Forwarding with in-packet Bloom Filters

T-110.6120

9.10.2012

Jimmy Kjällman

Ericsson Research, NomadicLab

Background

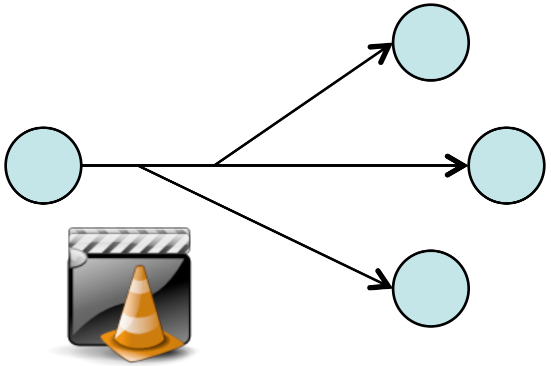
General Starting Points

- New *Future Internet* architecture
- Focus on *long-term* research
  - With feedback to short-term work
- *Clean slate* approach
  - Reconsidering old assumptions
- *Redesigning* the Internet architecture
  - Considering both technical and socio-economic aspects
- *Information-Centric Networking*
  - Various projects around the world

Choices and Goals (and Constraints)

- *Information-centric*
  - Not host centric
- *Publish/subscribe*
  - Instead of send/receive
- *Identify information*
  - No (global) node addresses

- Secure and efficient networking
  - DDoS protection, multicast, …



Projects

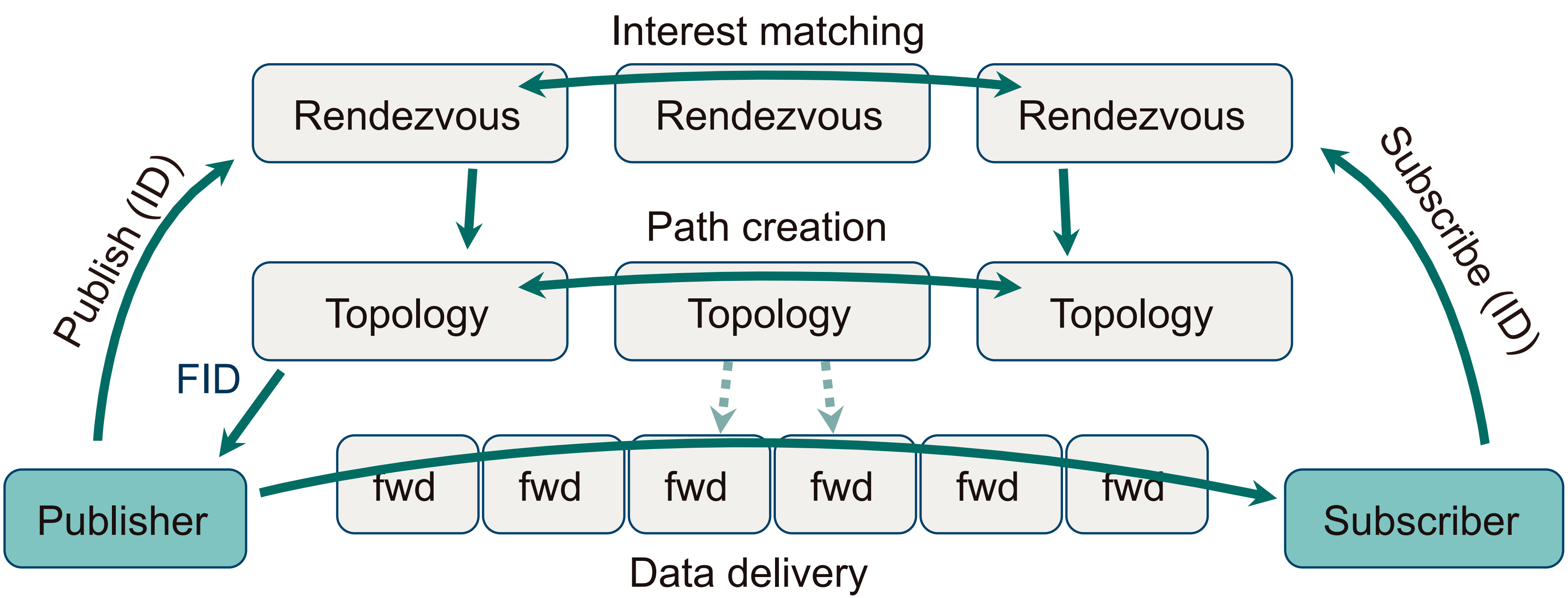
- **EU FP7 PSIRP 2008-2010**
  Publish/Subscribe Internet Routing Paradigm

- **EU FP7 PURSUIT 2010-2013**
  Publish/Subscribe Internet Technology

- **ICT SHOK FI WP3 2008-2012**

PSIRP/PURSUIT Basic Architectural functions

- **Rendezvous – matching publish and subscribe events**
- **Topology – network topology knowledge, path computation**
- ***Forwarding – fast data delivery***



Ideas about Forwarding

- Need for a new forwarding mechanism in PSIRP
- Some requirements
  - Multicast support
  - Security (receiver in control, DDoS protection)
  - Efficiency
- One of the initial ideas: MPLS-like labels

- Another idea: Bloom filters
  - Very little state and signaling required, native multicast support, no global addressing, path not revealed, no routing tables and lookups, no pushing/popping, …

LIPSIN

- <u>Li</u>ne Speed <u>P</u>ublish/<u>S</u>ubscribe <u>I</u>nter-<u>N</u>etworking

- **Petri Jokela(*), András Zahemszky, Christian Esteve, Somaya Arianfar, and Pekka Nikander, "LIPSIN: Line speed Publish/Subscribe Inter-Networking", ACM SIGCOMM 2009**

(* Original author of most of these presentation slides.)

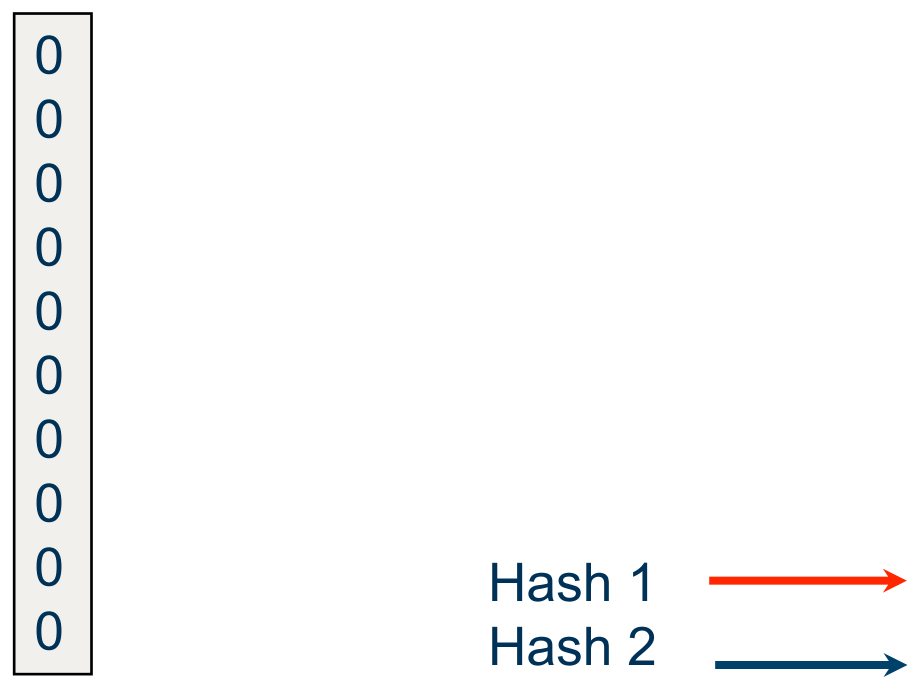
Bloom filters –
Burton Howard Bloom, 1970

Pursuing a Pub/Sub Internet PURSUIT

Bloom filters

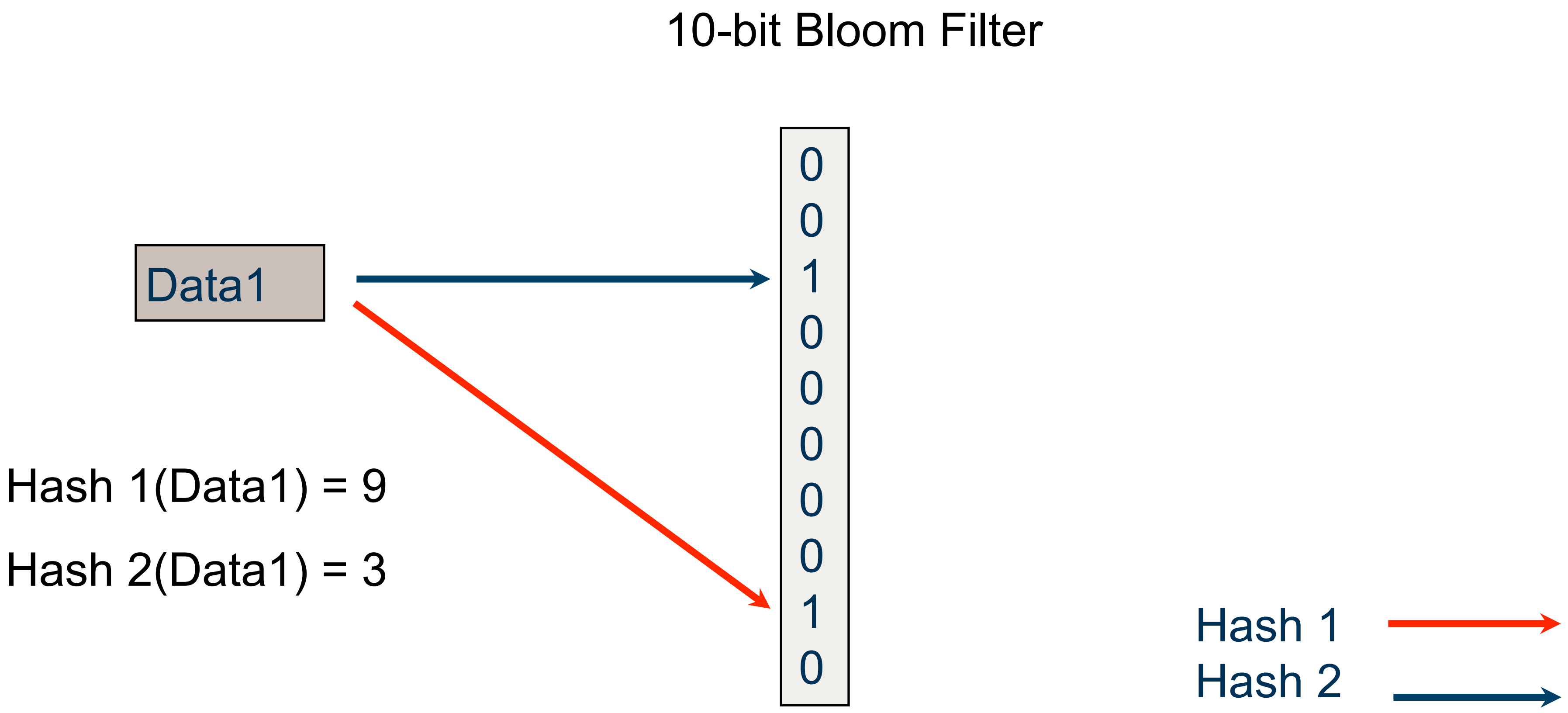
- Probabilistic data structure, space efficient
- Used to test if an element has been added to a set

