DISTRIBUTED WIRELESS BROADCAST PROTOCOLS WITH NETWORK CODING FOR SINGLE/MULTIPLE SOURCES

Cedric Adjih, Ichrak Amdouni, Oliver Hahm (Inria)
Claudio Greco, Michel Kieffer (CentraleSupelec)
IPR Statement

- No IPR from our side
- From IPR disclosures at IETF, look for:

<table>
<thead>
<tr>
<th>IETF IPR Disclosure</th>
<th>Patent</th>
<th>This presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID #2183</td>
<td>“Randomized distributed network coding”, US 7706365</td>
<td>Random linear [re]coding (slide 4, and following)</td>
</tr>
</tbody>
</table>
Wireless Multihop Broadcast

- Broadcast:
  - Single source: many packets to entire network
  - Multiple sources: one packet to entire network

- Use cases in Wireless Sensor Networks:
  - “OTA” (over-the-air programming)
  - Data collection with “unmanaged” network

- Without NC: SMF (RFC 6221), Trickle (RFC 6206)
**Fully Distributed Protocols**

- **Fully Distributed Broadcast Protocol:**
  - No knowledge of the entire network (sources/dest.)

- **Ex: protocol DRAGONCAST/DragonNet**

- Every node retransmits coded payloads at a given packet rate per second (e.g. with random linear coding)

- Coded payloads are maintained in a (decoding) set

- Control plane: state piggybacked on coded payloads
  - Decoded payloads, number of neighbors, ex: …

- **This presentation:** sliding encoding window (single source), encoding vectors (multiple source)
Use Case 2: NC Transport, In-Network Coding,

- Assisted by multi-path (subgraph) routing
  - Usage: reliability, resilience to link and node outage.
  - Supports both Unicast and Multicast

Single source: Sliding Window

- **Sliding Encoding Window (SEW in DRAGONCAST):**
  - Each node transmits decoding state: “first undecoded”
  - Node generates packets considering neighbor state

Node A

- Simple functioning
Coding Interval-based Sliding Encoding Window


Redesign of SEW, aware of:
- Heterogeneous decoding rate at nodes
- Introduce « losses »
- Limited buffer size (overflow)
  - Choice between throwing decoded or undecoded packets
  - Combinations may become useless: \( P_{11} + \ldots \) if \( P_{11} \) dropped

Encoding strategy:
- Fit at best neighbors needs in terms of payloads
- Needs more information about the state of neighbors
  - state advertisement from signaling
CISEW: Finer Signaling

- **SEW signaling = state:**
  - Decoded
  - Being Decoded
  - Unseen

- **CISEW state:** each original payload is one of:
  - Decoded (or « lost ») but no longer available
  - Decoded and available
  - Not yet decoded but received in one/some linear combinations
  - Not yet decoded but never received

Diagram:
- **Unwanted:** $P_0$
- **Decoded:** $P_3, P_4, ..., P_7, P_8, ..., P_3, P_{23}, ...$
- **Undecoded:** $P_0, P_4, ..., P_7, P_8, ..., P_3, P_{23}, ...$
- **Decoded:** $P_3, P_4, ..., P_7, P_8, ..., P_3, P_{23}, ...$
- **Unseen:** $P_0, P_4, ..., P_7, P_8, ..., P_3, P_{23}, ...$
4 types of index intervals

- **Black**: unwanted indices (e.g. payloads that are no longer buffered)
- **Grey**: indices that the node is not interested in, but would not harm decoding
- **Gold**: indices that the node is interested in, in the near future
- **White**: indices that the node is interested in

*How to set these intervals: it is a **POLICY***
Question 1: How to set intervals?
Question 2: How to set encoding windows?

