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?

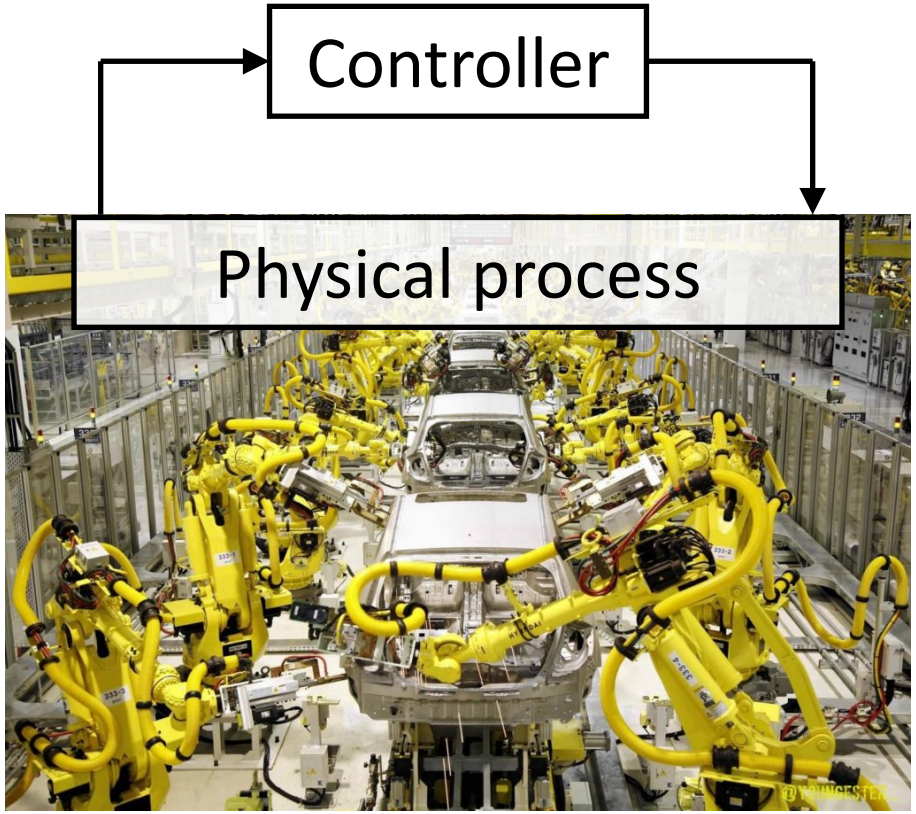
Requirements

Requirements

- **Dynamic multi-hop networks**

Requirements

Closed-loop control: **10 – 500 ms** [1]



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

[1] Akerberg et al., Future research challenges in wireless sensor and actuator networks targeting industrial automation, IEEE INDIN 2011

Requirements

- Dynamic multi-hop networks
- Low latency, high reliability
- **Efficiency (energy, costs, etc.)**

Current Solutions

Current Solutions

	Multi-Sink Routing [1]	Sequential Flooding [2]	Current Practice [3]
Multi-hop / Mesh	yes	yes	no

[1] Mottola et al., MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks, IEEE Transactions on Mobile Computing 2011
[2] Ferrari et al., Efficient network flooding and time synchronization with Glossy, ACM/IEEE IPSN 2011
[3] Preiss et al., Crazyswarm: A large nano-quadcopter swarm. IEEE ICRA 2017

Current Solutions

	Multi-Sink Routing [1]	Sequential Flooding [2]	Current Practice [3]
Multi-hop / Mesh	yes	yes	no
Latency	high	medium	low

[1] Mottola et al., MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks, IEEE Transactions on Mobile Computing 2011
[2] Ferrari et al., Efficient network flooding and time synchronization with Glossy, ACM/IEEE IPSN 2011
[3] Preiss et al., Crazyswarm: A large nano-quadcopter swarm. IEEE ICRA 2017

Current Solutions

	Multi-Sink Routing [1]	Sequential Flooding [2]	Current Practice [3]
Multi-hop / Mesh	yes	yes	no
Latency	high	medium	low
Dynamic	no	yes	yes

[1] Mottola et al., MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks, IEEE Transactions on Mobile Computing 2011
[2] Ferrari et al., Efficient network flooding and time synchronization with Glossy, ACM/IEEE IPSN 2011
[3] Preiss et al., Crazyswarm: A large nano-quadcopter swarm. IEEE ICRA 2017

Current Solutions

	Multi-Sink Routing [1]	Sequential Flooding [2]	Current Practice [3]
Multi-hop / Mesh	yes	yes	no
Latency	high	medium	low
Dynamic	no	yes	yes
Energy Efficiency	medium	high	low

[1] Mottola et al., MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks, IEEE Transactions on Mobile Computing 2011
[2] Ferrari et al., Efficient network flooding and time synchronization with Glossy, ACM/IEEE IPSN 2011
[3] Preiss et al., Crazyswarm: A large nano-quadcopter swarm. IEEE ICRA 2017

Our Contribution

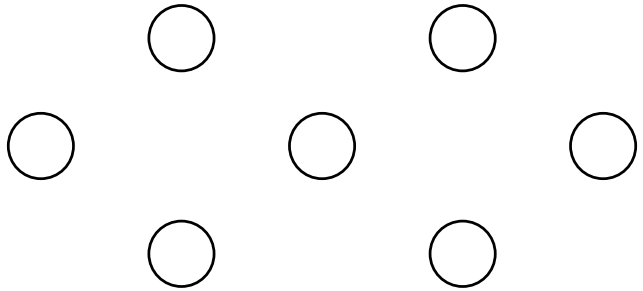
	Multi-Sink Routing [1]	Sequential Flooding [2]	Current Practice [3]	**Mixer**
Multi-hop / Mesh	yes	yes	no	**yes**
Latency	high	medium	low	**low**
Dynamic	no	yes	yes	**yes**
Energy Efficiency	medium	high	low	**high**