10-bit Bloom Filter



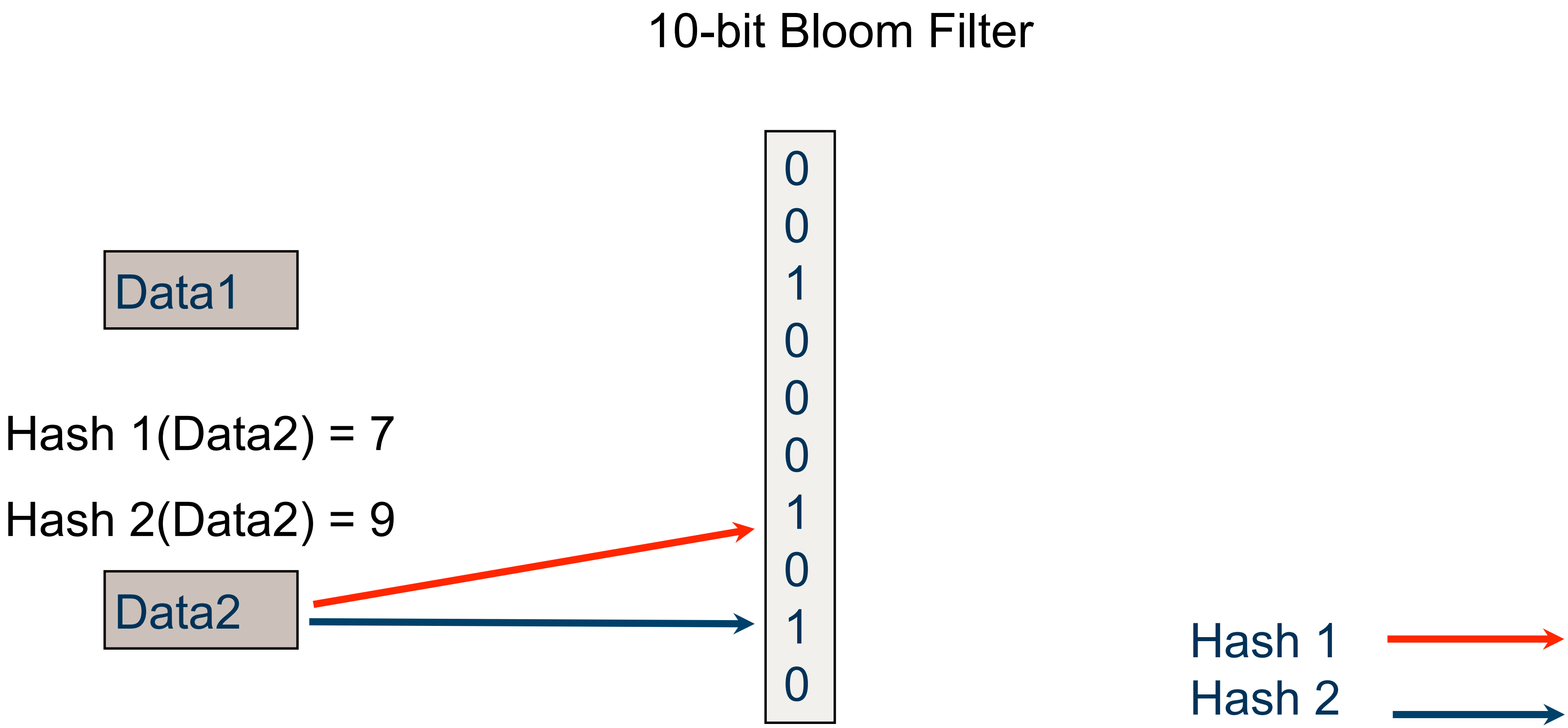
Bloom filters: Inserting items

- **Hash the data *k* times, get index values, and set the bits**



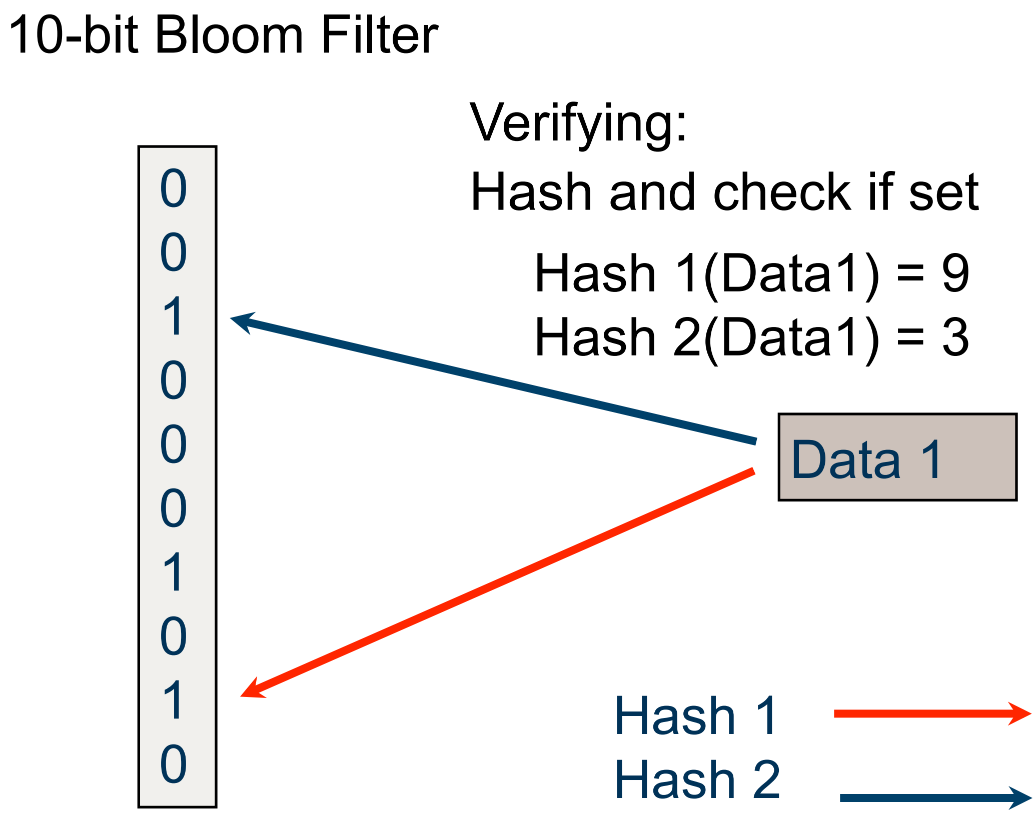
Bloom filters: Inserting items

- **Hash the data *k* times, get index values, and set the bits**



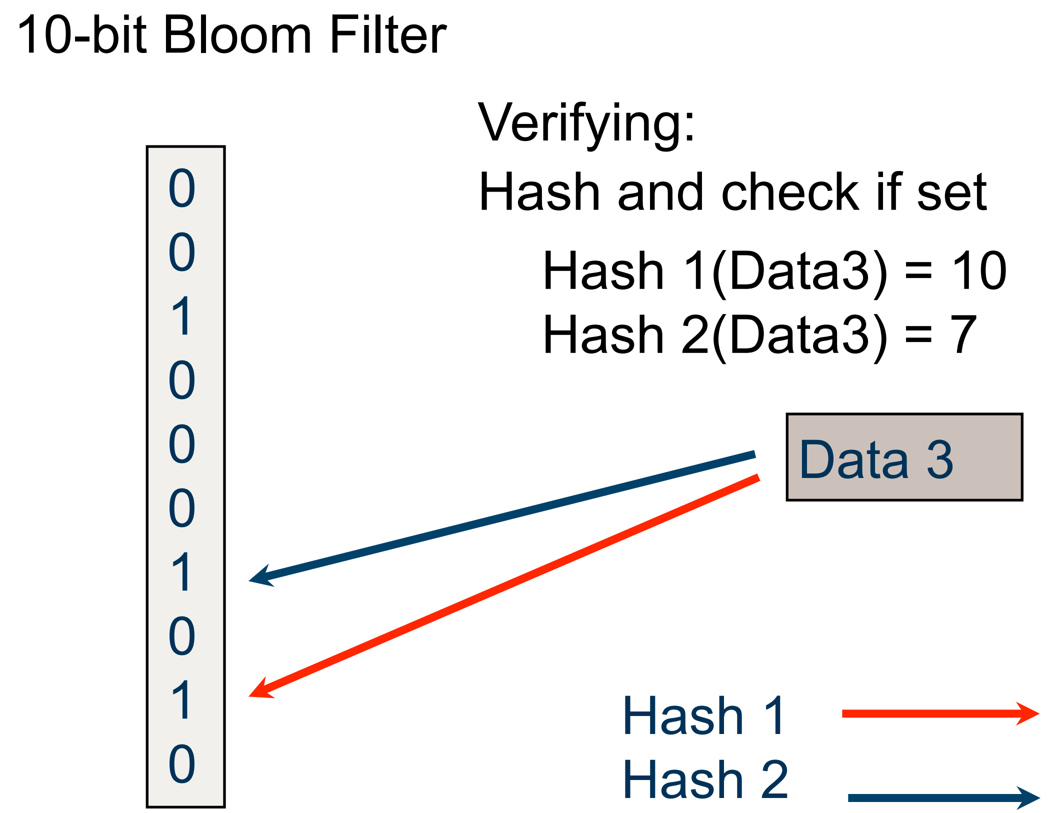
Bloom filters: Verifying (positive)

- **All corresponding bits have been set → positive response**



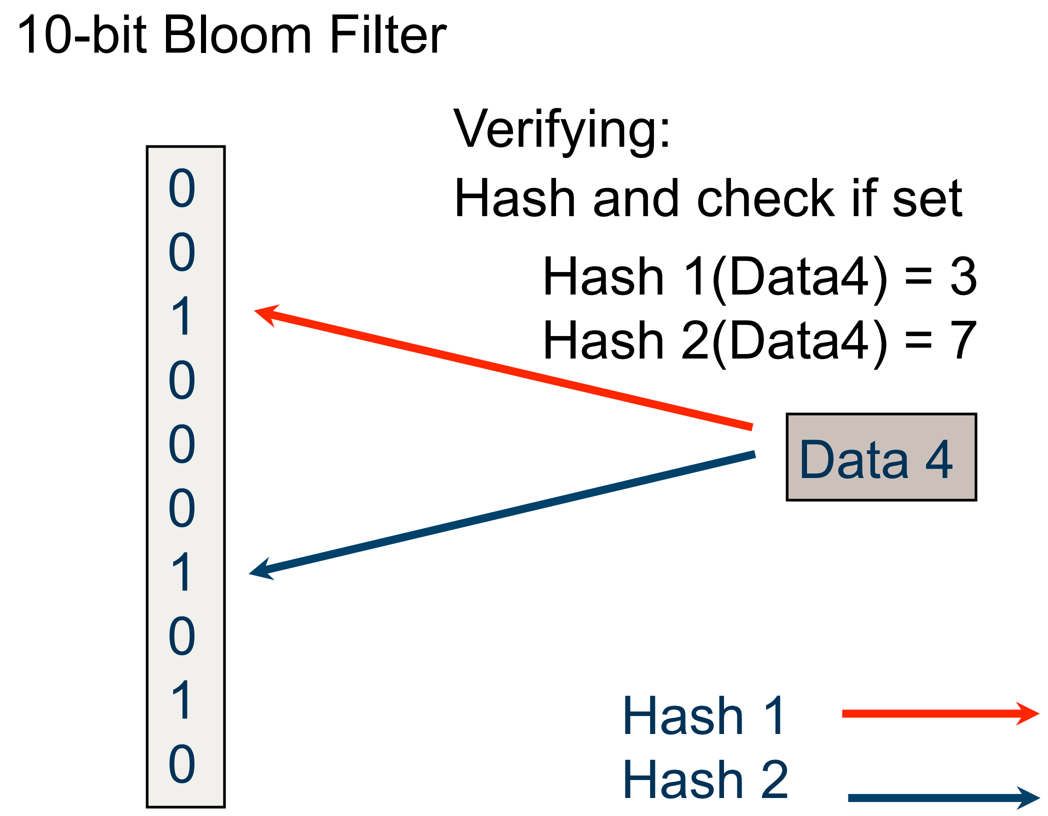
Bloom filters: Verifying (negative)

- **Some bits do not match → negative response**



Bloom filters: False positives

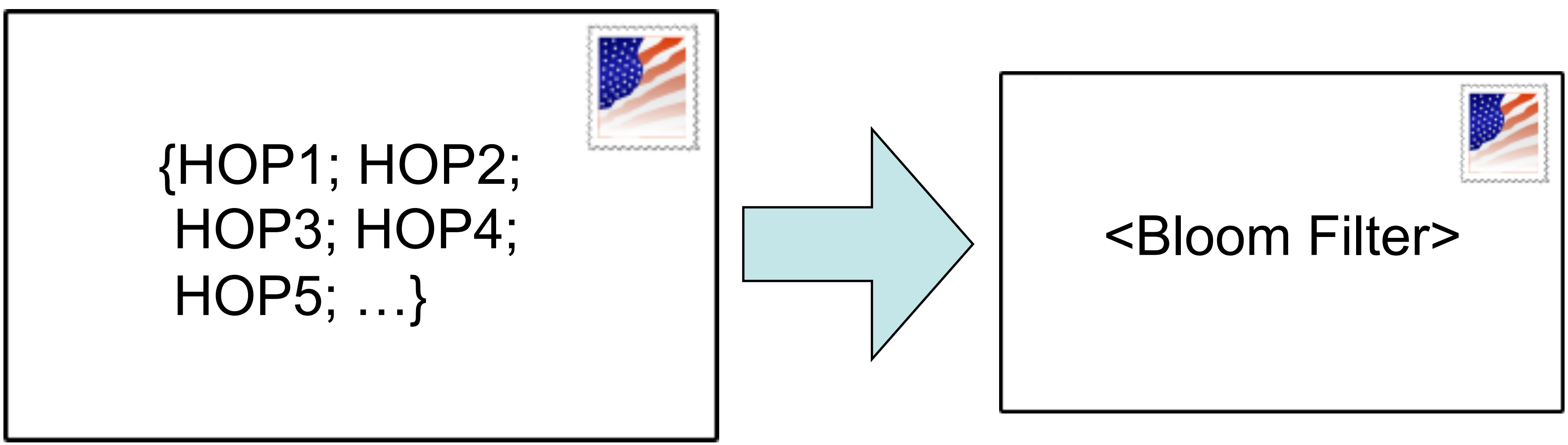
- **Bits match the BF although "Data 4" was never added**



