z-Formation

—or—

Excuse me, Sir, but can we deliver packets securely without addresses?

Presenter Mikko Särelä

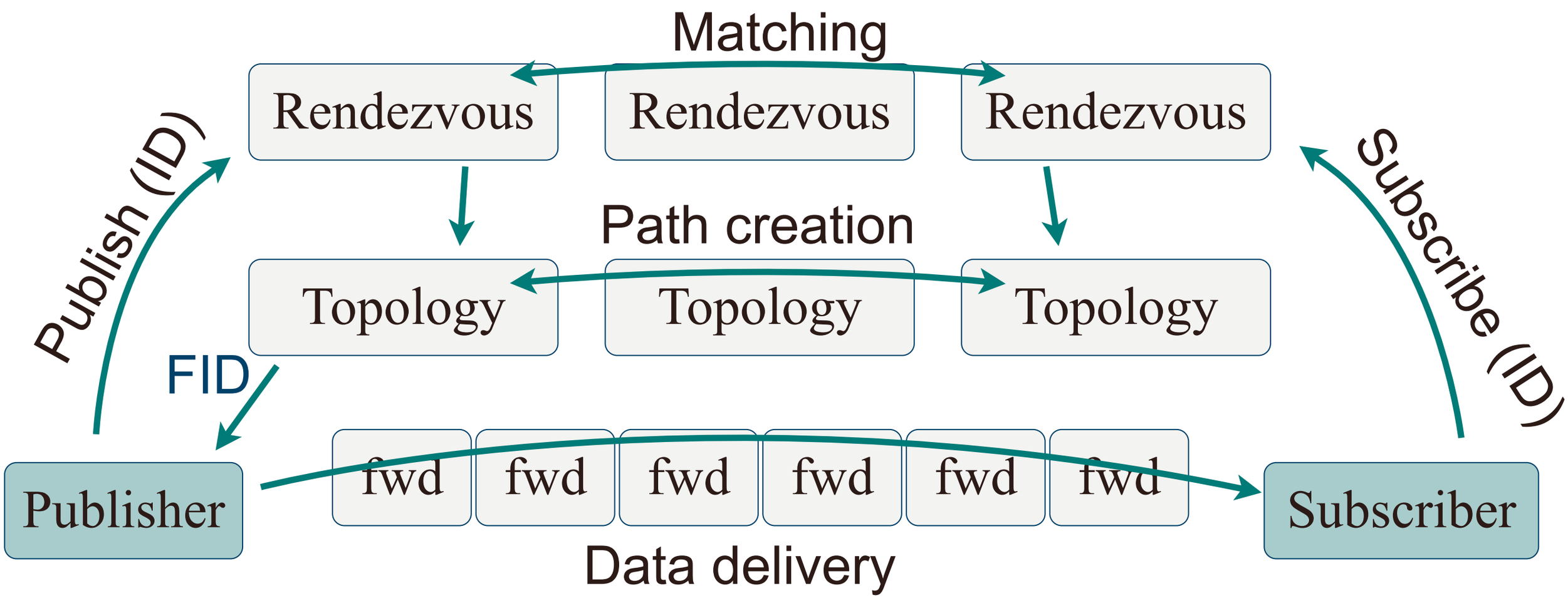
Christian Esteve, Petri Jokela, Pekka Nikander, Mikko Särelä, Jukka Ylitalo

Outline

- Context and motivation
- Potential benefits
- Background: zFilters
  - Forwarding without globally routable addresses
  - Optimisations for better performance (“base case”)
  - Simulation results
- z-Formation
  - Security enhancements (“secure case”)
- Security properties
- Summary

Context - RTFM architecture

- Rendezvous - matching publish and subscribe events
- Topology - network topology knowledge, path creation
- Forwarding - fast delivery



Context

- zFilters: A new *stateless forwarding method*
- z-Formation: A secure variant of z-Filters
- Architecturally *compatible with (G)MPLS* control plane
- Forwarding much **simpler** than today
- Prototypes of basic zFilters implemented for FreeBSD and NetFPGA
  - zFilters published at SIGCOMM 2009
  - Code available http://psirp.org/downloads

Forwarding	table	decision
zFilter	~2000 bit / port	~10 gates / port
z-Formation	~200 bits	~64 kgates

