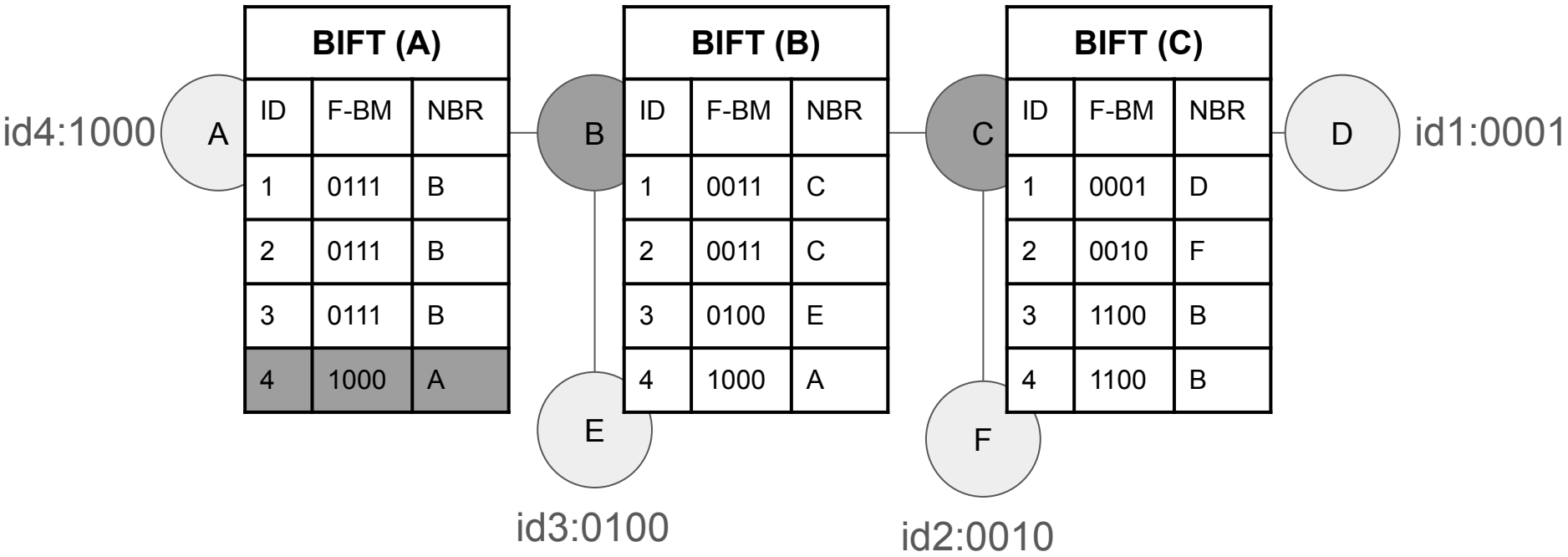
“Map-and-Encap” in Multicast

- Traditional multicast routing protocols
  - Setting forwarding states per-(Source, Group) tuple
  - Mcast group numbers is out of operator’s control
  - Application dynamics decide what and how many prefixes they announce
    - Applicable to both unicast and multicast
- Bit Index Explicit Replication (BIER)
  - Assuming one knows routers A, B C are interested to the group G ⇒ ***mapping***
  - Delivering to G ⇒ explicitly replicating the packets to the router A, B, C
    - Using the router reachability that IGP already establishes
    - **Encapsulating** packets with the list of egress routers