In-packet bloom filters– zFilters

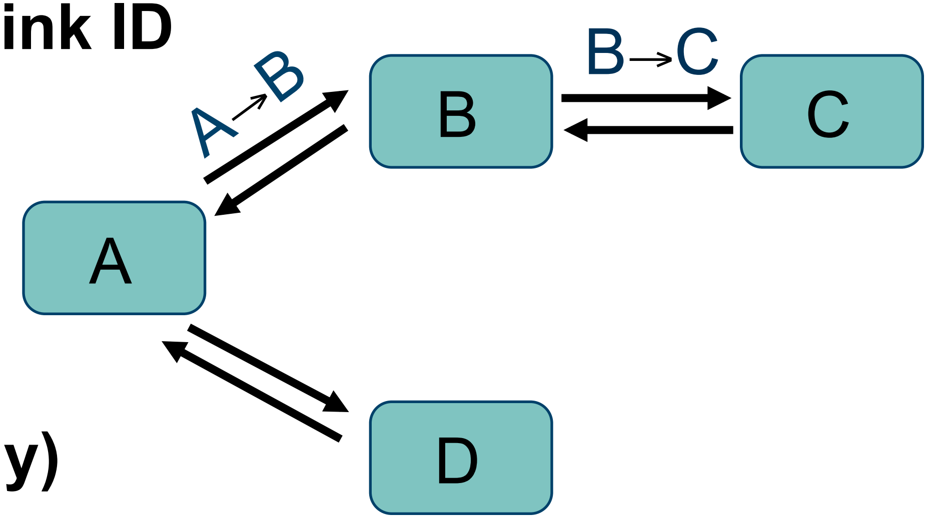
Forwarding with zFilters

- **Source routing**
- **Explicitly enumerating all hops requires a lot of space – so instead we encode this information into a Bloom filter**



Link IDs

- No names for nodes
  - Each link is identified with a uni-directional (outgoing) Link ID

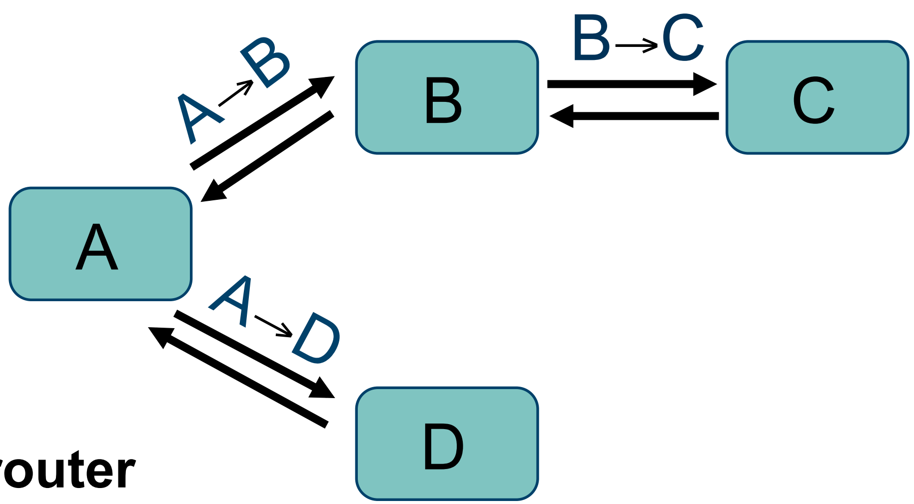


- Link IDs
  - No hashing required, generate the 1-bits otherwise (e.g. randomly)
  - Size e.g. 256 bits of which 5 bits set to 1
    - 2 x the size of an IPv6 addr
    - Statistically unique

A->B	0	1	0	0	0	1	0	0	1
B->C	1	0	0	0	0	1	1	0	0

Link IDs and zFilters

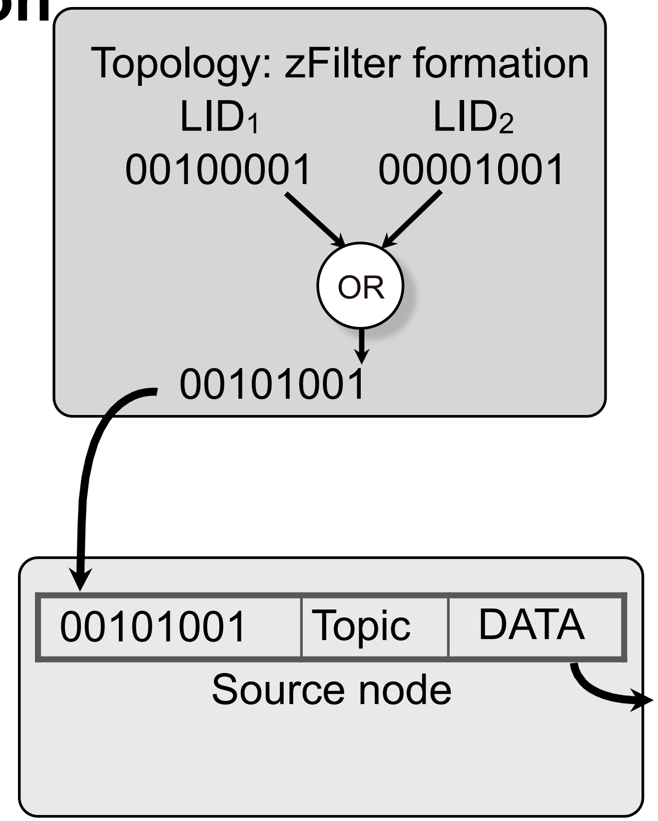
- **Strict source routing**
  - Create a path, collect all Link IDs
  - Include (OR) all path's/tree's Link IDs into a Bloom filter
- **Multicast support**
  - Include multiple outgoing links from one router
- **Stateless (almost)**
  - Only Link IDs stored on the router
- **Packet forwarding**
  - Always to the correct destination
  - False positives possible



A->B	0	1	0	0	0	1	0	0	1
B->C	1	0	0	0	0	1	1	0	0
A->C	1	1	0	0	0	1	1	0	1

Topology manager's role

- **Needs (intra-)network link information**
  - **Topology and Link IDs**
  - **E.g., OSPF, PCE**
- **Computes paths on request**
  - **Creates the zFilter using the Link ID information**
  - **Gives the zFilter to the source node**
    - **(Source adds zFilter to outgoing data packets)**

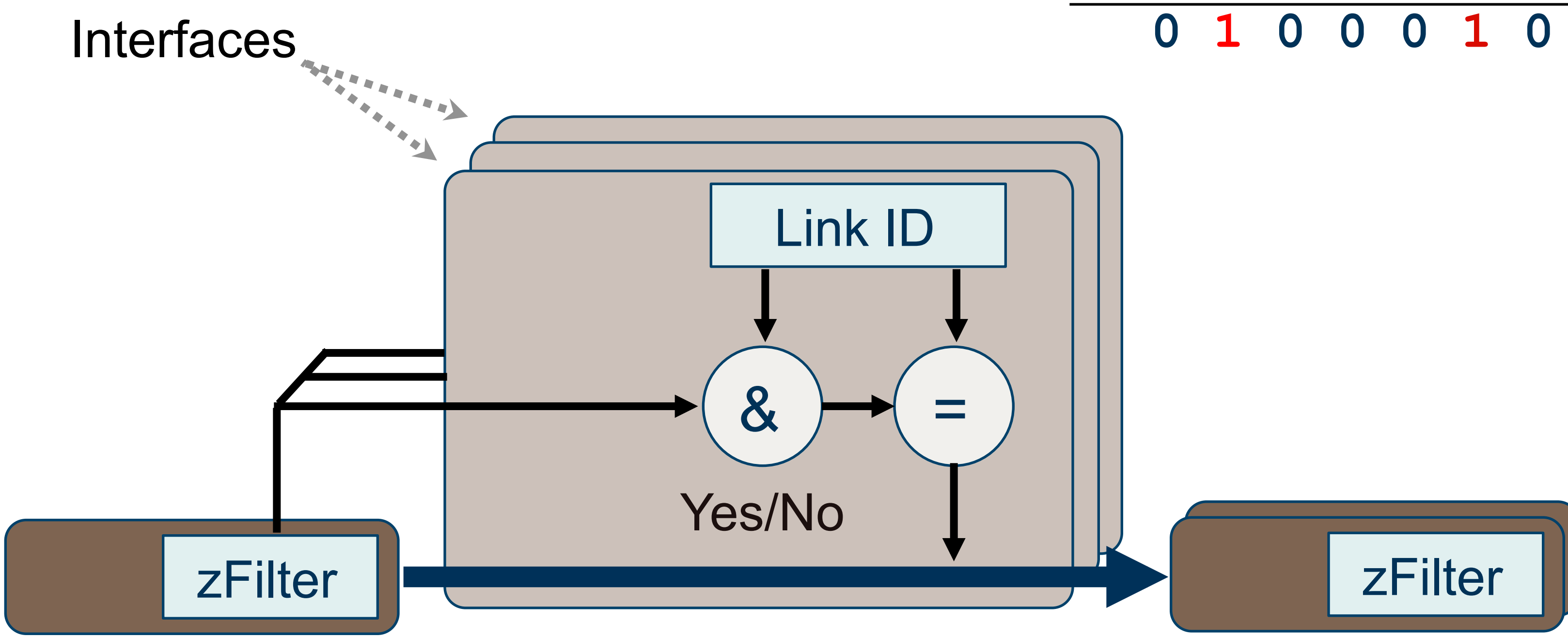


Forwarding decision

- **Forwarding decision based on binary AND and a comparison**
  - **zFilter in the packet matched with all outgoing Link IDs**
  - **Forward if: *zFilter AND LID = LID***

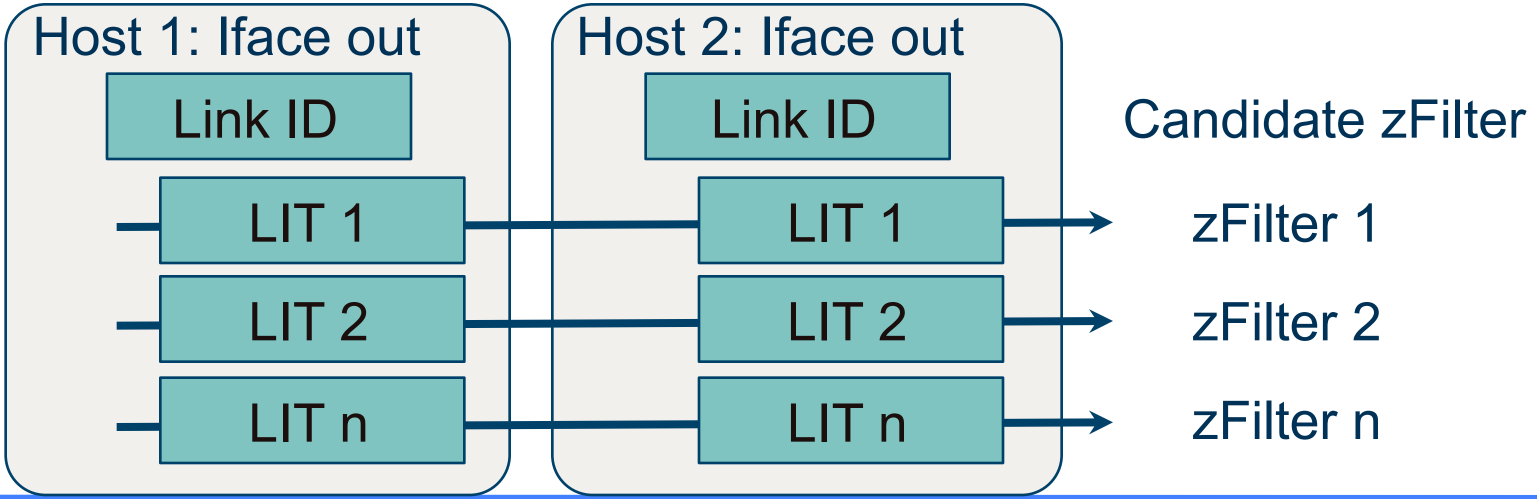
(⇔ *(zFilter AND LID) XOR LID = 0*)

