EXPERIMENTS WITH BROADCAST WITH NETWORK CODING

IETF/IRTF 89
LONDON

Cedric Adjih, Ichrak Amdouni, Hana Baccouch, Antonia Maria Masucci (Inria)

IPR Statement

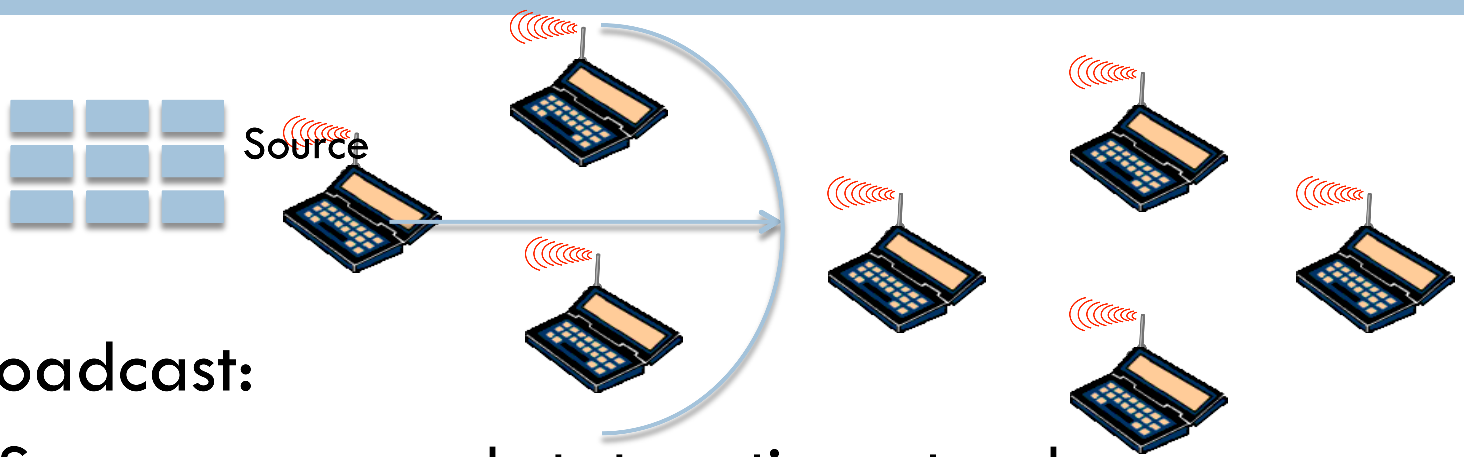
- No IPR from our side
- From IPR disclosures at IETF, some to look for:

IETF IPR Disclosure	Patent	This presentation
ID #2183	"Randomized distributed network coding", US 7706365	Random linear [re]coding (slide 5, and following)
ID #2183	"Feedback-based online network coding", US 8068426	Possibly: feedback for stopping transmitting decoded packets (slide 5, 8, and following)

Objective: evaluation of NC

- Q: What benefits can NC bring ?
  - (funding from French MoD, project ANR-ASTRID/GETRF)
- Steps for evaluation:
  - Focus on a realistic application
  - Propose a reasonable protocol
  - Evaluate it on real testbed (work in progress)
- Performance metrics:
  - Performance (number of transmissions, …)
  - Complexity/simplicity of protocol design, and implementation

Chosen application: broadcast



- Broadcast:
  - Source: many packets to entire network
- Broadcast in Wireless Sensor Networks:
  - Widely deployed wireless multi-hop networks
  - Actual use case: “OTA” (over-the-air programming)
- Advantages of NC: efficiency, natural robustness, simplified control plane

