# Secure Mobile Networking Lab

Winter 2017/18

## Random Network Coding based Broadcast

Student: Jan Sturm

Supervisor: Dingwen Yuan

# Content

- Introduction:
  - Random Network Coding (RNC)
  - Gaussian Elimination
  - Galois Fields
- Protocol Overview
- Optimizations
- Performance Evaluation
- Contiki Live Demo

# Introduction

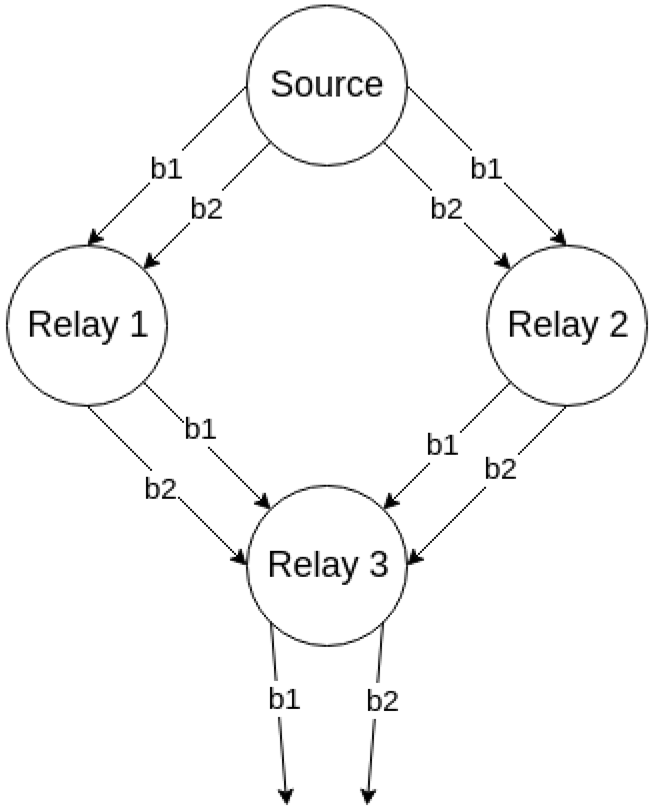
**Why RNC?**

# Broadcast scenario

- Node types:
  - single source
  - relay
  - sink
  - relay+sink
- Deployed sensor network in environment
  - environment-dependent errors
  - debug and fix errors with frequent software updates
  - Source node continuously broadcasts a batch of packets (firmware update)

# Broadcast scenario

- Naive approach
  - Flood packets through network
  - Not very energy efficient

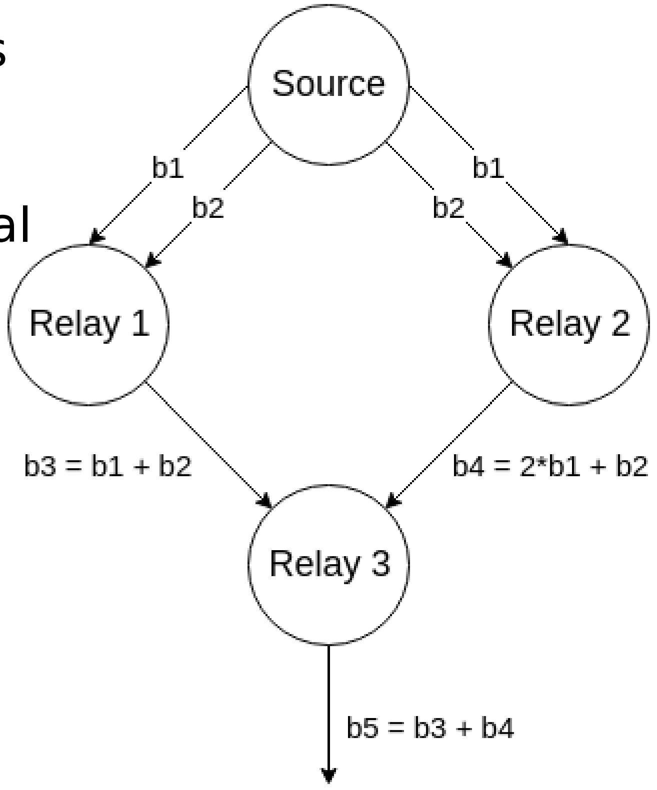


# Broadcast scenario

- Sensor nodes:
  - Small micro-controller, limited memory, battery power supply
  - Power is one of most critical resources
  - Packet transmission is very high energy-consuming action compared to computation
    → more computation rather than transmission