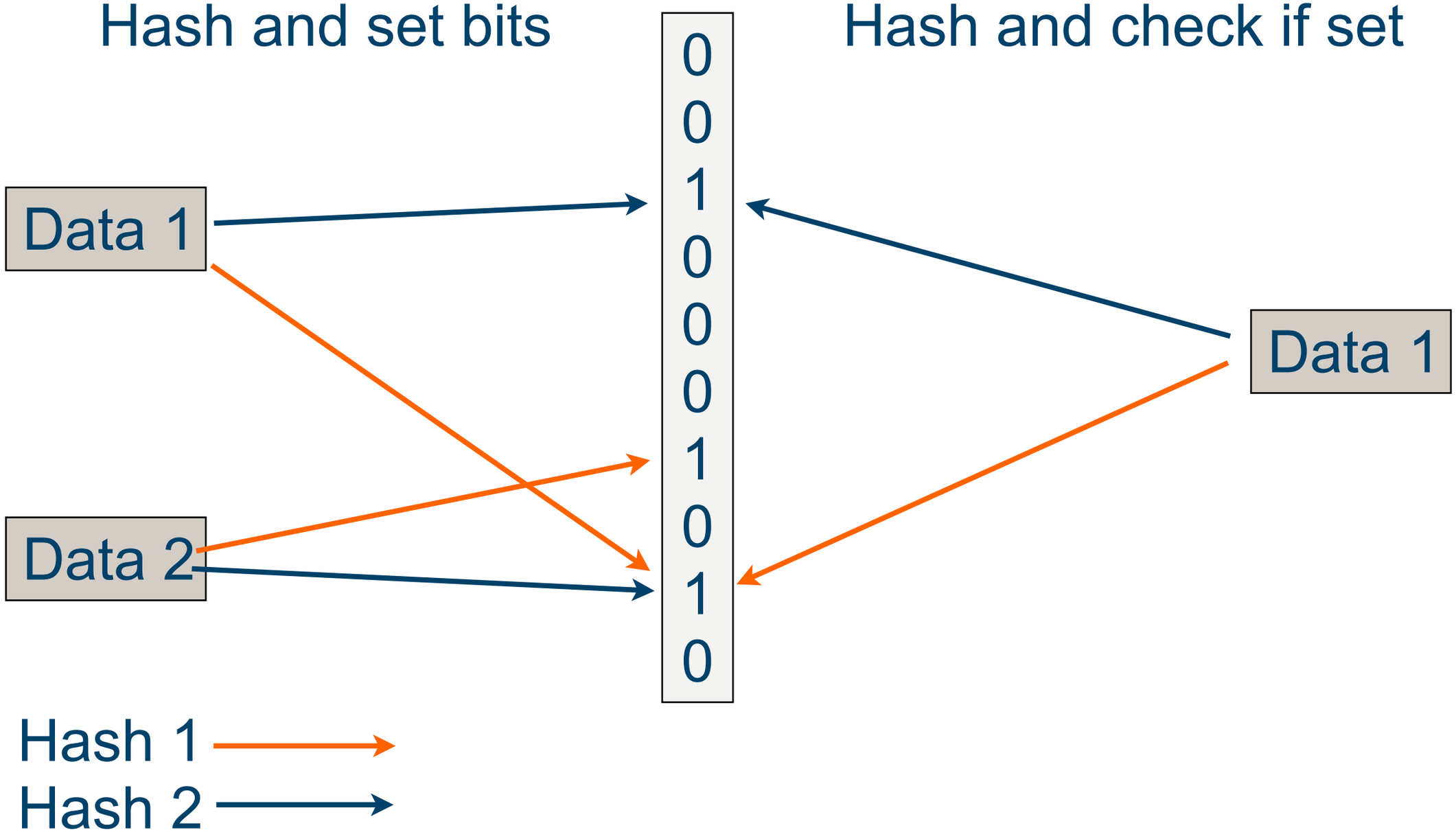
Motivation

- Simplicity
- Multicast support
- Efficiency
  - Fairly short, constant-delay switching time
  - Can likely be done at line speed
- Small forwarding tables
  - zFilters: ~2000 bits / port
  - z-Formation: Minimum ~256 bits / node
    - Trade-off between table size and gate logic
- Increase security over existing zFilter solution
  - source route identifier as a capability

