XORs in the Air: Practical Wireless Network Coding

Sachin Katti
HariharanRahul, WenjunHu, Dina Katabi,
Muriel Medard, Jon Crowcroft

Presented by Lianmu Chen

Outline

- Introduction
- Cope Overview
- Cope Gains
- Making it work
- Implementation details
- Experimental results
- Conclusions

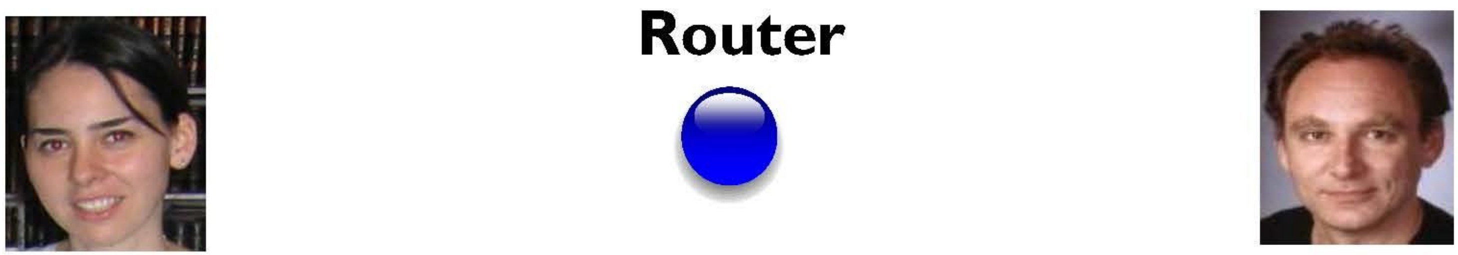
Introduction

Problem

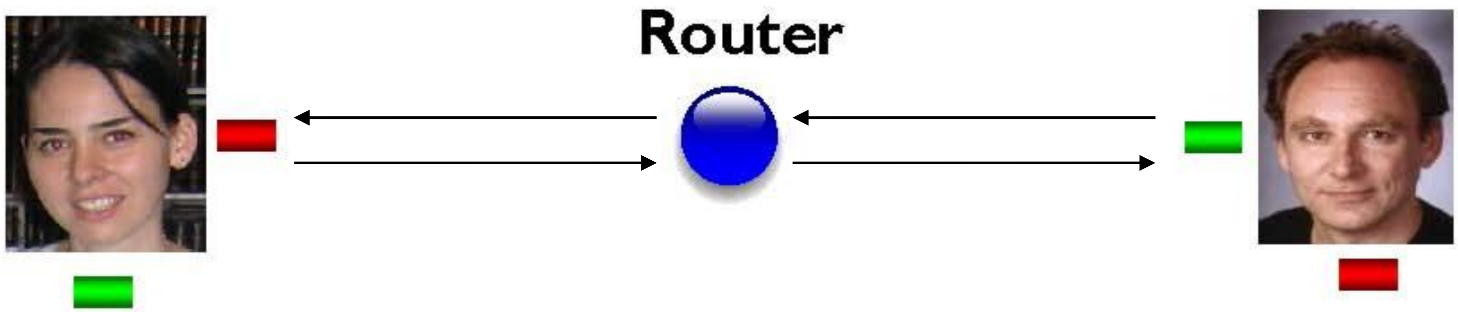
Current wireless implementation suffer from a severe throughput limitation and do not scale to dense large networks.

New architecture: COPE.