# Random Network Coding

- RNC approach
  - broadcast random linear combinations
  - Reduce traffic and increase reliability
  - Gaussian elimination to decode original packets




- decoding at relay 3 in example:

  b1 = b4 – b3

  b2 = 2*b3 - b4

# Random Network Coding

- Each node constructs a linear combination of K packets $p_1, \ldots, p_K$
- Choose K random coefficients $a_1, \ldots, a_K$
- Compute

$$b = a_1 \cdot p_1 + a_2 \cdot p_2 + \cdots + a_K \cdot p_K$$

- Packet consists of header and payload
  - header = coefficients
  - payload = encoded packets

$$b_i : \left[ \texttt{header}_i \;\middle|\; \texttt{pl}_i \right] = \left[ [a_1, \ldots, a_K]_i \;\middle|\; \texttt{pl}_i \right]$$

# Gaussian Elimination

- Nodes construct coefficient matrix $A$
- System of linear equations

$$A \cdot x = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1K} \\ a_{21} & a_{22} & \cdots & a_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K1} & a_{K2} & \cdots & a_{KK} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} = b$$

# Gaussian Elimination

- Perform elementary row operations on $A|b$
- Obtain upper triangular matrix $\tilde{A}$

$$\tilde{A} \cdot x = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1,K-1} & a_{1K} \\ 0 & \tilde{a}_{22} & \dots & \tilde{a}_{2,K-1} & \tilde{a}_{2K} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \tilde{a}_{K-1,K-1} & \tilde{a}_{K-1,K} \\ 0 & 0 & \dots & 0 & \tilde{a}_{KK} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K-1} \\ x_K \end{bmatrix} = \begin{bmatrix} b_1 \\ \tilde{b}_2 \\ \vdots \\ \tilde{b}_{K-1} \\ \tilde{b}_K \end{bmatrix} = \tilde{b}$$

- Exactly one solution for $rank(A) = K$
- Decode packets $x_1, \dots, x_K$ with back substitution

# Galois Fields

- Field =

  set of elements for which *add, mul, sub, div* results in another element of the same set

- Galois Field (finite field) =

  field with a finite number of elements

- e.g. $GF(2) = \{0,1\}$
- Practical interest: $GF(2^m)$

# Galois Fields: *GF($2^m$)*

- Can be represented
  - as polynomials of degree less than *m* over *GF(2)*
  - binary numbers

- Example for *GF($2^8$)*:
  - Hex: *0xA3*
  - Binary: *10100011*
  - Polynomial: $x^7+x^5+x+1$

# Galois Fields

- Advantage compared to $\mathbb{R}$ :

$a_1 \cdot p_1 + a_2 \cdot p_2 + \cdots + a_K \cdot p_K$ → constant size

- Linear combination has same size as each individual packet

# Protocol Overview

- Batch = $K$ packets (we combine $K$ packets)
- Source broadcasts packets in fixed interval
- Relay helps to spread packets
  - Cache data in memory
  - Run Gaussian elimination
  - new information (rank of $A$ changed)?

    → generate and send out new RNC packet after random delay
- Only sinks will decode data

# Protocol Overview

- 100% reliability with Negative Acknowledgements (NACKs)
  - Every node keeps countdown timer
    - If timer fires → broadcast NACK
- Which node to respond?
  - Many nodes respond → unnecessary transmissions + channel congestion
  - Solution:
    1. Check if requested information is available
    2. If yes, delay for random period
    3. Broadcast NACK-reply, if no NACK-reply is heard during period
- NACK-reply: random linear combination of all so far received packets

# Optimizations – Gaussian Elimination

- Iterative process
  - Run Gaussian Elimination on reception of new packet
  - Iteratively build parts of upper triangular matrix

→ smaller computations every time we receive a packet

better than

one big computation for a full rank matrix

- Lower latency
- Determine in-time whether received packet provides new information

