XORs in the air: Practical Wireless Network Coding

Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Medard, Jon Crowcroft

SIGCOMM '06

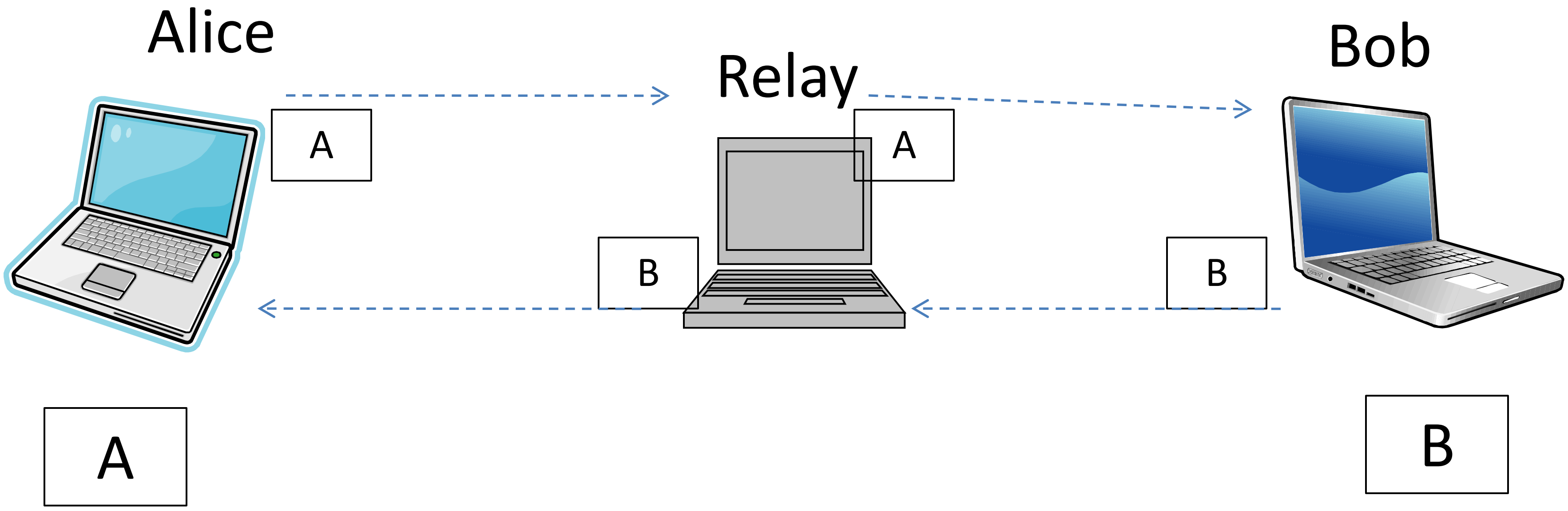
Presented by
Thangam Seenivasan

Problem

Increase the throughput of dense wireless networks

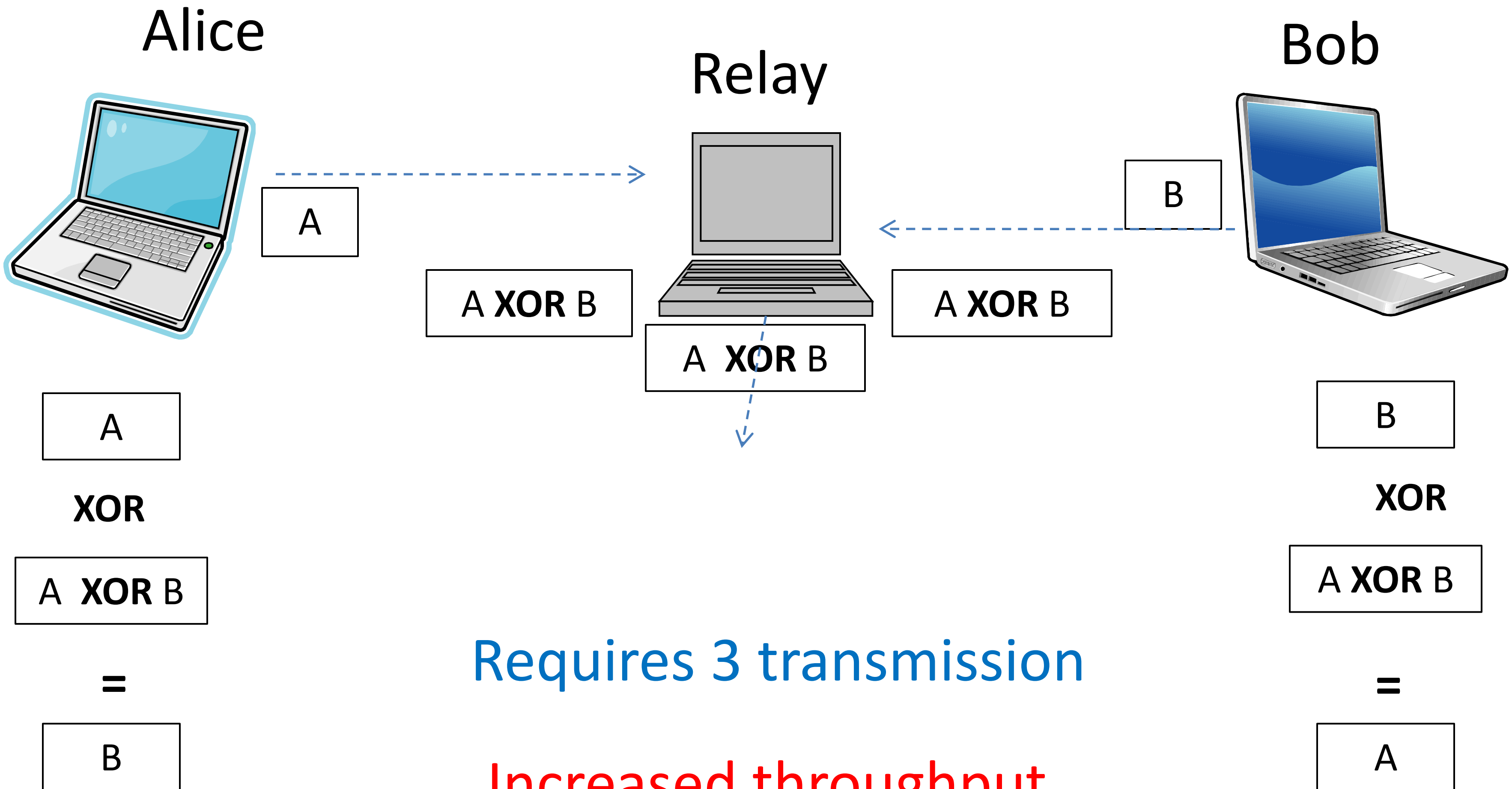
Network Coding

Current Approach



Requires 4 transmission

COPE Approach



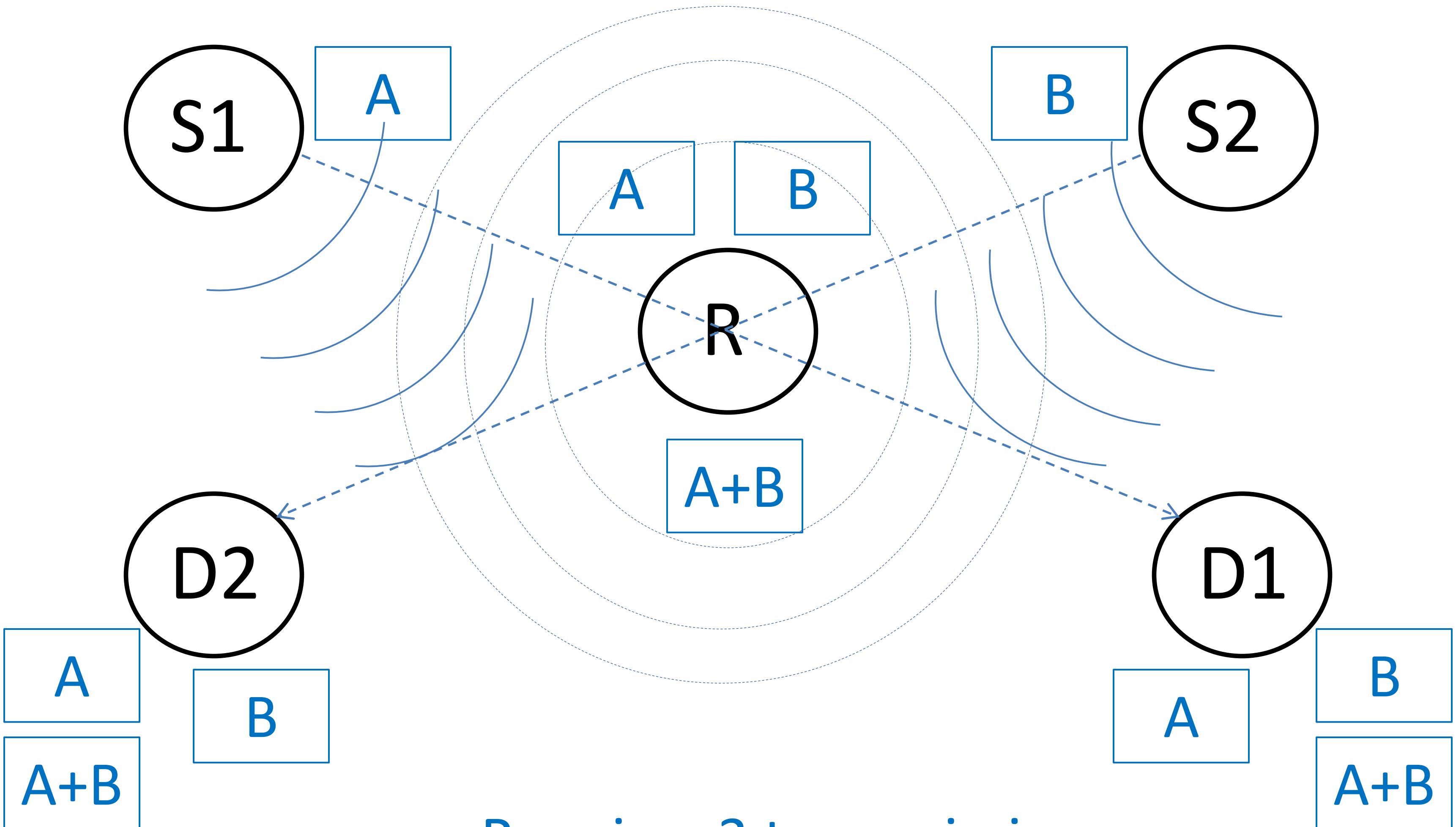
Requires 3 transmission

Increased throughput

COPE Approach

- Exploits shared nature of wireless medium
  - Every node snoops on all packets
  - A node stores all heard packets for a limited time
- Tell neighbors which packets it has heard
- Perform opportunistic coding
  - XOR multiple packets and transmit them as single packet
- Decode the encoded packet using stored packets