	1	1	0	0	0	1	1	0	1
&	0	1	0	0	0	1	0	0	1
	0	1	0	0	0	1	0	0	1

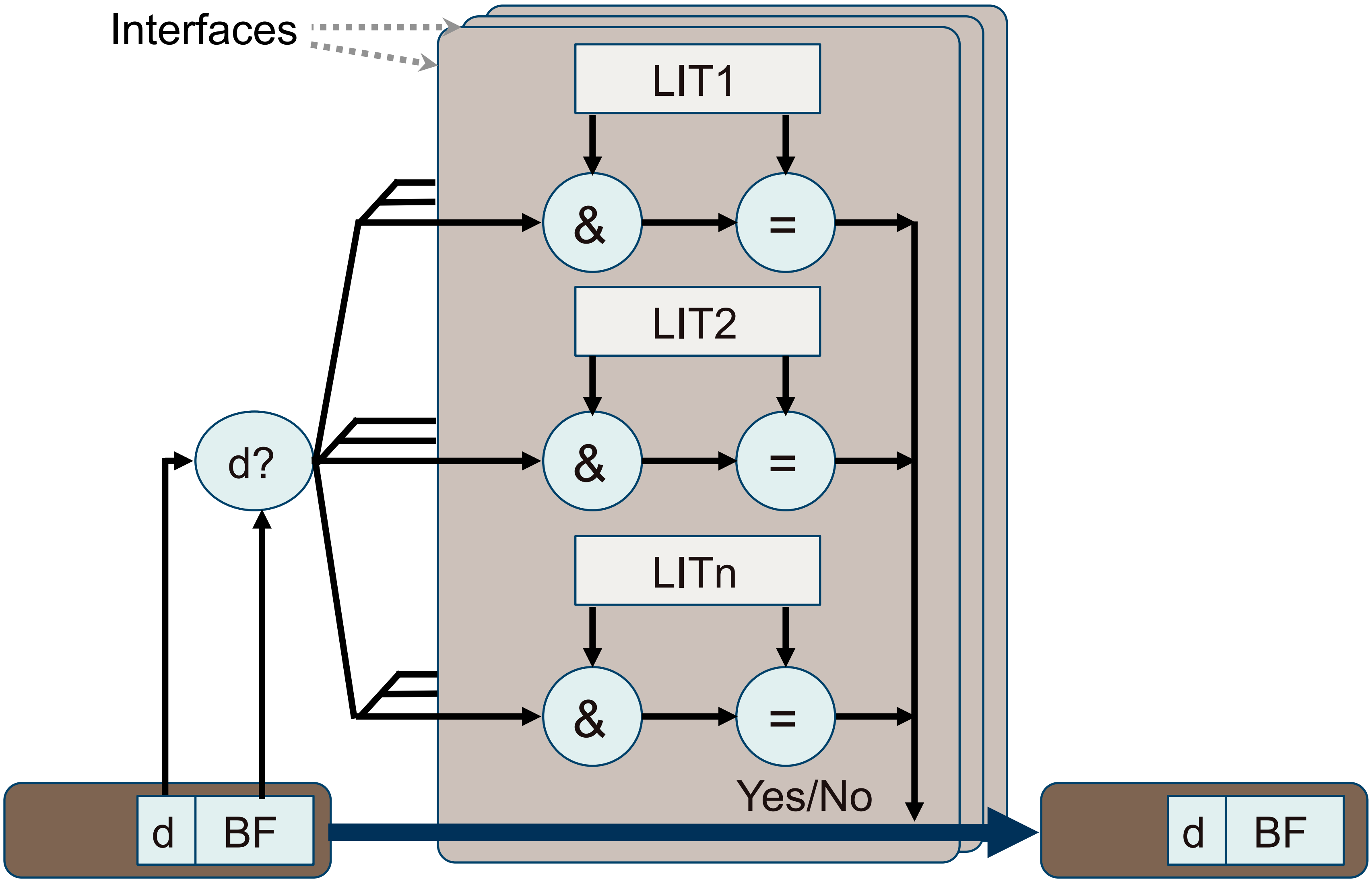


Using Link Identity Tags (LIT)

- **Goal: Better false positive rate**
  - **Define *n* different LITs instead of a single LID**
  - **LIT has the same size as LID, and also *k* bits set to one**
  - **Power of choices**
- **Route creation and packet forwarding**
  - **Calculate n different candidate zFilters**
  - **Select the best performing zFilter (index *d*) and use that**

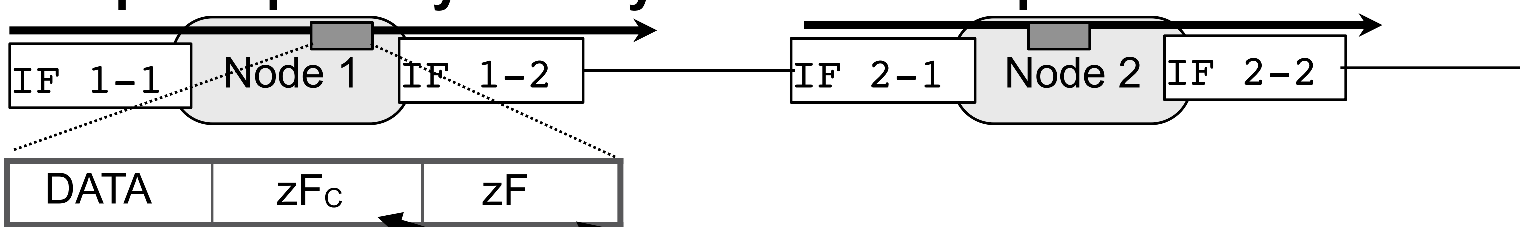


Using Link Identity Tags (LIT)



zFilter collection

- **During packet traversal, the reverse zFilter can be easily generated**
  - **Add a field in the packet for collected zF**
  - **All routers forwarding the packet add the incoming LID to the field**
  - **Once the packet arrives to the destination, the collected zF can be used to forward data to the reverse direction**
  - **Simple especially with symmetric links/paths**



$zF_C = zF_C \text{ OR } LID_{1\text{-}1}$

Add incoming, match outgoing

Interface	Link ID
IF 1-1	00110000
IF 1-2	00001001

Interface	Link ID
IF 2-1	01010000
IF 2-2	10000010

Evaluation

Forwarding speed

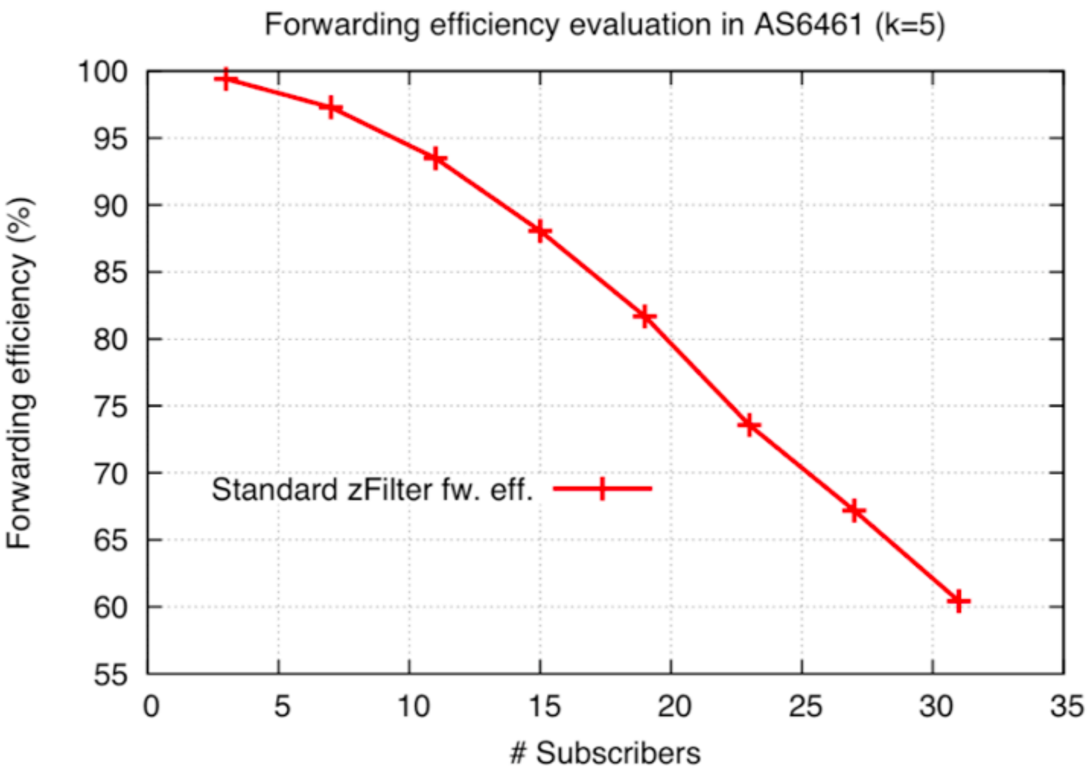
- **Measured on a NetFPGA**
- **Results**
  - **No routing table lookups → lower latency compared to IP**
  - **zF latency stays constant, independent of the network size**
  - **Line speed**

Path	Avg. latency	Std dev.
Plain wire	94 μs	28 μs
IP router	102 μs	44 μs
zFilter	96 μs	28 μs

- **Measurements in Blackadder (software)**
  - **Early results indicate that line speed forwarding over 10 Gbit/s links can be achieved**

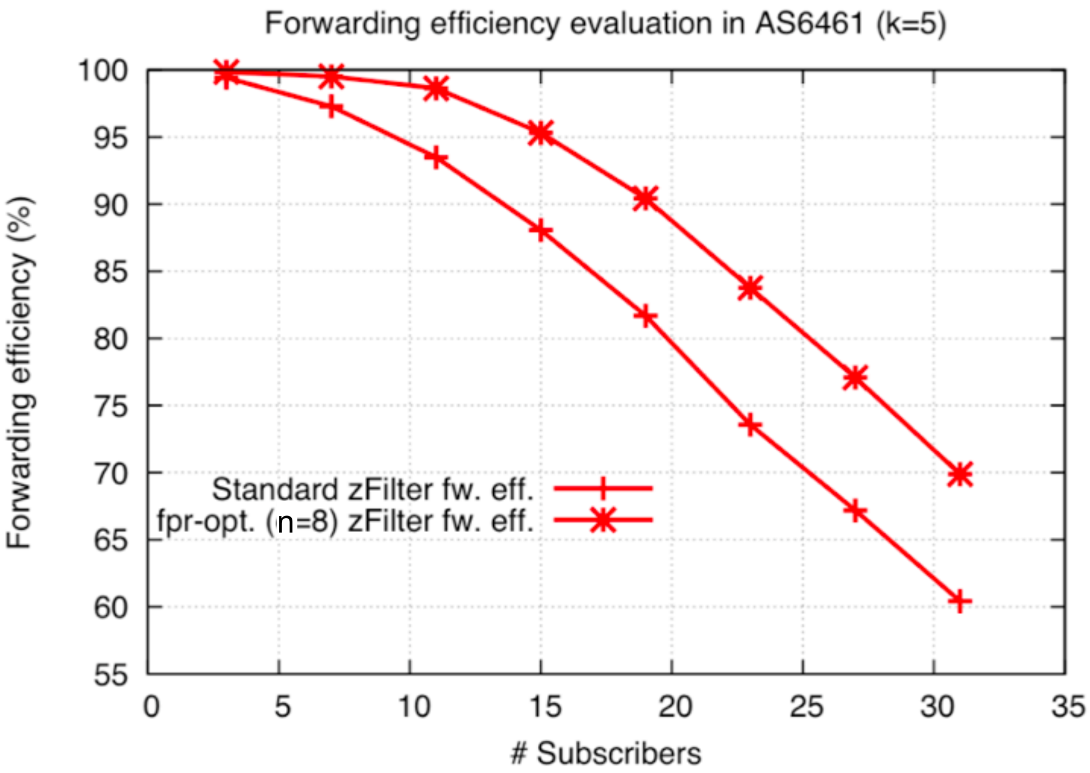
Forwarding efficiency

- **Simulations (ns-3) with**
  - **Rocketfuel**
  - **SNDlib**
- **Forwarding efficiency with 20 subscribers**
  - **~80%**
- **AS6461: 138 nodes, 372 links**



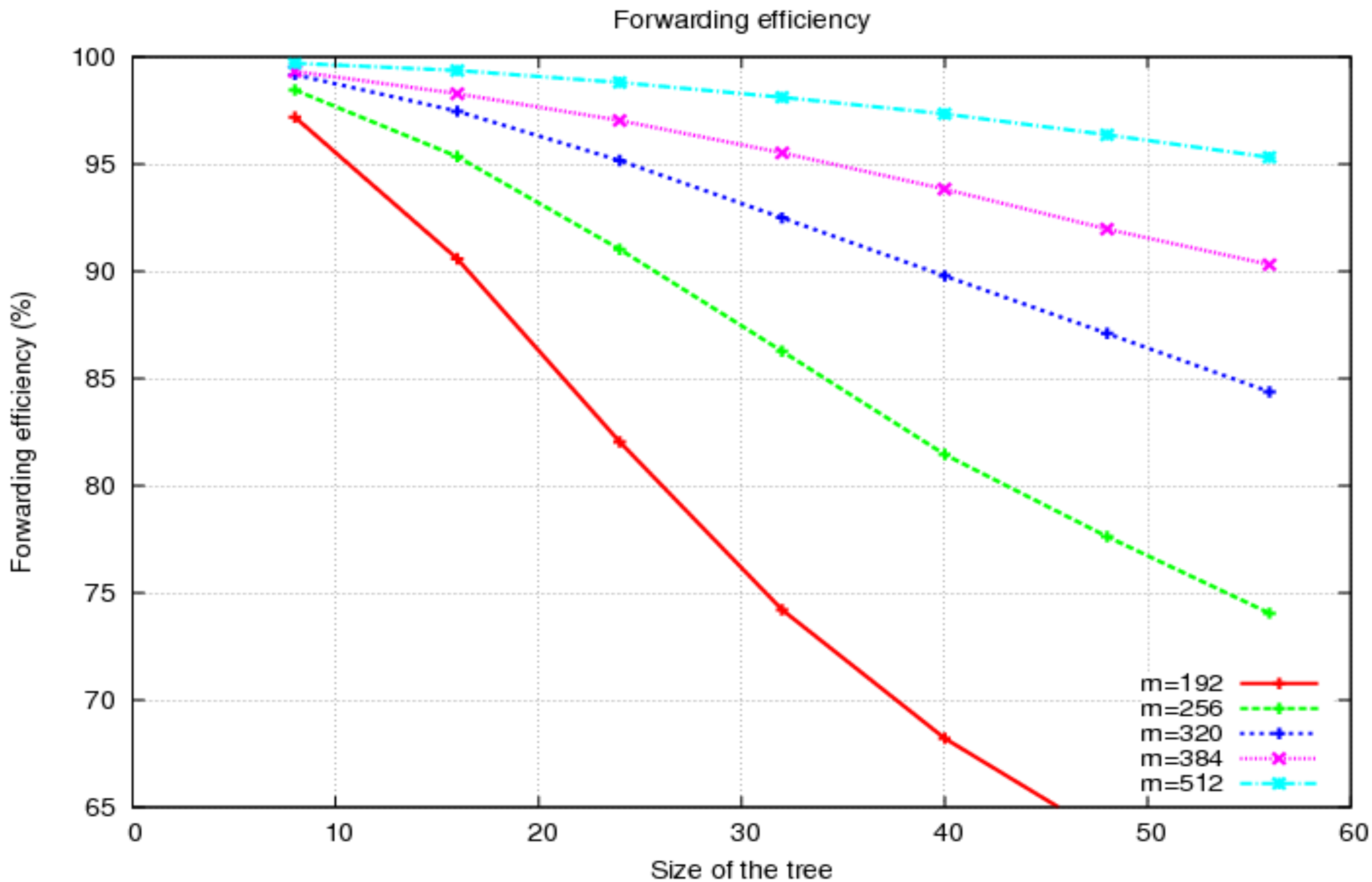
Forwarding efficiency

- **Simulations with**
  - **Rocketfuel**
  - **SNDlib**
- **Forwarding efficiency with 20 subscribers**
  - **~80%**
  - **LIT Optimized: 88%**