# Optimizations – *GF(2^m)* arithmetic

- Addition is fast (XOR)
- Multiplication / division is slow (polynomial multiplication)
  - Solution: look-up tables
    - two arrays $\log_g(x)$ and $g^{(x)}$ with $2^m$ elements each
      - Use generator $g$ of $GF(2^m)$
        → each non-zero element can be written as $g^i$

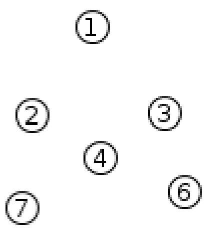
$$a \cdot b = g^{\log_g(a \cdot b)} = g^{(\log_g(a)+\log_g(b)) \mod |g|}$$ *→ 3 table look-ups for multiplication*

$$a^{-1} = g^{\log_g(a^{-1})} = g^{-\log_g(a)} = g^{|g|-\log_g(a)}$$ *→ 2 table look-ups for multiplicative inverse*

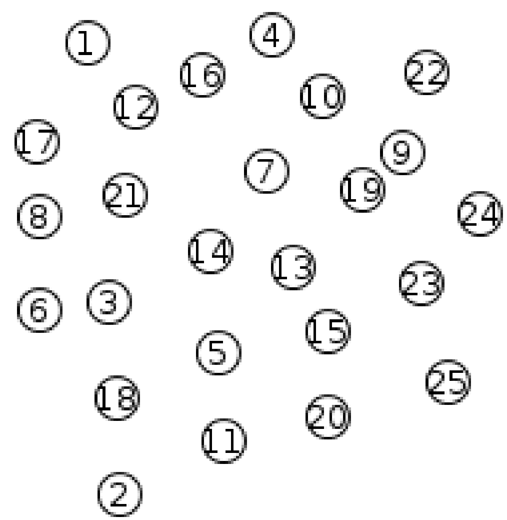
# Performance Evaluation



- Small and bigger network
- *GF(256)*, *GF(16)* and *GF(2)*
- batch size *K=6,7,8*
- Comparison with simple flooding scheme




small network (6 nodes)

