SECURE IN-PACKET BLOOM FILTER BASED FORWARDING NODE ON A NETFPGA

1ST EUROPEAN NETFPGA DEVELOPERS WORKSHOP

SEP 9-10TH, 2010

UNIVERSITY OF CAMBRIDGE, UK

ADNAN GHANI AND PEKKA NIKANDER

PRESENTATION OUTLINE

- Background
  - In-packet Bloom filter (iBF) based forwarding
  - Link IDs and Bloom Filters
  - Forwarding decision
  - Using Link Identity Tags (LITs)
  - False positives and forwarding efficiency
  - Algorithmic view
- Computational iBFs
  - Split key management
  - Flow diagrams
  - Implementation details
  - Latency measurements

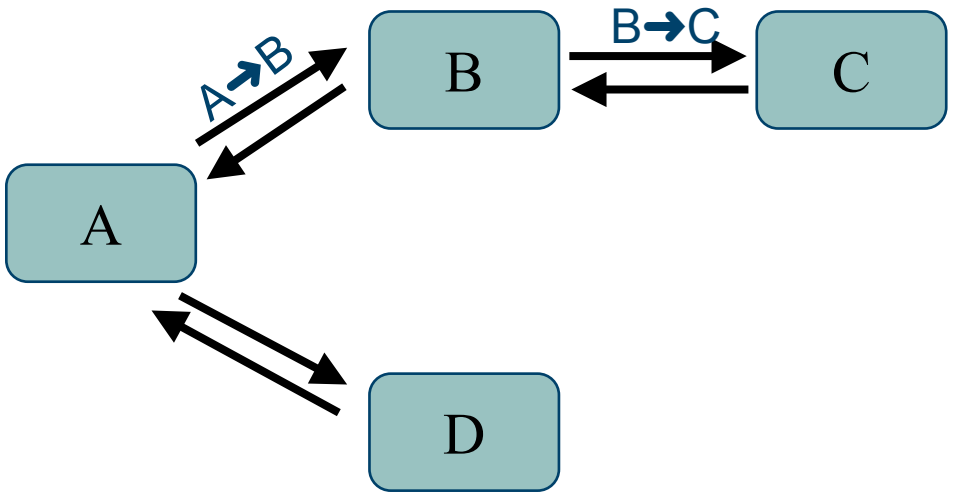
IBF-BASED FORWARDING

- Give names to links, not to nodes
- Form a source-route using the links 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
  - Enhancing with computational link identifiers
  - Virtual trees

LINK IDS AND BLOOM FILTERS

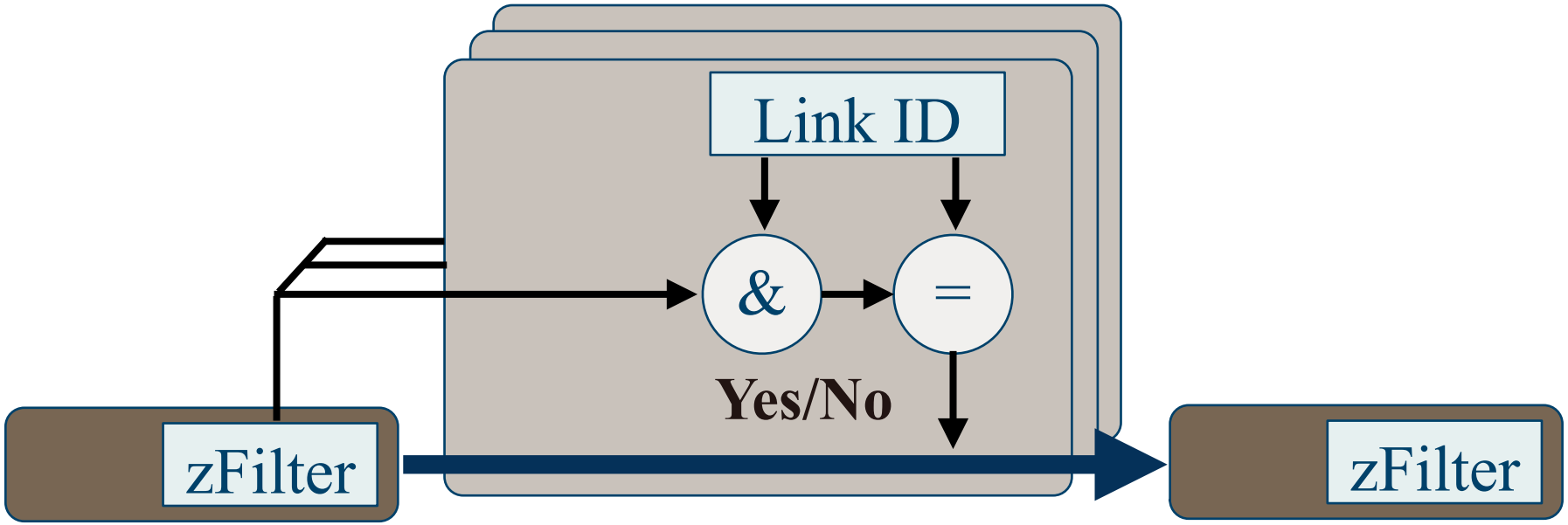
- No names for nodes
  - Each link identified with a unidirectional Link ID
- Link IDs (Bloom masks)
  - Statistically unique
  - Periodically changing
  - Size e.g. 256 bits
  - Local or centrally controlled
- Source routing
  - Encode Link IDs into a Bloom filter (zFilter)
  - Naturally multicast
-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