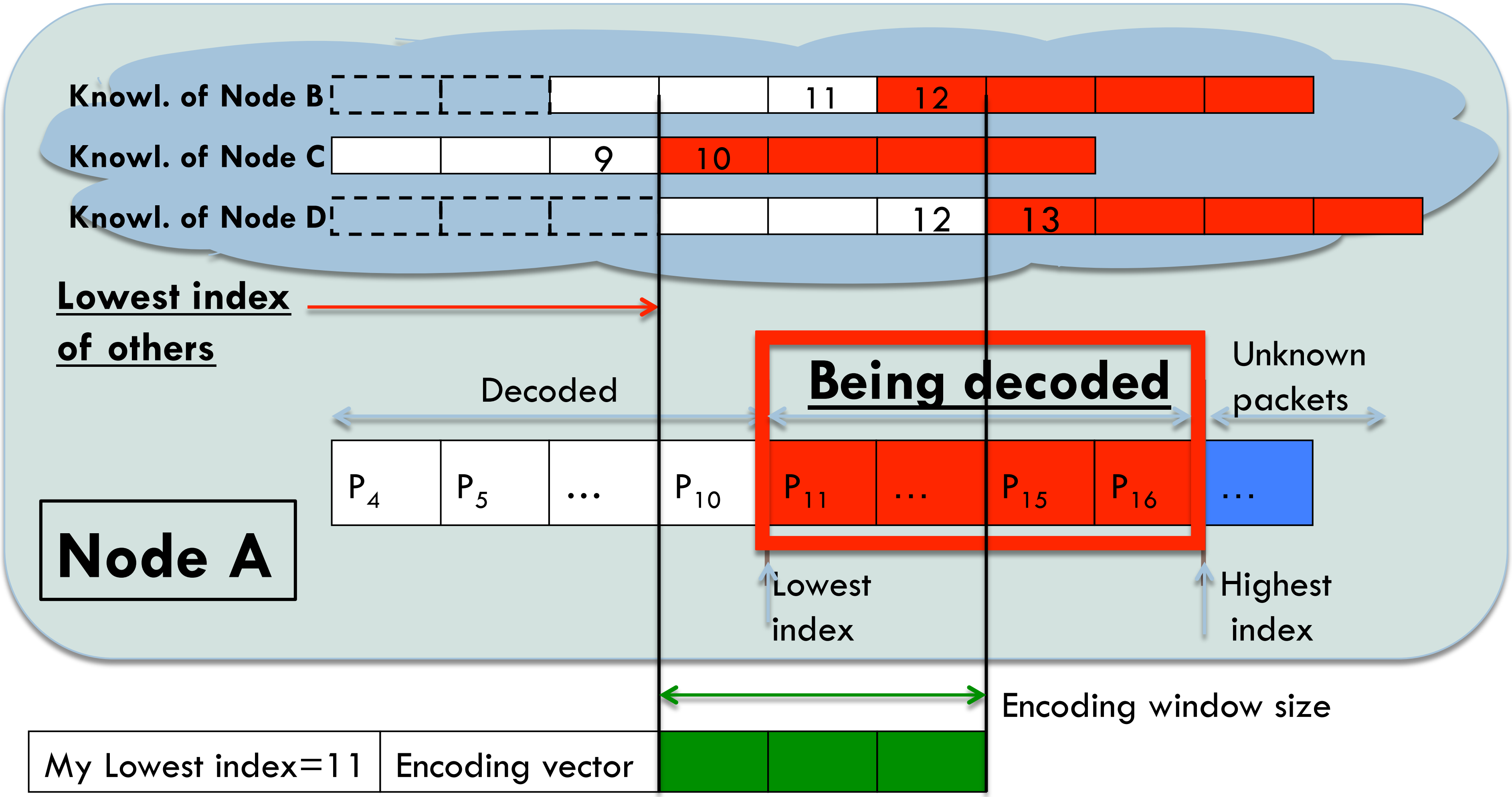
Chosen protocol

- Broadcast Protocol:
  - Based on protocol DRAGONCAST (draft-adjih-dragoncast-00.txt)
  - Cho, S-Y. and C. Adjih, "Wireless Broadcast with Network Coding: DRAGONCAST", Inria RR-6569, July 2008
  - Coded payloads are maintained in a decoding set
    - With Gaussian elimination (but "inverted" RREF)
  - Every node retransmits coded payloads at a given packet rate per second: with random linear coding
  - State of the each node is piggybacked
    - Rank, low index (=decoded payloads), number of neighbors
  - Sliding Encoding Window (SEW)
  - Dynamic Rate Adjustment *(not implemented yet)*