[1] Mottola et al., MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks, IEEE Transactions on Mobile Computing 2011
[2] Ferrari et al., Efficient network flooding and time synchronization with Glossy, ACM/IEEE IPSN 2011
[3] Preiss et al., Crazyswarm: A large nano-quadcopter swarm. IEEE ICRA 2017

Approach

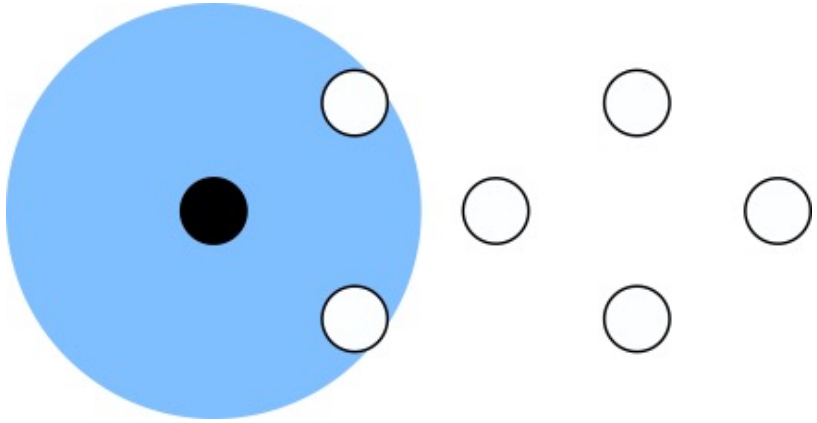
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



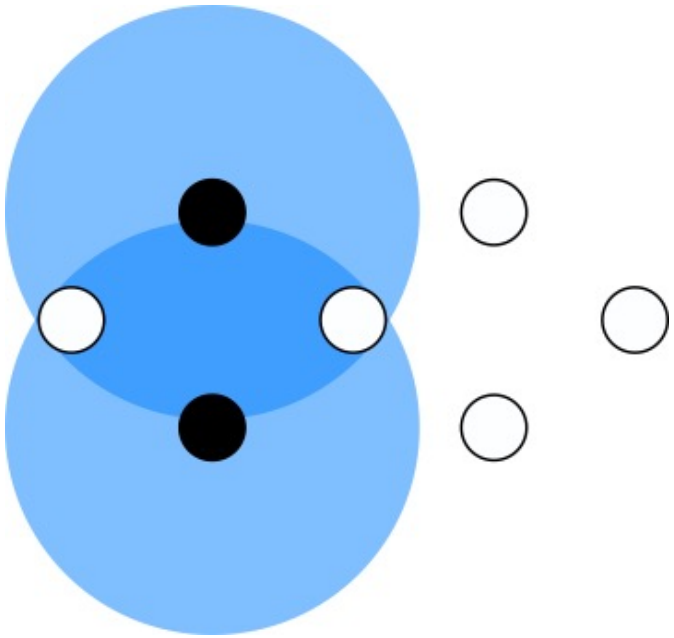
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



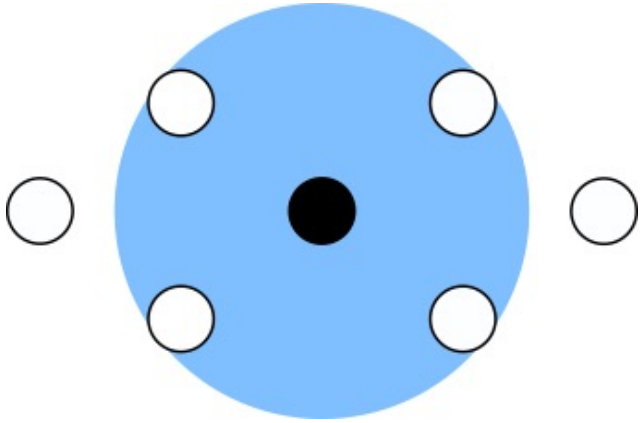
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



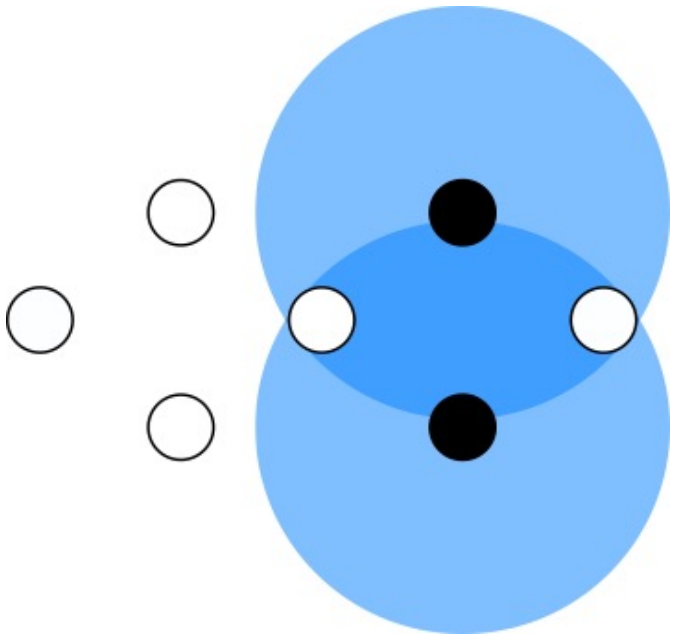
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