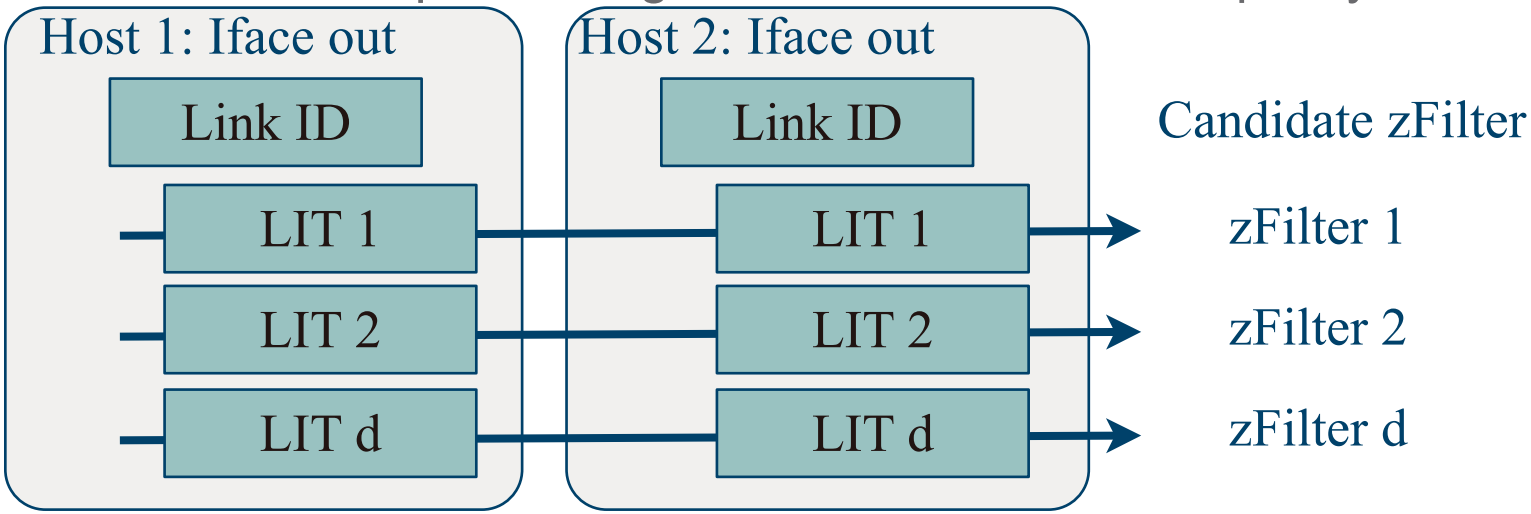
FORWARDING DECISION

- 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

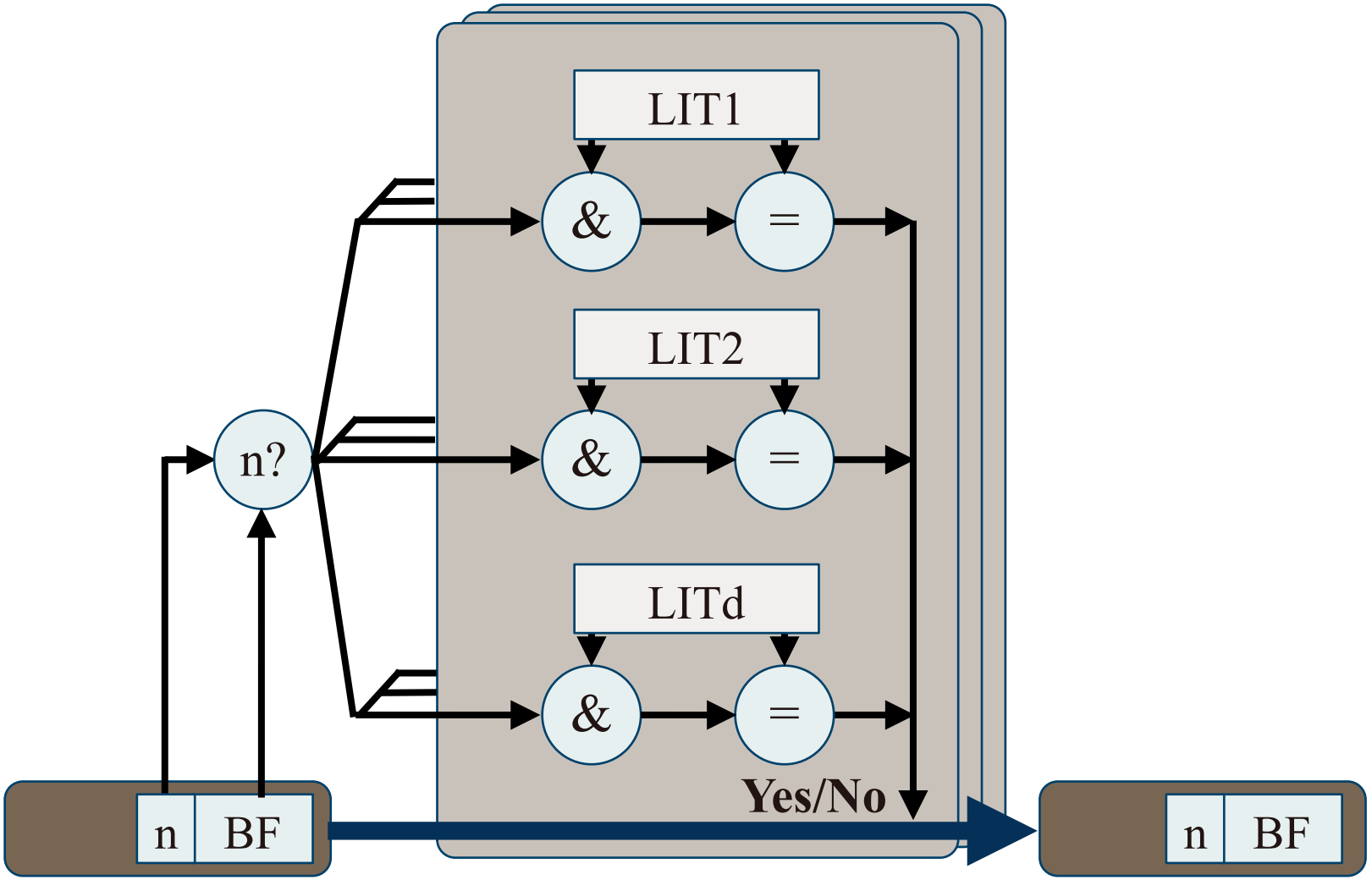


USING LINK IDENTITY TAGS (LIT)

- Better forwarding efficiency with a simple trick
  - Define *d* 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 *d* different candidate zFilters
  - Select the best performing zFilter, based on some policy

