Extreme Opportunistic Routing (ExOR)

- Basic idea: avoid duplicates by scheduling

- Instead of choosing a fix sequential path (e.g., src->B->D->dst), the source chooses a list of forwarders (a forwarder list in the packets) using ETX-like metric
  - a background process collects ETX information via periodic link-state flooding

- Forwarders are prioritized by ETX-like metric to the destination
ExOR: Forwarding

• Group packets into batches

• The highest priority forwarder transmits when the batch ends

• The remaining forwarders transmit in prioritized order
  – each forwarder forwards packets it receives yet not received by higher priority forwarders
  – status collected by batch map
Batch Map

• Batch map indicates, for each packet in a batch, the highest-priority node known to have received a copy of that packet
ExOR: Example

2nd round Tx: 3, 6
Batch map: 13032012

1st round Tx: 1, 2, 3, 4, 5, 6, 7, 8

Rx: 1, 2, 7, 8  Tx: 1, 7
Batch map: 13032012

N0
N1

Rx: 2, 5, 8  Tx: 5, 8
Batch map: 03032002

N2

Forwarder list:
N3(dst), N2, N1, N0 (src)

Rx: 2, 4  Tx: batch map only
Batch map: 03030000

N3
ExOR: Stopping Rule

• A node stops sending the remaining packets in the batch if its batch map indicates over 90% of this batch has been received by higher priority nodes
  – the remaining packets transferred with traditional routing
Evaluations

- 65 Node pairs
- 1.0MByte file transfer
- 1 Mbit/s 802.11 bit rate
- 1 KByte packets
- EXOR batch size 100
Evaluation: 2x Overall Improvement

- Median throughputs: 240 Kbits/sec for ExOR,
  121 Kbits/sec for Traditional
OR uses links in parallel

Traditional Routing
3 forwarders
4 links

ExOR
7 forwarders
18 links
OR moves packets farther

- ExOR average: 422 meters/transmission
- Traditional Routing average: 205 meters/tx
Comments: ExOR

• Pros
  – takes advantage of link diversity (the probabilistic reception) to increase the throughput
  – does not require changes in the MAC layer
  – can cope well with unreliable wireless medium

• Cons
  – scheduling is hard to scale in large networks
  – overhead in packet header (batch info)
  – batches increase delay
Outline

• Non-traditional routing
  – motivation
  – network coding: exploiting network broadcast
  – opportunistic routing
    • ExOR
    • MORE
MORE: MAC-independent
Opportunistic Routing & Encoding

• Basic idea:
  – Replace node coordination with network coding
  – Trading structured scheduler for random packets combination

• Previous network coding technique is for inter-flow
• MORE is for intra-flow network coding
Basic Idea: Source

• Chooses a list of forwarders (e.g., using ETX)
• Breaks up file into K packets (p1, p2, ..., pK)
• Generate random packets

\[ p_j' = \sum c_{ji} p_i \]