- No universal answer: flexibility
Multiple sources (inter-flow NC)

- Multiple sources in full distributed network:
  - How to index payloads? $7P_1 + 3P_2 + P_3 + 2P_4$: whose $P_1$?
    - Classical encoding vector (1 source)
    - Cope-style (complemented with packet ID)

- NeCoRPIA

- Just choose a Random Payload Index

- Problem:
  - “payload index collisions”
Multiple sources (inter-flow NC)

- **Ideas:**
  - Use knowledge of payload content to guide resolution
    - Crowdsensing application: time and position
  - Hash on content: mechanism to check decoded
  - Gaussian Elimination on the full packets

- **NeCoRPIA (ICC’15):**
  - constraint satisfaction problem defined over a finite field
  - Relaxation, in a sequence linear of linear programs
NeCoRPIA-lite

- Implementation (GF(2))
  - OS: RIOT [http://riot-os.org/]
  - Testbed, w/ WSN nodes
  - FIT IOT-lab [https://iot-lab.info/]

- Example with 10 nodes

```
01100000 10100000 a832659e9c6d7d01830c40
01100011110000 6cd02e2fa0fadab3892a20
00000001 11000000 5962010000000000000000
01100010000000 823d010000000000000000
01000010000000 9e1e000000100000000000
01000001110000 4e38000000100000000000
0000001110100000 c9c700000010000000000
01000010000000 48c300000010000000000
01000100100000 72cf000000100000000000
01100011100000 7e83000000100000000000
```

```plaintext
01000000000000 e2cd5a24f4f8e4bac2a579
01000000000000 3410eb0434ee9573d4e870
01000000000000 7cd3100000000000000000
01000000000000 54e0000000000000000000
00000000000000 b5b3700000001000000000
00000000000000 ecd1500000001000000000
00000000000000 cb5ca00000010000000000
00000000000000 3ecba00000010000000000
00000000000000 0d25e00000010000000000
00000000000000 a5dca00000010000000000
```
Conclusion

- Elements for NC in wireless multi-hop networks
  - Interest of the RG?
### DragonNet Packet Format

**Figure 4.1:** The format of DragonNet message as specified for WSNs.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Node ID | Sequence Number

#### Dragon Parameters

Flow Parameters

Coded Data

**Figure 4.2:** Dragon Related Parameters.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Rank of the Transmitter | High Index
Low Index | Number of Neighbors
LastTime

**Figure 4.4:** Coded Payload Related Parameters.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Message Interval | Sliding Encoding Window Size
Coded Block size

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>

Coded Payload Data
SEW: Sliding Encoding Window

- Principle: “real-time” robust decoding
- Variant of Gaussian elimination (“inverted” RREF)

\[
\begin{align*}
\mathbf{P}_1 & \rightarrow 1 & 0 & 0 \\
\mathbf{P}_2 & \rightarrow 0 & 1 & 0 \\
\mathbf{P}_3 & \rightarrow 0 & 0 & 1 \\
x\mathbf{P}_4 + y\mathbf{P}_5 + \mathbf{P}_6 & \rightarrow 0 & 0 & 0 \\
u\mathbf{P}_4 + v\mathbf{P}_5 + \mathbf{P}_7 & \rightarrow 0 & 0 & 0 \\
... + a\mathbf{P}_8 + b\mathbf{P}_9 + \mathbf{P}_{10} & \rightarrow 0 & 0 & 0 \\
\end{align*}
\]
CISEW Policy

- Depends on:
  - Computing and storage capacity of nodes:
    - E.g; a huge advertised Gold interval -> too many undecoded packets, no suitable for low capacity sensors
  - Application requirements:
    - E.g1: real time application: nodes decoding should evolve in parallel, nodes should have close intervals. A too late node should increase its interval even if some payloads are not decoded in order not prevent neighbors from progressing
    - E.g2: For code distribution (over the air reflashing in WSNs), all payloads are equally important, a node must still requesting the same undecoded indices if its neighbors are much more progressed.
Protocol Overview

State of node U (DRALIB)

“Decoding Information Base”
• $Q_1 = P_1 + 2P_2 + 4P_3$
• $Q_2 = P_1 + 2P_2 + P_3$
• $Q_3 = P_2 + P_3$
(rank=3)

[...]

State of node V (DRALIB)

“Decoding Information Base”
• $Q_k = P_1 + 3P_2 + 5P_4$

Neighbor Information Set
[...]

Node U:
rank = 3, #neigh = 3

Coded Packet (w/ DRAS)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Known neigh.</th>
<th>Encoding vector</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>$(1,3,5,0,...)$</td>
<td>$[P_1 + 3P_2 + 5P_4]$</td>
</tr>
</tbody>
</table>