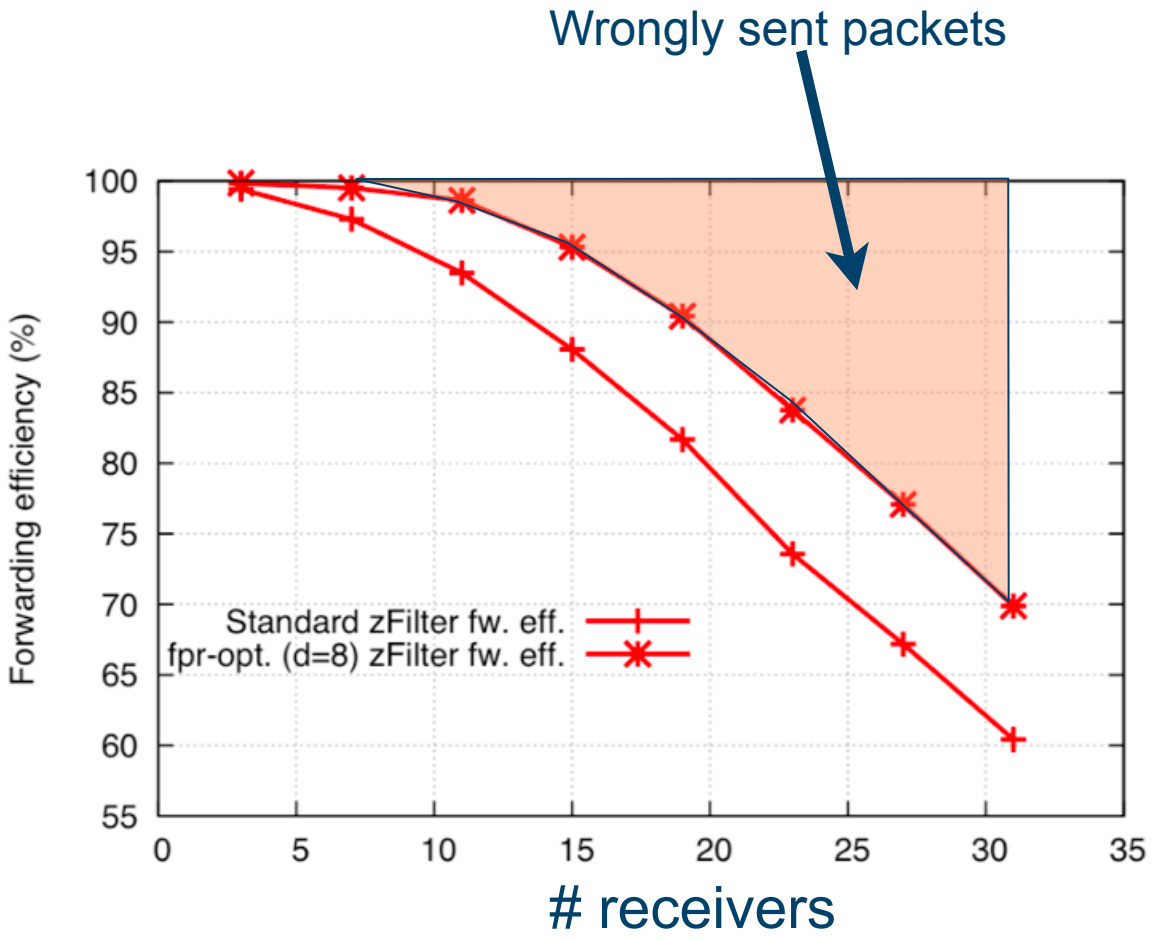


USING LINK IDENTITY TAGS (LIT)



FORWARDING EFFICIENCY

- Simulations with
  - Rocketfuel
  - SNDlib
- Forwarding efficiency
- 20 receivers
  - Basic LID: 80%
  - Optimised: 88%
    - with 8 LITs



ALGORITHMIC VIEW

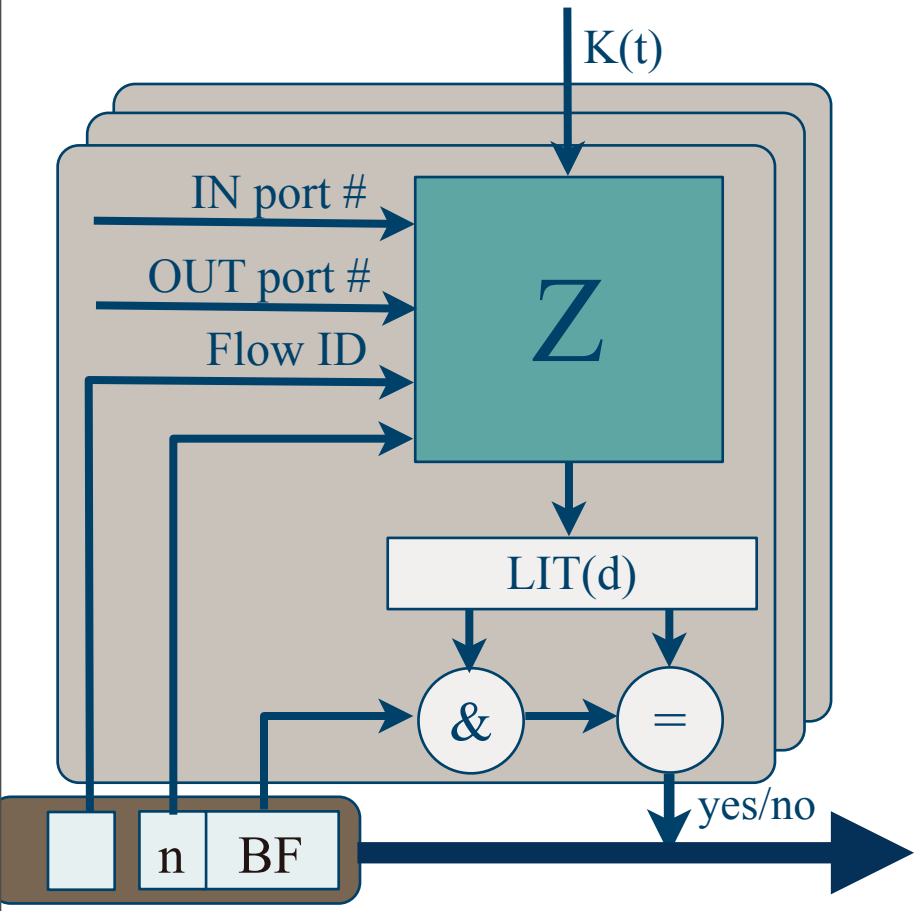
› Forwarding based on following algorithm

```
Input: LITs of the outgoing links;
       zFilter in the packet header
foreach LIT of outgoing interface do
            if (zFilter & LIT) = LIT then
                        Forward packet on the link
            end
end
```