Bloom Filters - basic idea

Inserting items
Hash and set bits

Verifying
Hash and check if set



zFilters

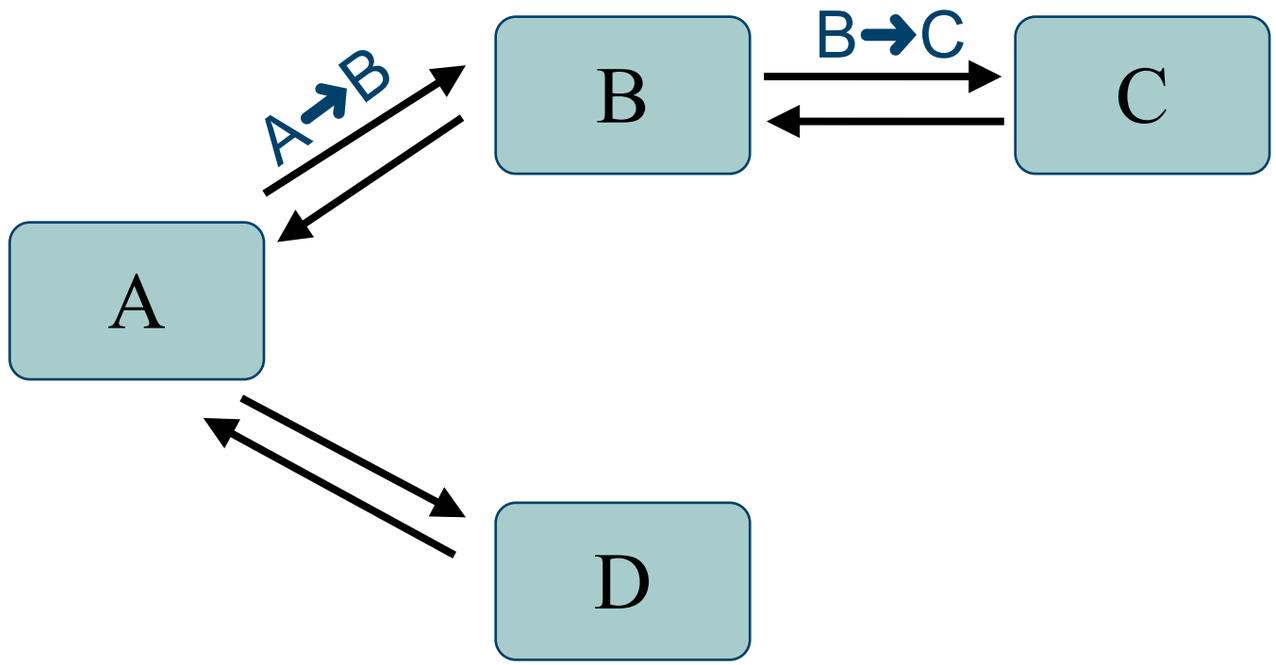
Technical details: Introduction

- Give names to links, not to nodes
- Form a source-route using the link names
- Encode the set, as a Bloom filter, into the packet header
- Main drawback: false positives due to using Bloom filters

- Details on next slides:
  - Link-identity-based source routing
  - Forwarding decisions
  - Optimising with multiple link identifiers
  - Simulation results

Link IDs and Bloom filters (zFilters)

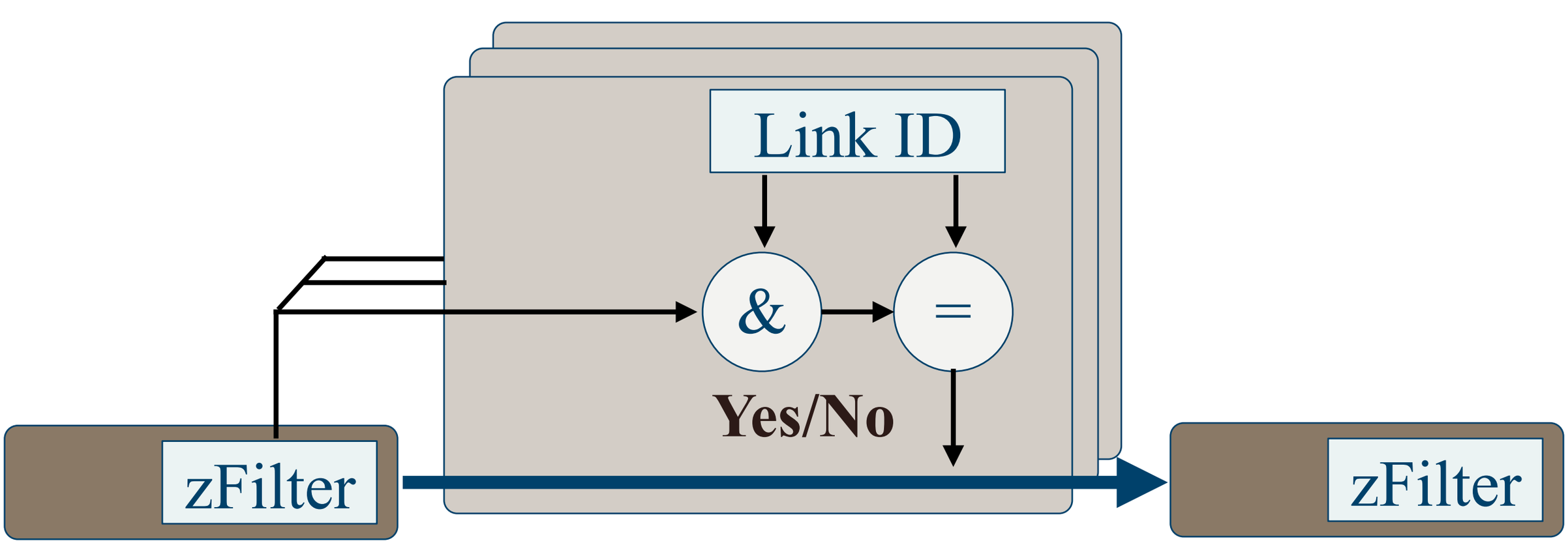
- No names for nodes
  - Each link identified with a unidirectional Link ID
- Link IDs
  - Statistically unique
  - Periodically changing
  - Size e.g. 256 bits
  - Local or centrally controlled
- Source routing
  - Encode Link IDs into a Bloom filter (zFilter)
  - Naturally multicast
- "Stateless"