Scenario



Requires 3 transmission

Outline

- Design
- Cope Gains
- Making it work
- Implementation details
- Experimental results

Overview

- Opportunistic listening
- Opportunistic coding
- Learning neighbor state

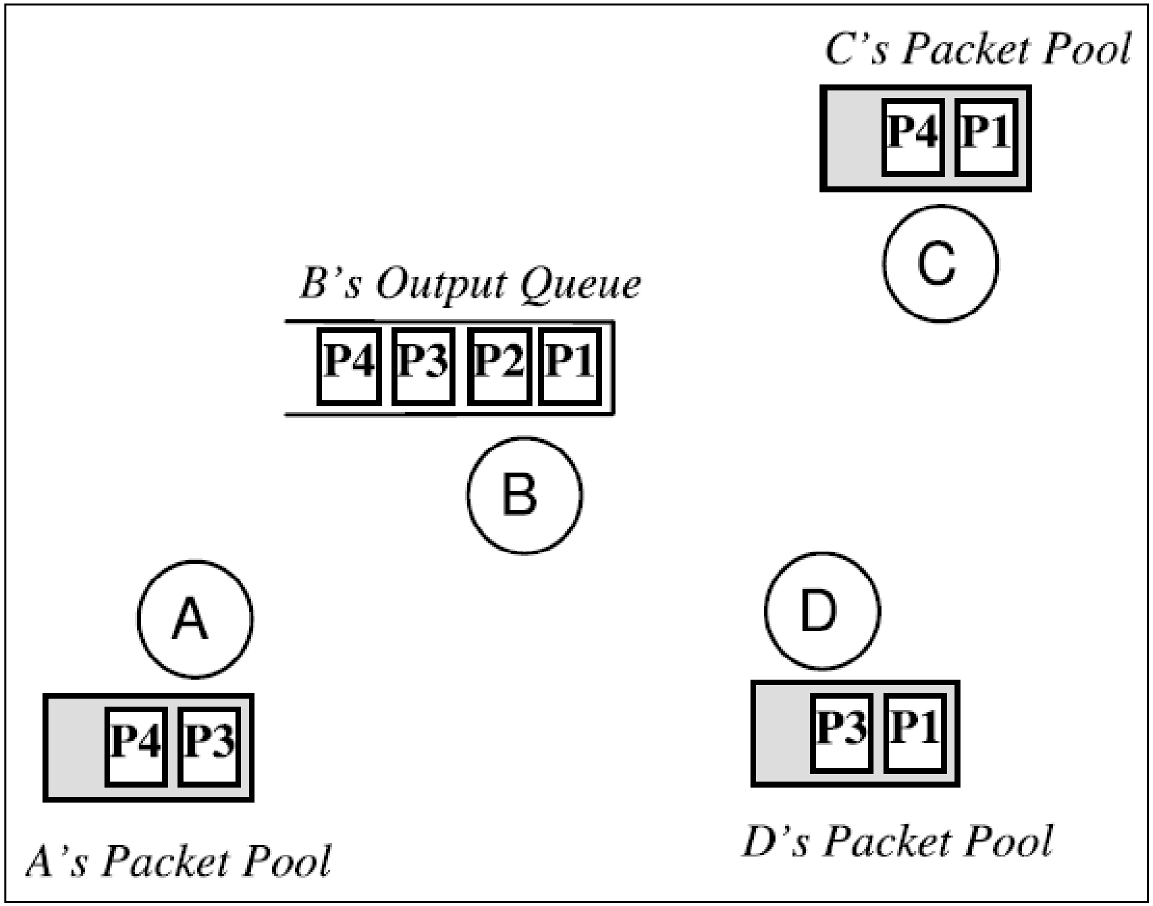
Opportunistic listening

- Exploit broadcast nature of wireless
  - Set nodes in promiscuous mode
  - Opportunities to overhear packets
- Store the overheard packets
  - Limited time period (T = 0.5s)
- Broadcast reception reports to tell neighbors which packets it has stored
  - Annotate with data packets
  - If no data packets, send reception reports periodically

Opportunistic coding

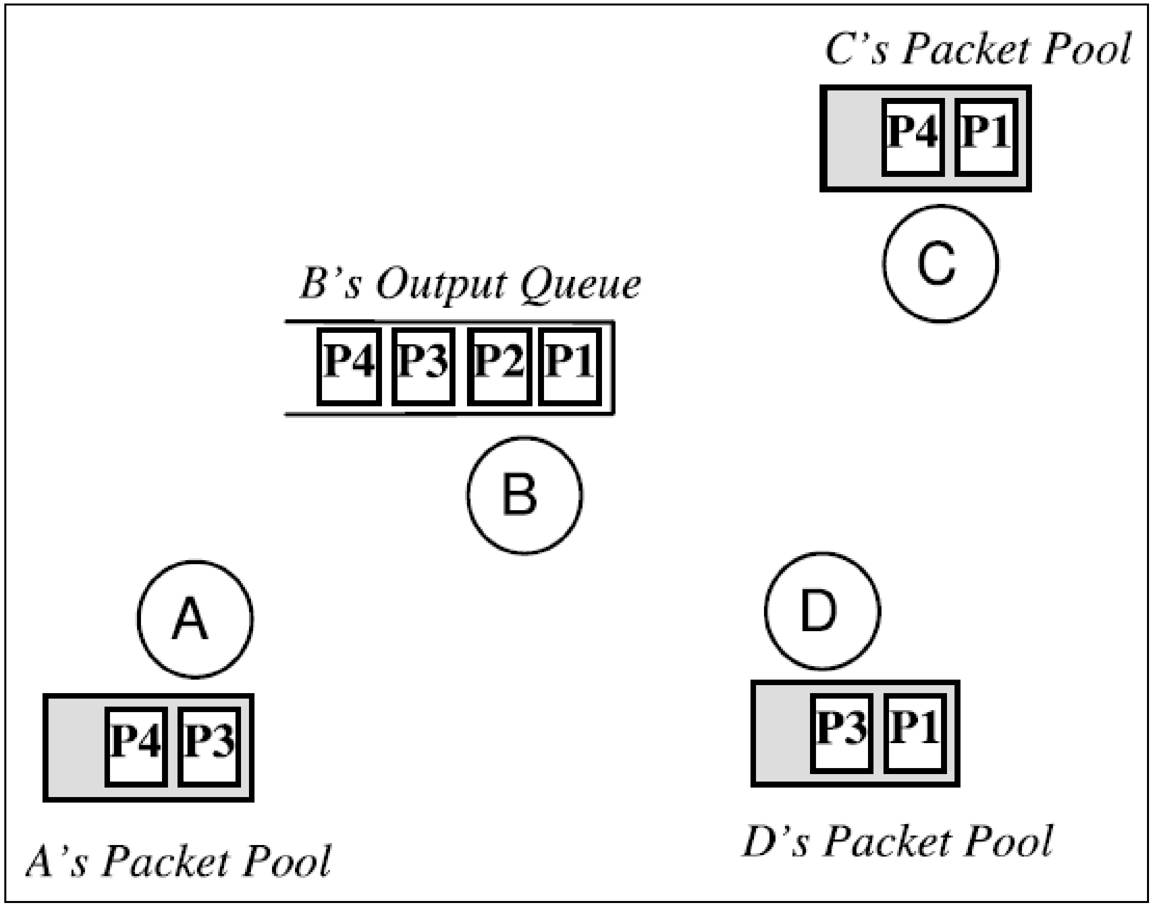
What packets to code together to maximize throughput?