Current Approach

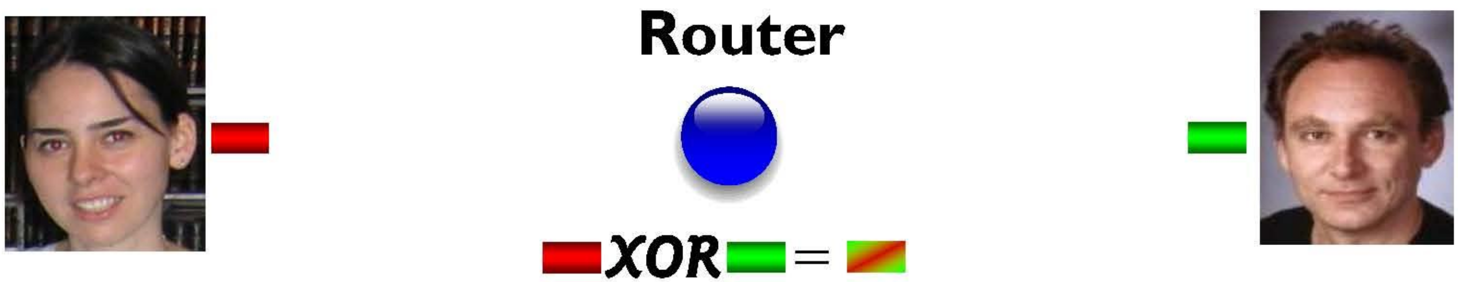


Current Approach

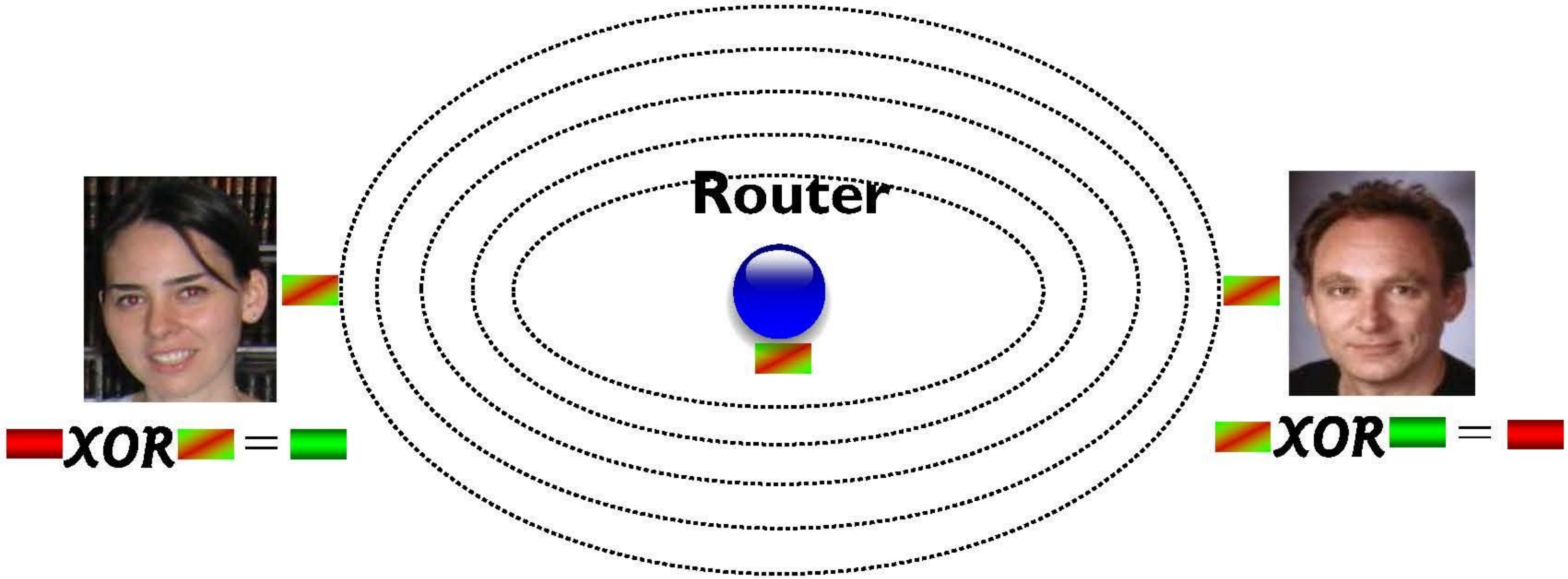


- Requires 4 transmissions
- Can we do it in fewer transmissions?

Our Approach



Our Approach



- Requires 3 transmissions instead of 4
- Increased throughput

Cope Overview

Cope Overview

Cope incorporates three main techniques:

(a) Opportunistic Listening

(b) Opportunistic Coding

(c) Learning Neighbor State

Opportunistic Listening

(a)sets the nodes in promiscuous mode

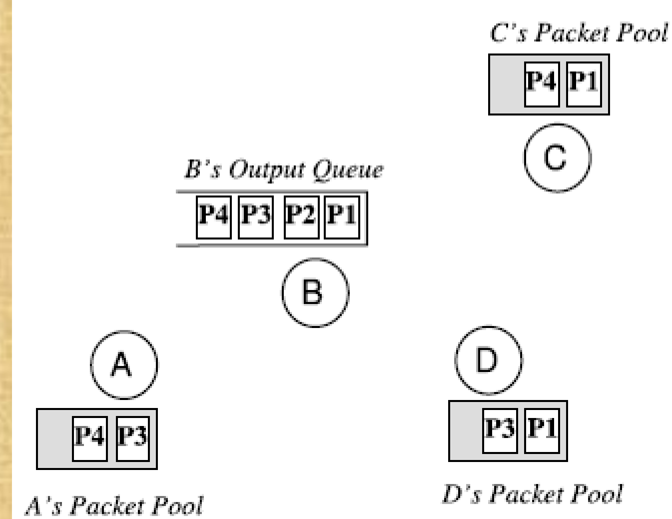
(b) snoop on all communications, store the overheard packets for a limited period *T*

(c) each node broadcasts ***reception reports***

Opportunistic Coding

Rule:

"A node should aim to maximize the number of native packets delivered in a single transmission, while ensuring that each intended next-hop has enough information to decode it's native packet."



(a) B can code packets it wants to send

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

(b) Nexthops of packets in B's queue

Coding Option	Is it good?
P1 + P2	Bad Coding (C can decode but A can't)
P1 + P3	Better Coding (Both A and C can decode)
P1 + P3 + P4	Best Coding (Nodes A, C, and D can decode)

(c) Possible coding options

Opportunistic Coding

Issues:

- Unneeded data should not be forwarded to areas where there is no interested receiver, wasting capacity.
- The coding algorithm should ensure that all next-hops of an encoded packet can decode their corresponding native packets.

Rule: To transmit n packets $p_1 \ldots p_n$ to n next-hops $r_1 \ldots r_n$, a node can XOR the n packets together only if each next-hop r_i has all n - 1 packets p_j for $j \neq i$

Learning Neighbor State

(a)Reception report

(b)guess whether a neighbor has a particular packet.

- *COPE estimates the probability that a particular neighbor has a packet, as the delivery probability of the link between the packet's previous hop and the neighbor.*
- incorrect guess： relevant native packet is retransmitted, encoded with a new set of native packets.