A→B	0	1	0	0	0	1	0	0	1
B→C	1	0	0	0	0	1	1	0	0
zF: A→B→C	1	1	0	0	0	1	1	0	1

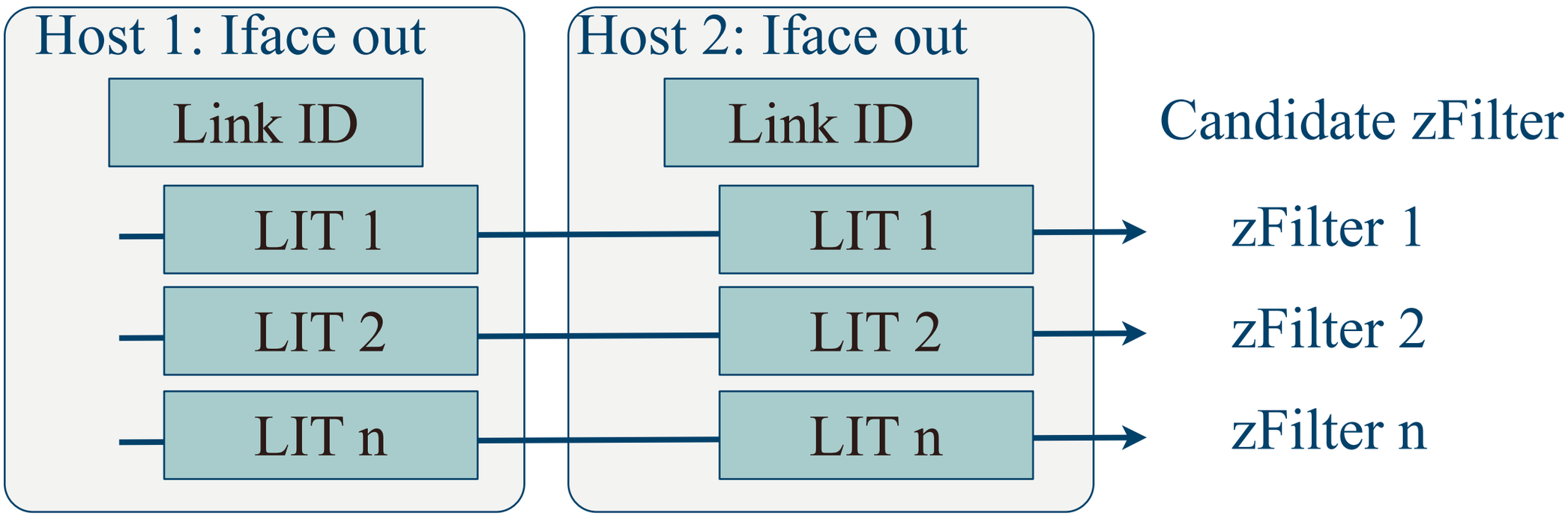
Forwarding Decision

- Forwarding decision based on binary AND and CMP
  - zFilter in the packet matched with all outgoing Link IDs
  - Multicasting: zFilter contains more than one outgoing links

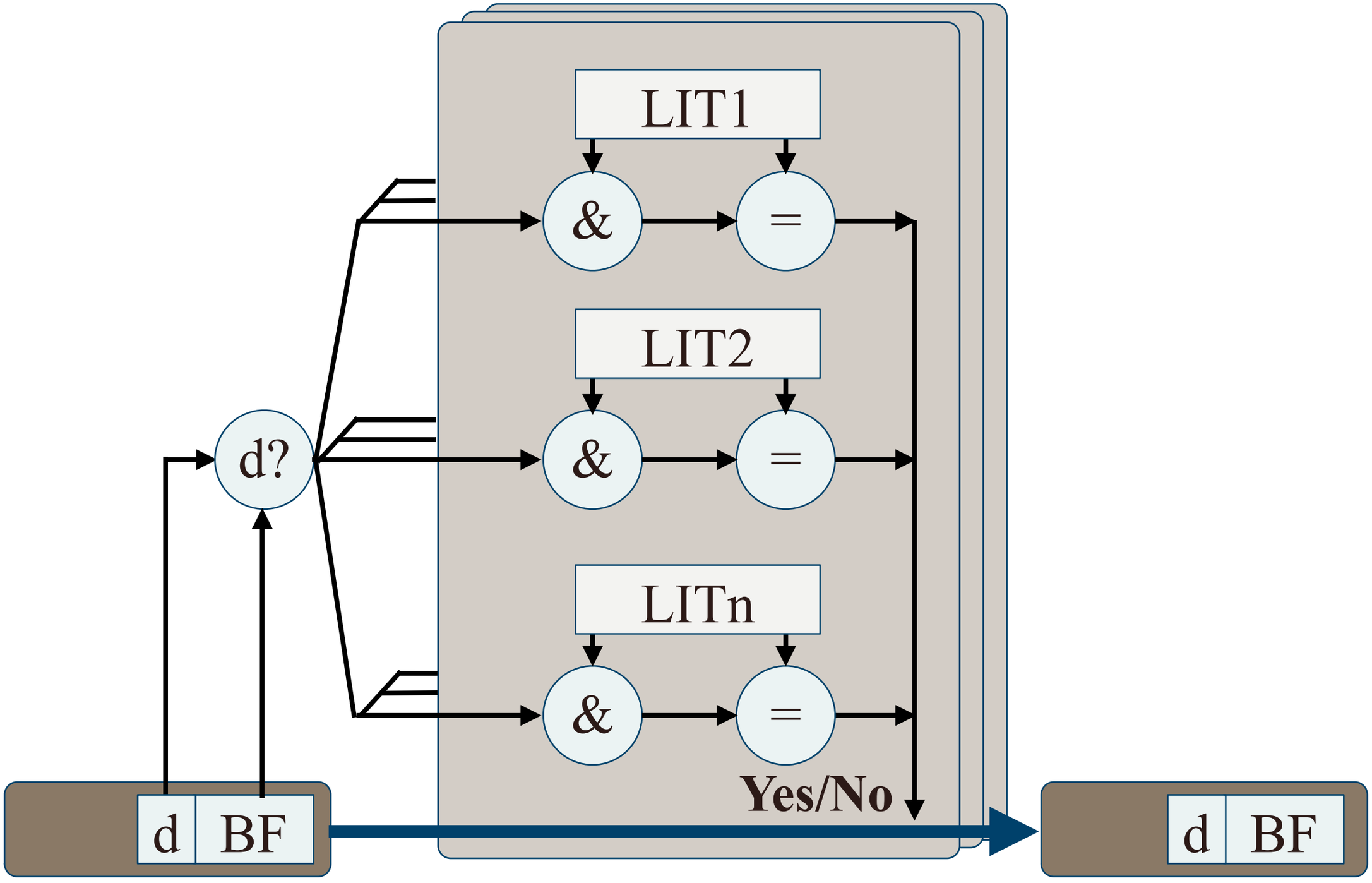


Using Link Identity Tags (LIT)

