Toward Efficient Many-to-Many Broadcast in Dynamic Wireless Networks

Fabian Mager, Carsten Herrmann, Marco Zimmerling
TU Dresden, Germany
Why Many-to-Many?
Why Many-to-Many?
Why Many-to-Many?
Why Many-to-Many?
Requirements
Requirements

• Dynamic multi-hop networks
Requirements

- Dynamic multi-hop networks
- Low latency, high reliability

Closed-loop control: 10 – 500 ms [1]

Requirements

• Dynamic multi-hop networks
• Low latency, high reliability
• Efficiency (energy, costs, etc.)
Current Solutions
## Current Solutions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Current Solutions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
</tbody>
</table>

## Current Solutions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-hop / Mesh</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Latency</td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>Dynamic</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

## Current Solutions

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-hop / Mesh</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Latency</td>
<td>high</td>
<td>medium</td>
<td>low</td>
</tr>
<tr>
<td>Dynamic</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>medium</td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>

## Our Contribution

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-hop / Mesh</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Latency</td>
<td>high</td>
<td>medium</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Dynamic</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Energy Efficiency</td>
<td>medium</td>
<td>high</td>
<td>low</td>
<td>high</td>
</tr>
</tbody>
</table>

Approach
Example: All-to-All Communication

• Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

• Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

• Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

• Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another
Our Key Ideas

• Overlay floods:
  Let nodes send combinations of previously received packets, built with **random linear network coding**
Our Key Ideas

• Overlay floods:
  Let nodes send combinations of previously received packets, built with *random linear network coding*
Our Key Ideas

• Overlay floods:
  Let nodes send combinations of previously received packets, built with random linear network coding
Our Key Ideas

• Overlay floods:
  Let nodes send combinations of previously received packets, built with random linear network coding

• Enable spatial reuse:
  Let multiple nodes transmit simultaneously and exploit the capture effect
Our Key Ideas

• Overlay floods:
  Let nodes send combinations of previously received packets, built with random linear network coding

• Enable spatial reuse:
  Let multiple nodes transmit simultaneously and exploit the capture effect
Main Challenges
Main Challenges

1. When should a node send?
2. What should a node send?
3. How to ensure synchronous transmissions without a global clock?
4. How to achieve an efficient runtime operation?
Main Challenges

1. When should a node send?
2. What should a node send?
3. How to ensure synchronous transmissions without a global clock?
4. How to achieve an efficient runtime operation?
When Should a Node Send?

• **Capture effect:**
  Correctly receive a packet despite interfering transmitters under physical layer specific conditions
  (e.g. 802.15.4: SINR >= 3dB, $\Delta t < 128$us)
When Should a Node Send?

• Capture effect:
  Correctly receive a packet despite interfering transmitters under physical layer specific conditions
  (e.g. 802.15.4: SINR ≥ 3dB, Δt < 128us)

• Too many → capture unreliable
When Should a Node Send?

• Capture effect:
  Correctly receive a packet despite interfering transmitters under physical layer specific conditions
  (e.g. 802.15.4: SINR >= 3dB, $\Delta t < 128$us)

  • Too many $\rightarrow$ capture unreliable
  • Too few $\rightarrow$ less spatial reuse
When Should a Node Send?

• Capture effect:
  Correctly receive a packet despite interfering transmitters under physical layer specific conditions
  (e.g. 802.15.4: SINR $\geq$ 3dB, $\Delta t < 128$us)

• Too many $\rightarrow$ capture unreliable
• Too few $\rightarrow$ less spatial reuse

• **Adaptive transmission policy**
  Choose transmit probability based on local node density
Main Challenges

1. When should a node send?
2. **What should a node send?**
3. How to ensure synchronous transmissions without a global clock?
4. How to achieve an efficient runtime operation?
What Should a Node Send?

• **Innovative** (linearly independent) packets are stored in a matrix

| ... || packet_1 || ... |
|-----|---------|------|
| ... || packet_2 || ... |
| ... || packet_3 || ... |
| ... || packet_4 || ... |
| ... || ...     || ... |
| ... || packet_n || ... |
What Should a Node Send?

- **Innovative** (linearly independent) packets are stored in a matrix
- Nodes send randomly chosen combinations of stored packets
What Should a Node Send?

• **Innovative** (linearly independent) packets are stored in a matrix

• Nodes send randomly chosen combinations of stored packets

• Several rules to make packets more useful, e.g.:
  • Immediate relay of innovation
  • Boost dissemination of own message

```
... packet_1 ...  
... packet_2 ...  
... packet_3 ...  
... packet_4 ...  
... ... ...  
... packet_n ...  
```
Evaluation
Setup

• Mixer prototype on TelosB
  • 4 MHz, 16 bit, 10 KB RAM
  • Radio: IEEE 802.15.4

• FlockLab testbed, ETH Zurich
  • 27 TelosB nodes

• All-to-all, each node 1 message
Reliability

100%

Mixer delivered all messages in every experiment
Latency

![Graph 1: Latency vs Payload size in slots]

![Graph 2: Latency vs Payload size in seconds]
Latency

Mixer outperforms sequential flooding by up to 3.5x
Conclusion
Conclusion

• Mixer, a many-to-all communication primitive
  • Made for dynamic wireless multi-hop networks
  • Combines synchronous transmissions and network coding
  • Complete spectrum from 1-to-all to all-to-all
  • Any initial message distribution

• Versatile, fast, efficient, reliable