BIER in Operation

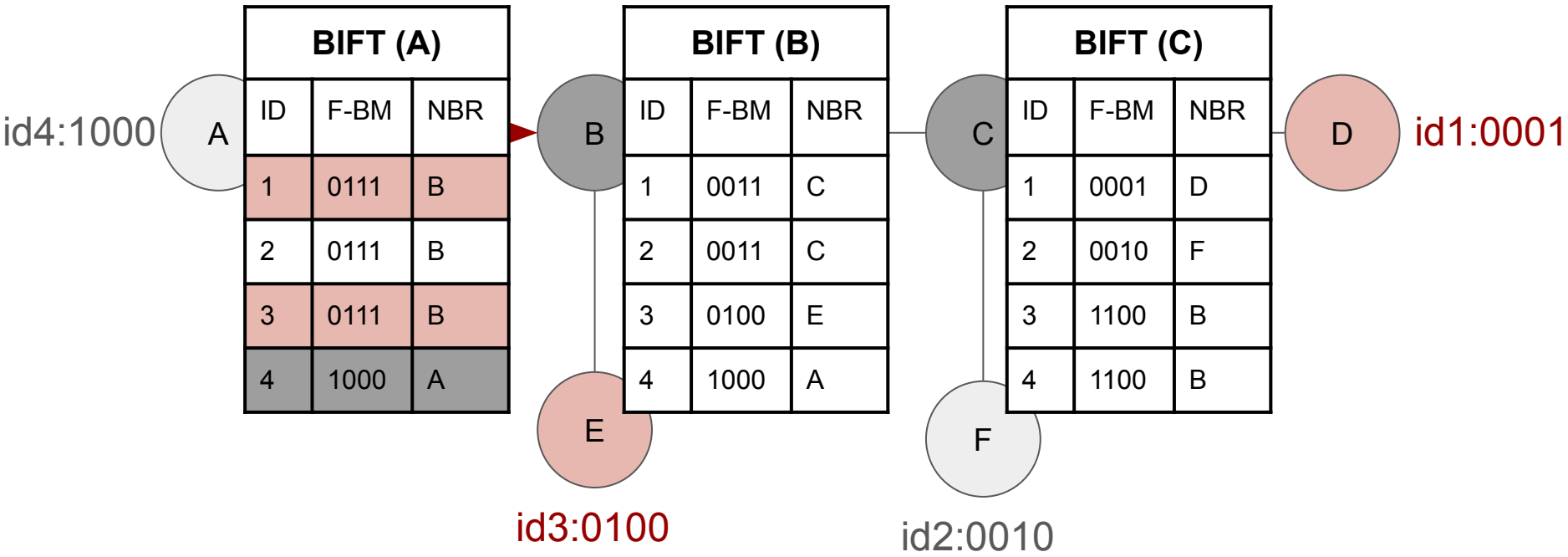


BIER in Operation



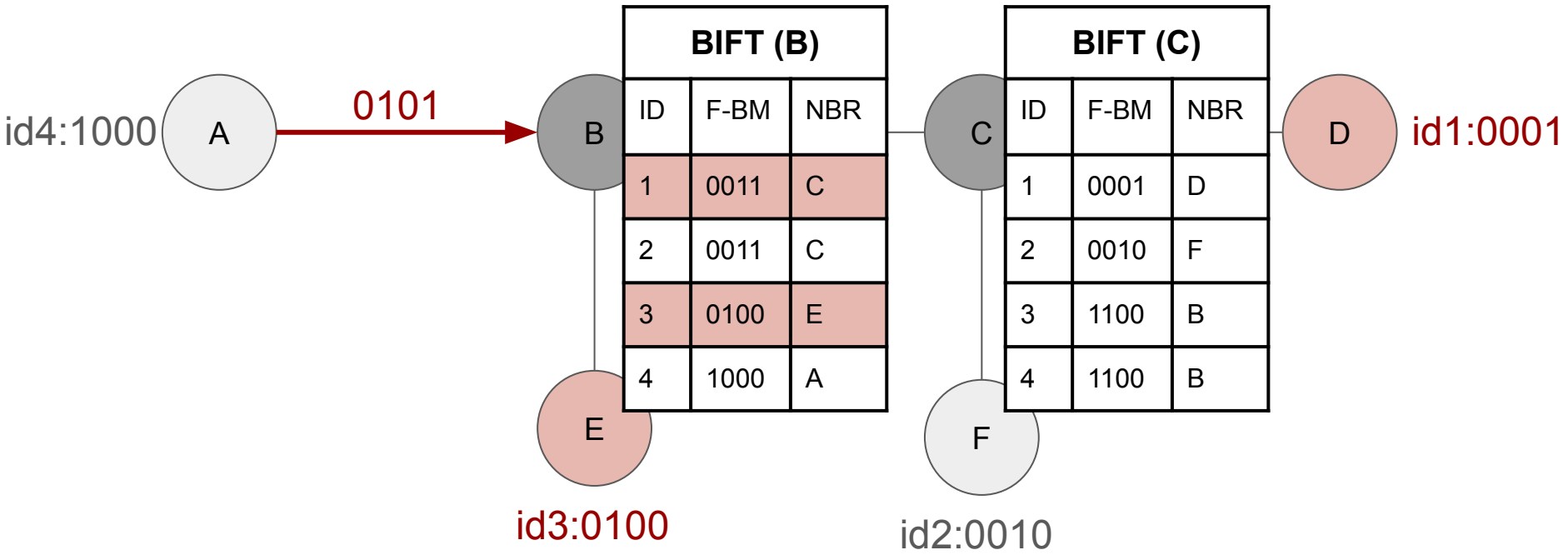
BIFT: BIER Index Forwarding Table
F-BM: Forwarding Bitmask
NBR: Nexthop BIER router