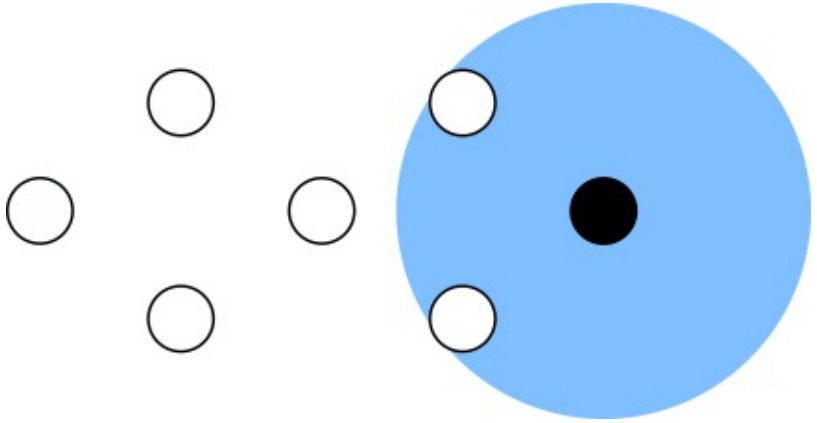
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



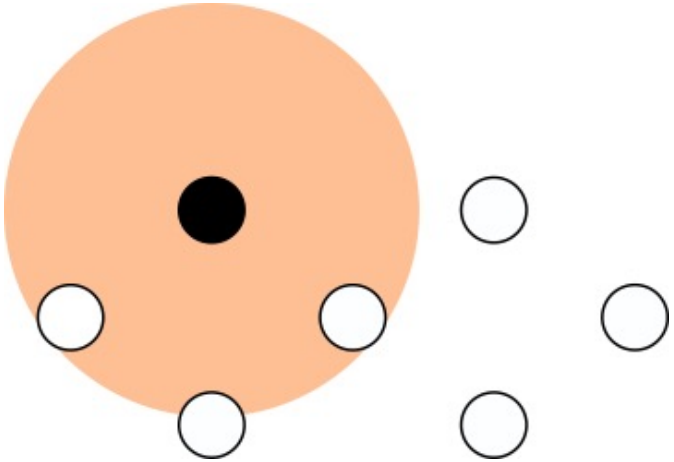
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



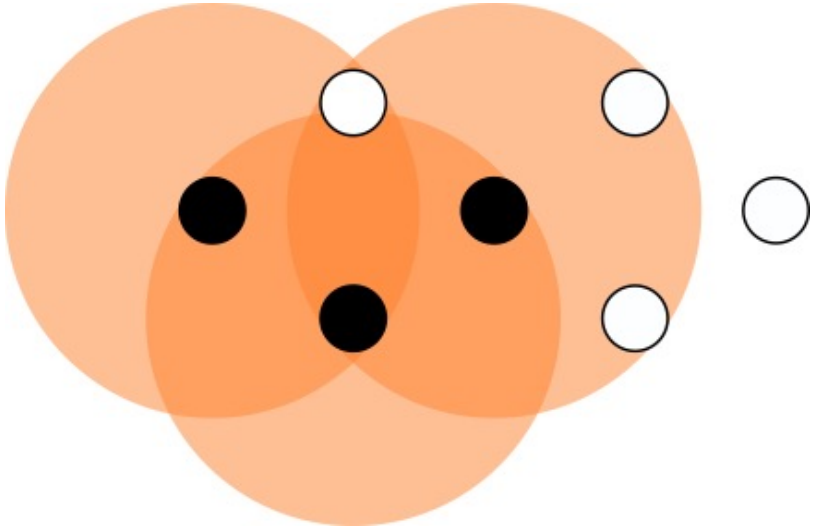
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



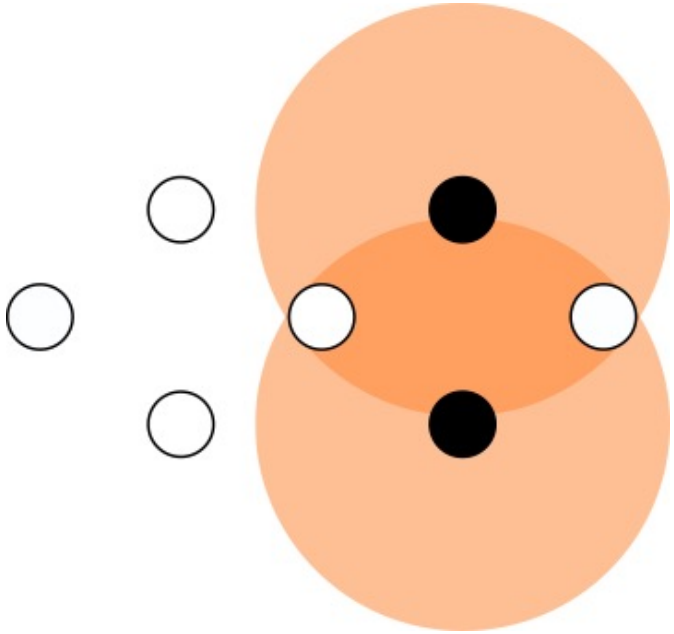
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