› Security problem: An attacker may try to determine bits set to one in forwarding identifier.

› Solution: Computational Bloom masks

SECURE CASE: COMPUTATIONAL IBFS



- Form LITs algorithmically
  - at packet handling time
- Secure periodic key *K*
- Input port index
- Output port index

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

COMPUTATIONAL IBFS

- O = Z(K, M, I)

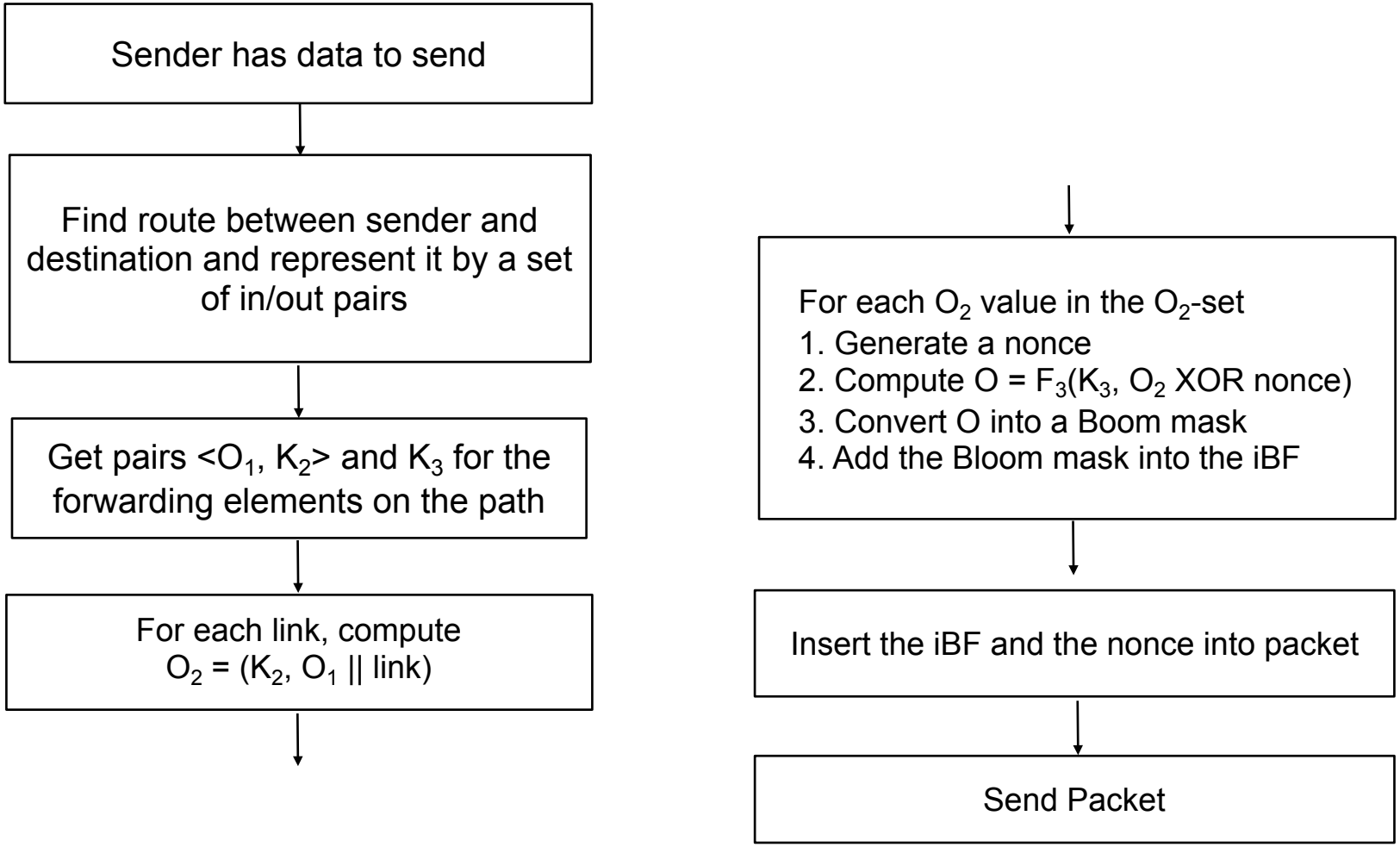
- K = semi static secret key
  - varies every few minutes or hours or days
- M = medium dynamic data
  - e.g. captures a session, link indices, etc
- I = dynamic, i.e. varies per packet