Opportunistic coding



Packets in B's Queue	Next Hop
P1	A
P2	C
P3	C
P4	D

Opportunistic coding

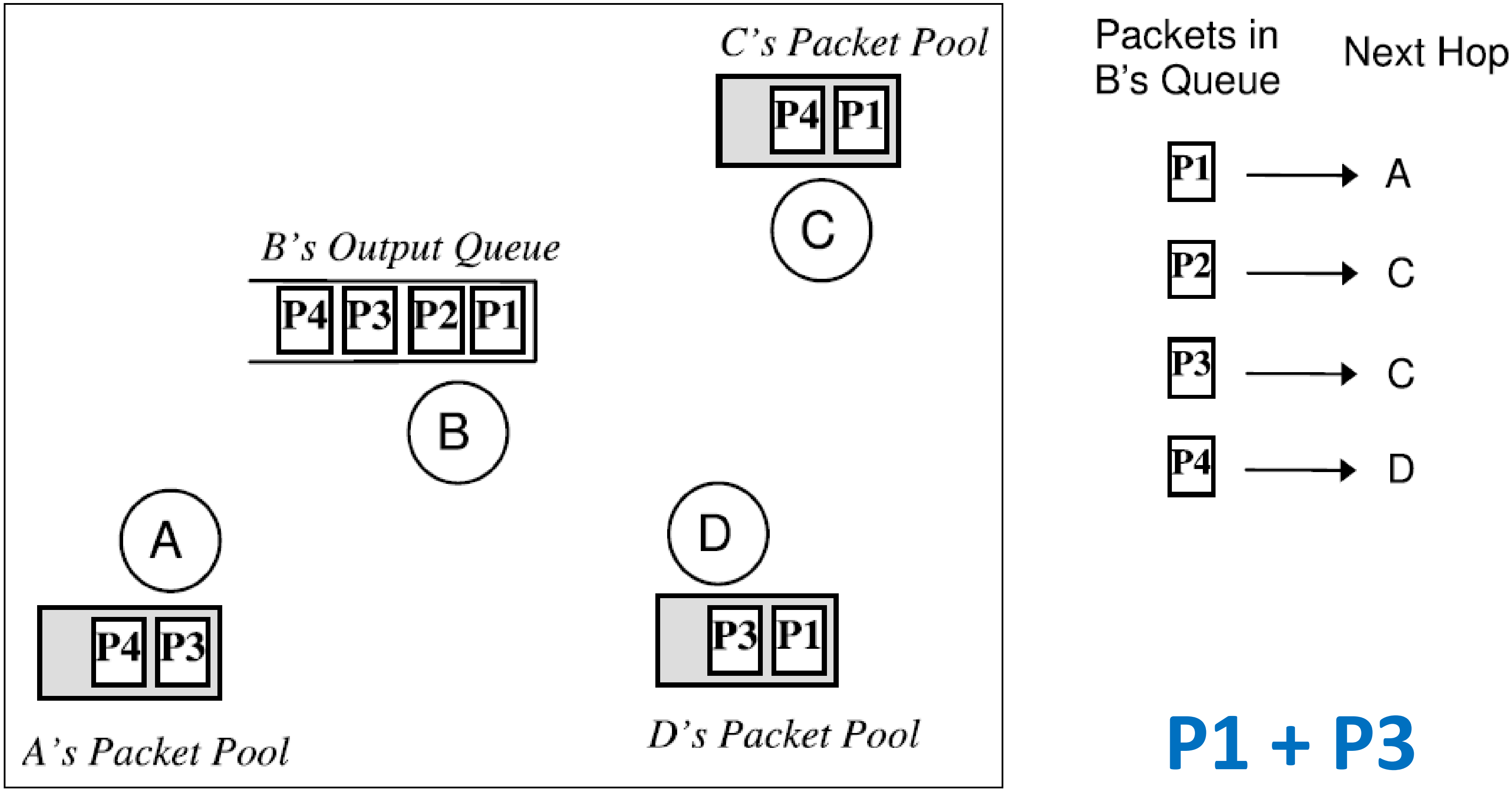


Packets in B's Queue	Next Hop
P1	A
P2	C
P3	C
P4	D

P1 + P2

Bad Coding – C can decode but A can't

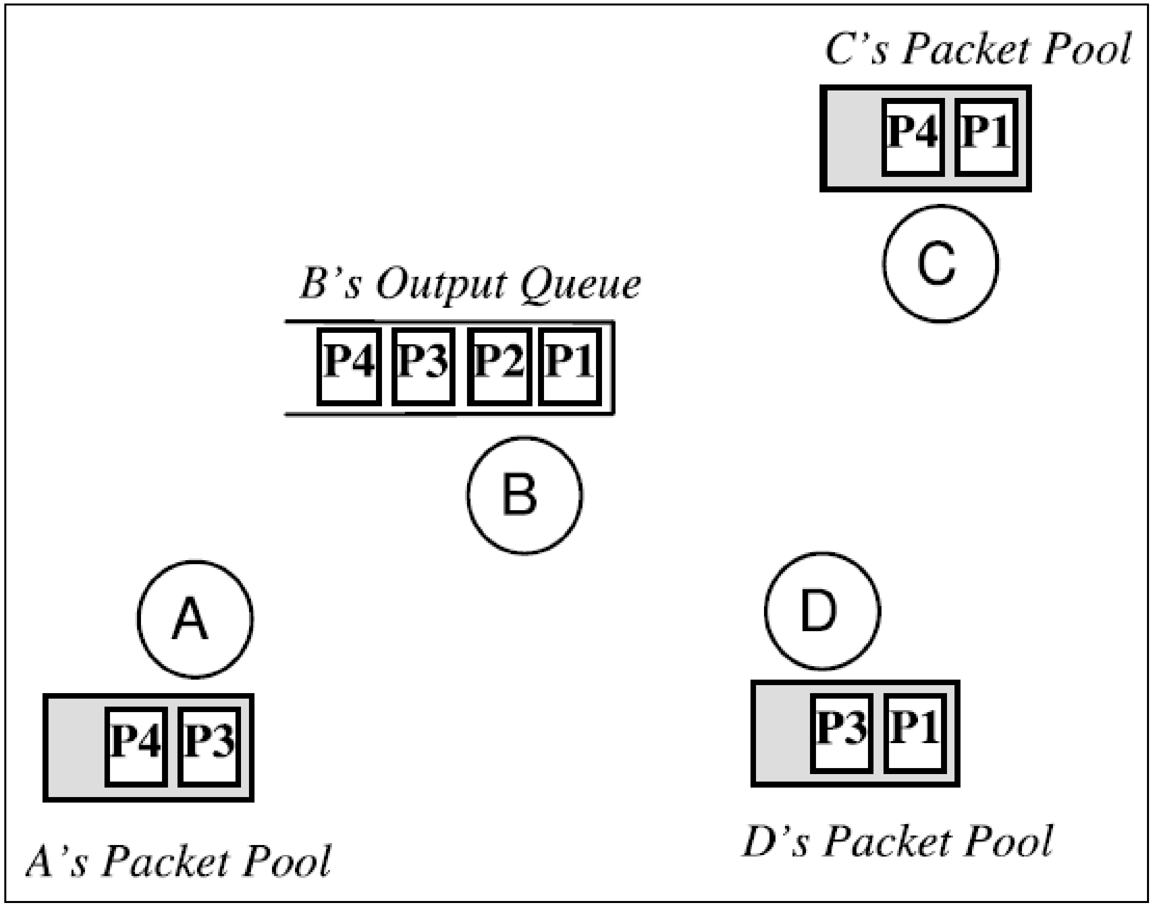
Opportunistic coding



P1 + P3

Better Coding – Both A and C can decode

Opportunistic coding



Packets in B's Queue	Next Hop
P1	A
P2	C
P3	C
P4	D

P1 + P3 + P4

Best Coding – Nodes A, C, D can decode

Opportunistic coding

- Maximize the number of native packets delivered in a single transmission
- While ensuring that each intended next hop has enough information to decode its native packet

Opportunistic coding

To transmit n packets: $p_1, ..., p_n$

To n next hops: $r_1, ..., r_n$

A node can XOR the n packets together only if each next hop r_i has all n-1 packets p_j for j!=i

Choose the largest n that satisfies the above rule

Learning Neighbor State

How does a node know what packets its neighbors have?

- Send reception reports
- During congestion, reports may get lost in collisions or may arrive late

Learning Neighbor State

- Wireless routing protocols compute delivery probability between every pair of nodes and broadcast them
  - E.g.: ETX

- Using these weights,
  - Estimate the probability that a particular neighbor has a packet

Outline

- Design
- Cope Gains
- Making it work
- Implementation details
- Experimental results

COPE Gains

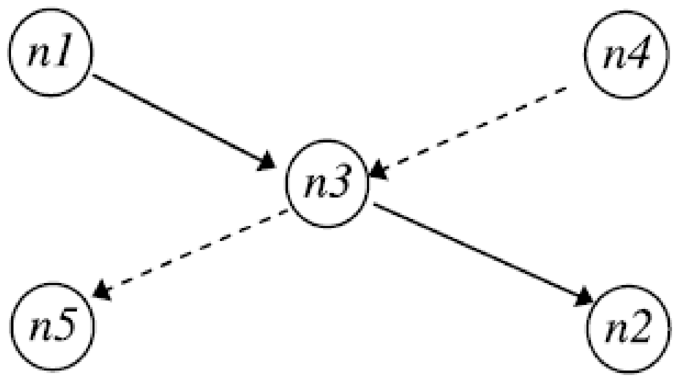
- Coding Gain
- Coding + MAC Gain

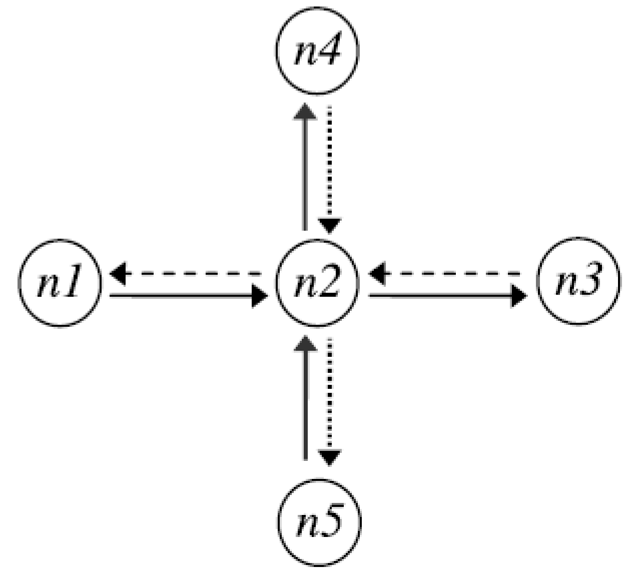
Coding Gain

$$\text{Coding Gain} = \frac{\text{Number of transmissions required by non-coding approach}}{\text{Minimum number of transmissions used by COPE}}$$