SEW: Sliding Encoding Window

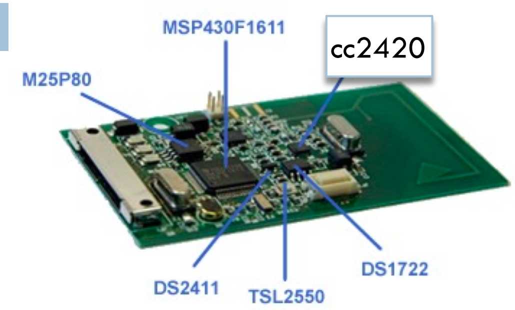
- Principle: “real-time” robust decoding
- Variant of Gaussian elimination (“inverted” RREF)

SEW: knowledge of neighbor state



Experimental settings

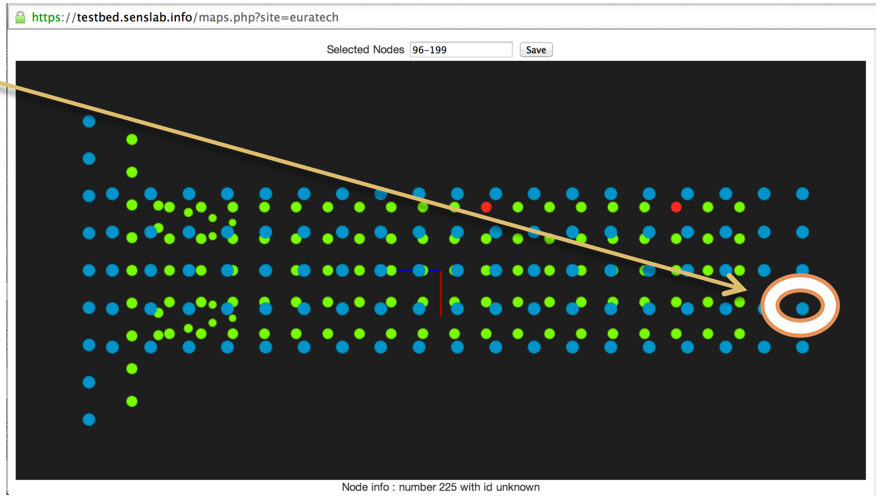
- Implementation on sensor nodes:
  - MSP430, 16 bits, 8 MHz
    - 48 kB Flash, 10 kB RAM
  - Radio 802.15.4 (CC2420)
- Used testbed with 200+ nodes
  - Site of Inria Lille (Euratech)
  - Part of a federation of large scale, open, testbeds: IoT-Lab/Senslab (FIT)
  - http://senslab.info/

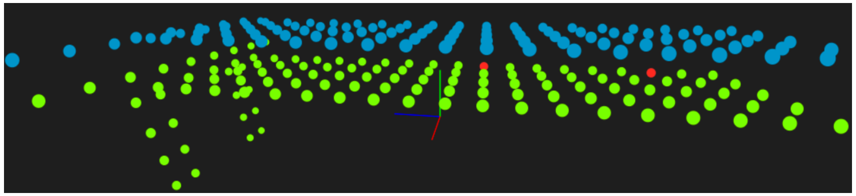


Experiment parameters

- 219 nodes
- Lowest power (-25 dBm)
- NC Parameters:
  - GF(4)
  - Coding window=15
  - Gaussian elim. win.=64 (regen. decoded packets)
  - Source, interval = 2 sec
  - Node, interval = 4 sec
  - Test: payload = 8 bytes (but on radio, 60+ bytes)