• MORE header includes the code vector [c_{j1}, c_{j2}, ..., c_{jK}] for coded packet p_j'}
Basic Idea: Source

```
src -- 10% -- A -- 100% -- dst
<table>
<thead>
<tr>
<th></th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>10%</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
src -- 10% -- B -- 100% -- dst
<table>
<thead>
<tr>
<th></th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
src -- 10% -- C -- 100% -- dst
<table>
<thead>
<tr>
<th></th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
src -- 10% -- D -- 100% -- dst
|          | 100%         |
```

```
Basic Idea: Forwarder

- Check if in the list of forwarders
- Check if linearly independent of new packet with existing packet
- Re-coding and forward
Basic Idea: Destination

• Decode

\[
\begin{pmatrix}
p_1 \\
\vdots \\
p_K \\
\end{pmatrix}
= \begin{pmatrix}
c_{11} & \ldots & c_{1K} \\
\vdots & \ddots & \vdots \\
c_{K1} & \ldots & c_{KK} \\
\end{pmatrix}^{-1}
\begin{pmatrix}
p'_1 \\
\vdots \\
p'_K \\
\end{pmatrix}
\]

• Send ACK back to src if success
Key Practical Question: How many packets does a forwarder send?

• Compute $z_i$: the expected number of times that forwarder $i$ should forward each packet
Computes $z_s$

$\varepsilon_{ij}$: loss probability of the link between $i$ and $j$

Compute $z_s$ so that at least one forwarder that is closer to destination is expected to have received the packet:

$$z_s = \frac{1}{(1-\prod_j \varepsilon_{sj})}$$
Compute $z_j$ for forwarder $j$

- Only need to forward packets that are
  - received by $j$
  - sent by forwarders who are further from destination
  - not received by any forwarder who is closer to destination

$$L_j = \sum_{i \text{ is further}} \left[ z_i (1 - \epsilon_{ij}) \prod_{k \text{ closer to } d} \epsilon_{ik} \right]$$
Compute $z_j$ for forwarder $j$

$$L_j = \sum_{i \text{ is further}} [z_i(1-\varepsilon_{ij}) \prod_{k \text{ closer to } d} \varepsilon_{ik}]$$

- To guarantee at least one forwarder closer to $d$ receives the packet

$$z_j = \frac{L_j}{(1- \prod_{k \text{ closer to } d} \varepsilon_{jk})}$$
• 20 nodes distributed in a indoor building
• Path between nodes are 1 ~ 5 hops in length
• Loss rate is 0% ~ 60%; average 27%
Figure 6—Unicast Throughput. Figure shows the CDF of the unicast throughput achieved with MORE, ExOR, and Srcr. MORE’s median throughput is 22% higher than ExOR. In comparison to Srcr, MORE achieves a median throughput gain of 95%, while some source-destination pairs show as much as 10-12x.
Problem of MORE?
Mesh Networks API So Far

<table>
<thead>
<tr>
<th>Network</th>
<th>Forward correct packets to destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHY/LL</td>
<td>Deliver correct packets</td>
</tr>
</tbody>
</table>
Motivation

570 bytes; 1 bit in 1000 incorrect  →  Packet loss of 99%
Implication

Opportunistic Routing → 50 transmissions
Outline

- Non-traditional routing
  - motivation
  - network coding: exploiting network broadcast
  - opportunistic routing
    - ExOR
    - MORE
    - MIXIT
## New API

<table>
<thead>
<tr>
<th>PHY + LL</th>
<th>Deliver <strong>correct symbols</strong> to higher layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Forward <strong>correct symbols</strong> to destination</td>
</tr>
</tbody>
</table>
What Should Each Router Forward?
What Should Each Router Forward?

1) Forward everything ➔ Inefficient
2) Coordinate ➔ Unscalable
Symbol Level Network Coding

Forward random combinations of correct symbols
Symbol Level Network Coding

Routers create random combinations of correct symbols
Symbol Level Network Coding

Destination decodes by solving linear equations
Symbol Level Network Coding

Routers create random combinations of correct symbols
Symbol Level Network Coding

Destination decodes by solving linear equations
Destination needs to know which combinations it received

Use run length encoding

Original Packets  Coded Packet
Destination needs to know which combinations it received

Use run length encoding

Original Packets

Coded Packet
Destination needs to know which combinations it received
Use run length encoding

Original Packets

Coded Packet
Destination needs to know which combinations it received

Use run length encoding

Original Packets

Coded Packet
Destination needs to know which combinations it received

Use run length encoding
Symbol-level Network Coding

Forward random combinations of correct symbols

Original Packets  Coded Packet

R1
Symbol-level Network Coding

Forward random combinations of correct symbols

Original Packets  Coded Packet
Symbol-level Network Coding
Forward random combinations of correct symbols

Original Packets

Coded Packet
Symbol-level Network Coding

Forward random combinations of correct symbols

Original Packets

Coded Packet
Evaluation

- Implementation on GNURadio SDR and USRP
- Zigbee (IEEE 802.15.4) link layer
- 25 node indoor testbed, random flows

Compared to:
1. Shortest path routing based on ETX
2. MORE: Packet-level opportunistic routing
Throughput Comparison

Throughput (Kbps) vs. CDF

- **MIXIT**
- **MORE**
- **Shortest Path**

Throughput Comparison:
- **MIXIT** is 3x faster than **Shortest Path**.
- **MORE** is 2.1 times faster than **Shortest Path**.