Alice & Bob experiment – Coding gain = 4/3 = 1.33

Coding Gain

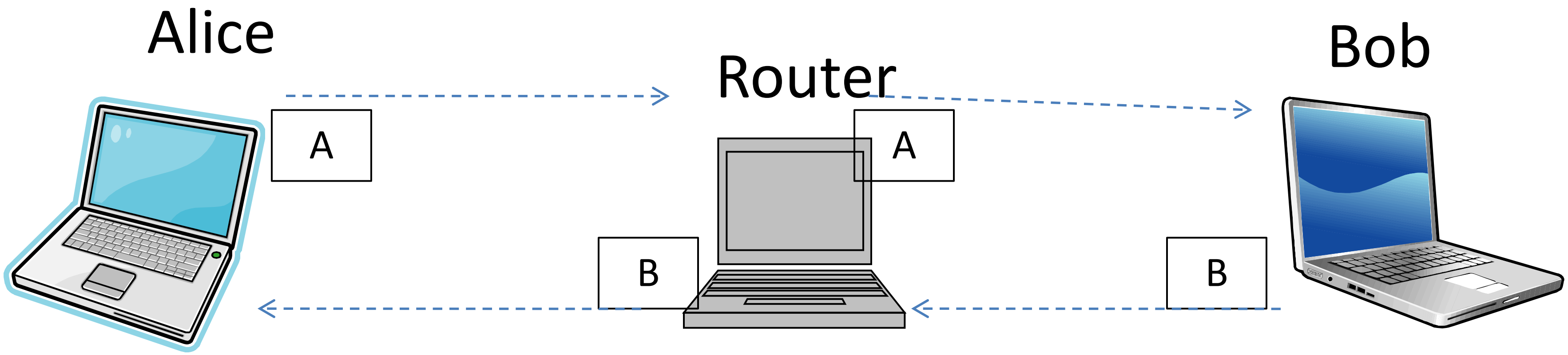




Coding gain = 4/3 = 1.33

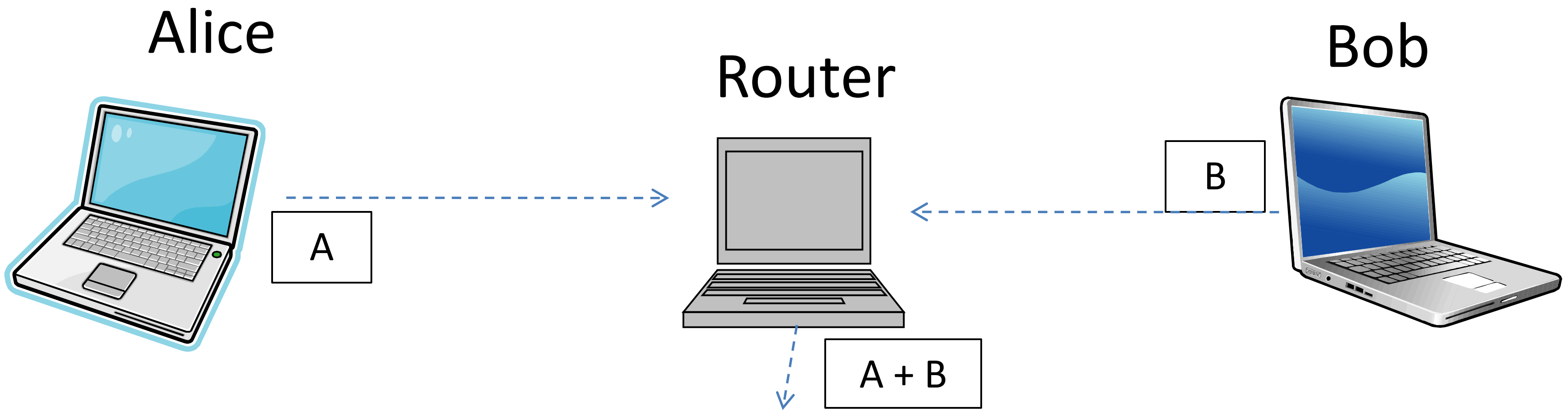
Coding gain = 8/5 = 1.6

Coding + MAC Gain



- MAC divides the bandwidth equally between the 3 contending nodes
- The router needs to transmit twice as many packets
- Hence router is a bottleneck
  - Half the packets are dropped as routers queue

Coding + MAC Gain



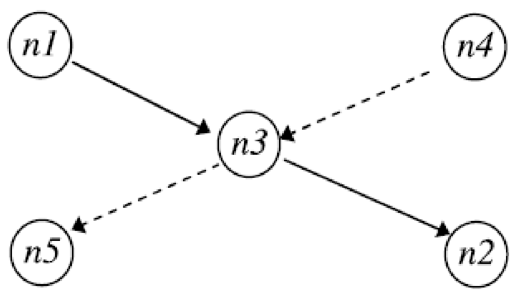
- COPE – XOR pairs of packets
  - router drains packets twice as fast

Coding + MAC gain = 2

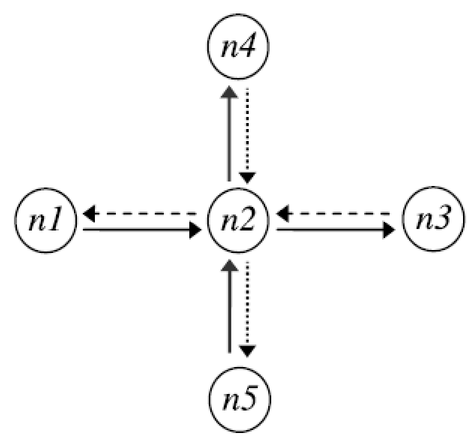
Coding + MAC Gain

- For topologies with single bottleneck

$$\text{Coding + MAC Gain} = \frac{\text{Draining rate with COPE}}{\text{Draining rate without COPE}}$$



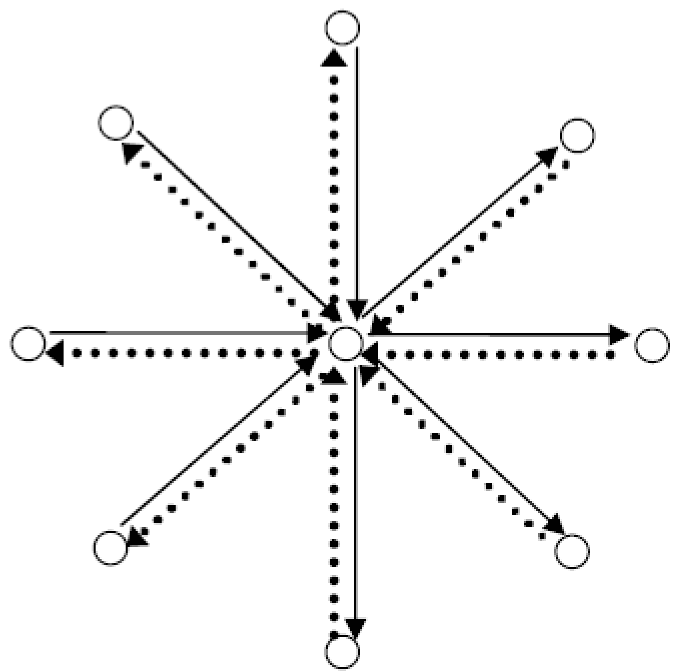
Coding + MAC gain = 2



Coding + MAC gain = 4

Coding + MAC Gain

- In the presence of opportunistic listening, COPE's maximum Coding + MAC gain is unbounded.



N -> ∞

Outline

- Design
- Cope Gains
- Making it work
- Implementation details
- Experimental results

Making it work

- Packet Coding Algorithm
- Packet Decoding
- Pseudo-broadcast
- Hop-by-hop ACKs and Retransmissions
- Preventing TCP packet reordering

Packet Coding Algorithm

- Never delaying packets
  - Does not wait for additional codable packets to arrive
- Preference to XOR packets of similar lengths
  - Pad zeros if different lengths
- Maintain two virtual queues per neighbor
  - One for small, one for large packets
- Dequeue the packet at the head of the FIFO
  - Look only at the head of the virtual queues
- Each neighbor has a high probability of decoding the packet – Threshold probability

Packet Coding Algorithm

1 Coding Procedure

```
Pick packet p at the head of the output queue.
Natives = {p}
Nexthops = {nexthop(p)}
if size(p) > 100 bytes then
    which_queue = 1
else
    which_queue = 0
end if
for Neighbor i = 1 to M do
    Pick packet p_i, the head of virtual queue Q(i, which_queue)
    if ∀n ∈ Nexthops ∪{i}, Pr[n can decode p ⊕ p_i] ≥ G then
        p = p ⊕ p_i
        Natives = Natives ∪{p_i}
        Nexthops = Nexthops ∪{i}
    end if
end for
which_queue = !which_queue
for Neighbor i = 1 to M do
    Pick packet p_i, the head of virtual queue Q(i, which_queue)
    if ∀n ∈ Nexthops ∪{i}, Pr[n can decode p ⊕ p_i] ≥ G then
        p = p ⊕ p_i
        Natives = Natives ∪{p_i}
        Nexthops = Nexthops ∪{i}
    end if
end for
return p
```

Packet Decoding

- Each node maintains a *Packet Pool*
  - Packets it received or sent out
- Packets are stored in a hash table keyed on packet id
- Encoded packet with n packets
  - XOR with n – 1 packets from packet pool

Pseudo-broadcast

- Broadcast
  - No ACKs
  - No retransmissions
  - Poor reliability and lack of back-off
- Unicast
  - ACKed as soon as received
  - Sender back-off exponentially if no ACKs
  - Retransmissions
  - More Reliable