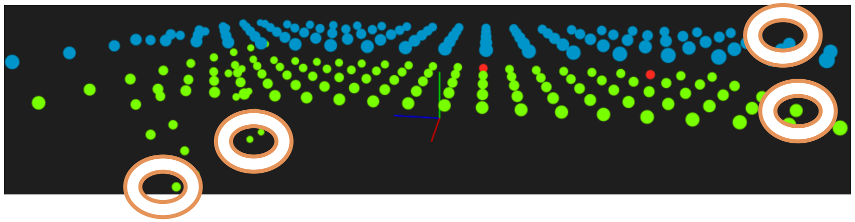
source

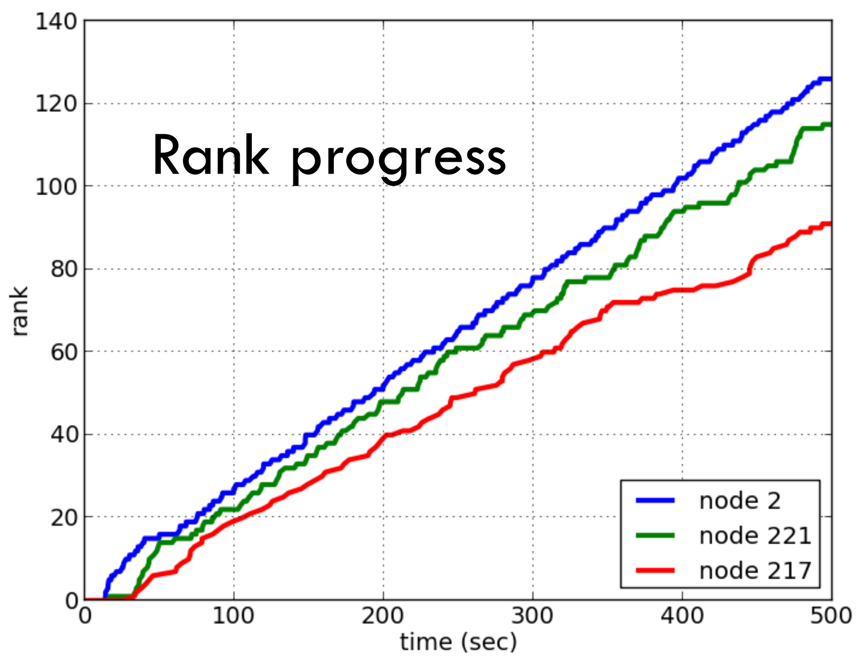


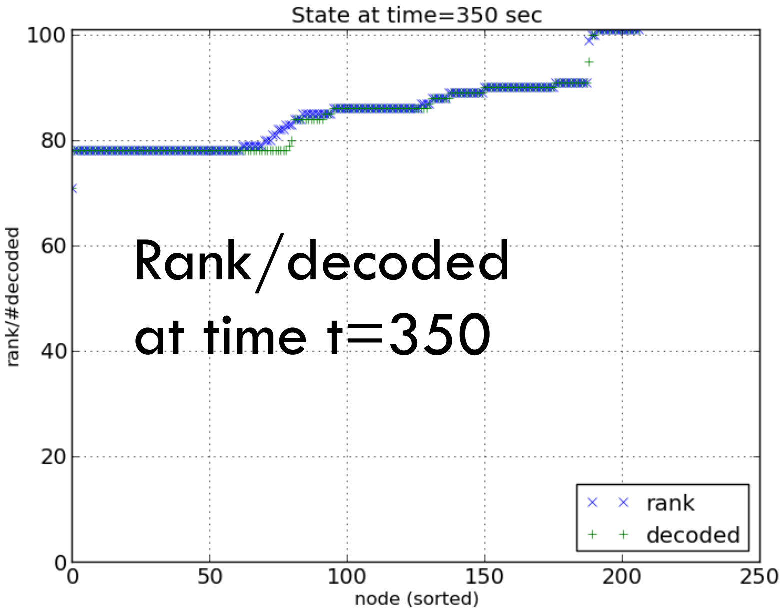


NC broadcast progress (rank)