Cope's Gains

Understanding COPE's Gains

Coding Gain

Definition： the ratio of no. of transmissions required without COPE to the no. of transmissions used by COPE to deliver the same set of packets.

Theorem*: In the absence of opportunistic listening, COPE's maximum coding gain is 2, and it is achievable.*

Obviously, this number is greater than 1

And 4/3 for Alice-Bob Example

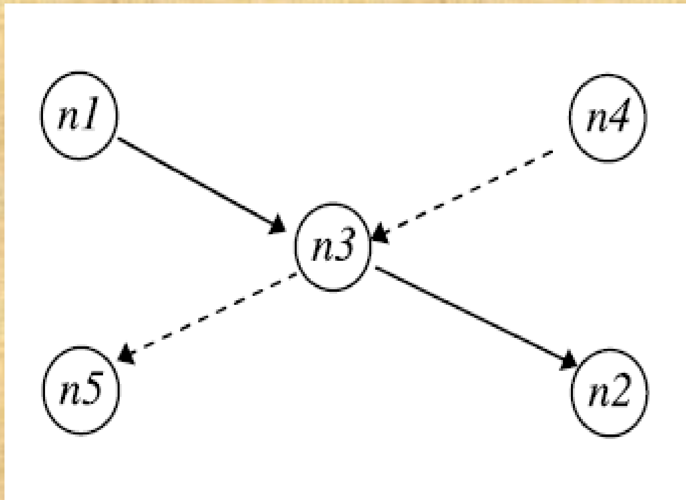
Coding Gain

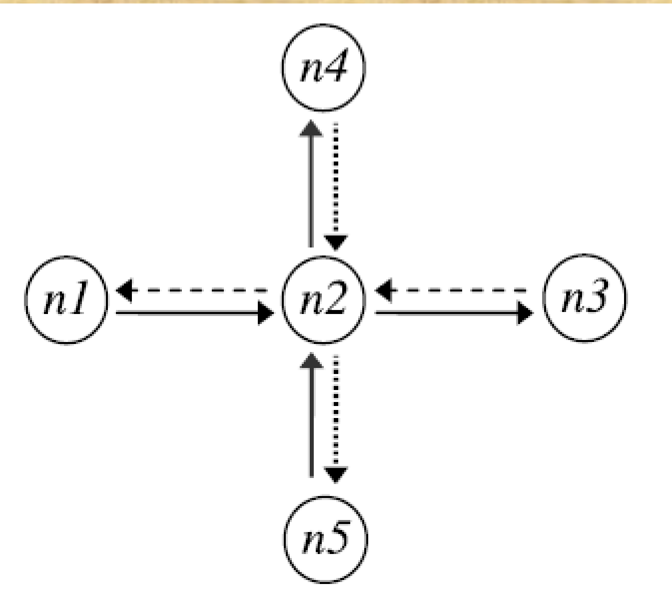
Chain topology; 2 flows in reverse directions.

Coding gain of the chain tends to 2 as the number of intermediate nodes increases.
The complete proof is in Appendix A.

Coding Gain

Obviously, the coding gain in Alice and Bob example is 4/3.





Coding gain = 4/3 = 1.33

Coding gain = 8/5 = 1.6

In the presence of opportunistic listening

Understanding COPE's Gains

Coding+MAC Gain

- **Definition**： the ratio of the bottleneck's draining rate with COPE to its draining rate without COPE.
- ***Theorem 2****: In the absence of opportunistic listening, COPE's maximum Coding+MAC gain is 2, and it is achievable.*

COPE+MAC Gains

Theorem 3*: In the presence of opportunistic listening, COPE's maximum Coding+MAC gain is unbounded.*

Topology	Coding Gain	Coding+MAC Gain
Alice-and-Bob	1.33	2
"X"	1.33	2
Cross	1.6	4
Infinite Chain	2	2
Infinite Wheel	2	∞

Table 2—Theoretical gains for a few basic topologies

Making it work

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
  - Distinguish between small and large packets
- Never code together packets headed to the same next-hop
  - maintains two virtual queues per neighbor; one for small packets and another for large packets, an entry is added to the appropriate virtual queue based on the packet's nexthop and size
- Dequeue the packet at the head of the FIFO
  - Look only at the head of the virtual queues, determine if it is a small or a large packet
- Each neighbor has a high probability of decoding the packet – Threshold probability

Packet Decoding

- Each node maintains a packet pool
- When a node receives an XORed collection of packets, it searches for the corresponding native node from it's pool
- It ultimately XORs the *n* - 1 packets with the received encoded packet to retrieve it's own native packet.

Pseudo-broadcast

802.11 MAC modes: unicast and broadcast

Unicast:

• packets are immediately *ack*ed by next-hops
•back-off if an *ack* is not received

Broadcast: Since COPE broadcasts encoded packets to their next hops, the natural approach would be to use broadcast
•Low reliability (In the absence of the acks, the broadcast mode offers no retransmissions)
•cannot detect collisions, does not back off
•high collision rates, poor throughput

Solution: ***Pseudo-broadcast***

Pseudo-broadcast

- Pseudo-Broadcast
  - Piggybacks on 802.11 Unicast

    it Unicasts packets meant for Broadcast.
  - Link-layer *dest* field is sent to the MAC address of one of the intended recipients, with an XOR-header added afterward, listing all the next-hops. (All nodes hear this packet)
  - If the recipient receives a packet with a MAC address different from it's own and if it is a next-hop, it processes it further. Else, it stores it in a buffer.
  - Since this is essentially Unicast, collisions are detected, and back-off is possible as well.