Pseudo-broadcast

- Pseudo-broadcast
  - Unicast packet to one of its recipients
  - That node ACKs and hence the transmission is reliable
  - Since others listen in promiscuous mode they receive the packet as well
  - An XOR header is added after the link-layer header listing all next hops
    - Each node checks the XOR header if it is a recipient and processes the packet

Hop-by-hop ACKs and Retransmissions

- Encoded packets require all next hops to ack the receipt of the associated native packet
  - Only one node ACKs (pseudo-broadcast)
  - There is still a probability of loss to other next hops
  - Hence, each node ACKs the reception of native packet
  - If not-acked, retransmitted, potentially encoded with other packets
  - Overhead - highly inefficient

Hop-by-hop ACKs and Retransmissions

- Asynchronous ACKs and Retransmissions
  - Cumulatively ACK every T_a seconds
  - If a packet is not ACKed in T_a seconds, retransmitted
  - Piggy-back ACKs in COPE header of data packets
  - If no data packets, send periodic control packets (same packets as reception reports)

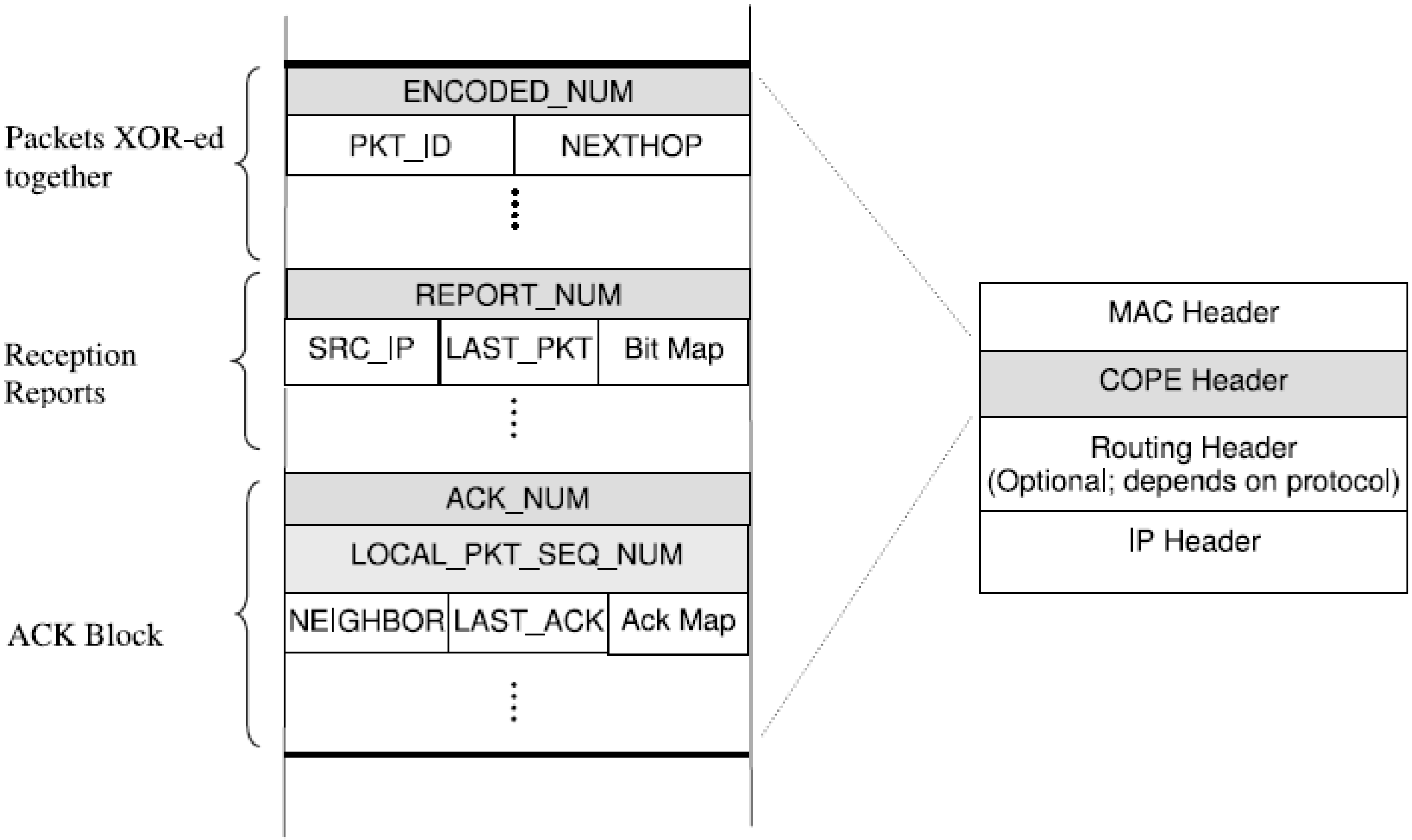
Preventing TCP Packet Reordering

- Asynchronous ACKs can cause packet reordering
  - TCP can take this as a sign of congestion
- Ordering agent
  - Ensures TCP packets are delivered in order
  - Maintains packet buffer

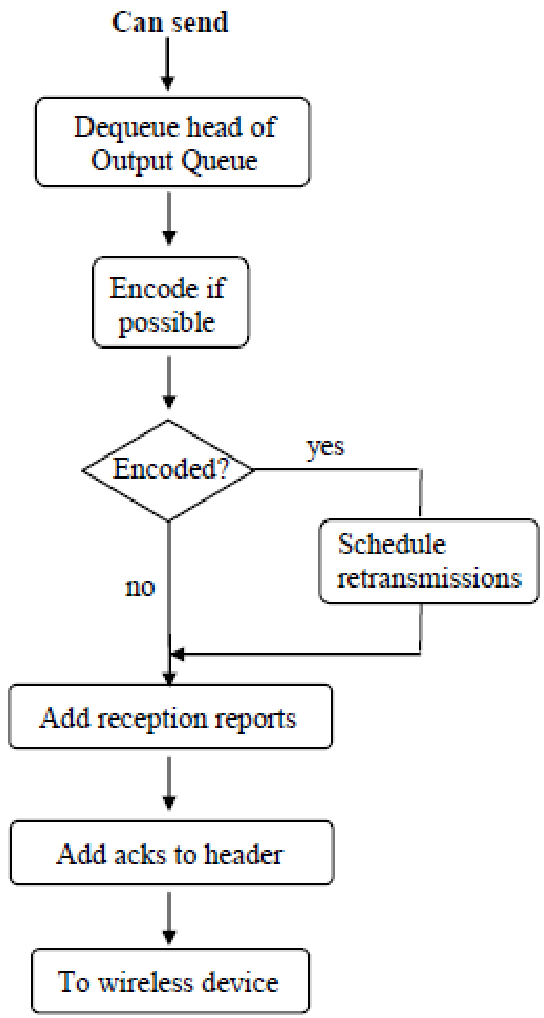
Outline

- Design
- Cope Gains
- Making it work
- Implementation details
- Experimental results

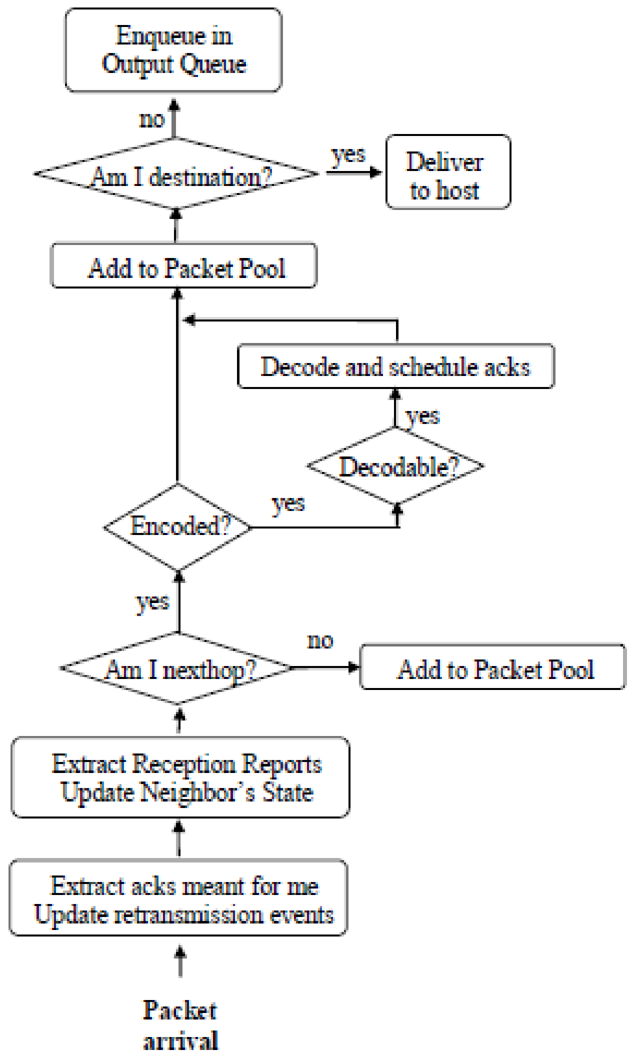
Packet Format



Control flow - Sender



Control flow - Receiver



Outline

- Design
- Cope Gains
- Making it work
- Implementation details
- Experimental results