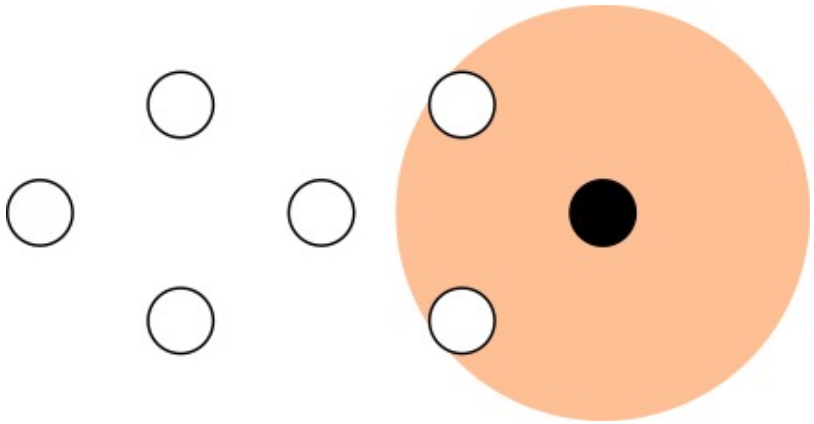
Example: All-to-All Communication

- Using sequential flooding, nodes flood one after another



Example: All-to-All Communication

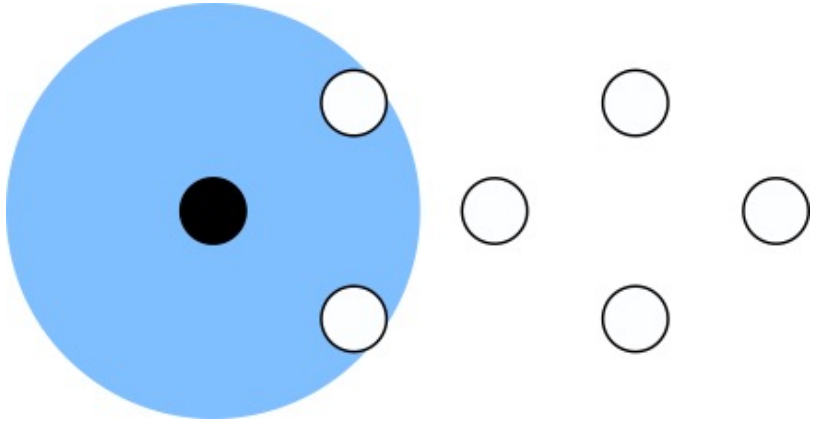
- Using sequential flooding, nodes flood one after another



Our Key Ideas

- Overlay floods:

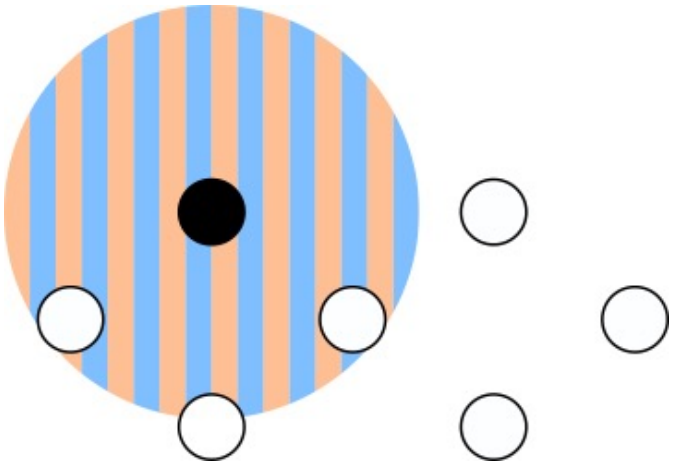
  Let nodes send combinations of previously received packets, built with **random linear network coding**



Our Key Ideas

- Overlay floods:

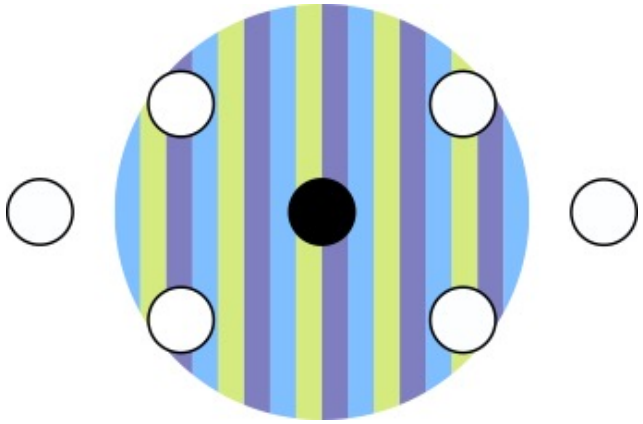
  Let nodes send combinations of previously received packets, built with **random linear network coding**



Our Key Ideas

- Overlay floods:

  Let nodes send combinations of previously received packets, built with **random linear network coding**