- Better forwarding efficiency with a simple trick
  - Define *n* different LITs instead of a single LID
  - LIT has the same size as LID, and also *k* bits set to 1
  - [Power of choices]
- Route creation and packet forwarding
  - Calculate *n* different candidate zFilters
  - Select the best performing zFilter, based on some policy

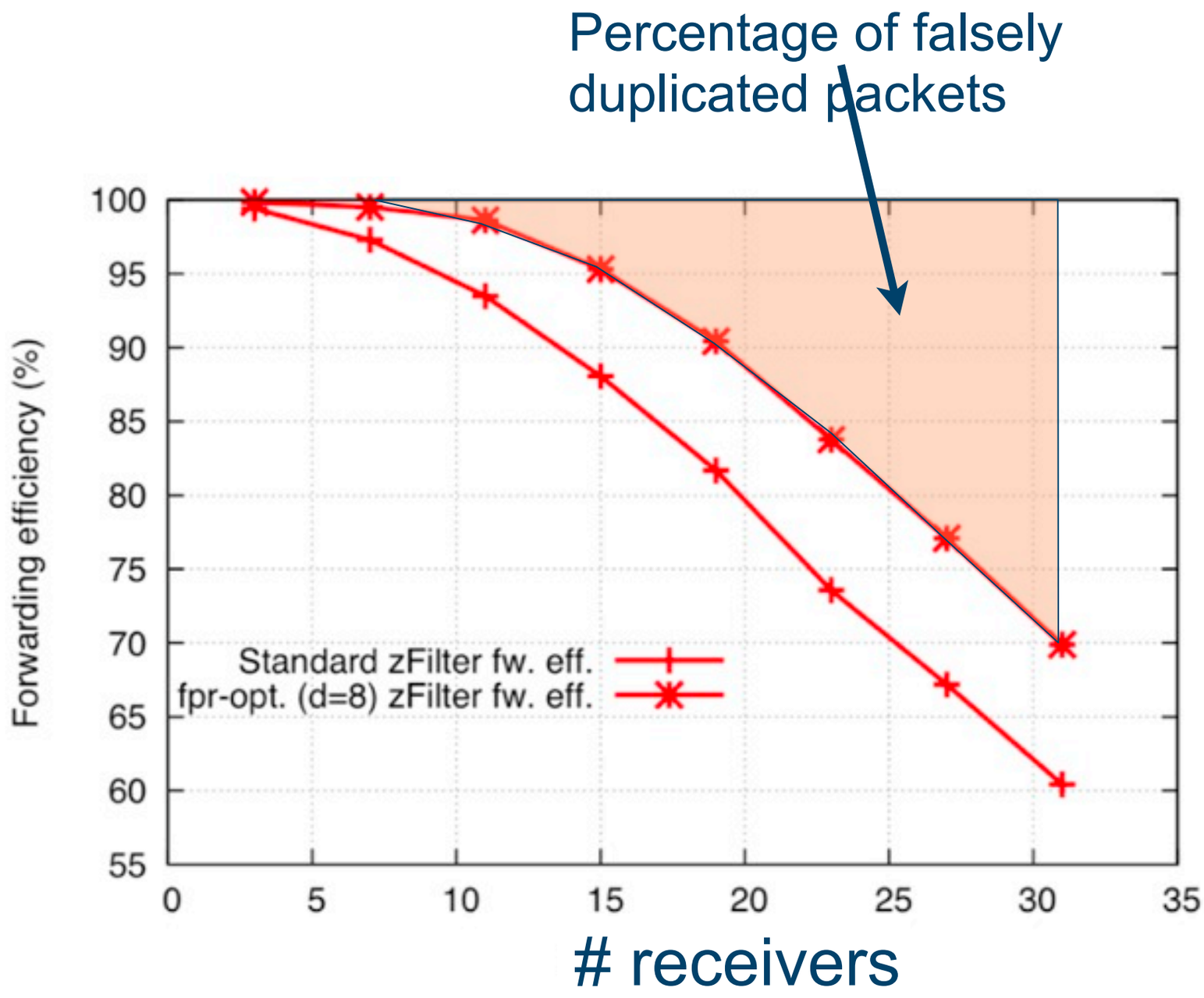


Using Link Identity Tags (LIT)



Forwarding efficiency

- Simulations with
  - Rocketfuel
  - SNDlib
- Forwarding efficiency
- 20 receivers
  - ~35 links
  - Basic LID: 80%
  - Optimised: 88%
    - with 8 LITs
- Unicast
  - easy
  - Internet paths < 14 AS hops



Technical details: Summary

- Name unidirectional links, not hosts
- Form a source route (path or tree)
- Encode source route as an in-packet Bloom filter