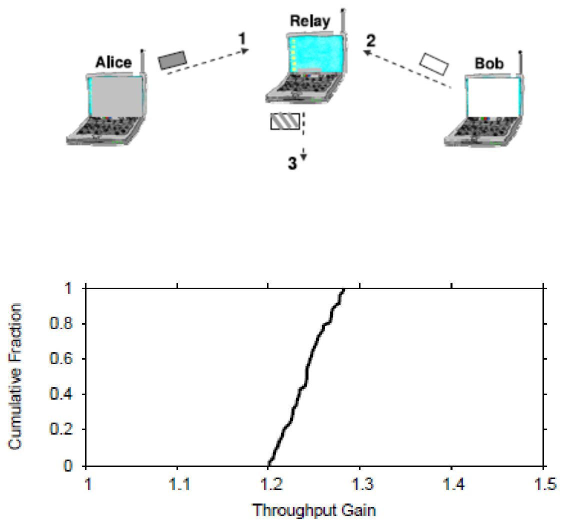
Testbed

- 20 nodes
  - Path between nodes are 1 to 6 hops in length
  - 802.11a with a bit-rate of 6Mb/s
- Software
  - Linux and click toolkit
  - User daemon and exposes a new interface
  - Applications use this interface
    - No modification to application is necessary
- Traffic model
  - *udpgen* to generate UDP traffic
  - *ttcp* to generate TCP traffic
  - Poisson arrivals, Pareto file size distribution

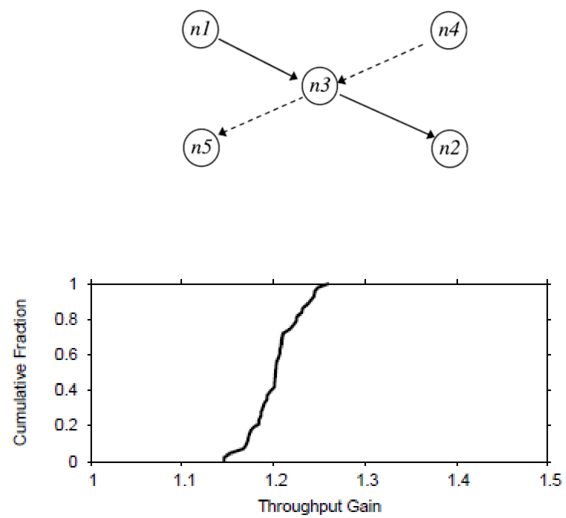
Metrics

- Network throughput
  - Total end-to-end throughput (sum of throughput of all flows in a network)
- Throughput gain
  - The ratio of measured throughput with and without COPE
  - Calculate from two consecutive experiments, with coding turned on and off

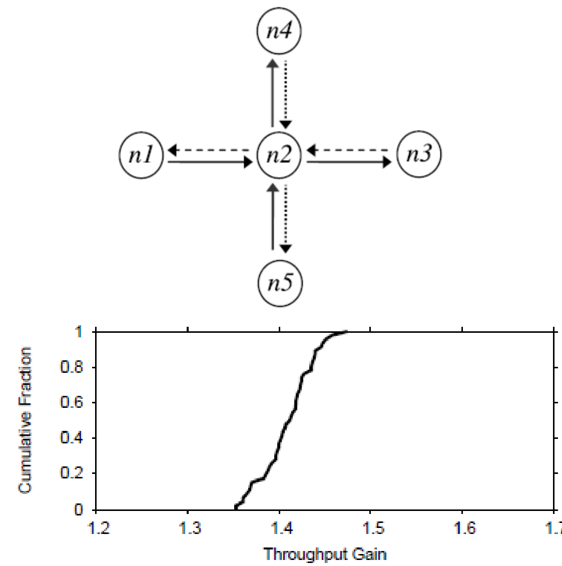
Long-lived TCP flows



(a) TCP gain in the Alice-and-Bob topology



(b) TCP gain in the X-topology

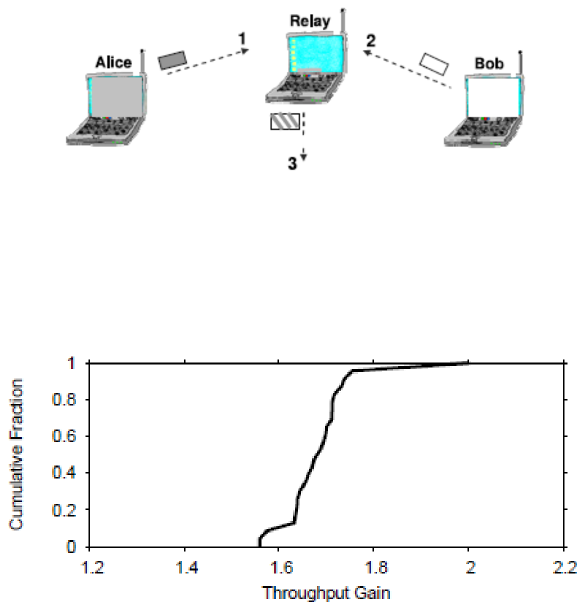


(c) TCP gain in the cross topology

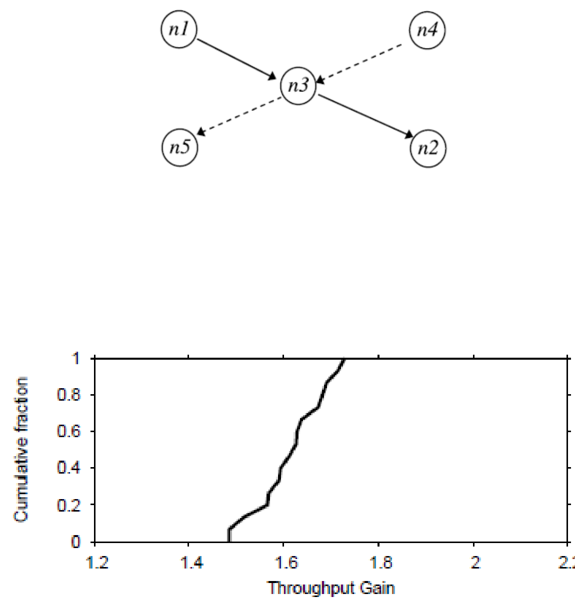
Close to 1.33 | Close to 1.33 | Close to 1.6

- Close to coding gain
  - TCP backs-off due to congestion control
  - To match the draining rate at the bottleneck

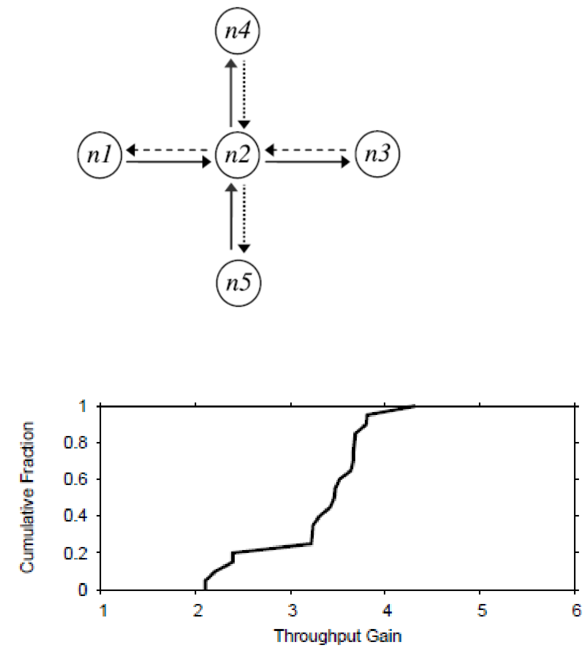
Long-lived UDP flows



(a) UDP gain in the Alice-and-Bob topology



(b) UDP gain in the X-topology



(c) UDP gain in the cross topology

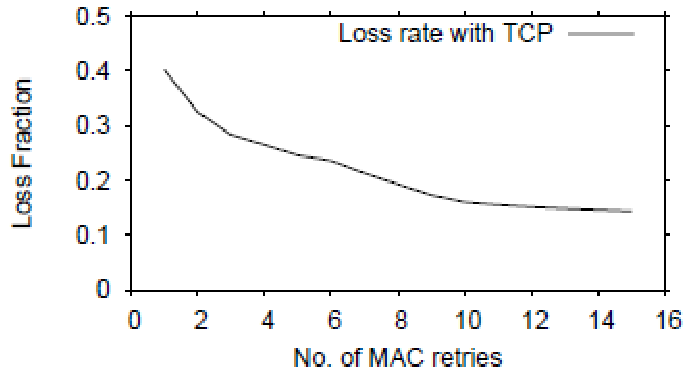
- Close to Coding + MAC gain
  - XOR headers add small overhead (5-8%)
  - The difference is also due to imperfect overhearing

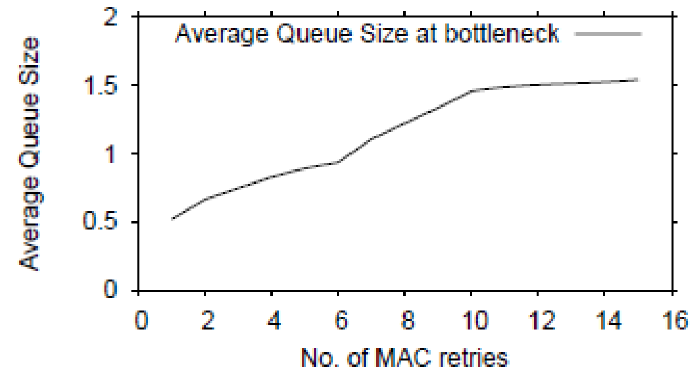
Ad-hoc network - TCP

- TCP flows
  - Arrive according to Poisson process
  - Pick sender and receiver randomly
  - Transfer files (size - Pareto distribution)

- Does not show any significant improvement
  - TCP's reaction to collision-related losses
  - Hidden terminals