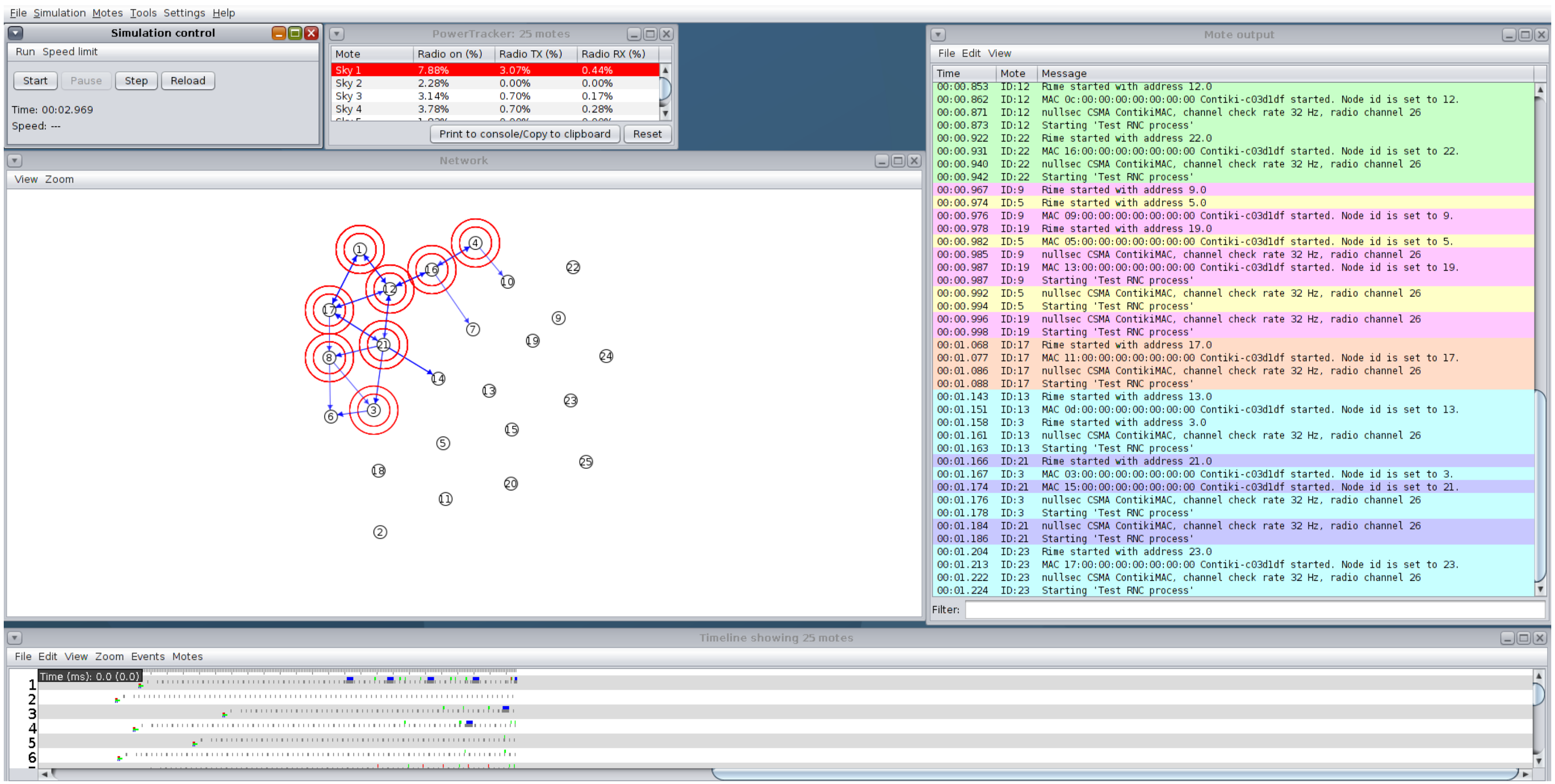



bigger network (27 nodes)

# Performance Evaluation

SECURE MOBILE NETWORKING

- Simulation of sensor network in Contiki / Cooja



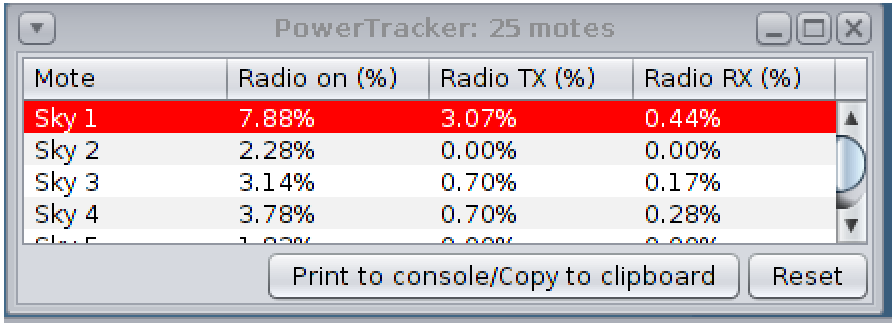
# Performance Evaluation
## Latency

- Latency (in ms) = time between source sending out a batch and the full recovery of a batch at a sink
- dominated by transmission time, not computation time
- *GF(256)* and *GF(16)*:
  - small network:
    - RNC ~**3x lower latency** than flooding
  - bigger network:
    - RNC ~**2x lower latency** than flooding for K=7,8
    - K=6 performs worse than flooding

      → need timeout adjustment for each K
- GF(2):
  - Small network: RNC ~1.5x lower latency than flooding
  - Bigger network: bad performance, too high due to many dependent packets