- small “stateless” forwarding table
- Simple constant-time forwarding decisions

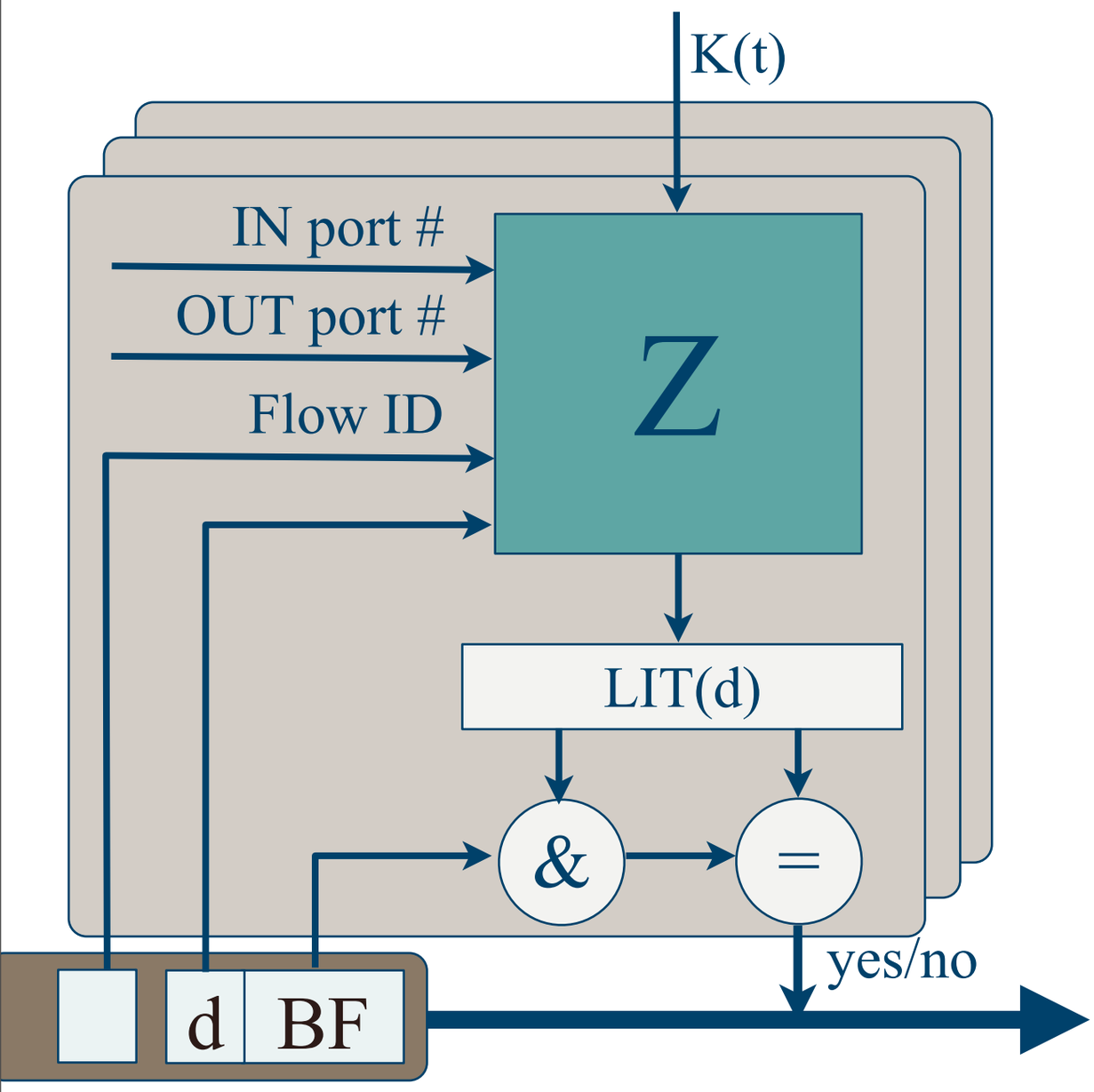
- Use multiple alternative link names (LITs)
  - Minimises forwarding inefficiency caused by false positives
- Generate LITs run-time on per-flow bases
  - Binds forwarding identifiers to specific flows and paths

z-Formation

Forwarding security

- Goal: to ensure (probabilistically) that hosts cannot send un-authorized traffic
- zFilter weaknesses
  - zFilter replay attacks
  - Computational attack
    - correlate zFilters to learn link IDs
  - Traffic injection attack
- Solution (z-Formation): Compute LIT in line speed and bind it to
  - path: in-coming and out-going port
  - time: periodically changed key
  - flow: flow identifier (e.g. IP 5-tuple / content id)

Secure case: z-Formation
aka Secure in-packet BFs



- Form LITs algorithmically
  - at packet handling time
  - LIT(d) = Z (I , K (t), In, Out, d),
- Secure periodic key *K*
- Input port index
- Output port index