Changing zFilter size

AS3967: 79 nodes, 147 bi-directional links



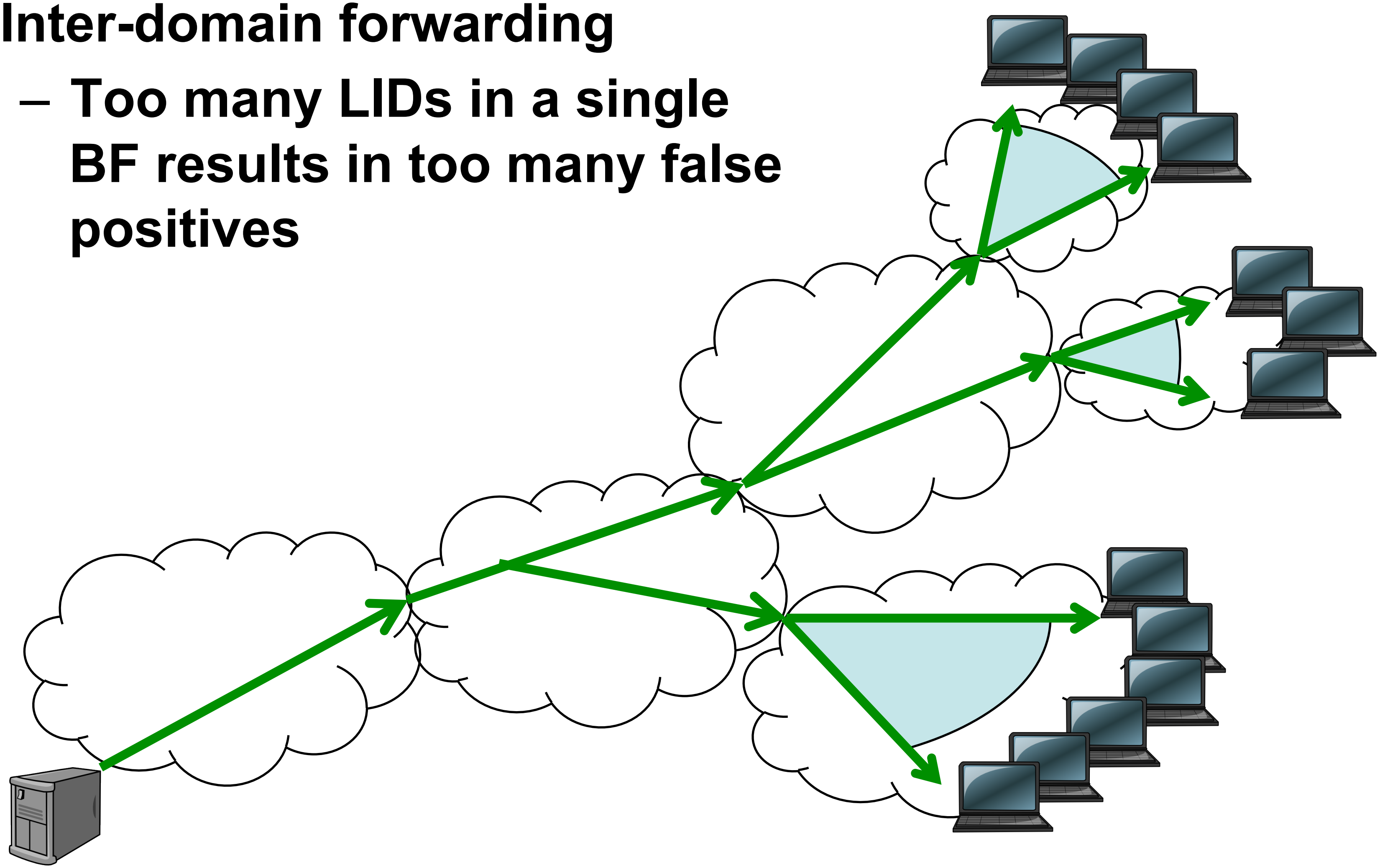
Security

- **A zFilter to a destination only works on a certain path, while IP addresses work from any source anywhere**
  → Better (although not complete) DDoS resistance
- **zFilter doesn't reveal (directly) which nodes are involved in the communication**
  → Better privacy

Scalability enhancements

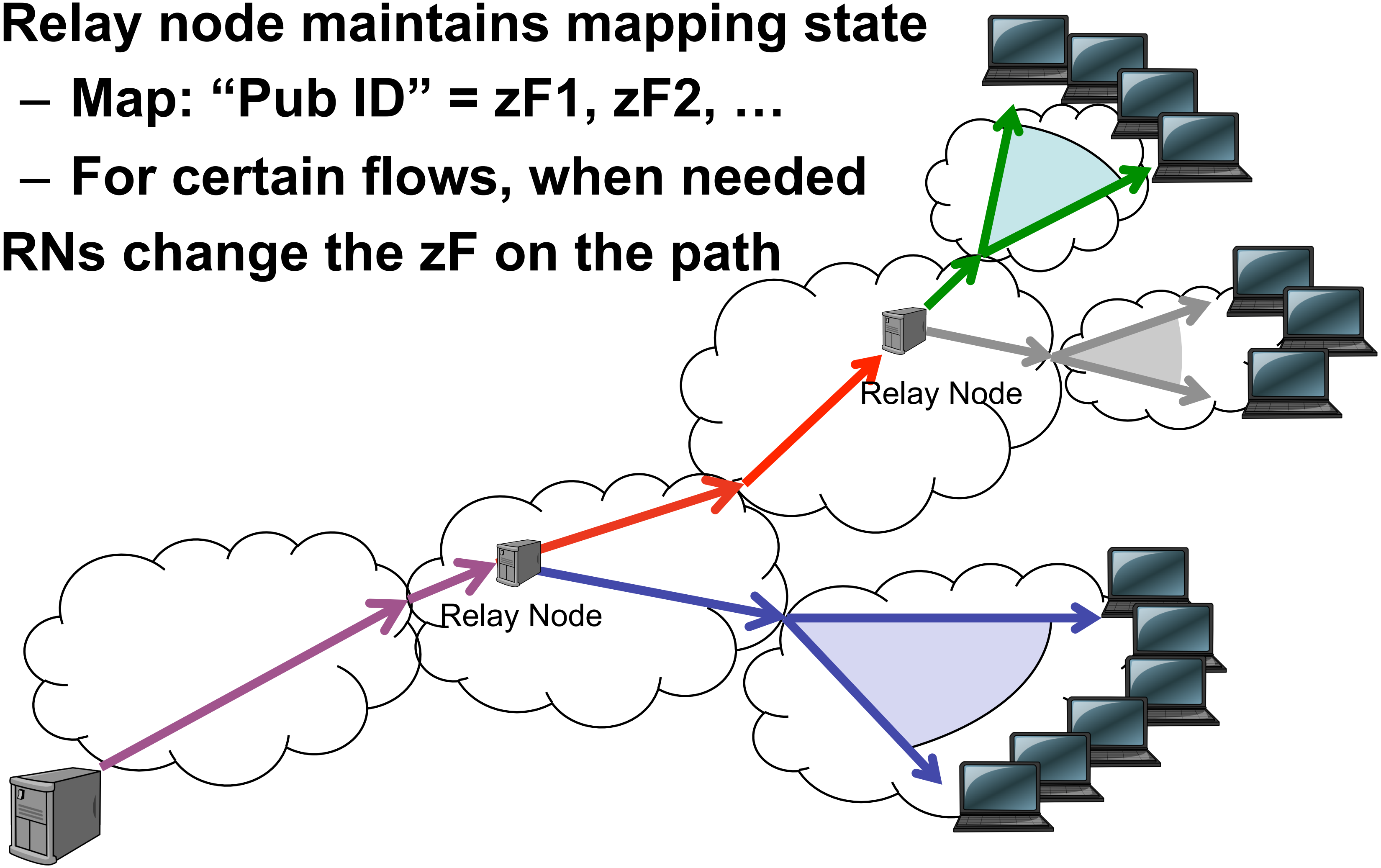
Scalability issues

- **Inter-domain forwarding**
  - **Too many LIDs in a single BF results in too many false positives**



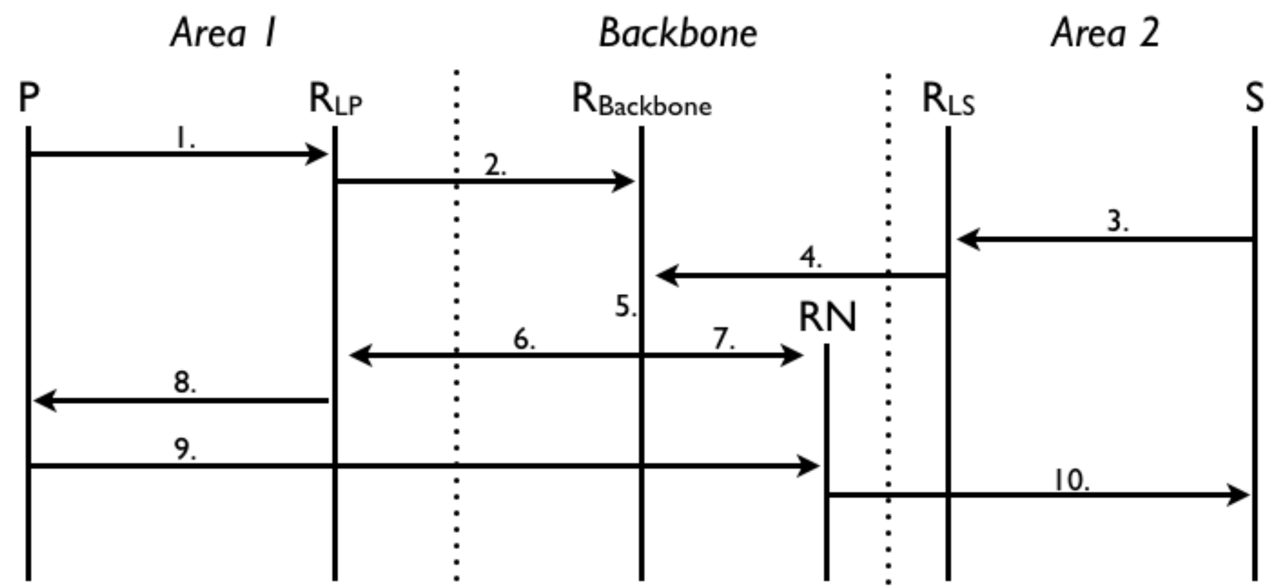
Scalability: Relay Nodes

- **Relay node maintains mapping state**
  - **Map: "Pub ID" = zF1, zF2, …**
  - **For certain flows, when needed**
- **RNs change the zF on the path**