BIER in Operation: Delivering to {D, E}



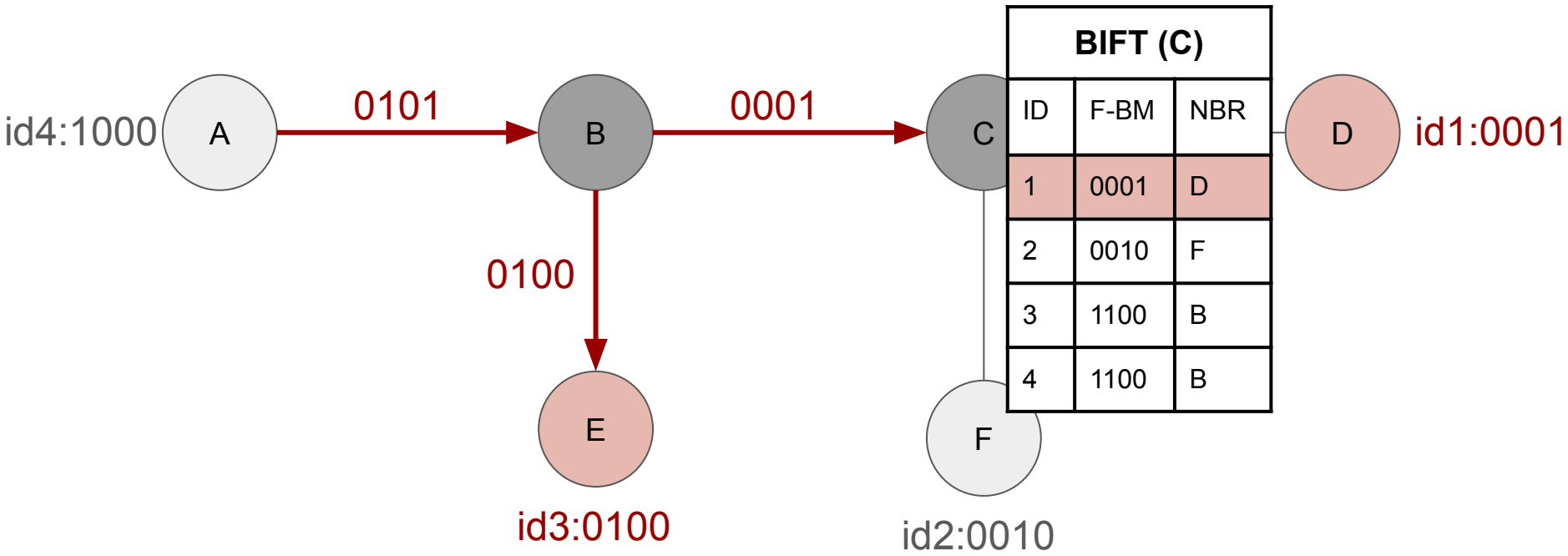
BIFT: BIER Index Forwarding Table
F-BM: Forwarding Bitmask
NBR: Nexthop BIER router

BIER in Operation: Delivering to {D, E}



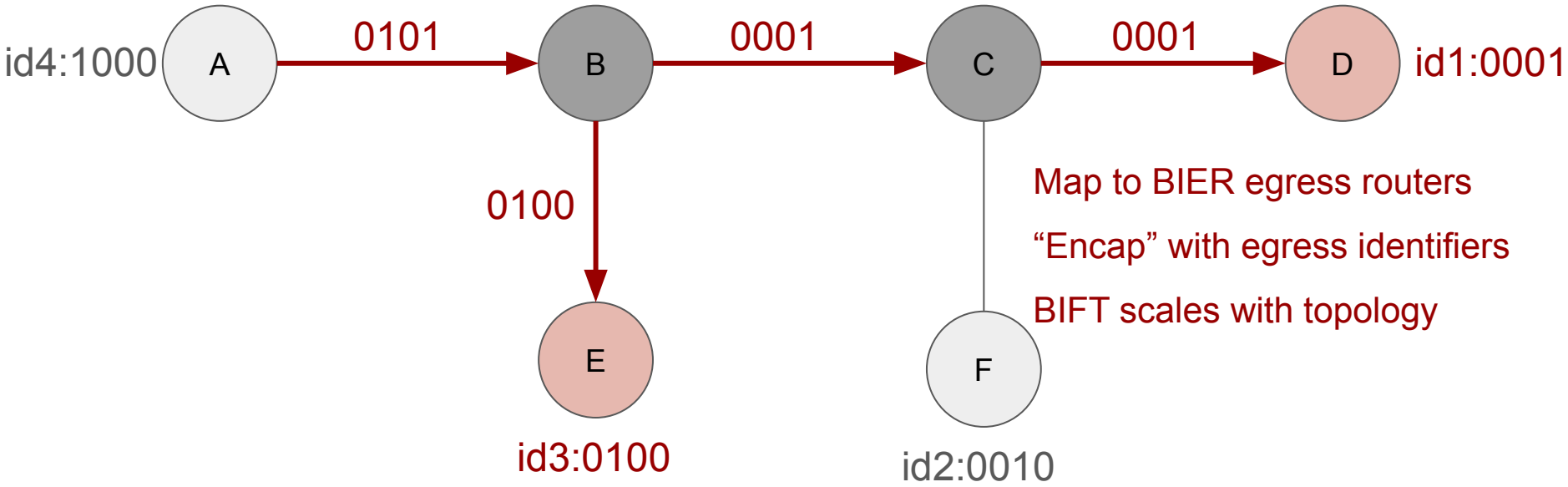
BIFT: BIER Index Forwarding Table
F-BM: Forwarding Bitmask
NBR: Nexthop BIER router

BIER in Operation: Delivering to {D, E}



BIFT: BIER Index Forwarding Table
F-BM: Forwarding Bitmask
NBR: Nexthop BIER router

BIER in Operation: Delivering to {D, E}



BIFT: BIER Index Forwarding Table
F-BM: Forwarding Bitmask
NBR: Nexthop BIER router