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)

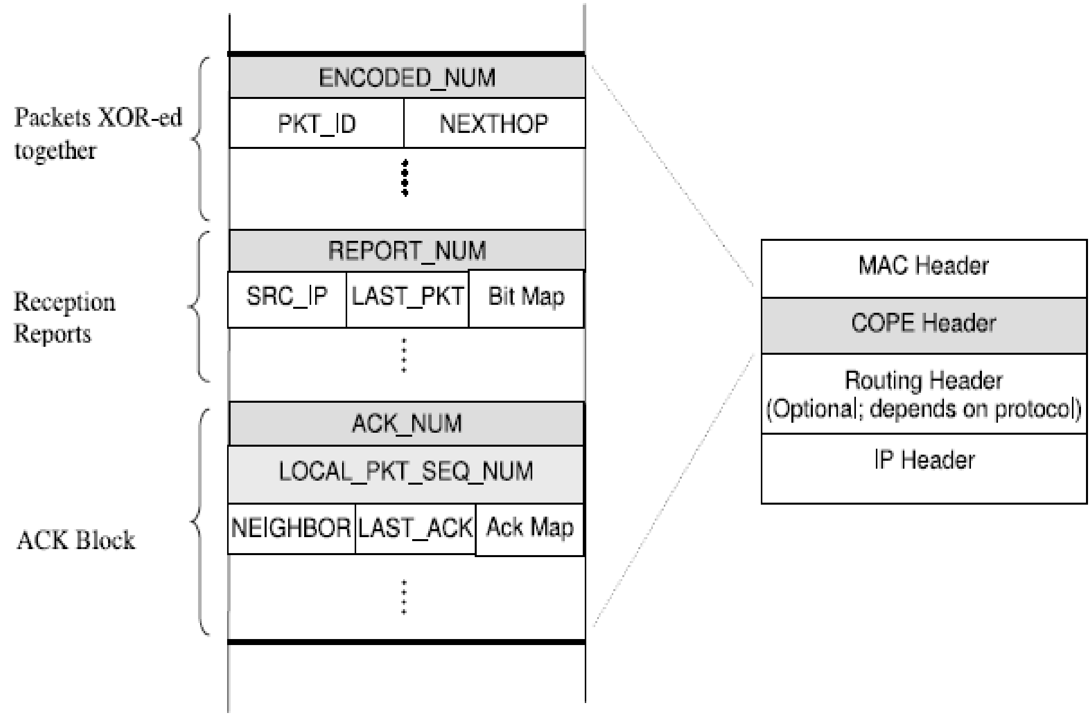
Preventing TCP Packet Reordering

- 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

Implementation

Implementation Details

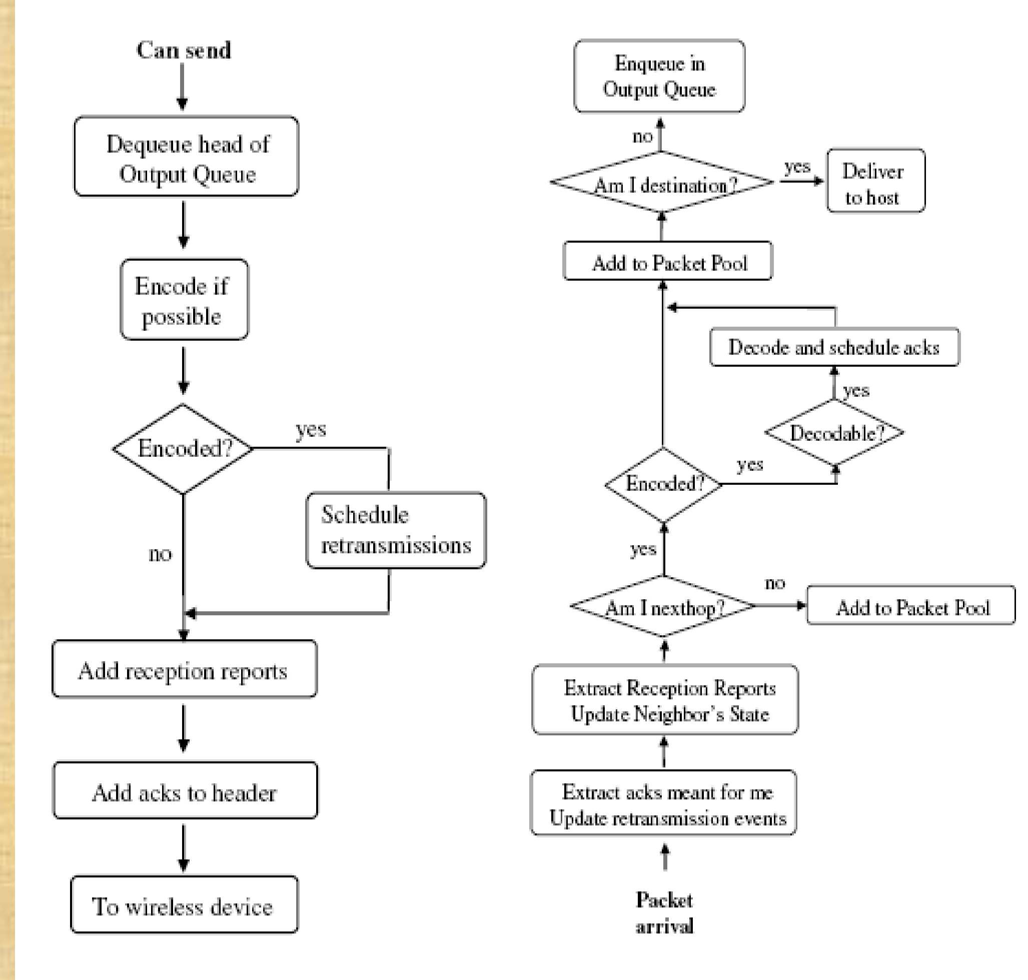
Packet Format:



The first block identifies the native packets XOR-ed and their nexthops. The second block contains reception reports. Each report identifies a source, the last IP sequence number received from that source, and a bit-map of most recent packets seen from that source. The third block contains asynchronous acks. Each entry identifies a neighbor, an end point for the ACK map, and a bit-map of ack-ed packets.

Implementation Details

Control Flow :



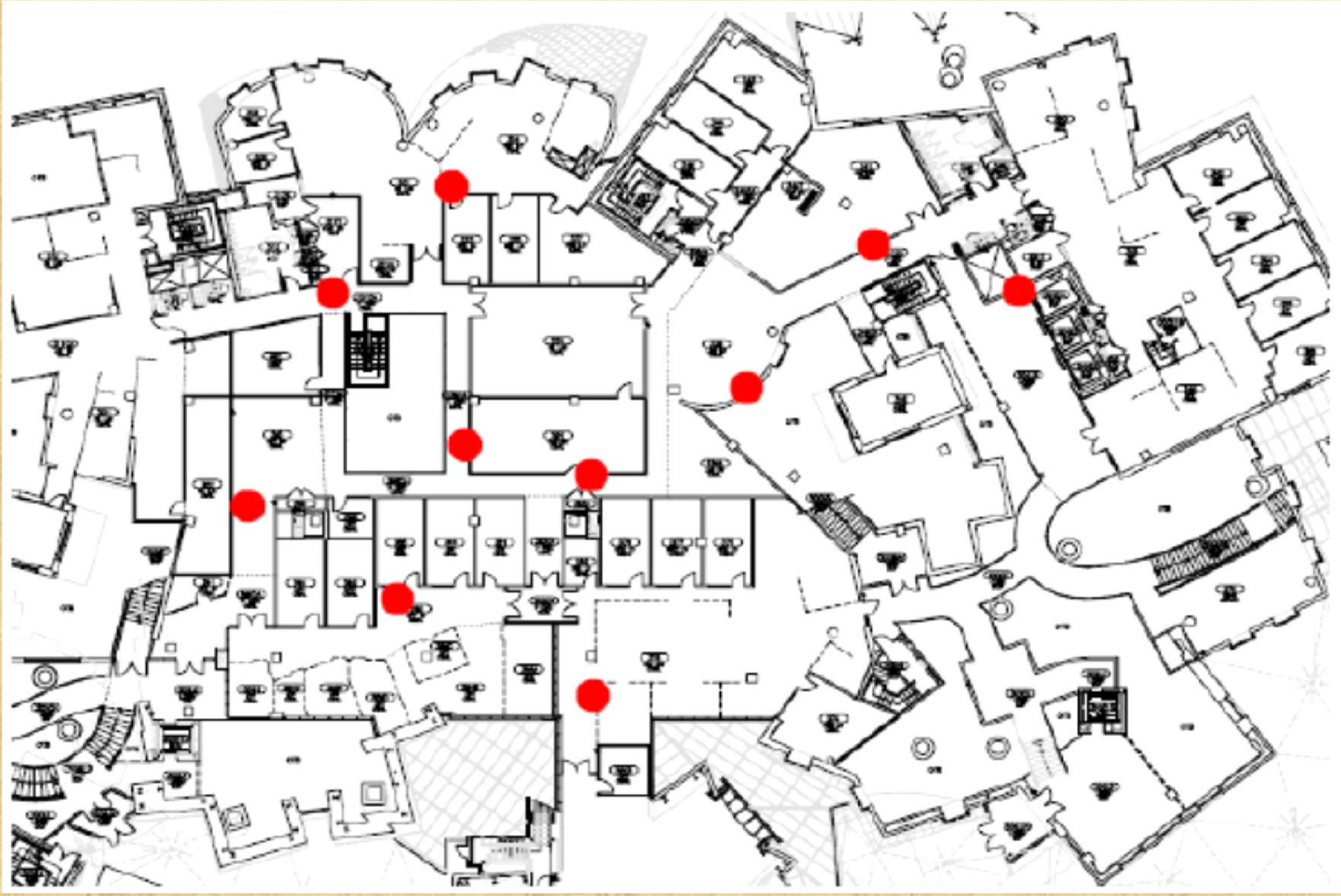
(a) Sender side (b) Receiver side

Experimental Result

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

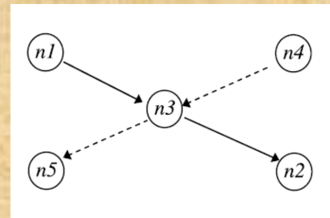
Experimental Results

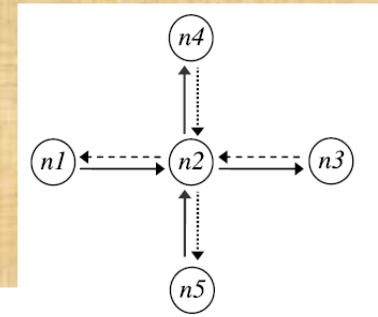


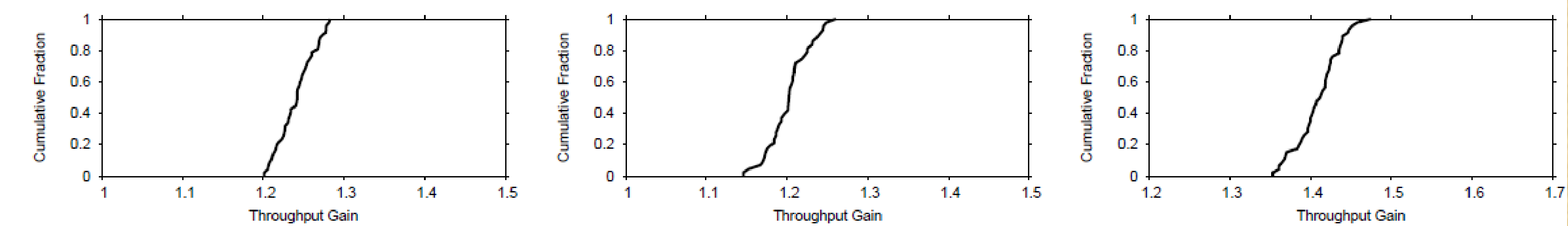
Metrics

- **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