Pursuing a Pub/Sub Internet PURSUIT

Setting up Relay Nodes

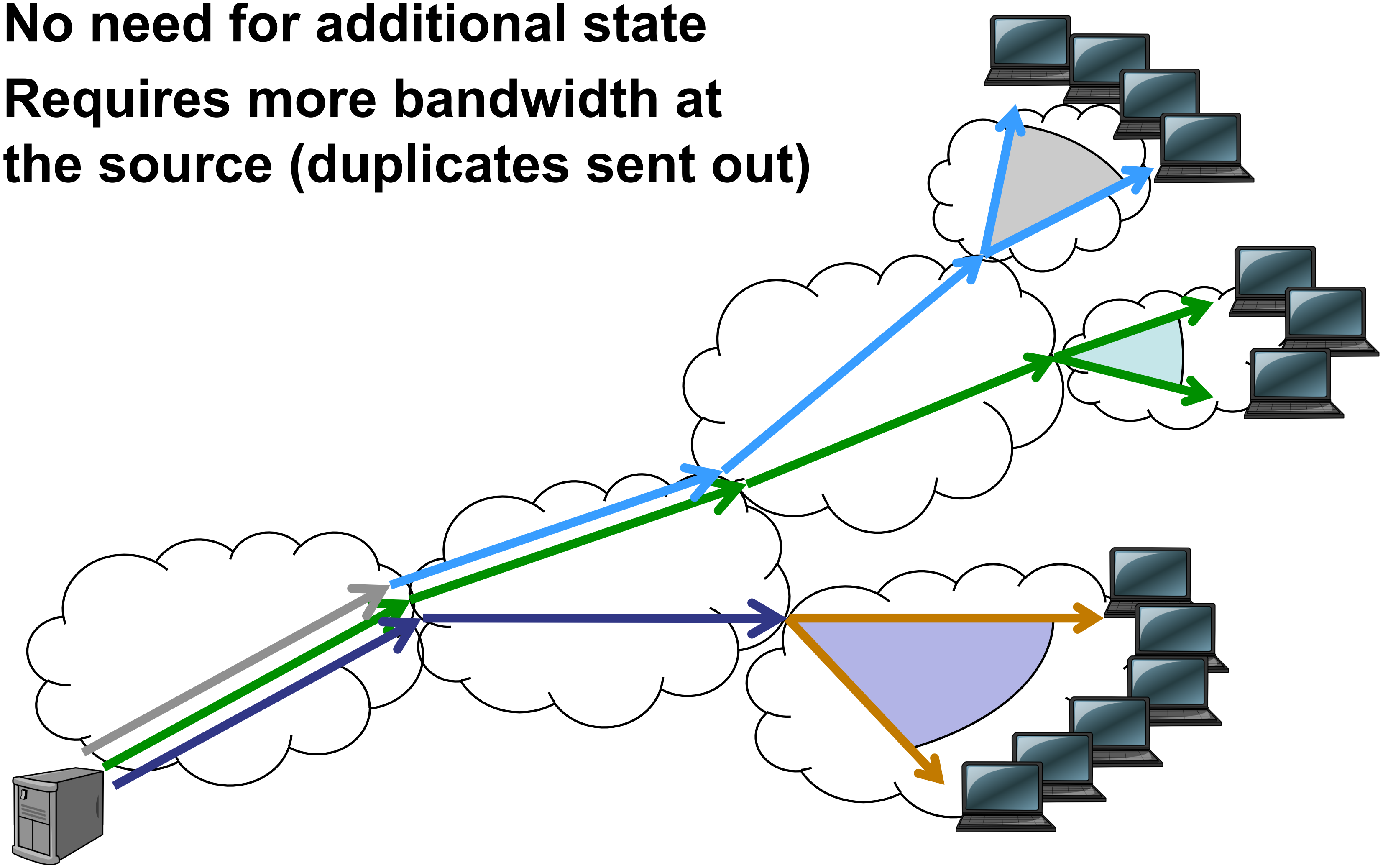


1. & 2. Publish(Pub ID)
3. Subscribe(Pub ID)
4. Subscribe(Pub ID, BF_{area2}, NofLinks, NofSubscribers)
5. Decision, create a new Relay point
6. Subscribe(Pub ID, $BF_{area1\text{-}to\text{-}RN}$, NofLinks, NofSubscribers)
7. CreateRN(Pub ID, $BF_{area2+RN\text{-}to\text{-}area2}$)
8. Subscribe(Pub ID, $BF_{area1+area1\text{-}to\text{-}RN}$)
9. Deliver Data (Pub ID) to $BF_{area1+area1\text{-}to\text{-}RN}$
10. Deliver Data (Pub ID) to $BF_{area2+RN\text{-}to\text{-}area2}$

Pursuing a Pub/Sub Internet

Scalability: Splitting the tree

- No need for additional state
- Requires more bandwidth at the source (duplicates sent out)

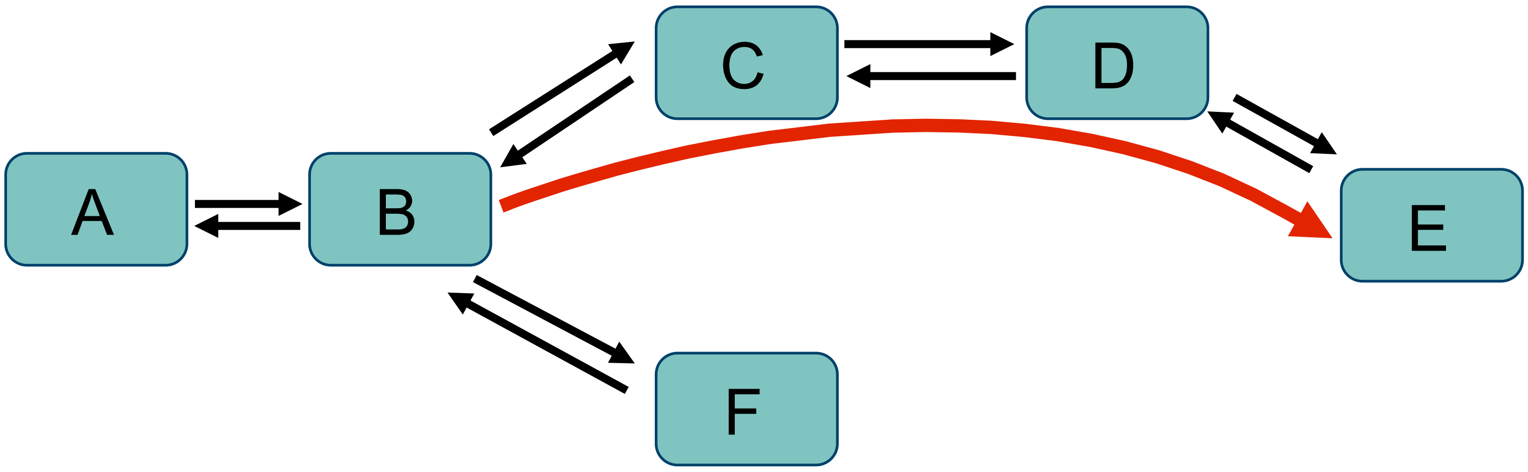


Scalability – stacking Bloom filters

- TM divides delivery tree into multiple parts along the paths
- Each part has its own BF
- These BFs are stacked into a packet, removed at boundaries
- BFs are variable size, chosen so that the probability of false positives is minimized

Scalability: Virtual trees

- Popular paths can be merged into virtual trees
  - A single Link ID for the tree
  - Additional state in the forwarding nodes
  - Increase scalability
- A virtual tree is not bound to a certain publication
  - E.g. a single tree for all AS transit traffic



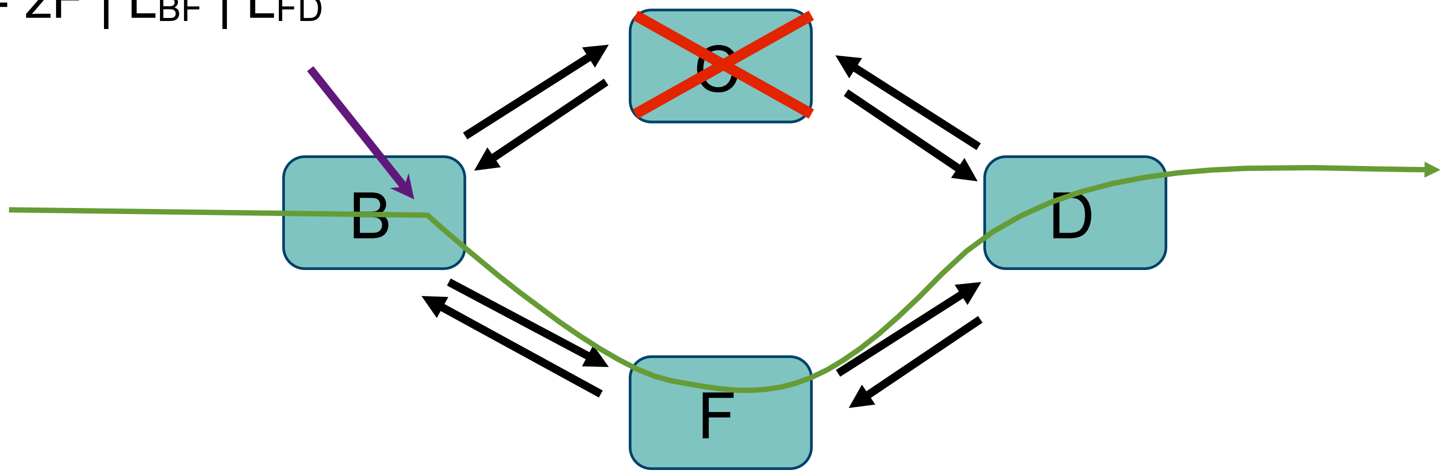
Virtual B->C->D->E   0 0 1 0 1 0 0 0 1

Failover enhancements

Fast reroute – Backup path

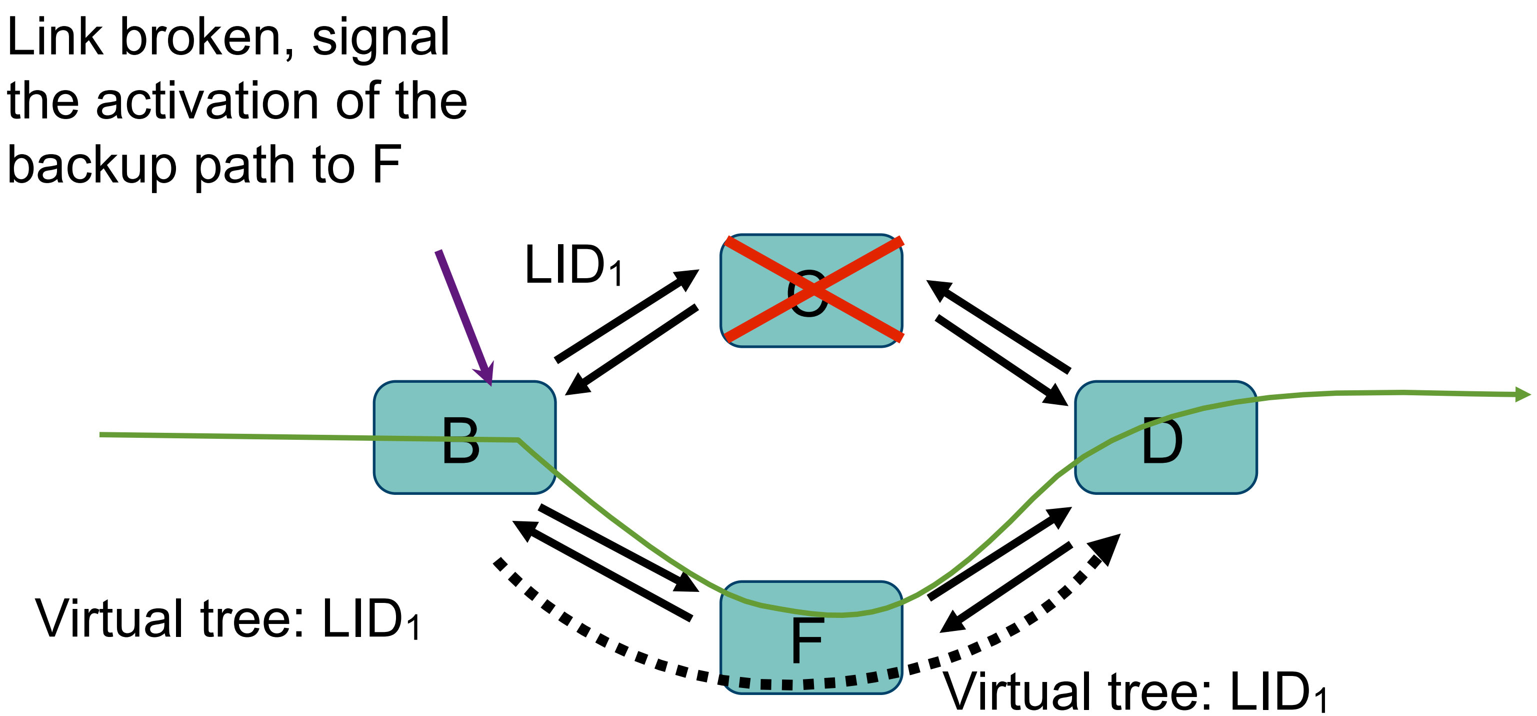
- **Node B maintains backup path information**
- **In case of broken link, add backup path**
  - **Increases temporarily the false positive probability until a new path is calculated at the topology manager**
  - **No additional signaling**

Add backup path:
$zF = zF \mid L_{BF} \mid L_{FD}$



Fast reroute – Virtual trees

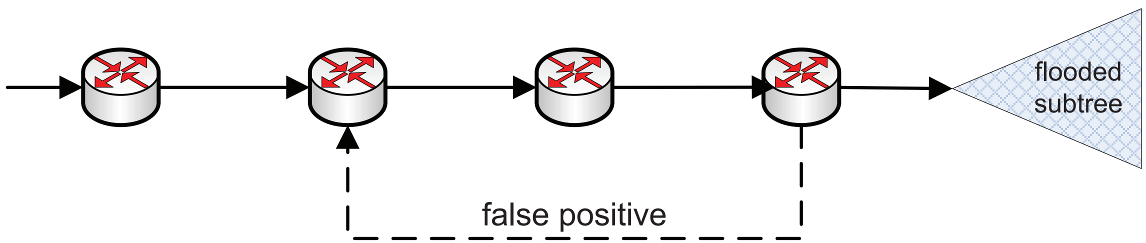
- **zFilter unmodified**
- **Activate backup path in case of node failure**
  - **Adds signaling**