# Performance Evaluation
## Energy consumption

- Average Radio Duty Cycle (RDC)
- Similar observations as for latency

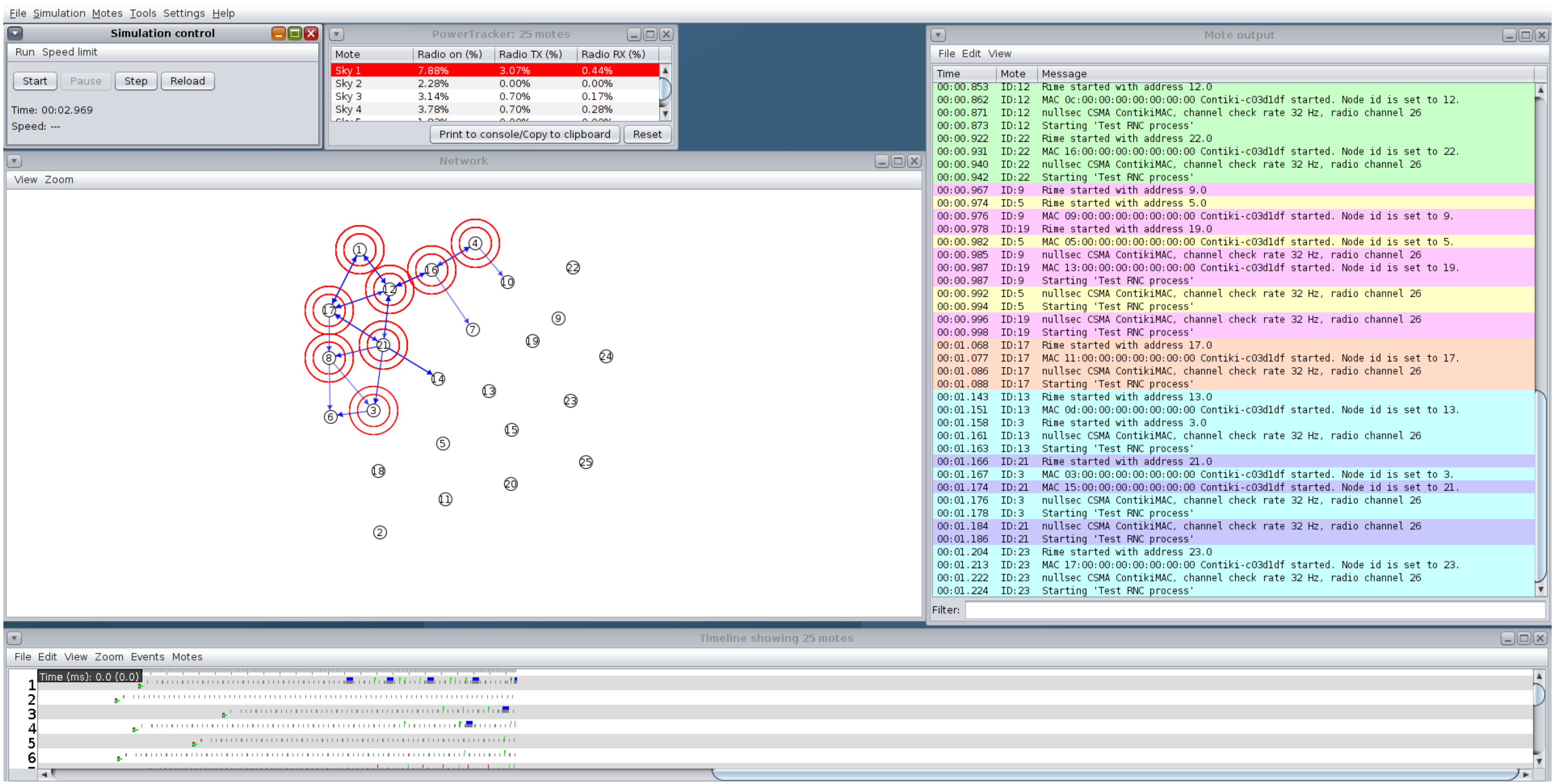


PowerTracker: 25 motes

| Mote | Radio on (%) | Radio TX (%) | Radio RX (%) |
|---|---|---|---|
| Sky 1 | 7.88% | 3.07% | 0.44% |
| Sky 2 | 2.28% | 0.00% | 0.00% |
| Sky 3 | 3.14% | 0.70% | 0.17% |
| Sky 4 | 3.78% | 0.70% | 0.28% |

Print to console/Copy to clipboard | Reset

# Contiki Live Demo



# Contact

**Prof. Dr.-Ing. Matthias Hollick**
matthias.hollick@seemoo.tu-darmstadt.de

Technische Universität Darmstadt
Secure Mobile Networking Lab – SEEMOO
Department of Computer Science
Mornewegstr. 32
D-64293 Darmstadt

Phone: +49 6151 16-25472
Fax: +49 6151 16-25471
Web: https://seemoo.de

# Copyright Notice