COPE in gadget topologies: Long-lived TCP flows





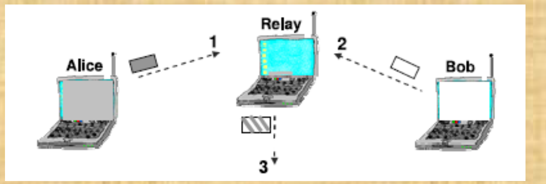


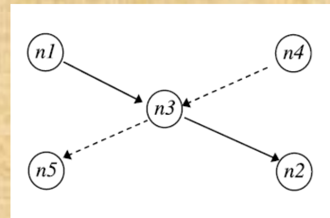
(a) TCP gain in the Alice-and-Bob topology (b) TCP gain in the X-topology (c) TCP gain in the cross topology

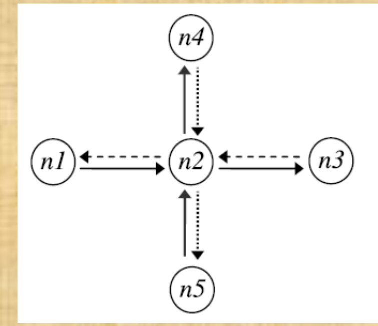
Close to 1.33 Close to 1.33 Close to 1.6

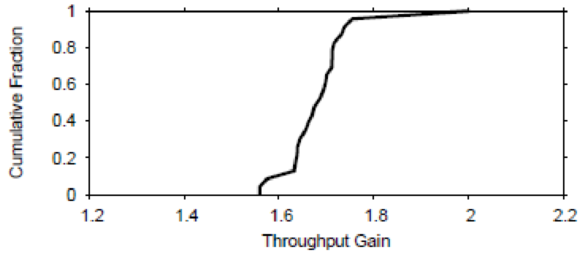
- **Throughput gain corresponds to coding gain， rather than Coding+MAC gain**
  - TCP backs-off due to congestion control
  - To match the draining rate at the bottleneck

Long-lived UDP flows

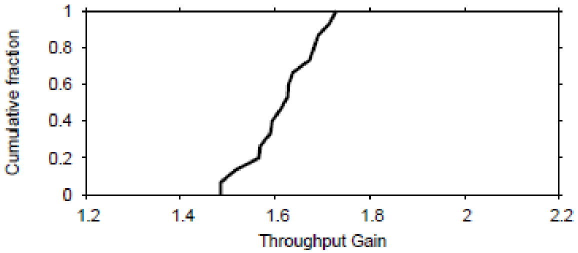




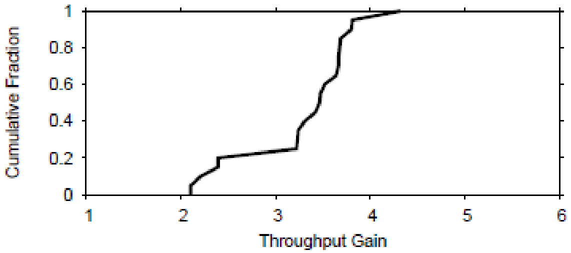




(a) UDP gain in the Alice-and-Bob topology



(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 , flow asymmetry

COPE in an Ad Hoc Network

TCP:

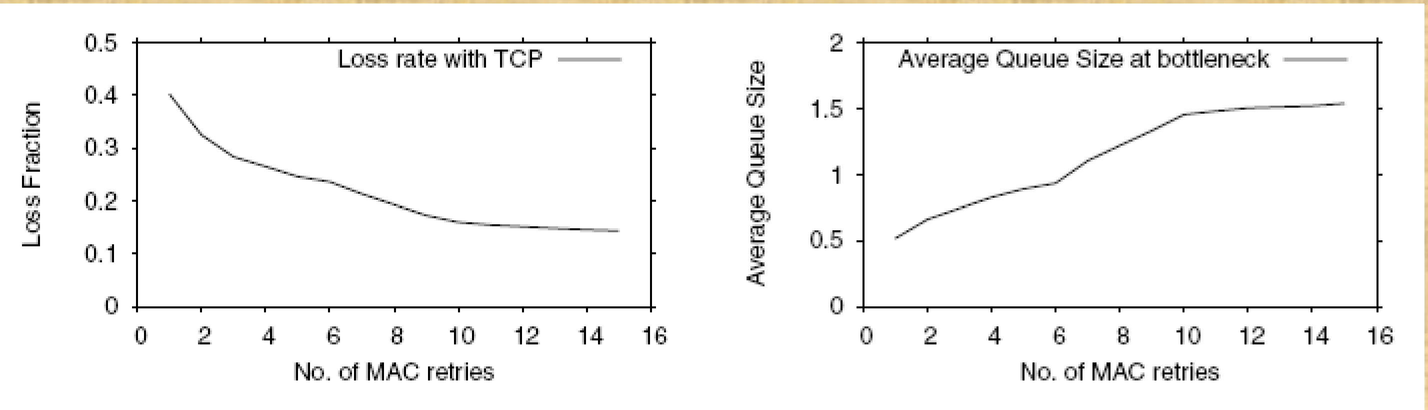
- TCP flows arrive according to a Poisson process, pick sender and receiver randomly, and the traffic models the Internet.
- TCP does not show significant improvement (average gain is 2-3%)

Why ?

Collision- related losses:

- Nodes are not within carrier sense of each other, resulting in **hidden terminal problems**

COPE in an Ad Hoc Network



•15 MAC retries , the TCP flows experience 14% loss
•TCP flows suffer timeouts and excessive back-off, unable to ramp up and utilize the medium efficiently.

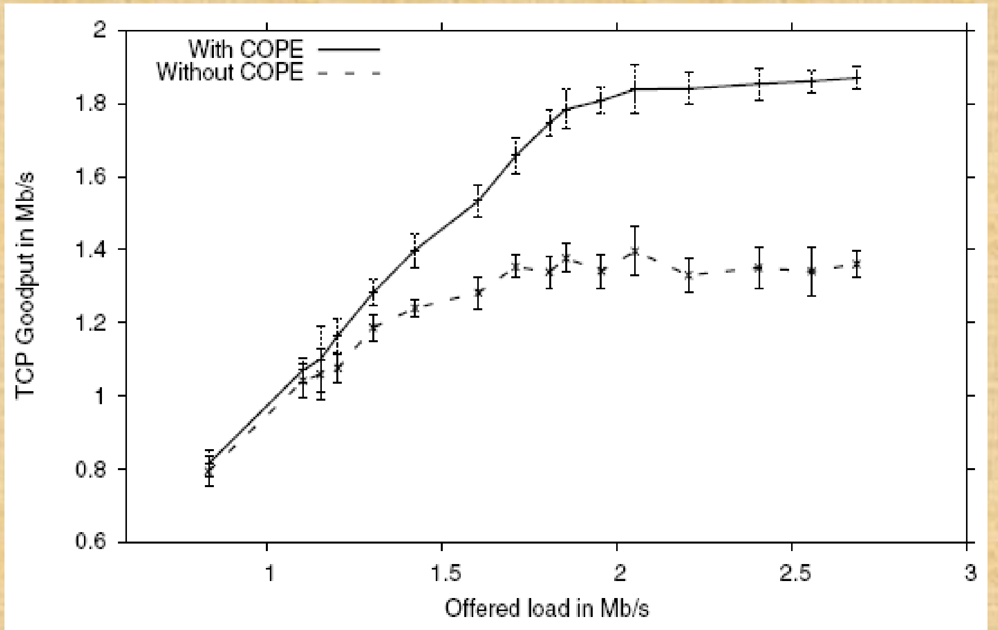
•Most of time: no packets in their queues or just a single packet.
•No enough traffic to make use of coding;
•Few coding opportunities arise

Hence, the performance is the same with and without coding

COPE in an Ad Hoc Network

TCP in a collision-free environment

- Bring the nodes closer together, within carrier sense range, hence avoid collisions.

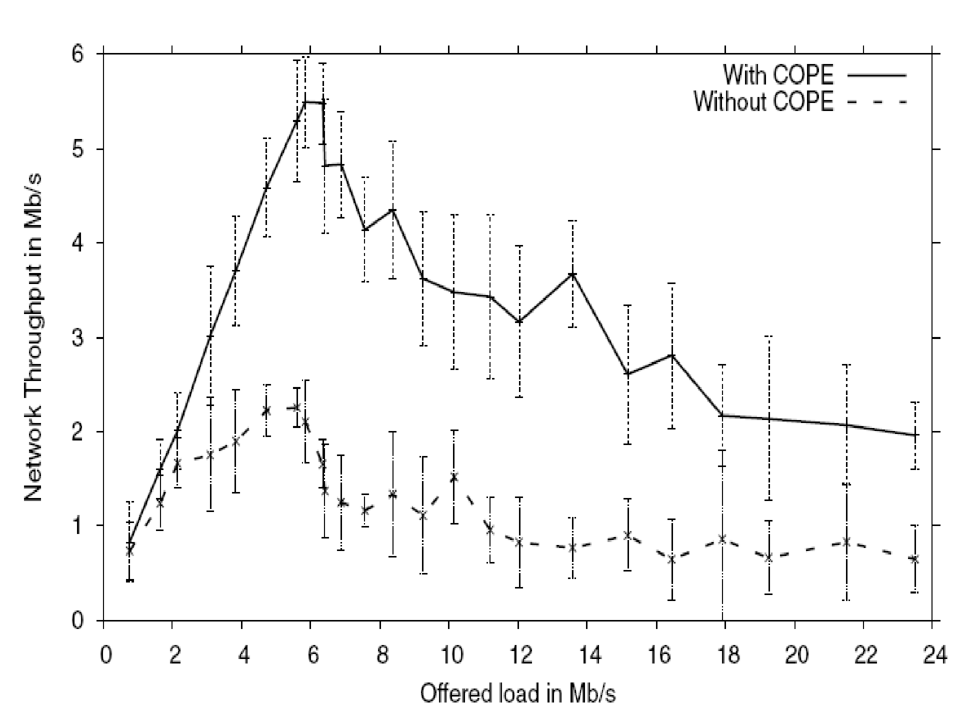


COPE performs well without hidden terminals!