- The key is split into three parts:

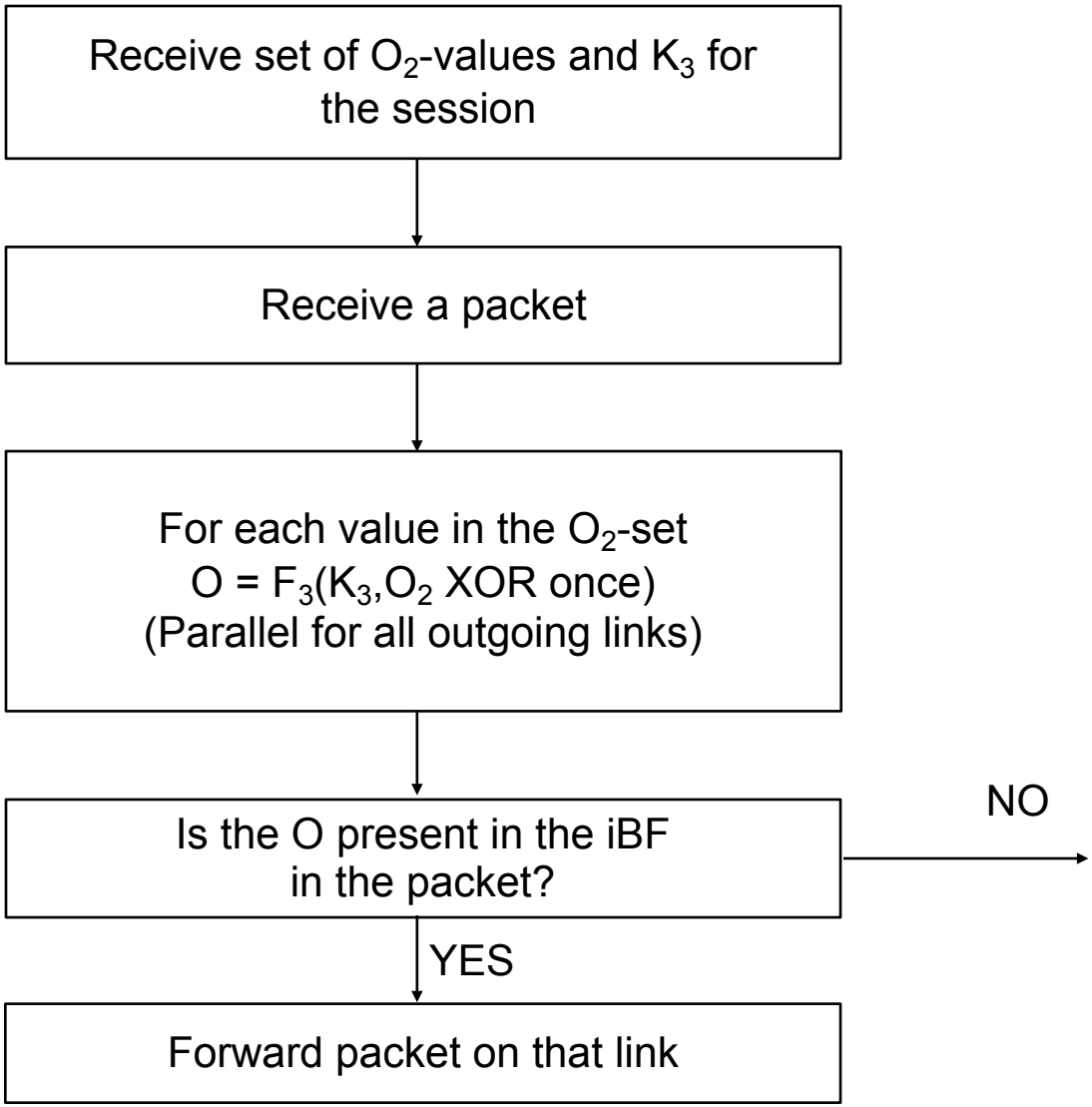
  $K_1 = KDF(K, "1"); K_2 = KDF(K, "2"); K_3 = KDF(K, "3");$