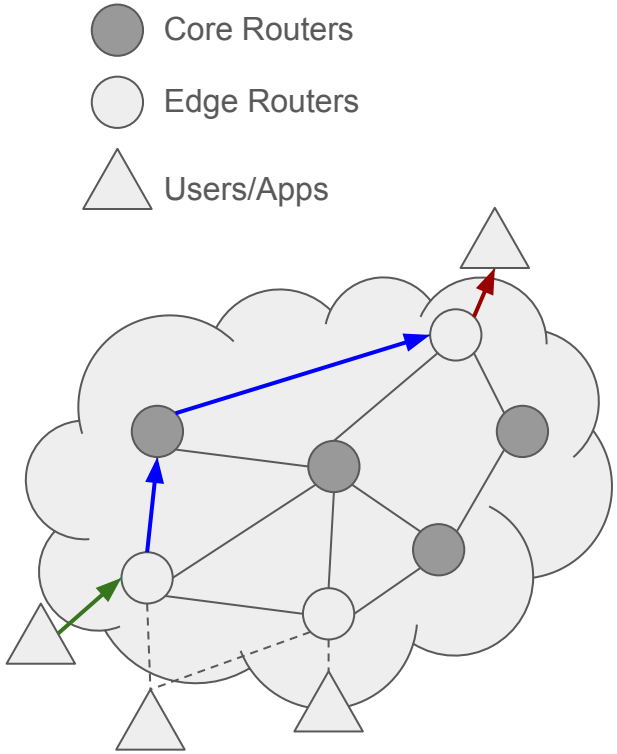
A **S**calable **R**outing **A**rchitecture for **N**DN

- An NDN native map-and-encap implementation
  - ***Mapping*** directly in routers, with synchronized content
  - ***Encapsulating*** via NDNLPv2, the L2.5 protocol for NDN forwarders
    - Extending NDNLPv2 with a new header
- Slow start: intra-domain routing
  - Assumption: IGP already established router reachability
  - Map-and-encap at AS level remains an open question[1]

1. From Map-and-Encap to BIER: Observations on Network Routing Scalability, just available on arXiv!

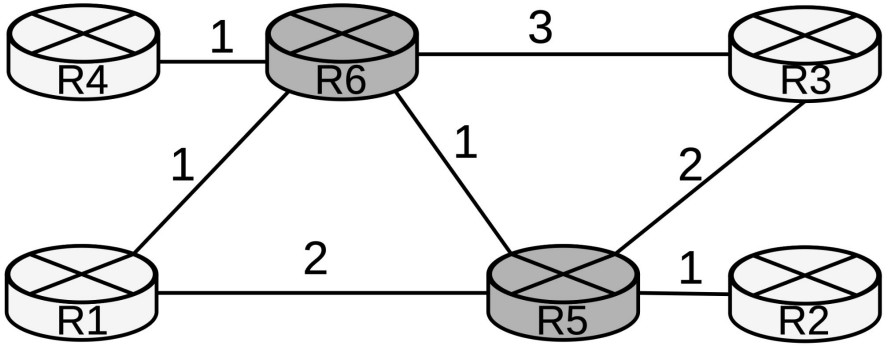
System Overview

- End Apps: register prefixes at edge routers
  - Source and sink for Interests
- Edge routers
  - Synchronizing on the mappings
    - Map a prefix to its **Egress Routers (ER)**
  - Ingressing/Egressing Interest based on the mapping
    - Add/Remove L2.5 encap header of **ER**
- Core routers
  - Forwarding Interest based on ER header



Mapping System

- Forwarding Plane: Prefix Egress Table
  - Router ⇒ {Prefix, Time-To-Live, Model}
  - Local FaceIds kept locally at per router
- Control Plane: Prefix State Database
  - Routers publishing PSD Data
    - Add/Remove/Update/Reset Ops
  - Synchronizing PSD Dataset via SVS
    - Discuss later in later slides



Example R1 View

Control Plane

Prefix State Database

R1	P1	TTL	U	Fid={...}
	P2	TTL	M	Fid={...}
	P3	TTL	M	Fid={...}
	.....			
R2	P2	TTL	M	Fid={}
	P3	TTL	M	Fid={}
	P4	TTL	U	Fid={}
	.....			

.....

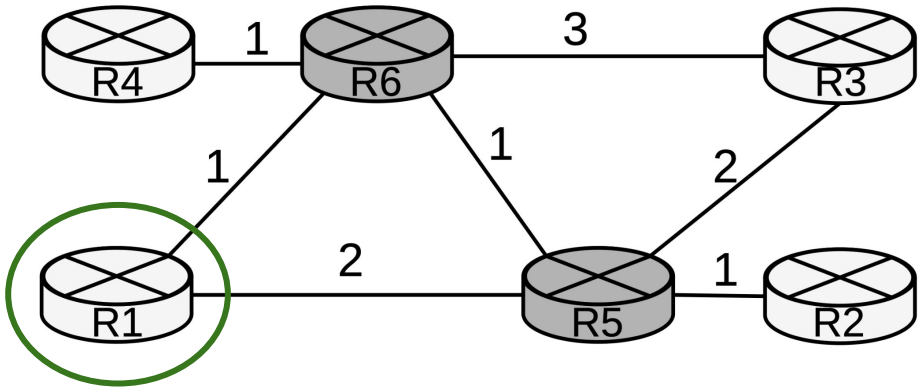
Forwarding Plane

Prefix Egress Table

P1	ER={R1}	Fid={...}	U
P2	ER={R1, R2}	Fid={...}	M
P3	ER={R1, R2, R3}	Fid={...}	M
P4	ER={R2, R4}	Fid={}	U

.....