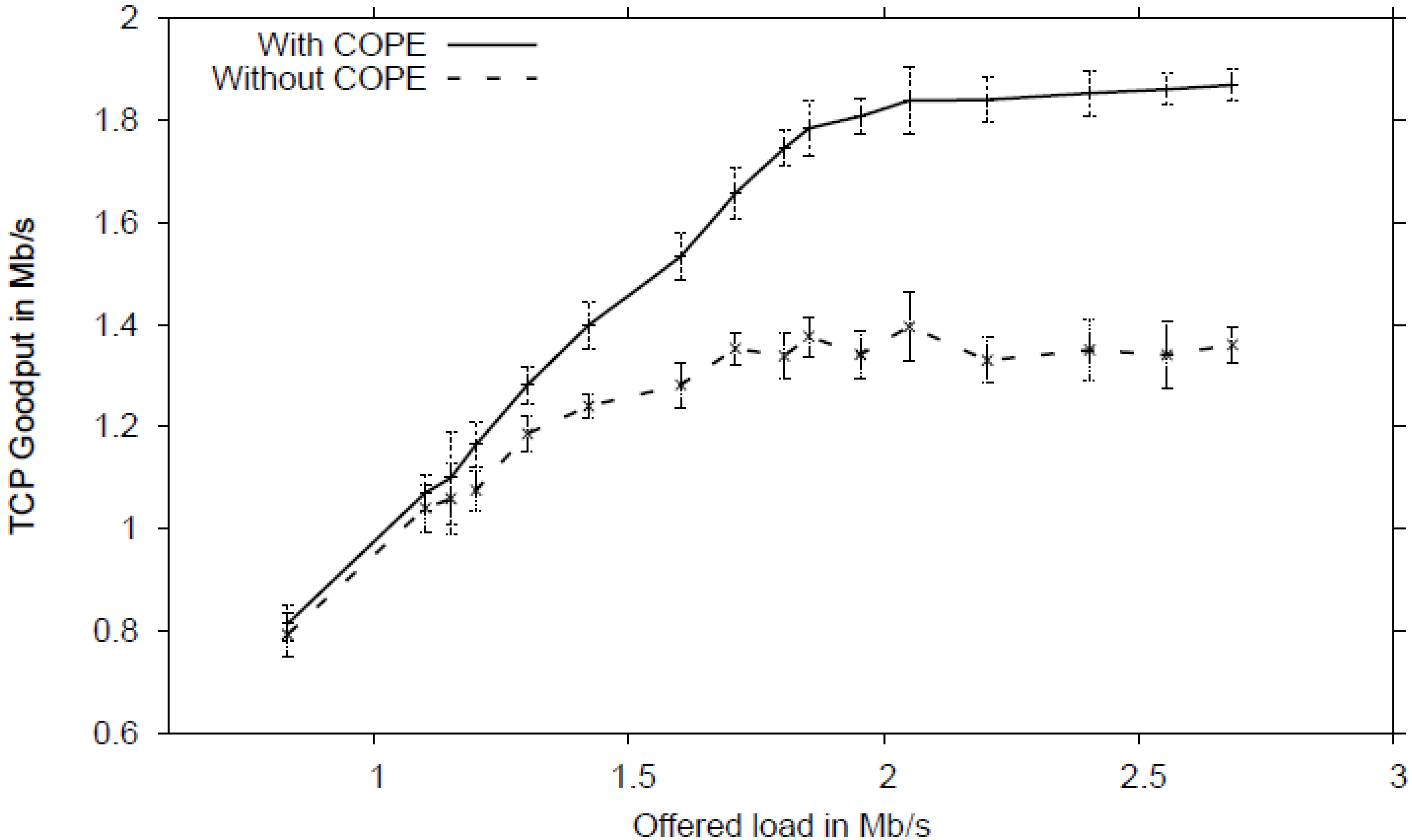
Ad-hoc network - TCP





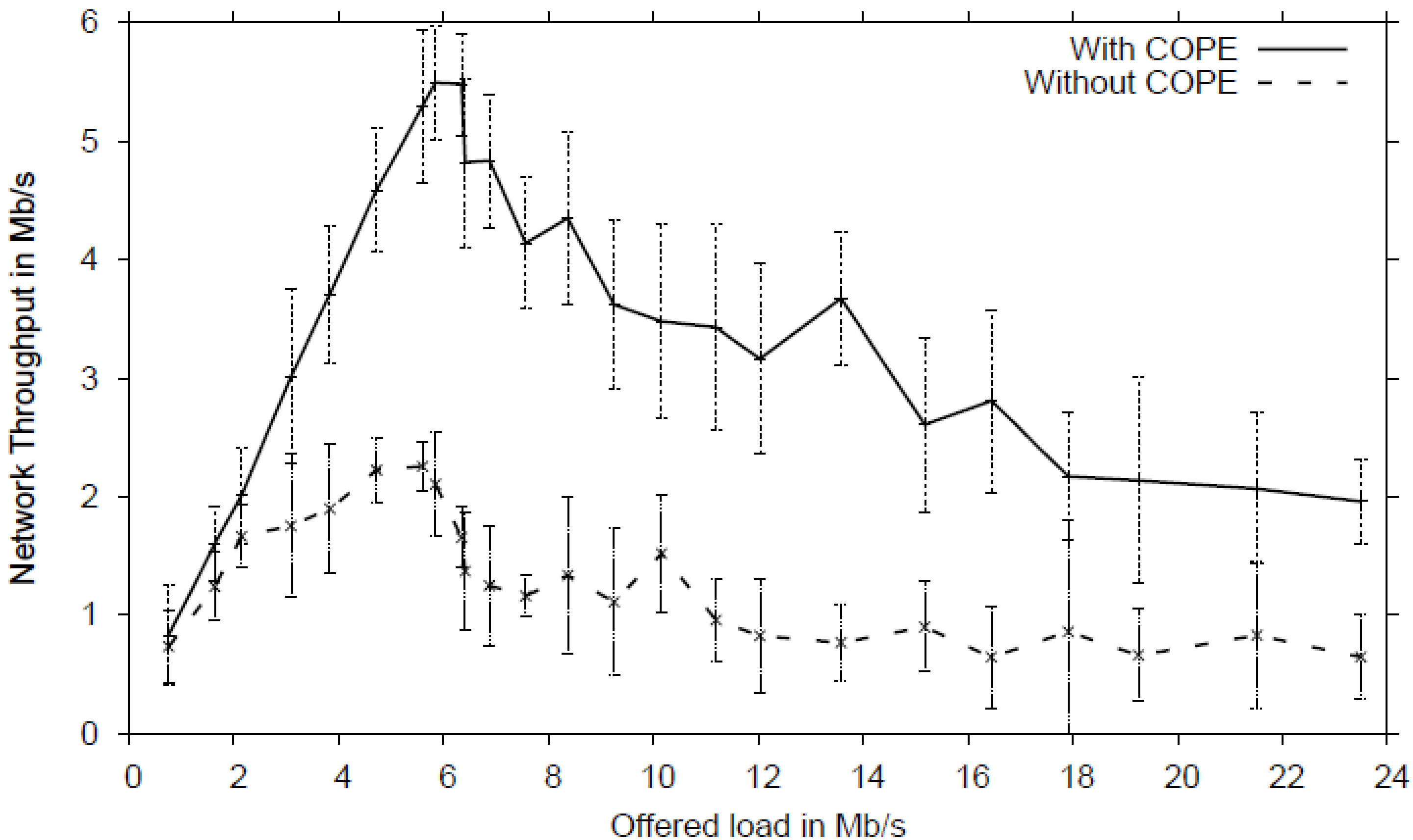
- Even with 15 MAC retries, 14% loss
  - Due to hidden terminals
- Bottleneck never see enough traffic to make use of coding
  - Few coding opportunities