- $O_1 =$ $F_1(K_1, <\text{other semi static inputs}>)$
- $O_2 =$ $F_2(K_2, O_1 \,||\, M)$
- $O = O_3 =$ $F_3(K_3, O_2 \,||\, I)$

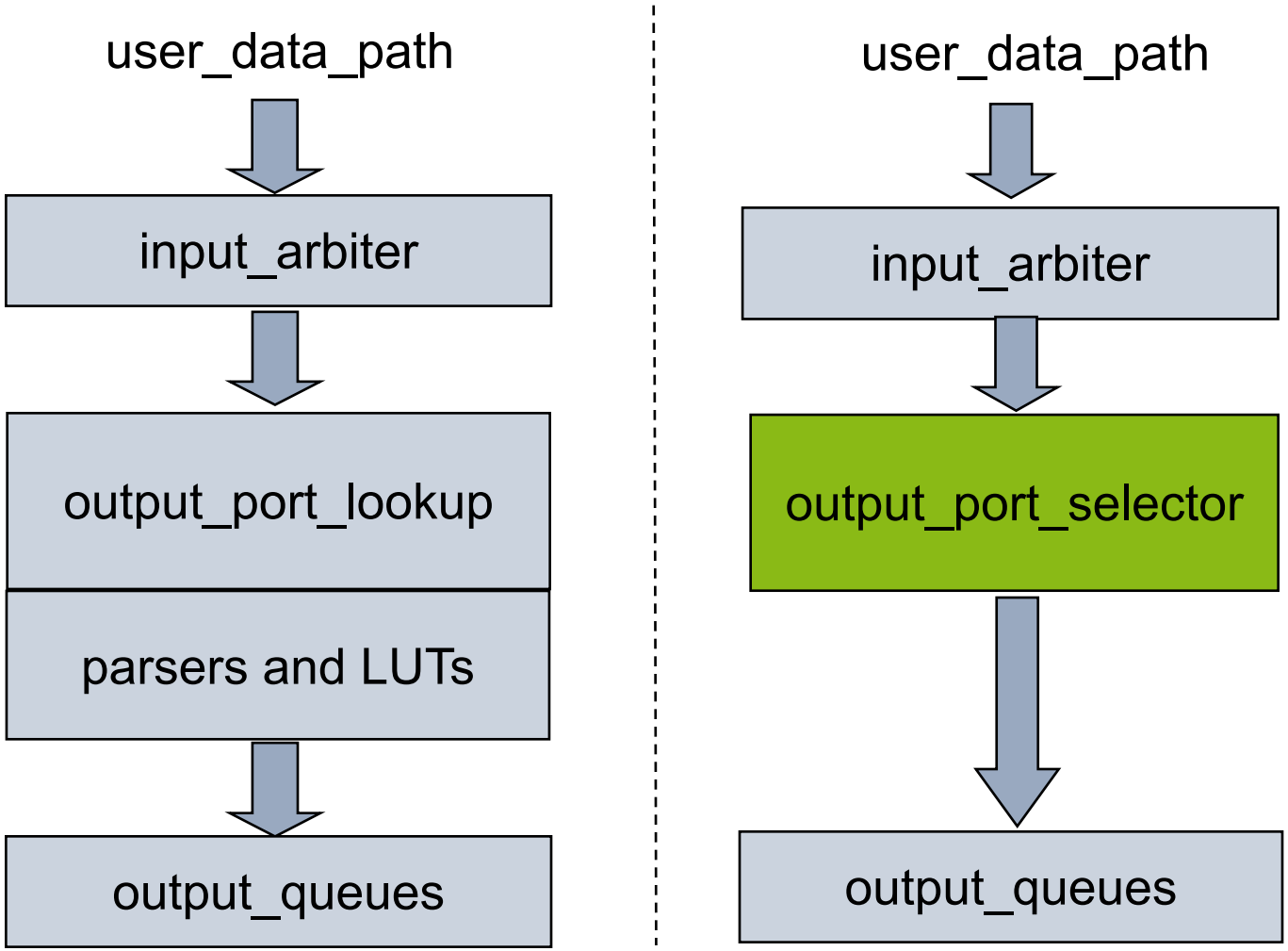
SENDER OPERATIONS (AS INFO)