Interest *Ingress* at R1



Prefix Egress Table

P1	ER={R1}	Fid={...}	U
P2	ER={R1, R2}	Fid={...}	M
P3	ER={R1, R2, R3}	Fid={...}	M
P4	ER={R2, R4}	Fid={}	U

.....

FIB

R1	/
R2	Face 1 (Cost 3)
R3	Face 1 (Cost 4)
R4	Face 2 (Cost 2)

.....

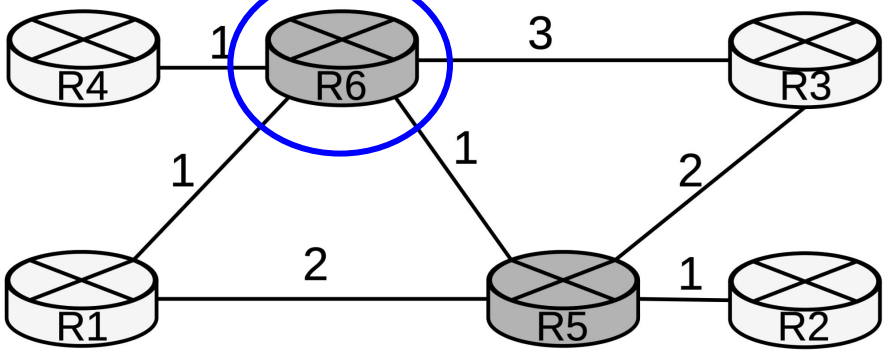
hasER

Incoming Interest /P4

R4

Outgoing Interest (forward)

Interest ***Transit*** at R1



isLocal (R6)

hasER

R4

Incoming Interest /P4

Prefix Egress Table

doesn't bother

FIB

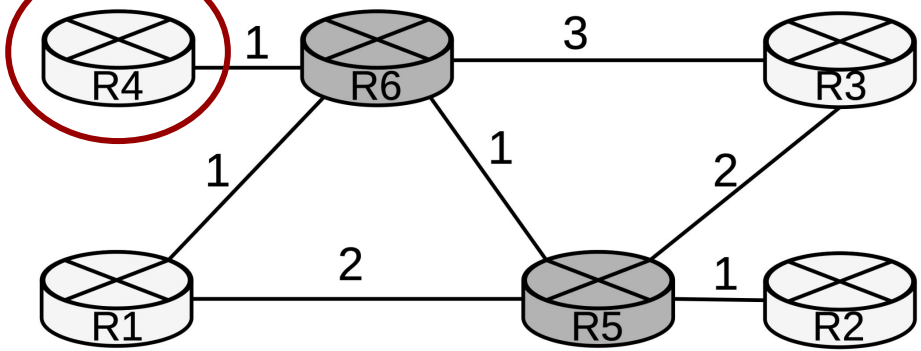
R6	/
R2	Face 2 (Cost 2)
R3	Face 1 (Cost 3)
R4	Face 4 (Cost 1)

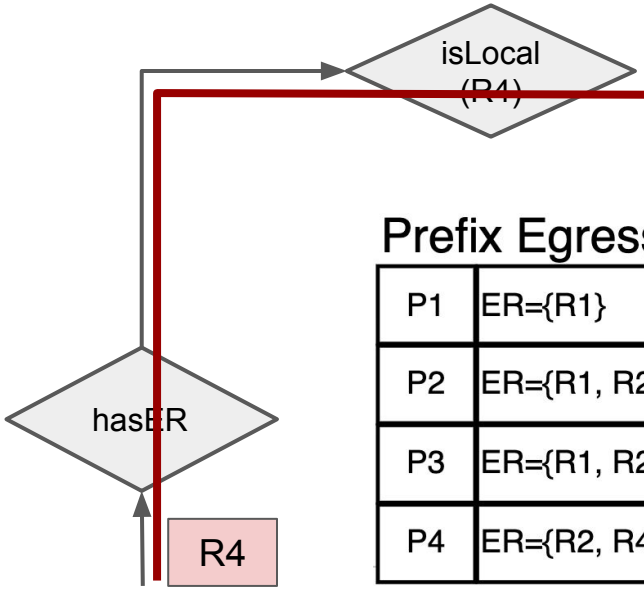
.....

R4

Outgoing Interest (forward)

Interest ***Egress*** at R1





Incoming Interest /P4

Prefix Egress Table

P1	ER={R1}	Fid={}	U
P2	ER={R1, R2}	Fid={}	M
P3	ER={R1, R2, R3}	Fid={}	M
P4	ER={R2, R4}	Fid={ }	U

.....