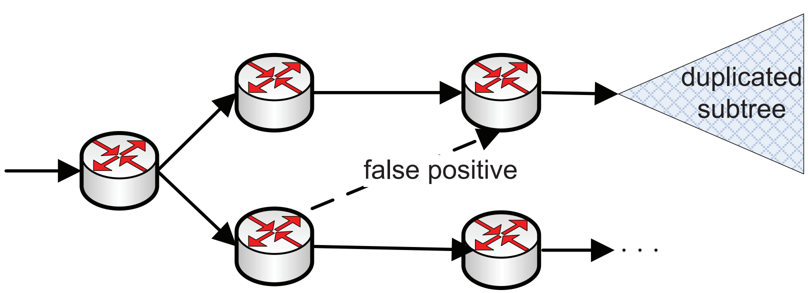


Loop prevention enhancements

Forwarding anomalies

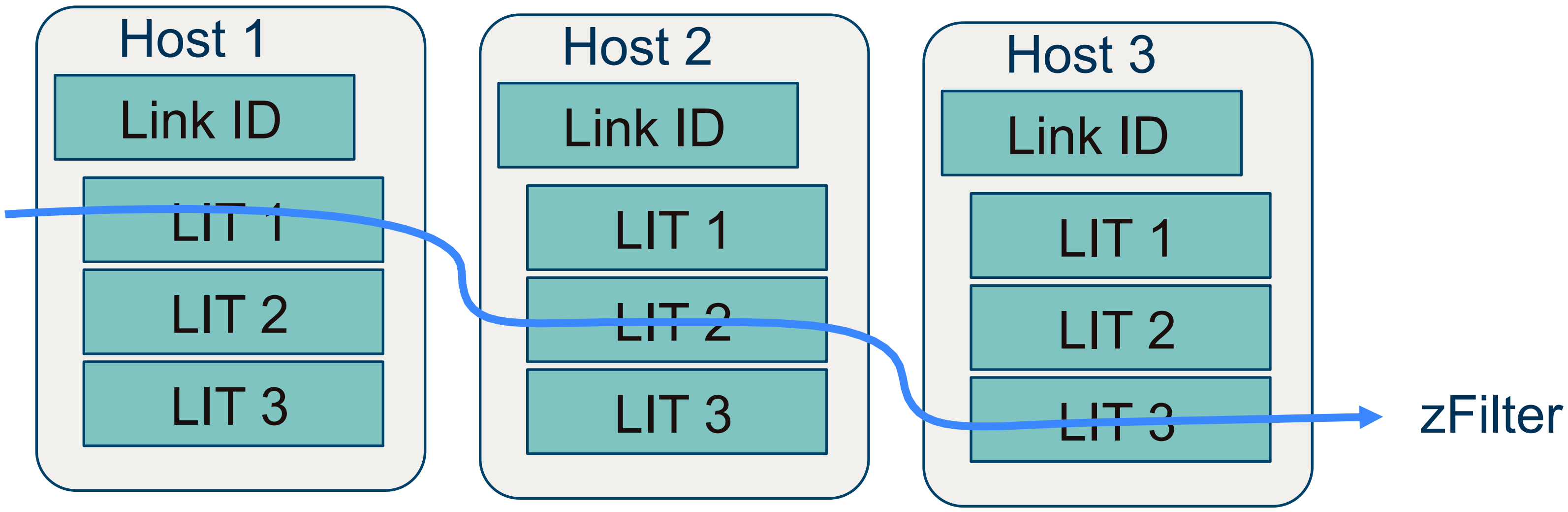
- **E.g. packet storms, forwarding loops, and flow duplication**
- **Accidental or maliciously created**





Avoiding loops

- Instead of fixed d determining the used LIT, change the d e.g. with d=(d+1) MOD e
- In case of a loop, the packet will have the same d only if the loop is e hops long
- Simple, stateless solution

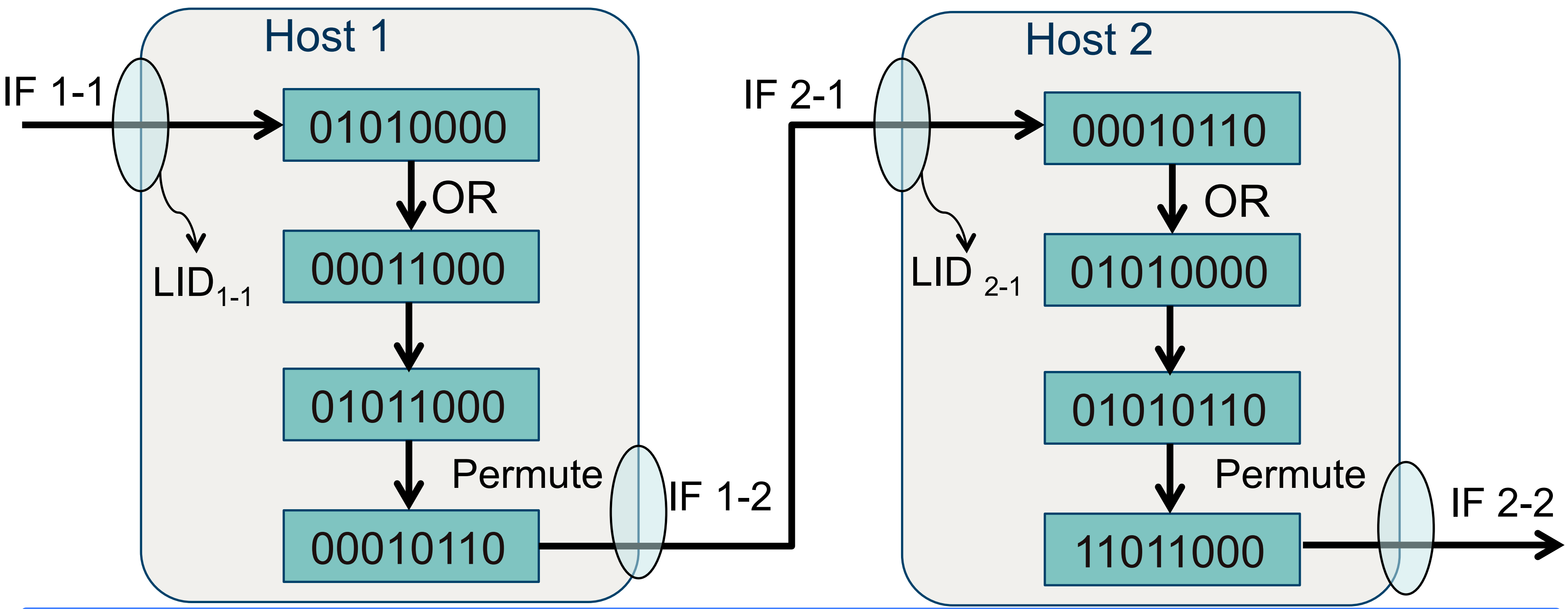


Permutations

- Goal: Prevent forwarding loops and flow duplication
- Idea: Make forwarding decision depend on the packet's path and hop-count
- Solution: Per-hop bit permutation
  - "Mix" the BF bits in incoming packets according to a function specific to the incoming interface
  - Simple to implement, no additional space in the packet, randomizes the BF in case of false positives
  - Requirements
    - Reversible operation, no significant increase in number of 1-bits
- Multicast zFilters
  - ORing is not enough, must be computed from the leaves of the tree

Forming the permuted zFilter

- **Especially suitable when zF is collected on reverse path**
- **zFilter verification to the other direction**
  - **Permute with the function**
  - **Match outgoing interfaces**



Pursuing a Pub/Sub Internet PURSUIT

Security enhancements

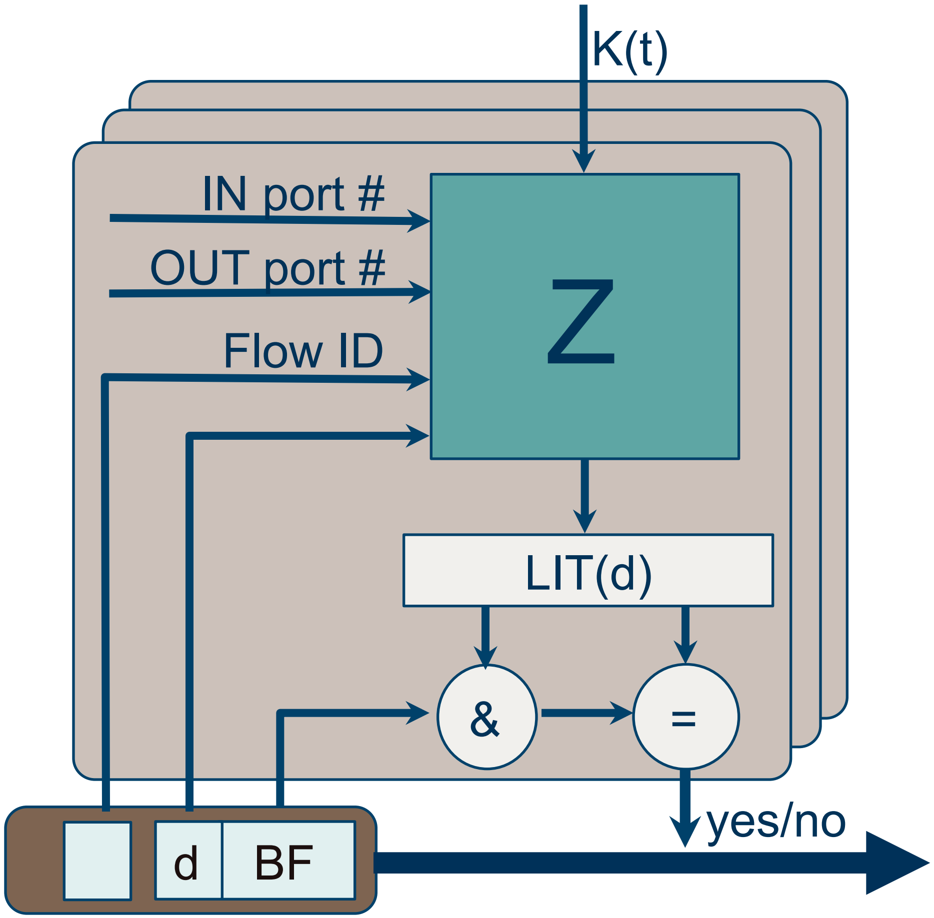
Security weaknesses with static LID/LITs

- zFilter replay attacks
  - Sending data with the same zFilter
- Traffic injection attack
  - Using existing zFilter, send data from the middle of the path
- Computational attack
  - Collect zFilters from packets
  - Correlate zFilters to learn link IDs

Secure forwarding

- Goal: to ensure (probabilistically) that hosts cannot send un-authorized traffic
- Solution (z-Formation): Compute LIT in line speed and bind it to
  - path: in-coming and out-going port
  - time: periodically changing keys
  - flow: flow identifier (e.g. content ID)

Secure case: z-Formation



- 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

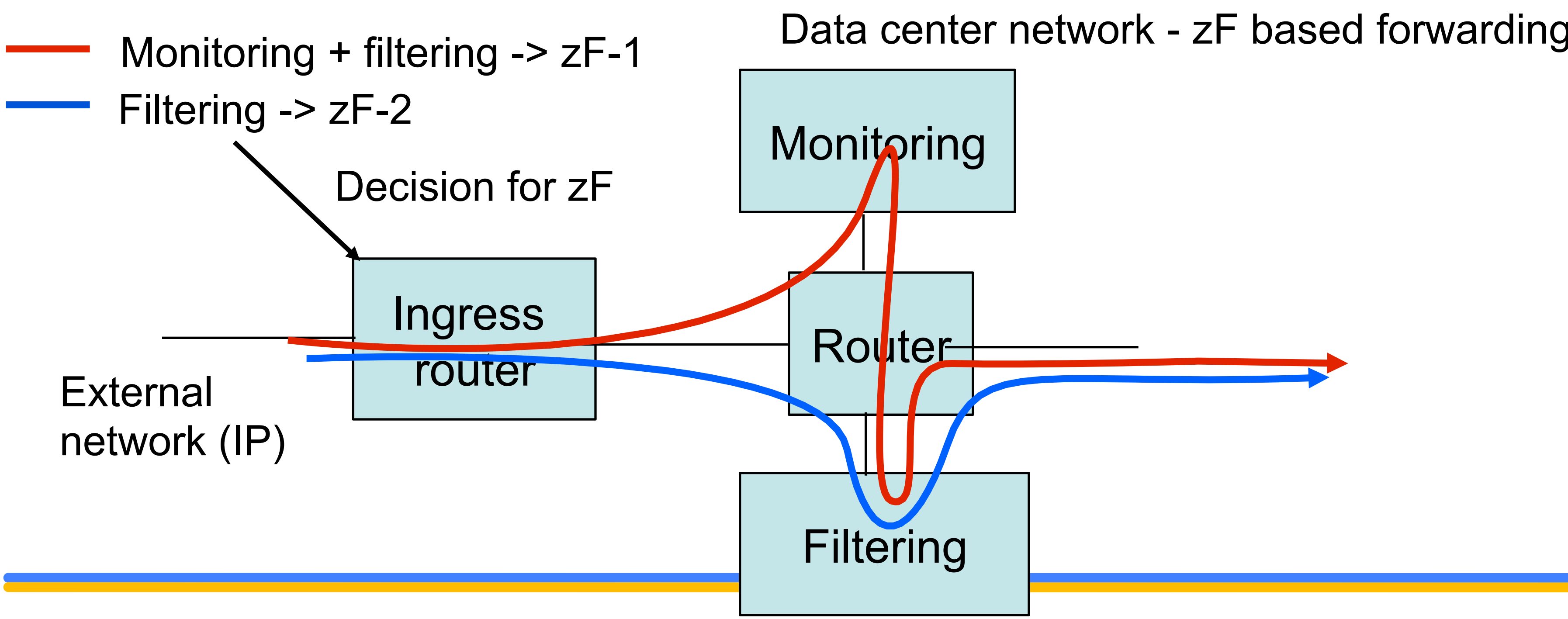
Security properties

- Binding a zFilter only to the outgoing port
  - Traffic injection possible
  - Correlation attacks possible
- Bind 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