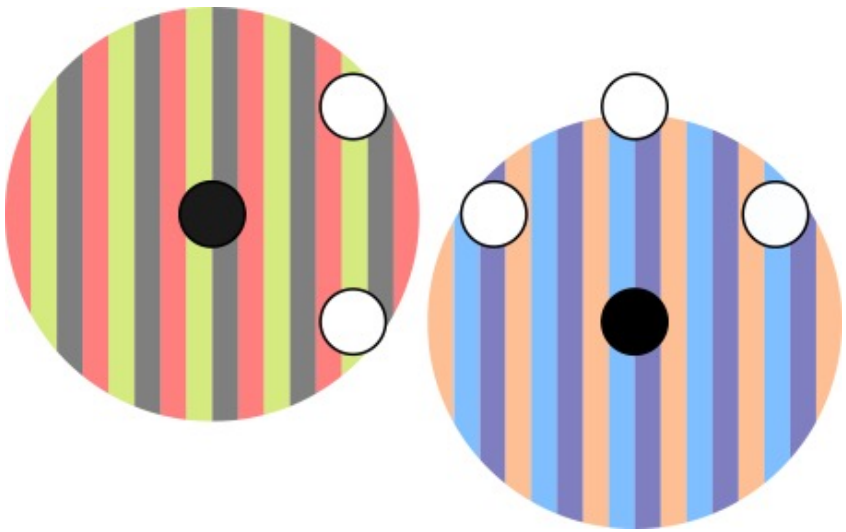
Our Key Ideas

- 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**



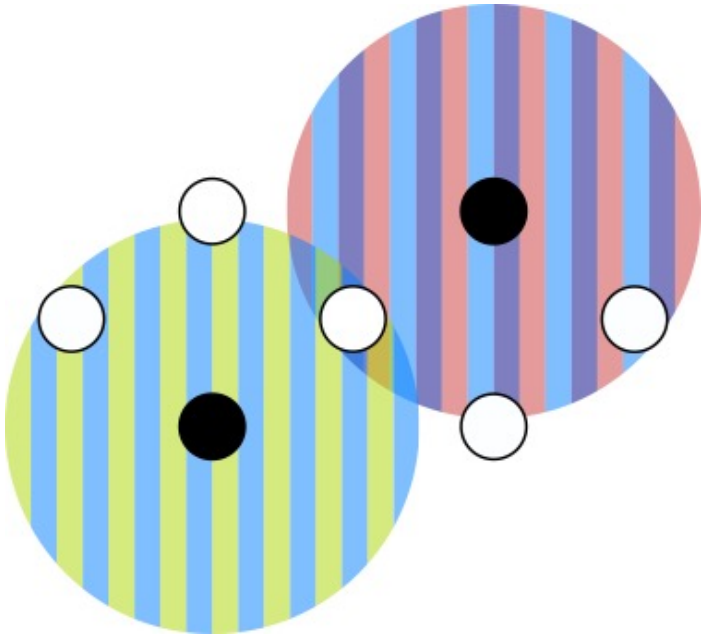
Our Key Ideas

- 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

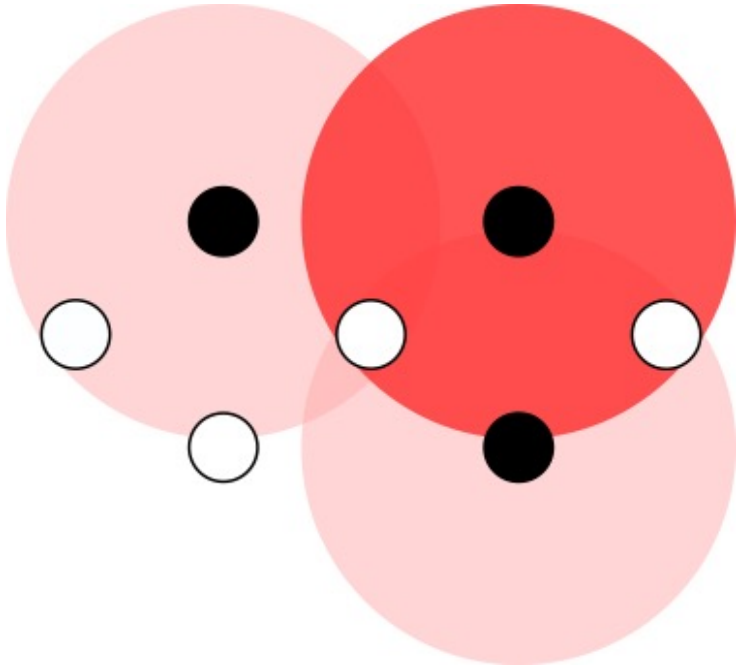
1. **When should a node send?**
2. What should a node send?
3. How to ensure synchronous transmissions without a global clock?
4. How to achieve an efficient runtime operation?

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, Δt < 128us)



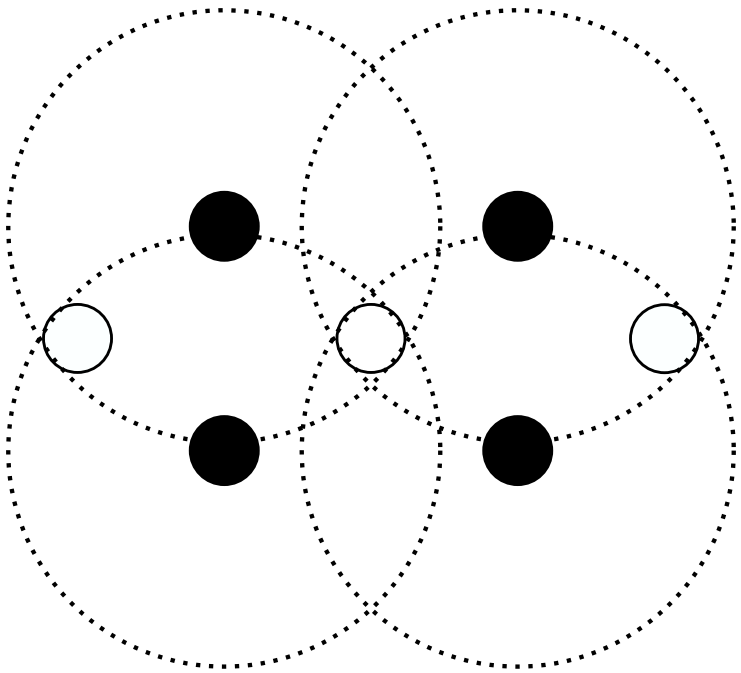
When Should a Node Send?