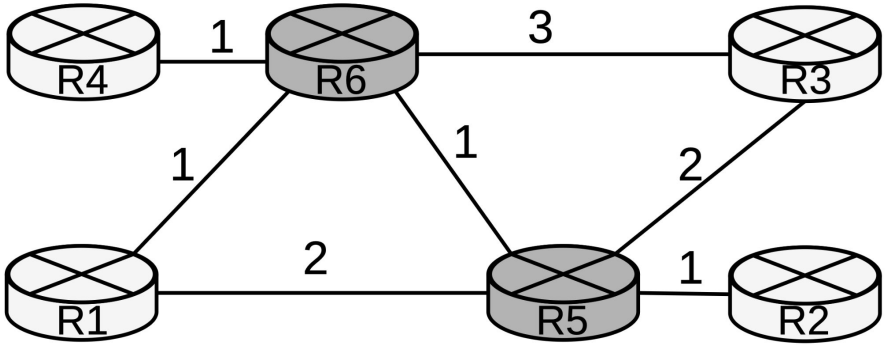
FIB

R4	/
R1	Face 1 (Cost 2)
R2	Face 1 (Cost 3)
R3	Face 1 (Cost 4)

.....

Outgoing Interest (app)

Local Forwarding at R1



Prefix Egress Table

P1	ER={R1}	Fid={...}	U
P2	ER={R1, R2}	Fid={...}	M
P3	ER={R1, R2, R3}	Fid={...}	M
P4	ER={R2, R4}	Fid={}	U

.....

FIB

R1	/
R2	Face 1 (Cost 3)
R3	Face 1 (Cost 4)
R4	Face 2 (Cost 2)

.....

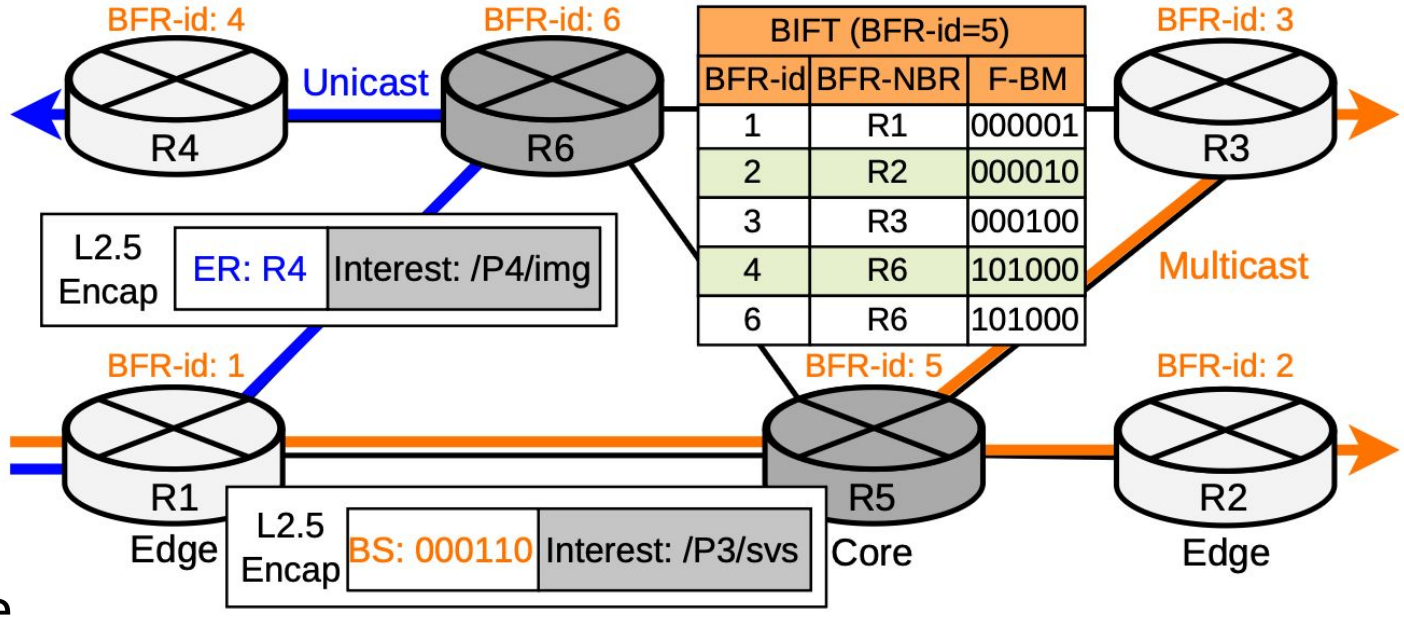
hasER

Incoming Interest /P4

Outgoing Interest (app)

Leveraging PET to Support BIER

- Operators configures BIER RouterID to all routers
  - ***PET provides the built-in mapping for BIER***
- A new L2.5 Encap header: BitString (BS)
  - One cannot carry both BS and ER --- an Interest cannot be both unicast or multicast
- Ingress routers
  - Looking at PET to see if multicast is needed
  - Identifying all egress routers, encoding BS, and do BIER forwarding
- Transit routers: BIER forwarding based on BIFT
- Egress routers: if my ID is included, egress



Prefix Egress Table

P1	ER={R1}	Fid={...}	U
P2	ER={R1, R2}	Fid={...}	M
P3	ER={R1, R2, R3}	Fid={...}	M
P4	ER={R2, R4}	Fid={}	U

.....

Explicitly replicating to member router R3 and R5

Stateless Interest Multicast!