Applications

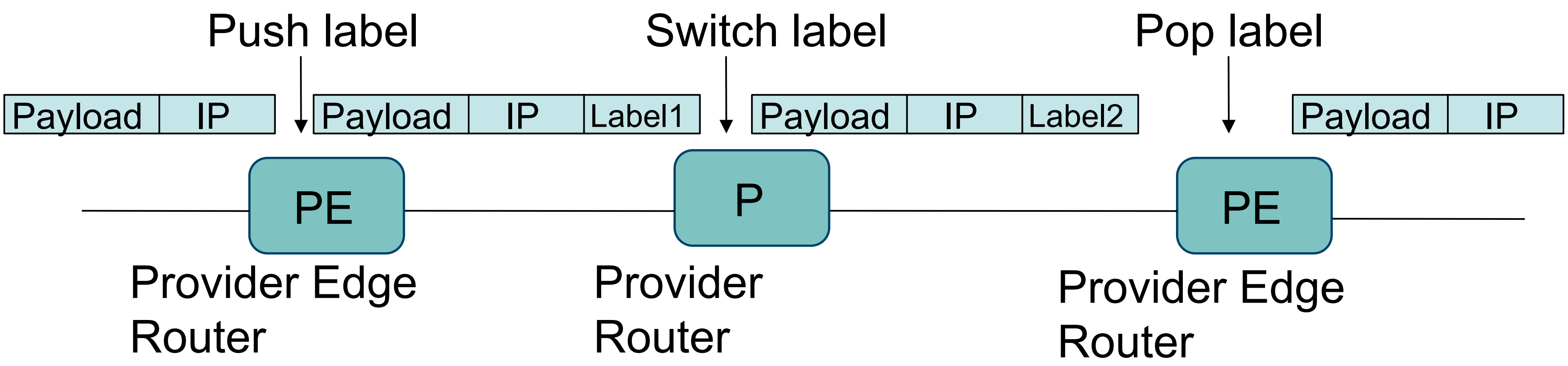
Data centers

- **zFilters only in the internal network**
- **Easier to modify the routing in the network**
  - **E.g. route packets via certain services: Load balancing, monitoring...**
  - **Binding the flow to input and output ports allows flexible path control at the ingress point**



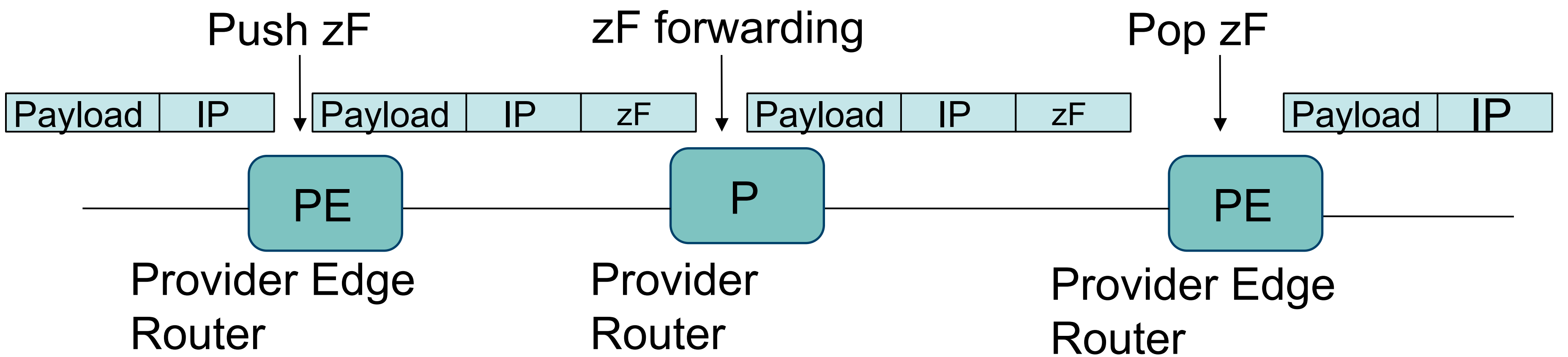
Background: (G)MPLS
Multiprotocol label switching

- **Evolution: MPLS->MPLS-TE -> GMPLS**
- **(G)MPLS is a rich set of protocols**
  - **Setting up Label Switched Paths**
  - **Forwarding on the Label Switched Paths**
  - **Traffic Engineering, resiliency (e.g. fast reroute)**
  - **Enabler of VPN services**
  - **Control plane for many different technologies**



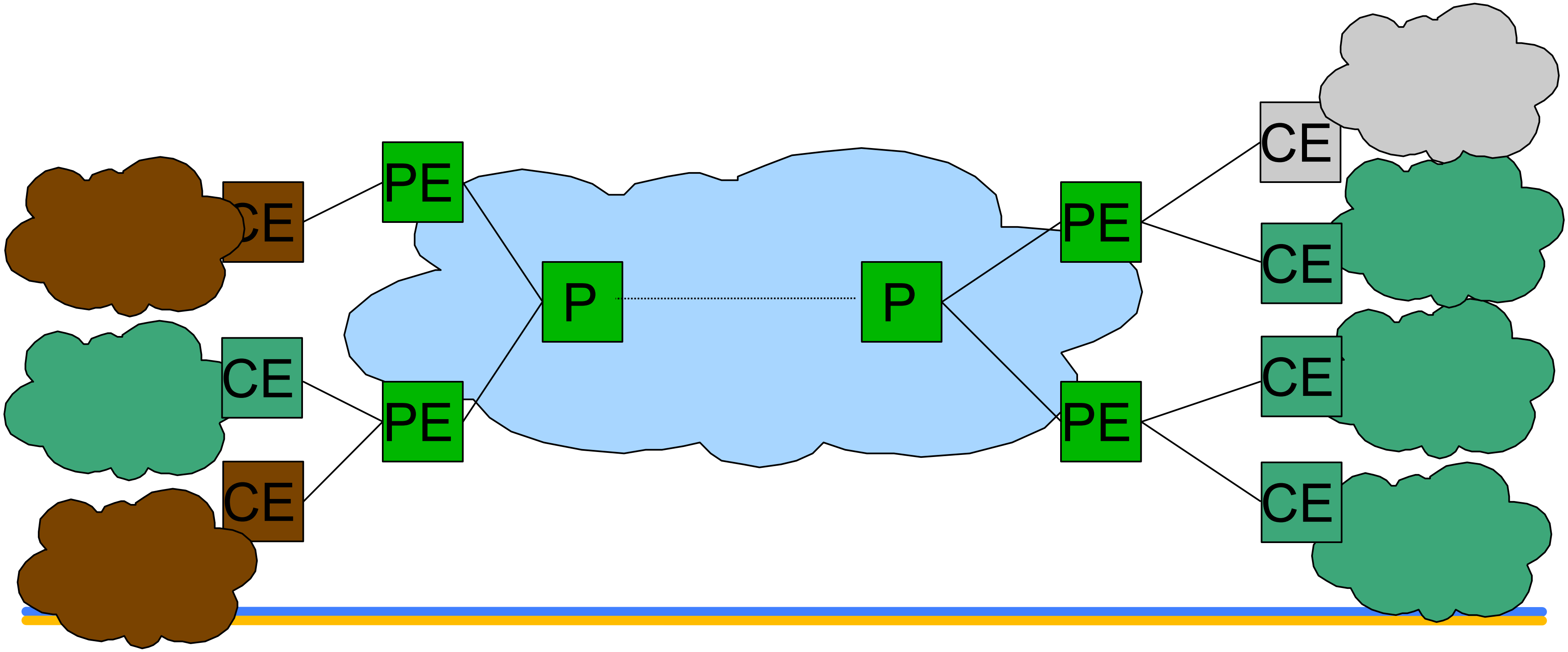
MPSS
Multiprotocol stateless switching

- **Advantages over label switching**
  - **There is not necessarily need for signaling**
  - **In simpler case, no state required**
  - **Multicast support (setup, maintenance) much simpler than with (G)MPLS**



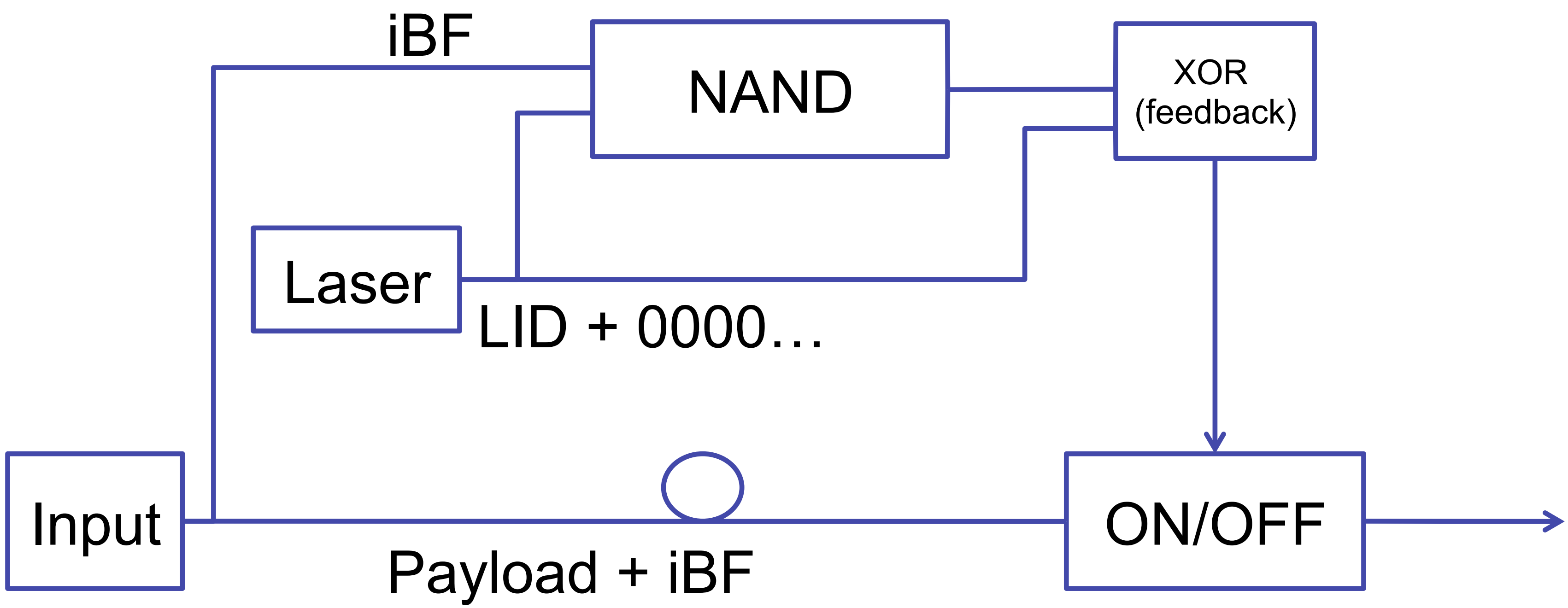
Multicast VPN with MPSS

- Effective support of point-to-multipoint communication
- The bandwidth efficiency vs. Multicast state trade-off eliminated
  - (Though longer header sizes)
- With zFilters: no multicast states, and acceptable bandwidth efficiency up to ~20 PEs



Optical packet forwarding

- **All-optical packet forwarding**
  - **Matching LID from laser with the iBF**
  - **XOR with feedback goes to zero if one bit fails (= no forwarding)**



Summary

- **New multicast forwarding mechanism**
  - Suits pub/sub and synchronous multicast very well
  - Can also be applied outside our pub/sub model
  - Almost stateless
  - Good security properties
- **But: Some scalability issues – especially due to false positives**
  - And also some security issues
- **Many enhancements/changes/additions to the basic LIPSIN mechanism have been proposed**
  - Tradeoffs
- **E.g., work on inter-domain forwarding ongoing**

Pursuing a Pub/Sub Internet PURSUIT

Some references

- Petri Jokela, András Zahemszky, Christian Esteve, Somaya Arianfar, and Pekka Nikander, "*LIPSIN: Line speed Publish/ Subscribe Inter-Networking*", ACM SIGCOMM 2009
- András Zahemszky and Somaya Arianfar, "*Fast reroute for stateless multicast*", IEEE RNDM 2009
- Christian Esteve et al., "*Self-routing Denial-of-Service Resistant Capabilities using In-packet Bloom Filters*", EC2ND, 2009
- Christian Esteve et al., "*Data center networking with in-packet Bloom filters*", SBRC, 2010
- András Zahemszky et al. "*MPSS: Multiprotocol Stateless Switching*", IEEE Global Internet Symposium 2010
- Mikko Särelä et al., "*Forwarding Anomalies in Bloom Filter Based Multicast*", IEEE INFOCOM 2010
- Dirk Trossen et al., "*PURSUIT Deliverable D2.2: Conceptual Architecture: Principles, patterns and sub-components descriptions*", Section 4.3: Forwarding, 2011
- Sajjad Rizvi, "*Performance analysis of bloom filter-based multicast*", Master's thesis, Aalto university, 2011