- 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**



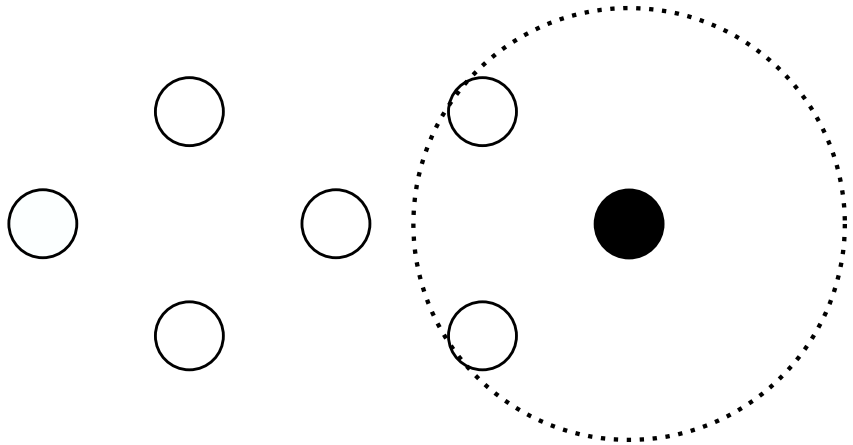
When Should a Node Send?

- Capture effect:

  Correctly receive a packet despite interfering transmitters under physical layer specific conditions

  (e.g. 802.15.4: SINR >= 3dB, Δt < 128us)

- Too many → capture unreliable
- **Too few → less spatial reuse**



When Should a Node Send?

- Capture effect:

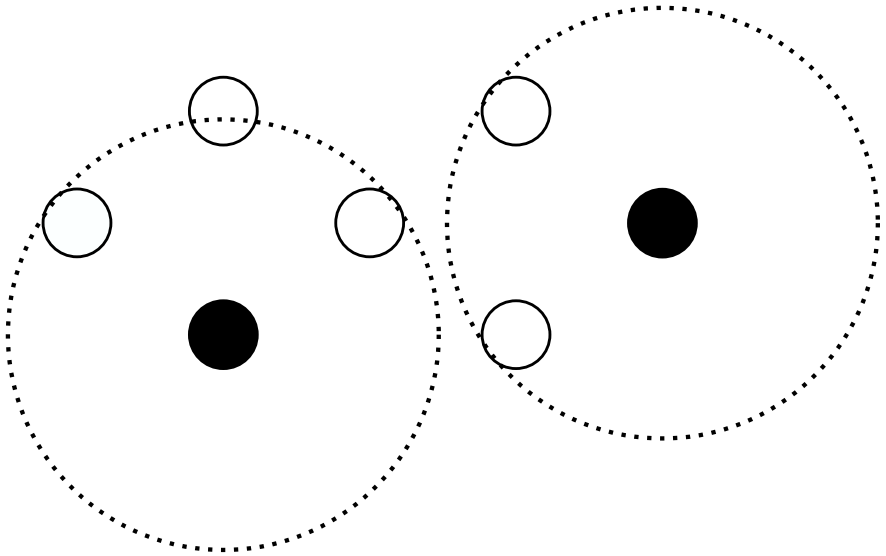
  Correctly receive a packet despite interfering transmitters under physical layer specific conditions

  (e.g. 802.15.4: SINR >= 3dB, Δt < 128us)

- Too many → capture unreliable
- Too few → 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?

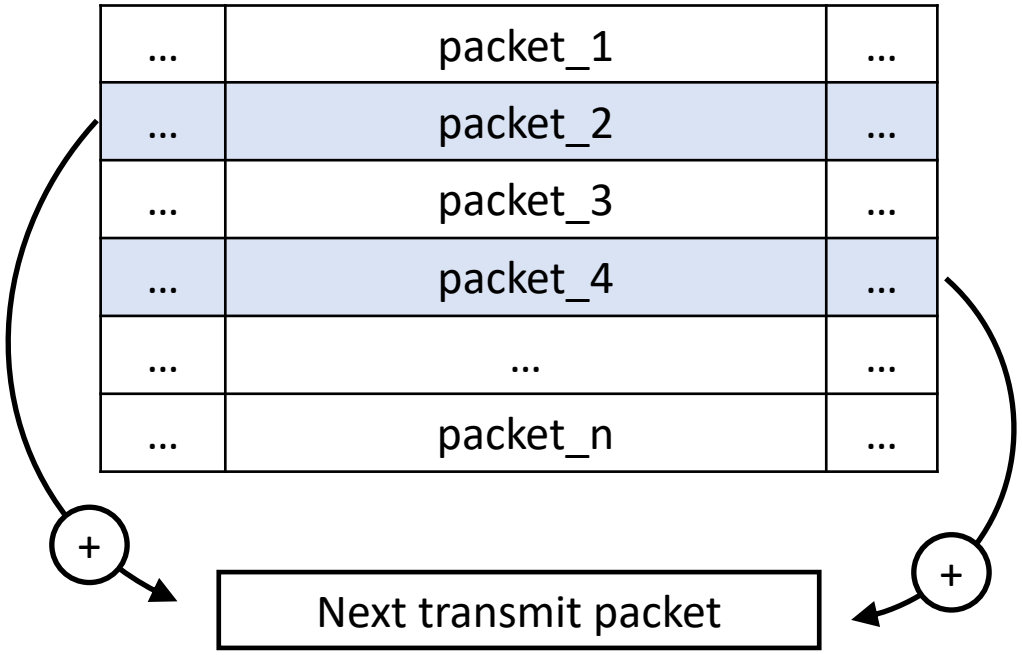
What Should a Node Send?