COPE in an Ad Hoc Network

UDP:

Aggregate end-to-end throughput as a function of the demands



Performance: COPE greatly improves the throughput of these wireless networks

COPE in a Mesh Access Network

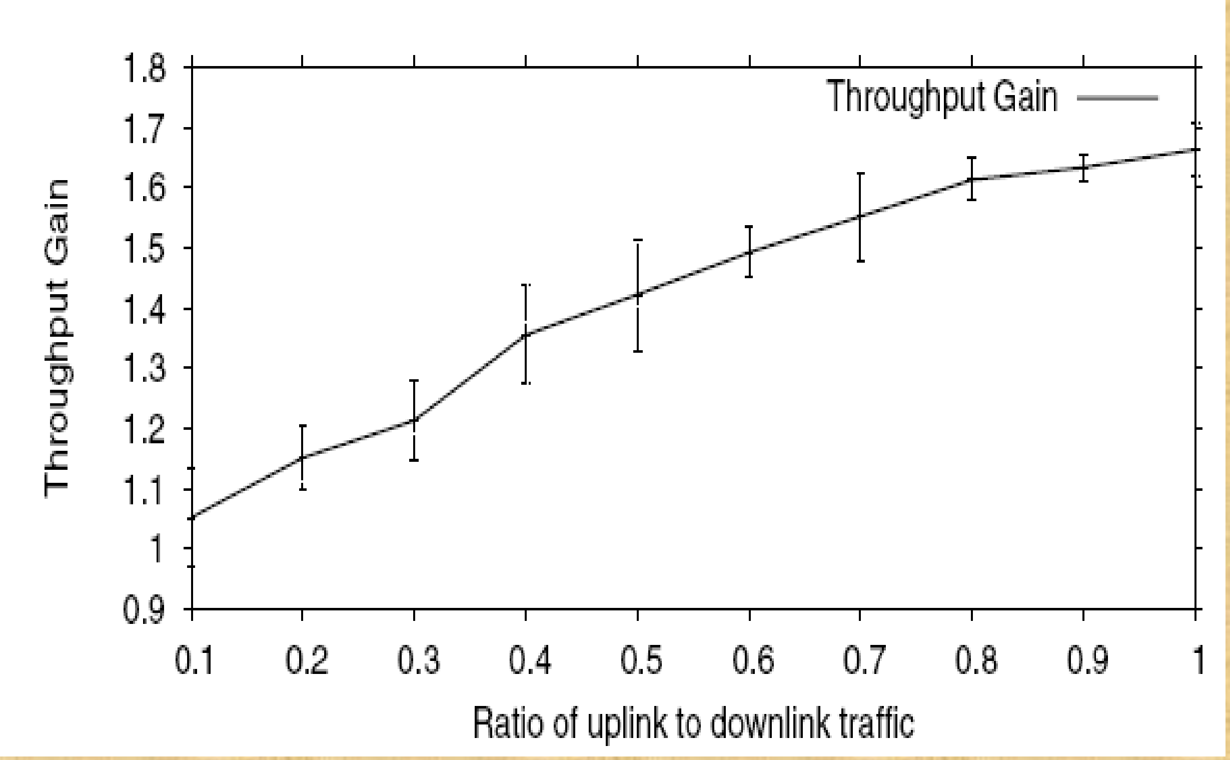
Internet accessing using Multi-hop Wireless Networks that connect to the rest of the Internet via one or more gateways/access points (Traffic flow to and from the closest gateway)

Settings:

UDP flows;
Four sets of nodes;
Each set communicates with the Internet via a specific node that plays the role of a gateway;

COPE in a Mesh Access Network

Throughput gains as a function of this ratio
of upload traffic to download traffic:



COPE's throughput gain relies on coding opportunities, which depend on the diversity of the packets in the queue of the bottleneck node.

Conclusions

Conclusion

- Findings:
  - Network Coding does have practical benefits
  - When wireless medium is congested and traffic consists of many random UDP flows, COPE increases throughput by 3 – 4 times.
  - For UDP, COPE's gain exceeds theoretical coding gain.
  - For a mesh access network, throughput improvement with COPE ranges from 5% - 70%
  - COPE does not work well with hidden terminals. Without hidden terminals, TCP's throughput increases by an average of 38%
  - Network Coding is useful for throughput improvement, but COPE introduces coding as a practical tool that can be integrated with forwarding, routing and reliable delivery.

Conclusion

- COPE: a new architecture to wireless networks
- Large throughput increase
- First implement network coding to wireless networks
- Simple and practical

Problems

- No experiments with mixed flows (Briefly mentioned)
- Other routing protocols?
- Almost no gain due to hidden terminal

WPI

Thank You

Questions?