Mapping Sync as Explicit Replication

- Whenever an edge router receives a prefix announcement
  - Explicitly replicating Sync Interest to all edge routers
    - PSD Sync group: /<network>/PSD/Sync
    - Exchanging ***[router, seq]*** state vector
  - Receiving edge router express Interest to fetch mapping update
    - PSD Data: /<network>/PSD/***<router>/<seq>***
- At PSD startup (router reachability converged)
  - In PET, adds all routers in RIB to PSD Sync ⇒ automatically includes edge
  - In PET, adds PSD Data prefix for every router ⇒ automatically includes edge
- All routers join PSD Sync ⇒ core routers help recover lost state vector
  - All edge routers fetch data

Implementation

- We prototyped SRAN in Golang forwarder ndnd
  - https://github.com/named-data/ndnd/tree/dv2
  - Using NDNLPv2 for encapsulation
  - Routing is merely another app
  - Security Policies (implemented with Light VerSec trust schema)
    - “Apps can only announce prefixes that they own”
    - “A router can only produce its own PSD Data”
- New CLI interfaces for local management
  - prefix-annnounce, prefix-withdraw
  - pet-list, bift-register, bift-list
  - …

Evaluation

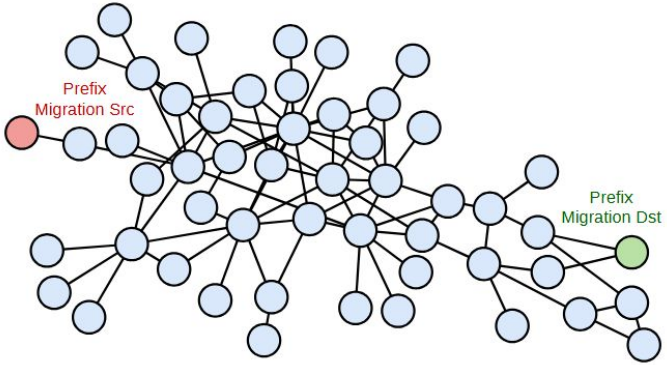
- Three metrics
  - Latency to announce/migrate a prefix in a network: PET ***convergence*** time
    - Migration: withdraw and announce between the farthest edge nodes
  - Packet overhead to announce/migrate a prefix in a network
    - Any difference compared to the existing approach?
  - Forwarding States
    - Legacy approach: FIB
    - SRAN: FIB in core, FIB and PET at edge
- Two topologies
  - AS1239 Sprint PoP: 52-router, 84-link, every node being an edge router
  - AS1755 EBONE: 111 backbone routers, 61 edge/access routers

AS1239: Sprint PoP

- Emulation: all link 10ms delay, 0.1% packet loss
- Negligible latency increase
  - Announce: median +2.06%
  - Migration: median +2.06%
- Significant packet overhead decrease
  - Announce: median -39.25%
  - Migration: median -8.88%

Legacy: network broadcast new announcement

SRAN: implicit multicast tree to the all routers



Operation	Min	Median	Q3 (P75)	Max
Announce (SRAN)	**98ms**	**99ms**	**109ms**	**4.107s**
Announce (Legacy)	95ms	97ms	98ms	4.109s
Migrate (SRAN)	**114ms**	**128ms**	**370ms**	**4.146s**
Migrate (Legacy)	110ms	113ms	320ms	4.123s

TABLE I: The Latency Distribution for Prefix Announcement and Migration in Sprint PoP Topology.

Operation	Min	Median	Q3 (P75)	Max
Announce (SRAN)	**2.43p**	**2.43p**	**2.43p**	**5.93p**
Announce (Legacy)	4.00p	4.00p	4.00p	4.24p
Migrate (SRAN)	**5.29p**	**7.29p**	**7.29p**	**7.57p**
Migrate (Legacy)	8.00p	8.00p	10.13p	13.58p

TABLE II: The Per-Link Packet Overhead for Prefix Announcement and Migration in Sprint PoP Topology.

Forwarding States

- Announcing 1000 unicast prefixes, 200 multicast prefixes
  - Unicast prefixes evenly distributed
  - Each multicast prefix randomly pick three fanout egress routers
- SRAN FIB only contains router reachability
  - Application prefixes all in PET
  - Transit routers only need to look at 52 entries
- Legacy forwarder keeps all prefix in FIB
  - Lookup 1200-entry FIB at each hop

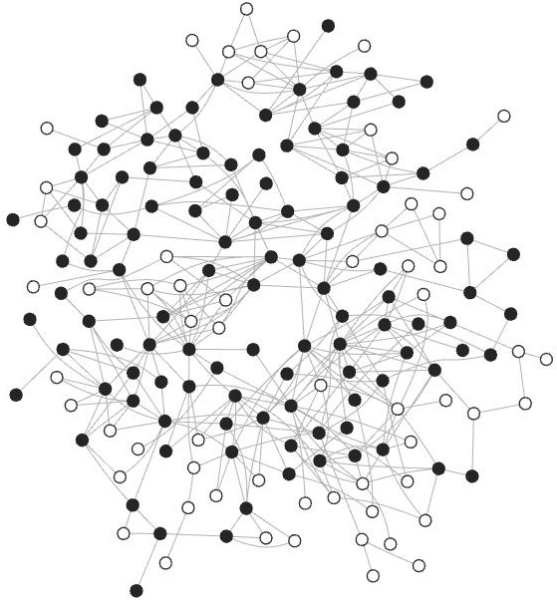
Router	FIB Entries		PET Entries	
	Min	Max	Min	Max
SRAN	52	52	1200 (+60)	1200 (+73)
Legacy	1200 (+60)	1200 (+73)	N/A	

TABLE III: Forwarding-State Comparison between SRAN and Legacy Forwarding.