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

…	packet_1	…
…	packet_2	…
…	packet_3	…
…	packet_4	…
…	…	…
…	packet_n	…

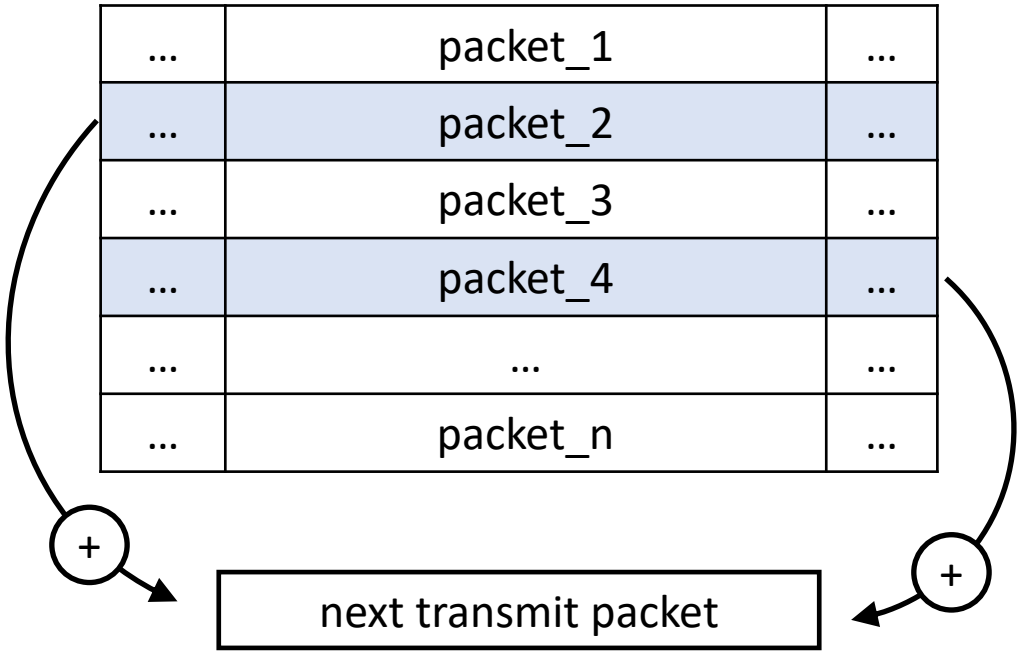
What Should a Node Send?

- **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



Evaluation

Setup

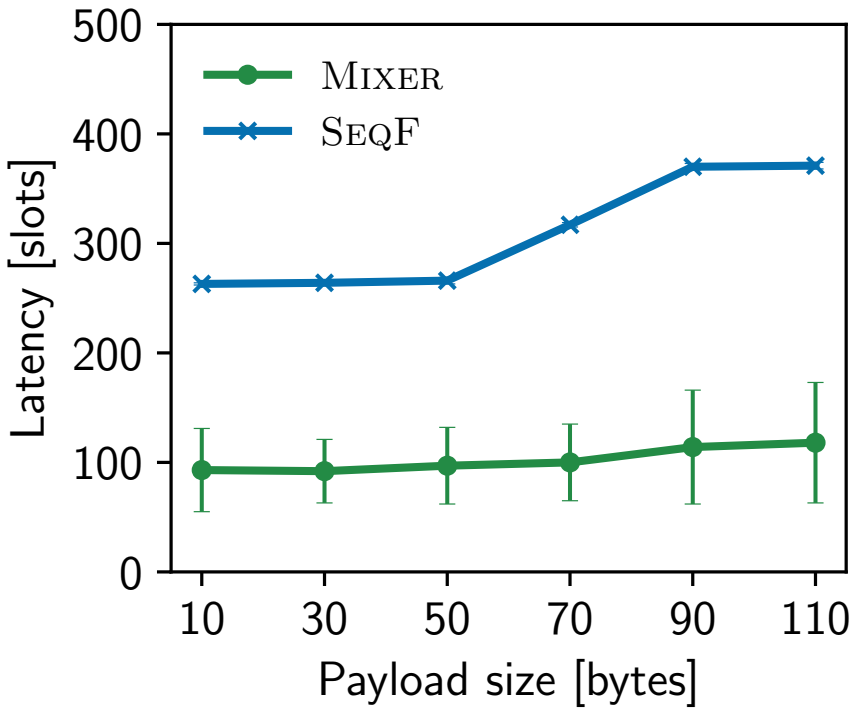
- Mixer prototype on TelosB
  - 4 MHz, 16 bit, 10 KB RAM
  - Radio: IEEE 802.15.4
- FlockLab testbed, ETH Zurich
  - 27 TelosB nodes
- All-to-all, each node 1 message