- Good progress
- Relies on SEW

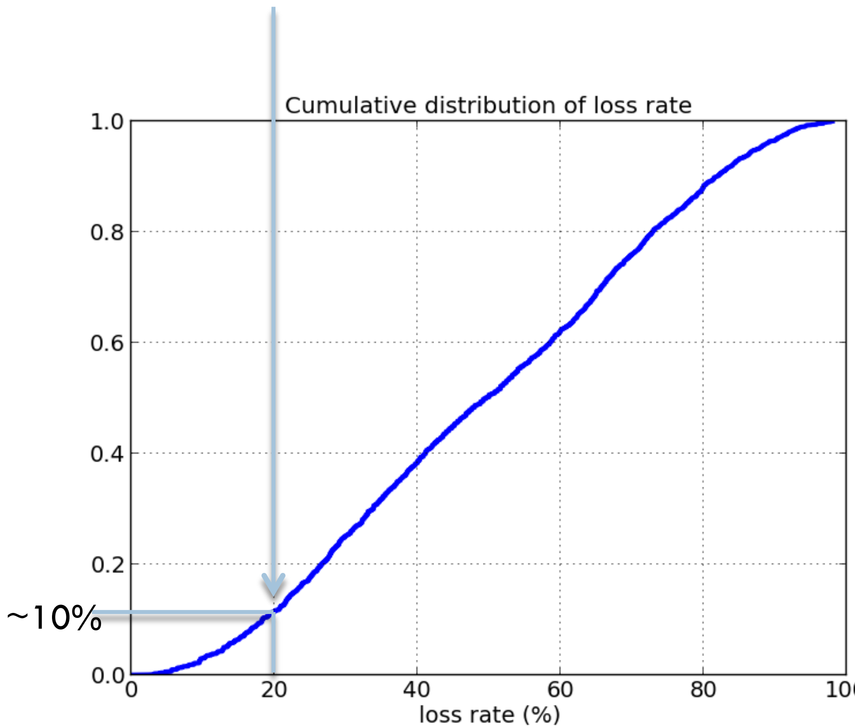


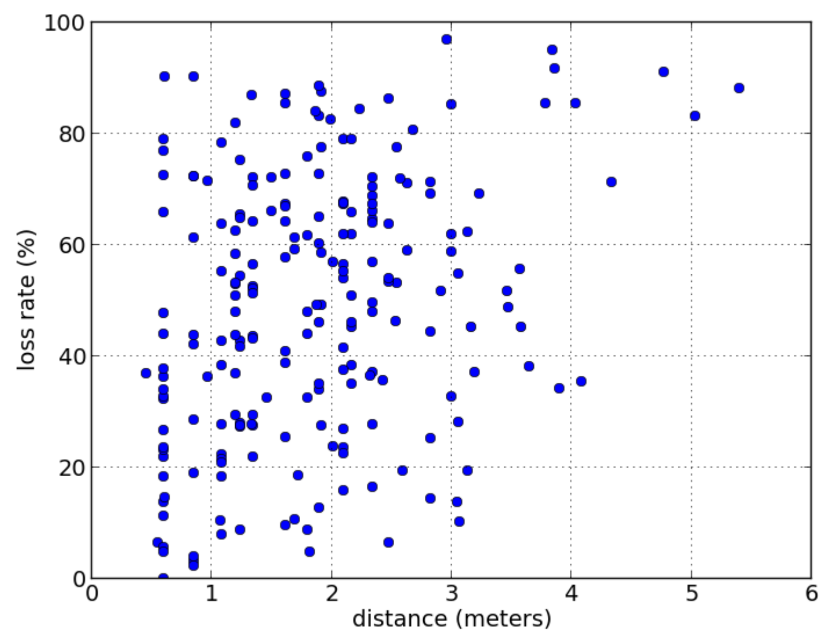




High loss rate

- NC is performing well despite high loss rate
- ~10% of the "links" have less than 20% loss rate