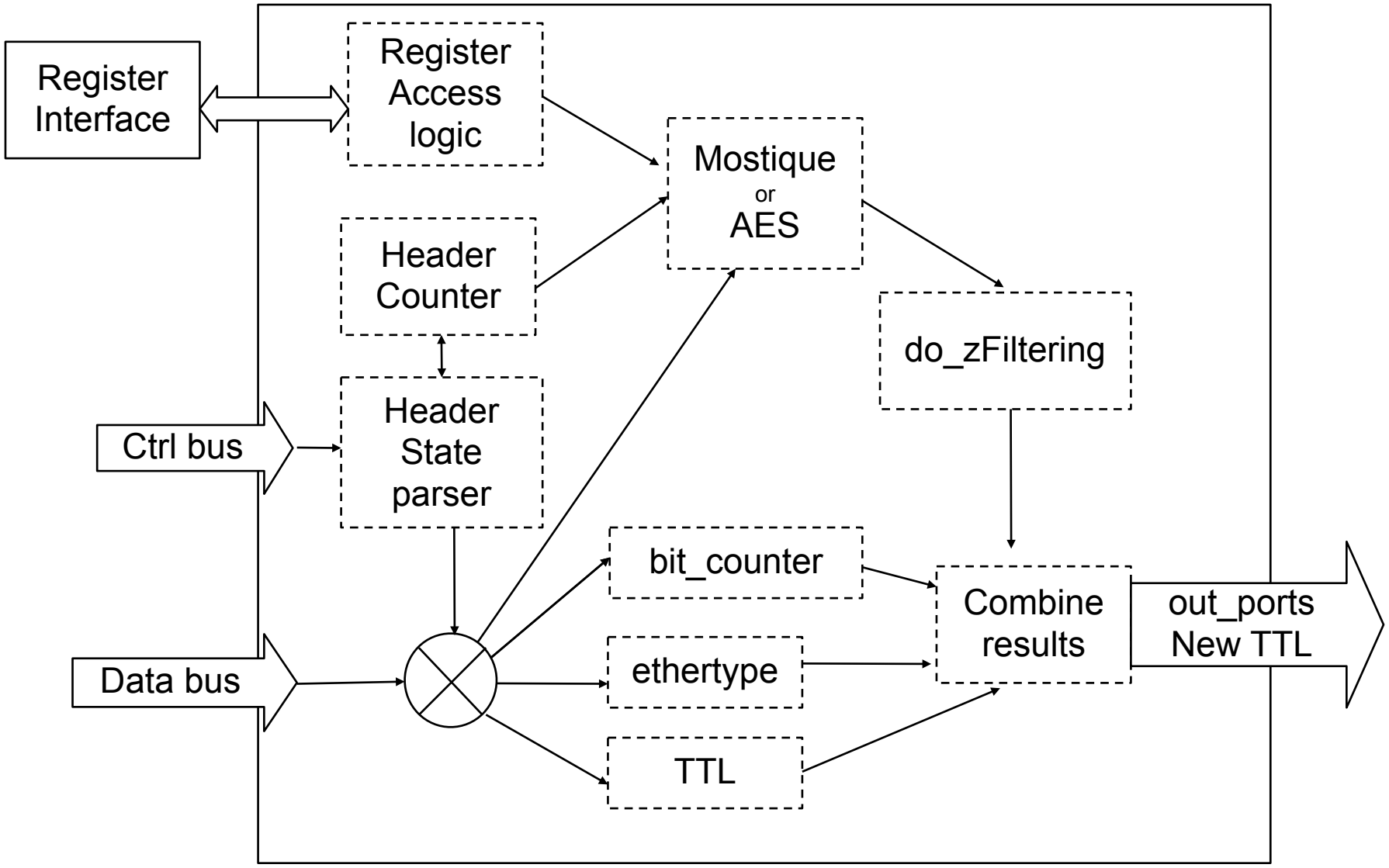
FORWARDING NODE OPERATION



REFERENCE DATAPATH AND MODIFIED DATAPATHS



OUTPUT_PORT_SELECTOR MODULE STRUCTURE



LATENCY MEASUREMENT RESULTS

Path and packet format	Average Latency	Standard Deviation
Wire (New)	12,784ns	4,448.96ns
NetFPGA with Moustique (New)	15,272ns	4991.28ns
NetFPGA with AES (New)	15,057ns	3,756.86ns
Wire (old)	12,549ns	4,867.34ns
NetFPGA with LIPSIN (old)	14,627ns	4,204.58ns