TCP with no hidden terminals



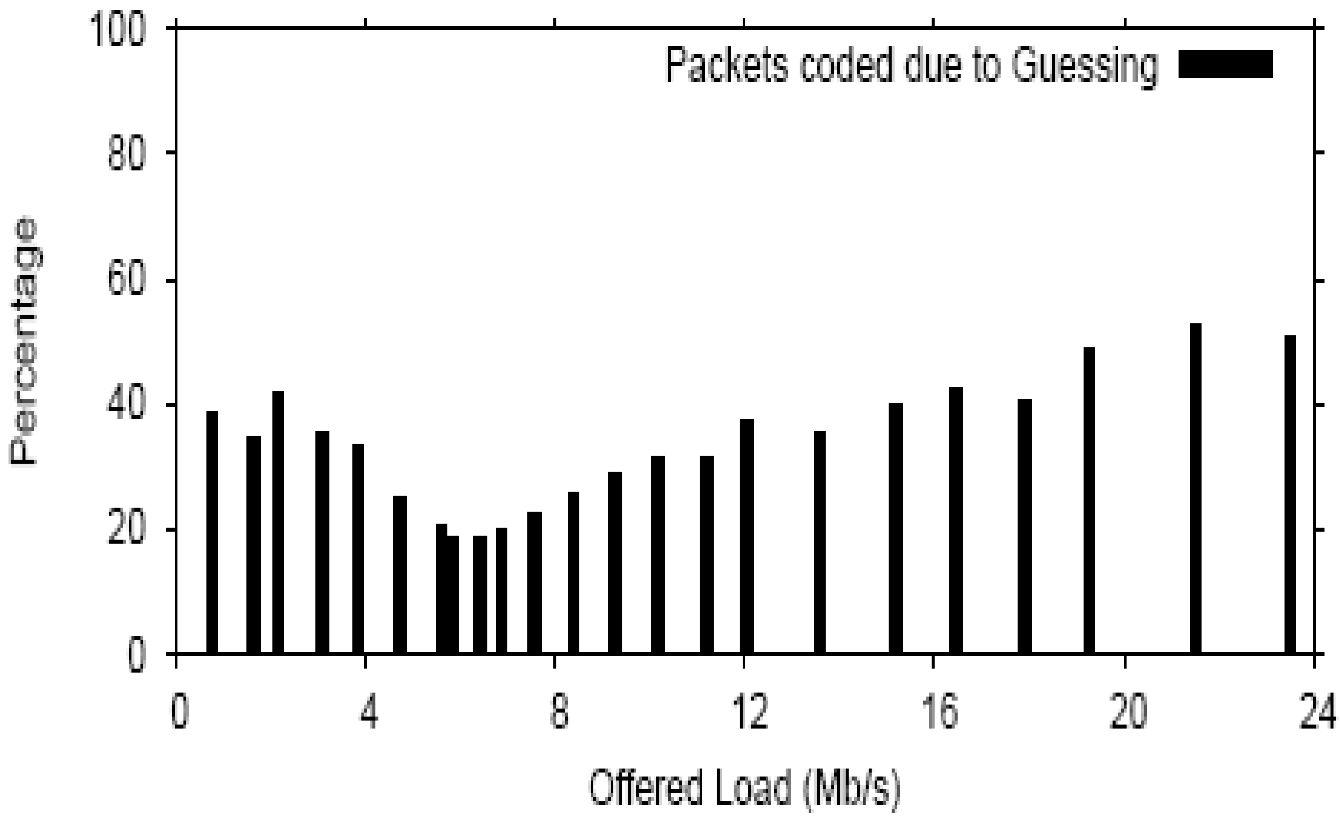
38% improvement in TCP goodput

Ad-hoc network - UDP

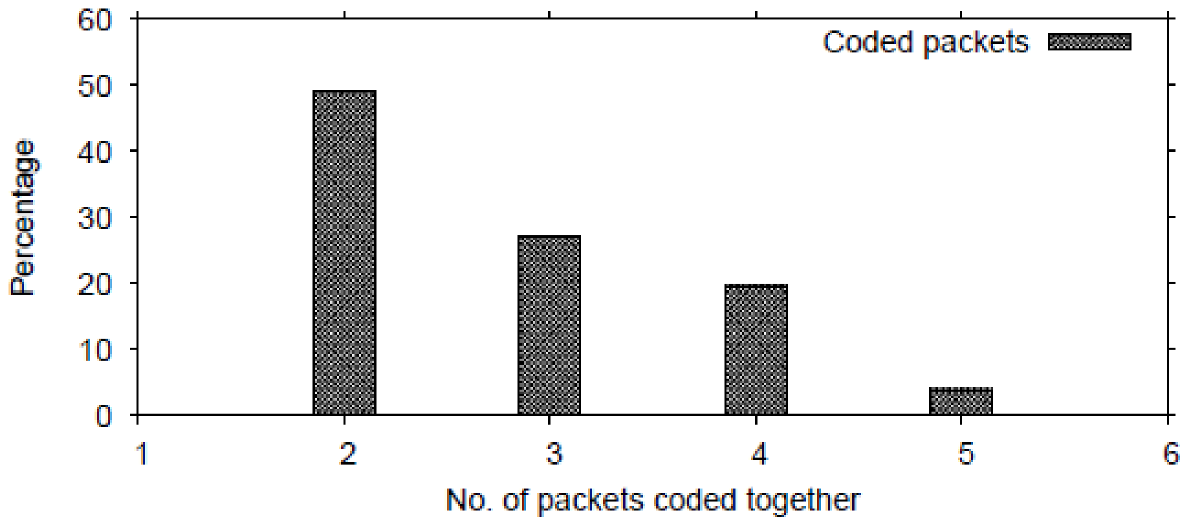


3-4x improvement in throughput

Ad-hoc network - UDP

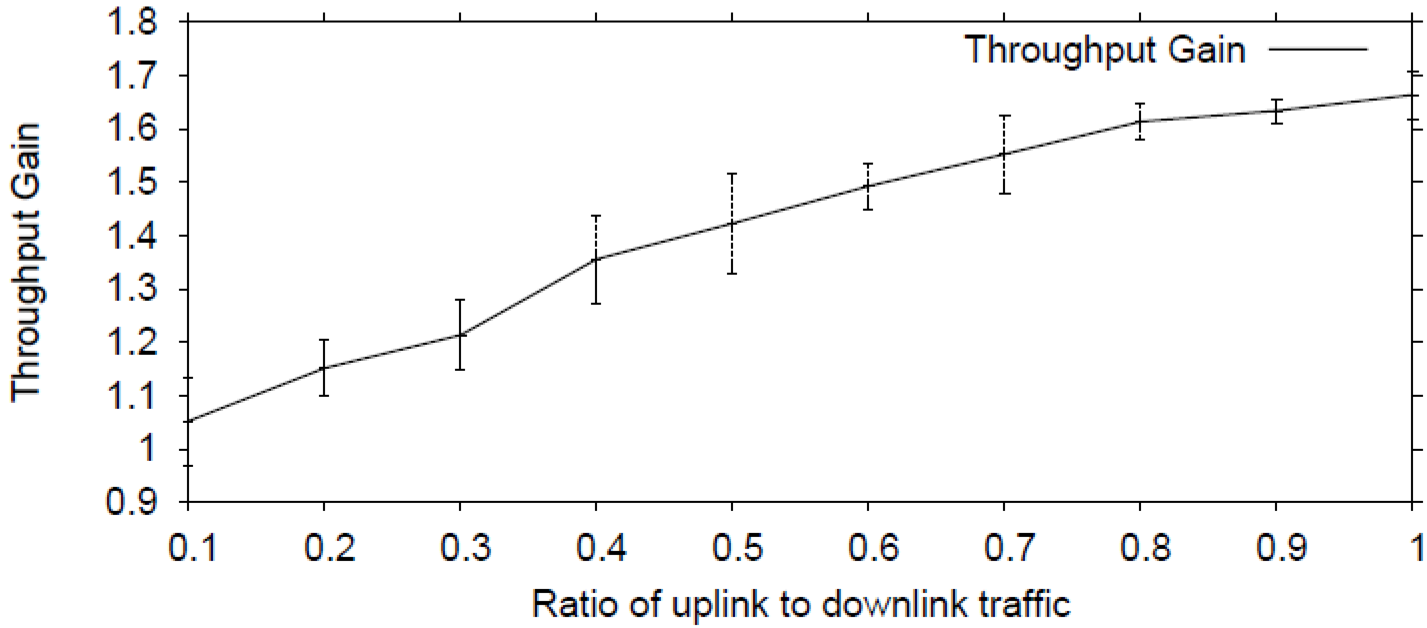


Ad-hoc network - UDP



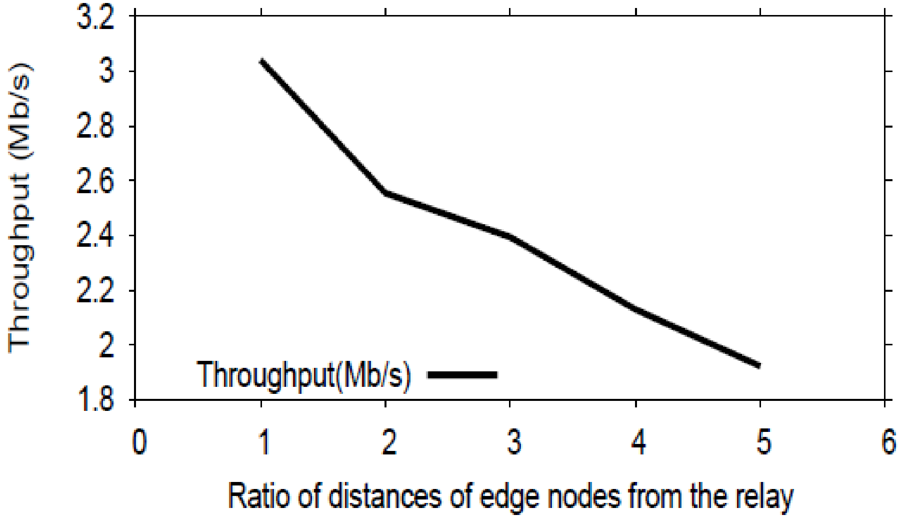
On an average 3 packet are coded together

Mesh network

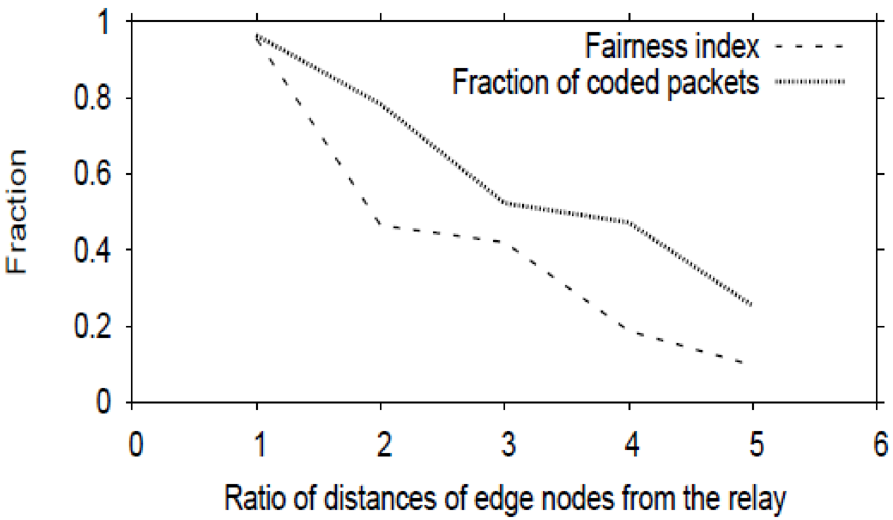


- COPE throughput gain relies on coding opportunities
  - Depends on diversity of packets in the queue of the bottleneck node

Fairness



More fair – more opportunities to code



Conclusion

- Network coding to improve the throughput of wireless networks
- COPE -Implementation of first system architecture for wireless network coding
- COPE improves the UDP throughput by 3-4x
- 5% to 70% throughput improvement in mesh networks depending on downlink-uplink ratio

Thank You

Questions?