Additional prefixes are for router sync and distance vector routing. They scale by topology. Variance due to router connectivity degree

AS1755: EBONE

- Simulation, focusing on packet overhead
  - All link 10ms delay
  - Edge-Core 0.1% packet loss
  - Core-Core 0.05% packet loss
- Significant packet overhead decrease
  - Announce: median -38.90%
  - Migration: median -55.86%

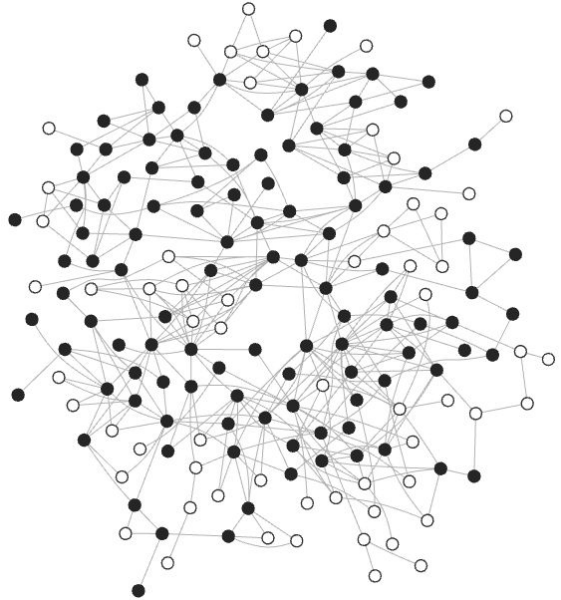


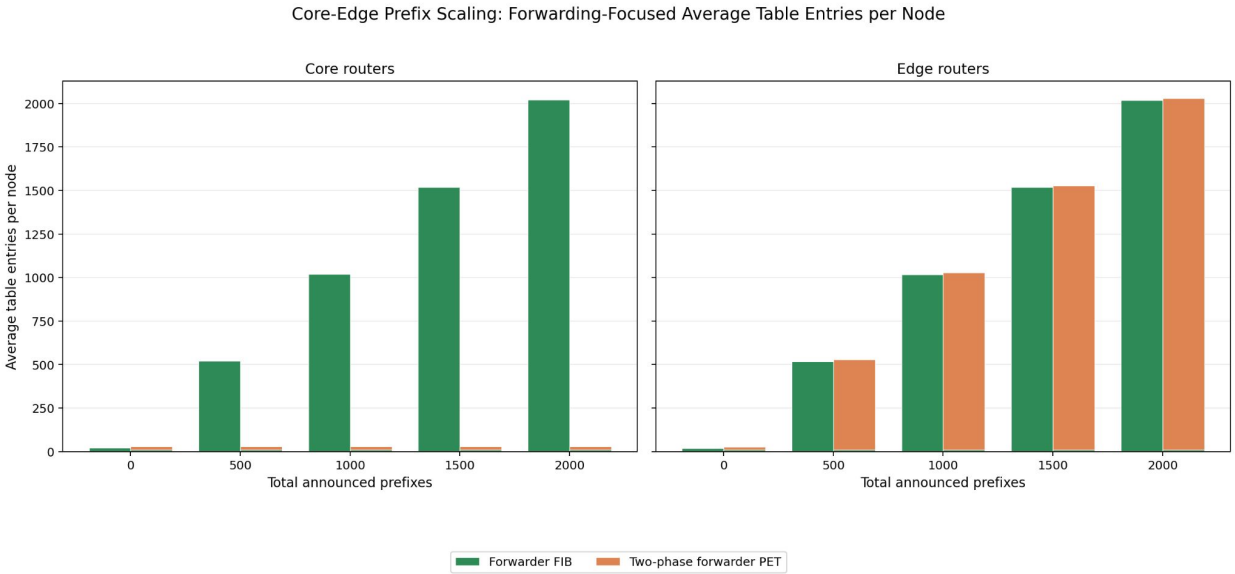
Operation	Min	Median	Q3 (P75)	Max
Announce (SRAN)	**2.45p**	**2.45p**	**2.45p**	**2.46p**
Announce (Legacy)	4.00p	4.01p	4.86p	10.21p
Migrate (SRAN)	**4.90p**	**4.90p**	**4.90p**	**4.91p**
Migrate (Legacy)	9.55p	11.10p	11.88p	25.08p

TABLE IV: The Per-Link Packet Overhead for Prefix Announcement and Migration in AS1755 Topology.

AS1755: EBONE

- Announcing 0-2K prefixes on edge routers





Future Work

- Extending to inter-domain routing
  - Design the “ingress/egress router” at AS level?
- Continue to push for code development
  - Ultimately roll out the new version of NDN Testbed