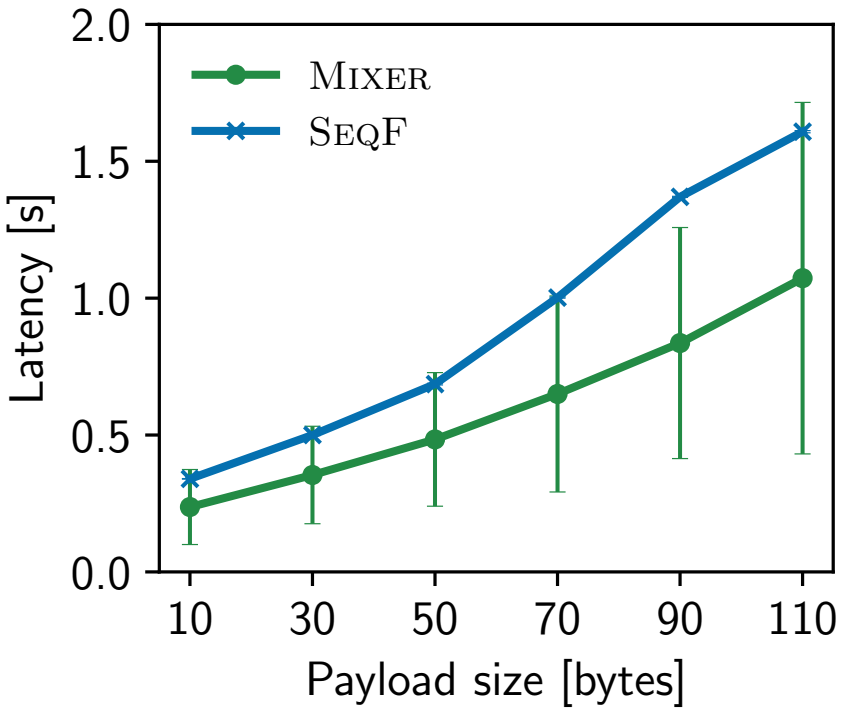
Reliability

100%

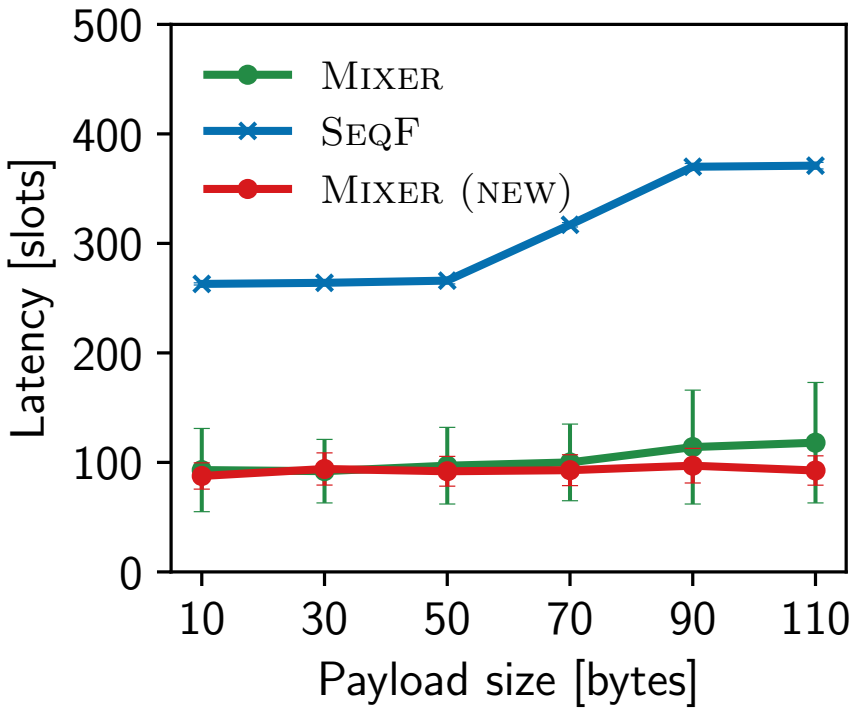
Mixer delivered all messages in every experiment

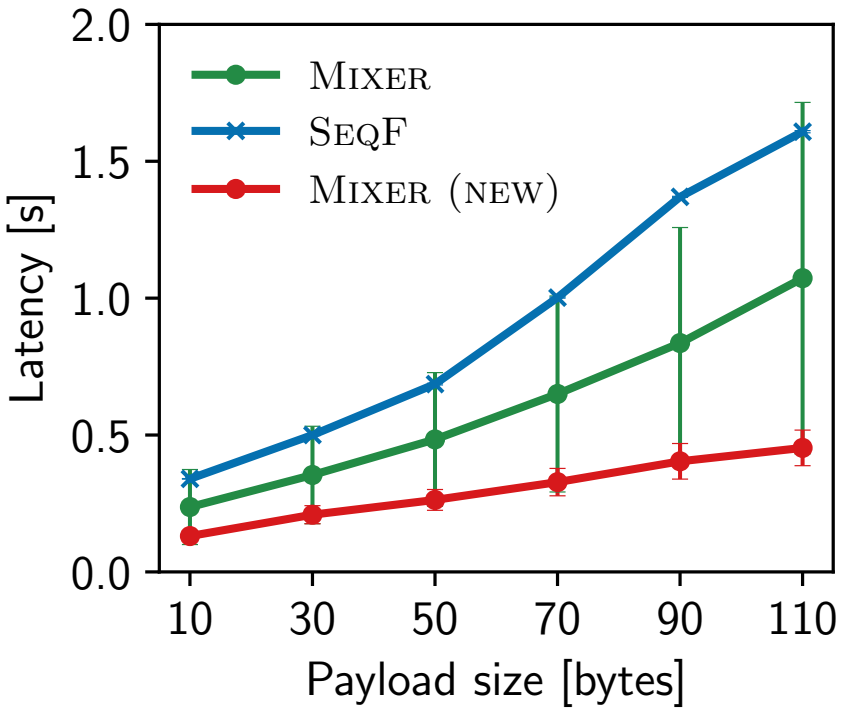
Latency


500
400
300
200
100
0
Latency [slots]
Mixer
SeqF
10
30
50
70
90
110
Payload size [bytes]


2.0
1.5
1.0
0.5
0.0
Latency [s]
Mixer
SeqF
10
30
50
70
90
110
Payload size [bytes]

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**