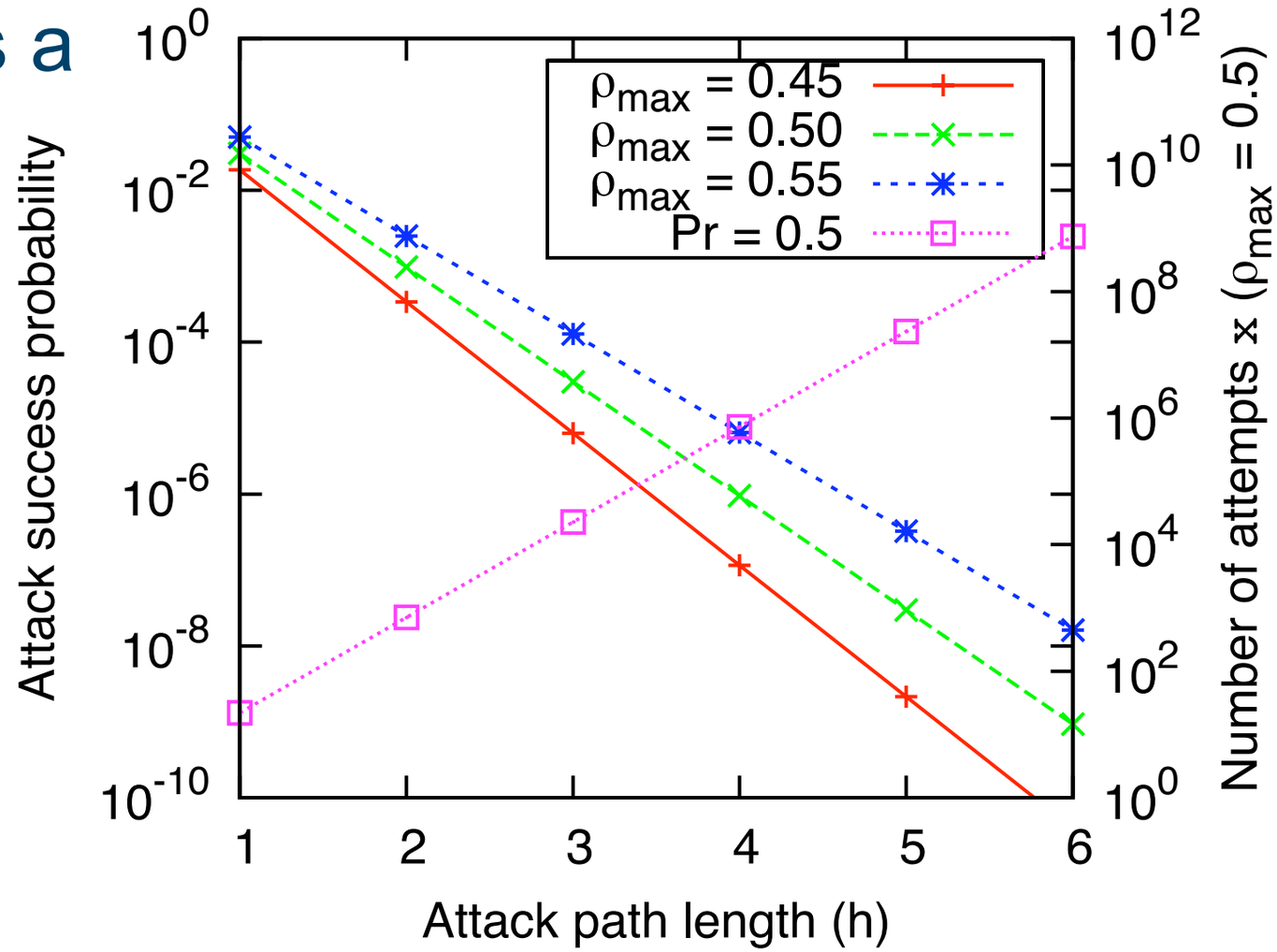
- *Flow ID* from the packet, e.g.
  - Information ID
  - IP addresses & ports
- *d* from the packet

Security properties

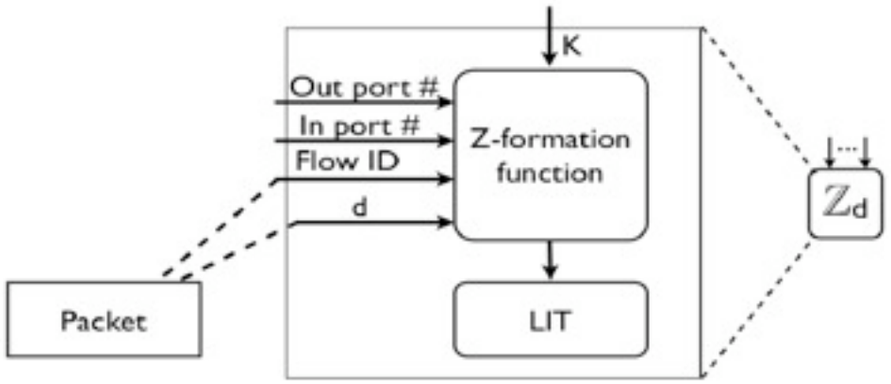
- z-Filter works **both** as a forwarding ID and a capability
  - To send, a host needs to know or guess a valid zFilter
- Base z-Filters
  - Bound to the *outgoing* ports along the path
    - Traffic injection possible
  - Hard to construct one without knowing LITs along the path
    - Correlation attacks possible
- z-Formation
  - Bound to the *incoming* and *outgoing* ports
    - Traffic injection difficult (due to binding to *incoming* port)
  - Very hard to construct one without knowing keys along the path
    - Correlation attacks possible only for a given flow ID
  - Bound to the packet stream (flow ID)
  - Need a cryptographically good Z algorithm