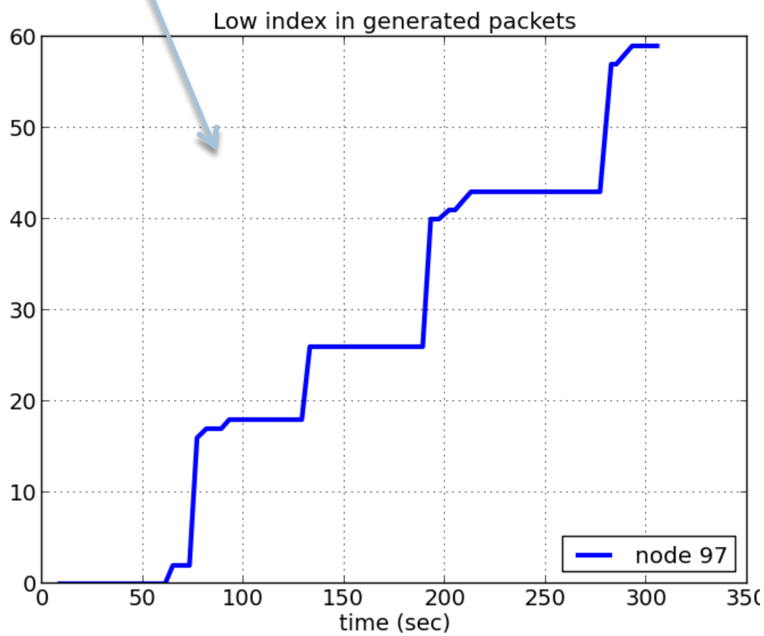




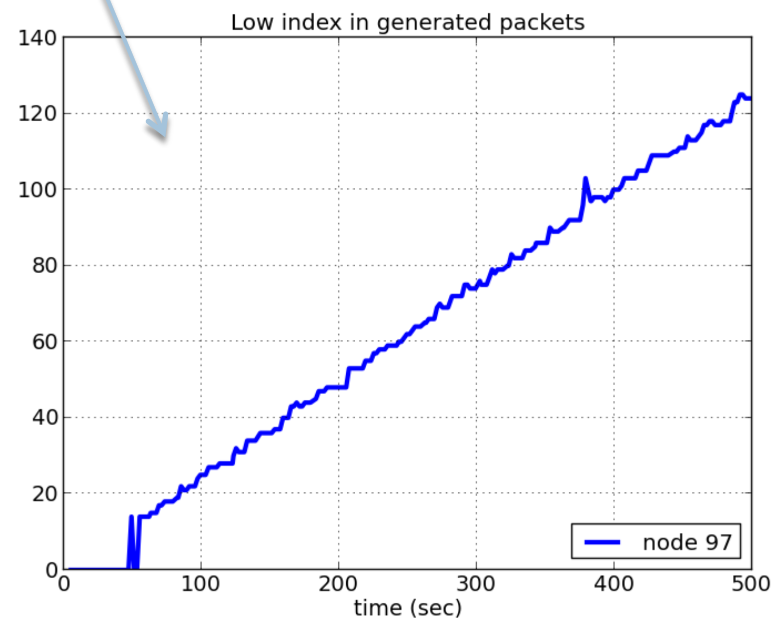
SEW, window sliding strategy

- Simple feedback of SEW is fine in multipath context

With two nodes



With 219 nodes

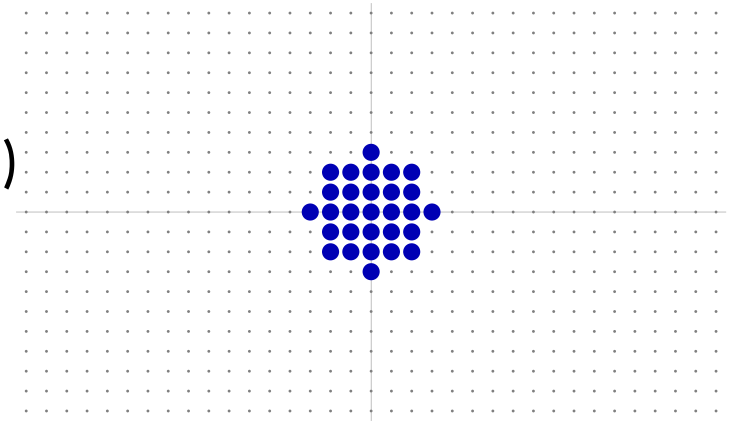


Some lessons

- Operates well even with high loss rate
- Complexity still present in neighbor state:
  - RAM: neigh. state (4608 bytes) vs packet set (3344 bytes)
  - Managing expiration (timers)
  - Packet loss = feedback loss
- Difficulties:
  - Sliding windows with limited “backlog”

Ongoing efforts

- Questions: **Right packet rate for every node ? Efficiency ? Coding nodes ?**
- Elements of answer (and benchmark):
  - Antonia Maria Masucci, C.A. « Efficiency of Broadcast with Network Coding in Wireless Networks », Inria RR-8490, Feb. 2014
  - Perfectly regular network
    - (remove problems on side: torus)
  - Rate=1, Source=#neigh (28)
  - "Nearly" every transmission is innovative



Conclusion

- Demonstrated NC broadcast
  - On most constrained hardware
  - High loss conditions
- Interest of the RG for:
  - This type of application ?
    - Broadcast (e.g. many nodes)
  - This type of architecture/ building blocks ?
    - Sliding windows
    - Neighbors state feedback
    - (Packet rate adaptation)

THANK YOU

SEW: Sliding Encoding Window

- Example

```
2 2 2 3 1 0 0 1 2 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 '\x08\x08\x08\x08'
1 1 1 2 2 2 1 2 0 0 3 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 '\x07\x07\x07\x07'
0 1 2 2 1 0 2 2 1 3 1 0 3 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 '\x06\x06\x06\x06'
1 2 3 3 1 0 2 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 '\r\r\r\r'
