15-744: Computer Networking

L-12 Wireless Broadcast

Taking Advantage of Broadcast
• Opportunistic forwarding
• Network coding
• Assigned reading
  • XORs In The Air: Practical Wireless Network Coding
  • ExOR: Opportunistic Multi-Hop Routing for Wireless Networks

Outline
• Opportunistic forwarding (ExOR)
• Network coding (COPE)
• Combining the two (MORE)

Initial Approach: Traditional Routing
• Identify a route, forward over links
• Abstract radio to look like a wired link
**Radios Aren’t Wires**

• Every packet is broadcast
• Reception is probabilistic

---

**Exploiting Probabilistic Broadcast**

• Decide who forwards after reception
• Goal: only closest receiver should forward
• Challenge: agree efficiently and avoid duplicate transmissions

---

**Why ExOR Might Increase Throughput**

• Best traditional route over 50% hops: 3/(1/0.5) = 6 tx
• Throughput ≅ 1/# transmissions
• ExOR exploits lucky long receptions: 4 transmissions
• Assumes probability falls off gradually with distance

---

**Why ExOR Might Increase Throughput**

• Traditional routing: 1/(0.25) + 1 = 5 tx
• ExOR: 1/(1 – (1 – 0.25)^4) + 1 = 2.5 transmissions
• Assumes independent losses
**ExOR Batching**

- Challenge: finding the closest node to have rx’d
- Send batches of packets for efficiency
- Node closest to the dst sends first
  - Other nodes listen, send remaining packets in turn
- Repeat schedule until dst has whole batch

**Reliable Summaries**

- Repeat summaries in every data packet
- Cumulative: what all previous nodes rx’d
- This is a gossip mechanism for summaries

---

**Priority Ordering**

- Goal: nodes “closest” to the destination send first
- Sort by ETX metric to dst
  - Nodes periodically flood ETX “link state” measurements
  - Path ETX is weighted shortest path (Dijkstra’s algorithm)
- Source sorts, includes list in ExOR header

---

**Using ExOR with TCP**

- Batching requires more packets than typical TCP window
Summary
- ExOR achieves 2x throughput improvement
- ExOR implemented on Roofnet
- Exploits radio properties, instead of hiding them

Outline
- Opportunistic forwarding (ExOR)
- Network coding (COPE)
- Combining the two (MORE)

Background
- Famous butterfly example:
  - All links can send one message per unit of time
  - Coding increases overall throughput

Background
- Bob and Alice
  - Require 4 transmissions
Background

- Bob and Alice

Coding Gain

- Coding gain = 4/3

Throughput Improvement

- UDP throughput improvement ~ a factor 2 > 4/3 coding gain

Coding Gain: more examples

With opportunistic listening, coding gain=2N/(1+N) \rightarrow 2.
With opportunistic listening, coding gain + MAC gain \rightarrow \infty
COPE (Coding Opportunistically)

- Overhear neighbors’ transmissions
- Store these packets in a Packet Pool for a short time
- Report the packet pool info. to neighbors
- Determine what packets to code based on the info.
- Send encoded packets

Opportunistic Coding

Packet Coding Algorithm

- When to send?
  - Option 1: delay packets till enough packets to code with
  - Option 2: never delaying packets -- when there’s a transmission opportunity, send packet right away
- Which packets to use for XOR?
  - Prefer XOR-ing packets of similar lengths
  - Never code together packets headed to the same next hop
  - Limit packet re-ordering
  - XORing a packet as long as all its nexthops can decode it with a high enough probability

Packet Decoding

- Where to decode?
  - Decode at each intermediate hop
- How to decode?
  - Upon receiving a packet encoded with n native packets
    - find n-1 native packets from its queue
    - XOR these n-1 native packets with the received packet to extract the new packet
Prevent Packet Reordering

- Packet reordering due to async acks degrade TCP performance

- Ordering agent
  - Deliver in-sequence packets immediately
  - Order the packets until the gap in seq. no is filled or timer expires

Summary of Results

- Improve UDP throughput by a factor of 3-4

- Improve TCP by
  - wo/ hidden terminal: up to 38% improvement
  - w/ hidden terminal and high loss: little improvement

- Improvement is largest when uplink to downlink has similar traffic

- Interesting follow-on work using analog coding

Reasons for Lower Improvement in TCP

- COPE introduces packet re-ordering
- Router queue is small ➔ smaller coding opportunity
  - TCP congestion window does not sufficiently open up due to wireless losses
- TCP doesn't provide fair allocation across different flows

Outline

- Opportunistic forwarding (ExOR)
- Network coding (COPE)
- Combining the two (MORE)
Use Opportunistic Routing

Opportunistic routing promises large increase in throughput

ExOR

- State-of-the-art opp. routing, ExOR imposes a global scheduler:
- Requires full coordination; every node must know who received what
- Only one node transmits at a time, others listen

But

- Overlap in received packets → Routers forward duplicates

Global Scheduling?

- Global coordination is too hard
- One transmitter
Global Scheduling?

Does opportunistic routing have to be so complicated?

MORE (Sigcomm07)
- Opportunistic routing with no global scheduler and no coordination
- We use random network coding
- Experiments show that randomness outperforms both current routing and ExOR

Go Random
Each router forwards random combinations of packets
- Randomness prevents duplicates
- No scheduler; No coordination
- Simple and exploits spatial reuse

Random Coding Benefits Multicast
Without coding → source retransmits all 4 packets
Random Coding Benefits Multicast

Random combinations

Random coding is more efficient than global coordination

MORE

- Source sends packets in batches
- Forwarders keep all heard packets in a buffer
- Nodes transmit linear combinations of buffered packets

Can compute linear combinations and sustain high throughput!

MORE

- Source sends packets in batches
- Forwarders keep all heard packets in a buffer
- Nodes transmit linear combinations of buffered packets
- Destination decodes once it receives enough combinations
  - Say batch is 3 packets

  1 \( P_1 + 3 \)
  2 \( P_2 + 2 \)
  3 \( P_3 + 6 \)

  4 \( P_1 + 5 \)
  5 \( P_2 + 5 \)
  6 \( P_3 + 10 \)

- Destination acks batch, and source moves to next batch