Injection attacks

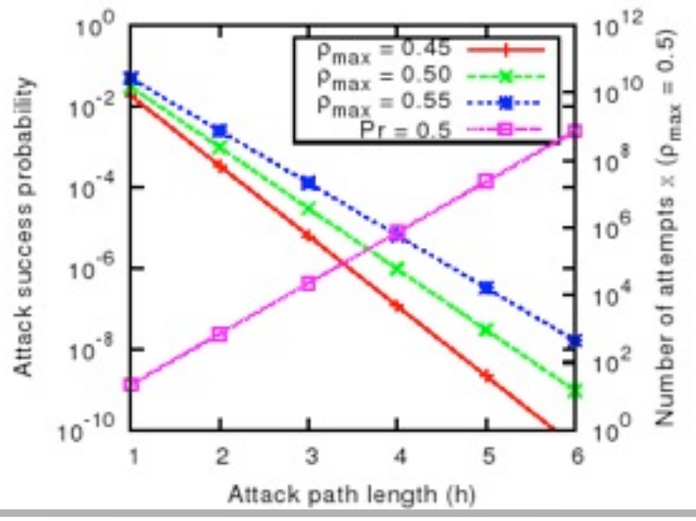
- Assuming attacker knows a zFilter passing at h hops distance from attacker
- Left y-axis shows the probability of a single packet reaching target for various fill factors
- Right y-axis shows the average number of attempts for one successful injection with probability 0.5



Discussion



- Replay attacks: limited to the key lifetime
  - As zFilters are tied to periodically changing keys (K(t)), one per node, the capabilities become expirable
- Brute force attack: “Best” attack strategy
  - Assuming attack traffic of 1M pps (1Gbps / 1000 bits pp)
    > 40min to guess (with Pr=0.5) one 5-hop working zFilter (which is only usable for single host)
- Re-keying time?
  - Trade-off between minimizing duration of unwanted traffic vs. overhead of zFilter renewal e.g., 1 min enough to complete transactional traffic + protect short paths
- Attack detection and mitigation:
  - fpr increase triggers detection plus
    e.g. blacklist mechanism on FlowID (I)



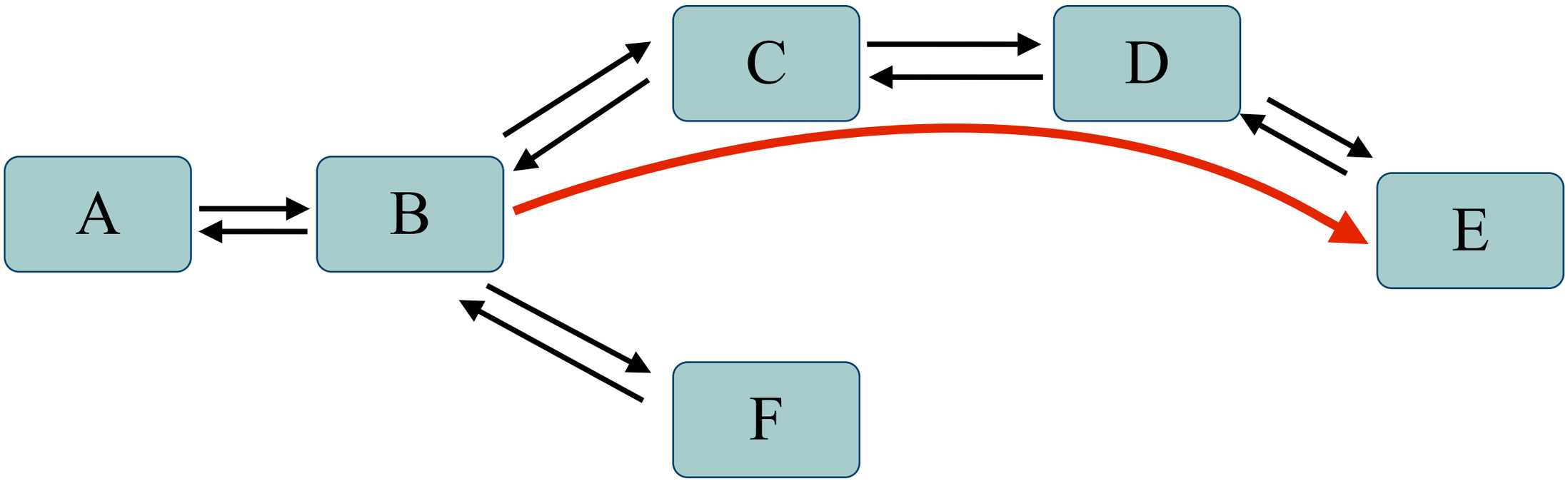
Conclusions

- z-Formation is a compact and secure source routing method
- Forwarding identifiers are
  - small and efficient to compute
- Capability like properties
  - expirable,
  - bound to packet flow,
  - content/communication intention
- Stateless
  - No need for per flow state
  - Forwarding with zero FIB table lookups

Comments? Questions?

Scalability beyond 20: Virtual links

- Popular paths/large trees represented as virtual links
  - A single Link ID for the tree
  - Additional state in the forwarding nodes
  - Increases scalability



Virtual B→C→D→E 0 0 1 0 1 0 0 0 1