2 0 1 1 3 0 2 0 2 3 3 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 '\x14\x14\x14\x14'
3 2 3 3 2 0 0 2 3 1 2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 '\x13\x13\x13\x13'
2 1 0 1 0 2 2 3 3 0 2 0 2 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 '\x10\x10\x10\x10'
1 0 3 0 0 0 3 1 3 2 0 0 3 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 '\x1a\x1a\x1a\x1a'
0 0 0 2 0 1 0 1 3 3 0 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 '\x10\x10\x10\x10'
1 2 0 0 0 3 2 0 1 1 0 0 2 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 '\x16\x16\x16\x16'
0 3 2 2 3 0 3 0 1 2 3 0 2 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 '\x11\x11\x11\x11'
2 0 0 1 2 3 0 2 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 '\x17\x17\x17\x17'
2 0 2 0 2 2 3 1 1 3 1 0 3 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 '\x1e\x1e\x1e\x1e'
3 2 3 1 2 3 3 0 1 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 '\x13\x13\x13\x13'
3 2 0 0 3 1 0 2 3 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 '\x19\x19\x19\x19'
3 0 0 2 0 2 3 1 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 '\x19\x19\x19\x19'
1 2 3 3 2 1 2 0 2 1 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 '\x16\x16\x16\x16'
2 1 0 3 0 3 0 2 3 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 '\x10\x10\x10\x10'
2 0 1 2 2 2 3 0 0 1 3 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 '\x12\x12\x12\x12'
3 3 0 1 3 2 0 3 1 3 3 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 '\x11\x11\x11\x11'
```

Implementation

- RAM for dragoncast = 7978 bytes
  - RAM for dragon = 4608 bytes
  - RAM for packet_set = 3344 bytes
- Code for MSP430: 20430 bytes

Rank vs decoded (window)