Practical Implementations of Network Coding

Frank Fitzek // Summer Semester 2019
Lecture / Exercise Dates - goo.gl/Lr7Xz4

- Here you find all the information for the lecture and the exercise can be found. Please check every week in case of changes (might happen)
- Slides
- Links
Aim of this lecture module

Explain network coding in theory and practice

Explain the uniqueness of network coding

Describe a wide range of applications of NC in current and future communication systems
  — 5G
  — Storage as well as transportation

Important is the “hands on” parts aligned with the theory
  — Please bring your laptop to all lectures and exercises
    – Preinstall software needed
    – Get KODO license from steinwurf.com/license
Research Methodology: Theory that matters!

Fundamental Research:
Theory, Concepts, Mathematical Analysis, Modelling

Research Area I
Research Area II
Research Area III

Algorithms & Protocols & Parameterization

Simulation
Software in the Loop
Hardware in the Loop
Testbed

Performance Evaluation

Exploitation:
Publications
Patents
Teaching
Demonstrators
Open Source
Tech Transfer
Publications

Research Task 1.1
Research Task 1.2
Research Task 1.3
...
Research Task C
...
Research Task 3.1
Research Task 3.2
...
Research Task 2.1
Research Task 2.2
...

Implement Theory

Theory
Implementation
Research Methodology: Example

2007: 120kB/s

2012: 27 MB/s

Coding and decoding throughput

Throughput [kB/s]

Packets coded together

0 10 20 50 100 200 400

GF(2^8) encoding
GF(2^8) decoding
GF(2^16) encoding
GF(2^16) decoding

Throughput [MB/s]

Generation Size

16 32 64 128

Benchmark
GF(2)
GF(2^8)
GF(2^16)
GF(2^{32} - 5)
How fast are we today?

Kodomark
Steinwurf ApS Libraries & Demo

This app is compatible with all of your devices.

http://tinyurl.com/z7vsp4c

Please try it out and support our research! If you have an Android device simply install and press START! Change the parameters to learn about network coding.
How do we approach NC?

Extreme application of NC! In general the same ideas as before but more gains!

Let’s have fun! We play around with some smart ideas!

The real deal! Versatile code for all application fields! Complex but powerful!
5G Motivation
5G atom definition
Mesh and Network Coding
5G atom definition

- Machine learning
- Block Chaining
- Compressed Sensing
- SDN
- NFV
- ICN
- Network Coding
- SDR

5G concepts
- Latency
- Security
- Energy
- Mobile Edge Cloud
- Content Delivery Networks
- Novelty

5G requirements
- Heterogeneity
- Mobility
- Compressed Sensing
- Energy

5G technologies
- Machine learning
- Block Chaining
- Compressed Sensing
- SDN
- NFV
- ICN
- Network Coding
- SDR
Digital Inter-Flow Network Coding: The Basics
Lecture 1
The Butterfly
Network Coding: The Butterfly

- Two packets a and b must be conveyed to two destinations over a given network.
- Assumption, capacity per link can handle one packet per time slot.
- Bottleneck in the middle.
- Either packet a or b will path the bottleneck.
- One destination will receive one unique packet, the other two packets.
Network Coding: The Butterfly

- Let's try b instead of a
- Same old problem
Network Coding: The Butterfly

- Ahlswede et. al. In 2000
- Coding the packet
- Other ideas were around
- Max-flow min-cut theorem

Kirchhoff versus Network Coding

Kirchhoff

\[ O = I_1 + I_2 \]

All engineers follow this principle!

Network Coding

\[ O = f(I_1, I_2) \]

Now we are alone ...!
Max-flow min-cut theorem

The existence of polynomial time algorithms is remarkable because the maximal rate without coding can be much smaller and finding the routing solution that achieves that maximum is NP-hard.

Network Coding: The Butterfly

- XOR operation
- Bitwise operation
- Same bit value results in "0"
- Different bit value results in "1"

<table>
<thead>
<tr>
<th>Input 1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Result</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Network Coding: The Butterfly

- Adding complexity at some nodes of the network
- Adding overhead in order to know what was coded together (encoding vector)
Network Coding: The Butterfly

Source: transmitting two information entities

Receiver: receiving two coded information entities

\[ x_1 \longrightarrow x_2 \]

\[ y_1 \quad y_2 \]

\[ y_1 \quad y_2 \]
Network Coding: The Butterfly

\[ x_1, x_2 \]

\[ (c_1, c_2) \]

\[ c_1 x_1 + c_2 x_2 \]

\[ y_1, y_2 \]
Network Coding: The Butterfly

\[
\begin{align*}
y_1 &= x_1 \\
y_2 &= c_1 x_1 + c_2 x_2 \\
(\begin{array}{c} y_1 \\ y_2 \end{array}) &= \left( \begin{array}{cc} 1 & 0 \\ c_1 & c_2 \end{array} \right) \left( \begin{array}{c} x_1 \\ x_2 \end{array} \right)
\end{align*}
\]

\[
\begin{align*}
y_1 &= c_1 x_1 + c_2 x_2 \\
y_2 &= x_2 \\
(\begin{array}{c} y_1 \\ y_2 \end{array}) &= \left( \begin{array}{cc} c_1 & c_2 \\ 0 & 1 \end{array} \right) \left( \begin{array}{c} x_1 \\ x_2 \end{array} \right)
\end{align*}
\]
Network Coding Basics - Fun
Network Coding Basics - Fun
Network Coding Basics - Fun

\[
\text{Apple} + \text{Apple} + \text{Banana} = 24
\]

\[
\text{Apple} + \text{Banana} + \text{Banana} = 18
\]

\[
\text{Banana} - \text{Coconut} = 2
\]
Network Coding Basics - Fun
Min-Cut
Cuts of Flow Networks

Prof. Dr.-Ing. Dr. h.c. Frank H.P. Fitzek
Practical Implementations of Network Coding Lecture
Technische Universität Dresden, Deutsche Telekom Chair of Communication Networks
Cuts of Flow Networks

Capacity = 20
Cuts of Flow Networks

Capacity = 8
Cuts of Flow Networks

Capacity = 16
Cuts of Flow Networks

Min Cut!

Capacity = 8
Ford-Fulkerson
Ford-Fulkerson Max Flow

Diagram showing a network with nodes S, A, B, C, D, E, F, and T. The edges and capacities are labeled with numbers to represent the maximum flow through the network.
Ford-Fulkerson Max Flow
Ford-Fulkerson Max Flow

Graph with nodes S, A, B, C, D, E, F, and T. Edges and capacities as follows:
- S to A: 8
- A to D: 3
- A to B: 3
- D to T: 9
- E to T: 1
- B to E: 3
- C to B: 3
- C to F: 3
- B to C: 1
- B to E: 4
- F to E: 5
- C to E: 5
- F to T: 1
- A to D: 2
- D to E: 3
- E to A: 3
- T to D: 6
- T to F: 3
- A to F: 3
- S to B: 7
Ford-Fulkerson Max Flow

![Graph with labeled nodes and edges showing max flow](image-url)
Ford-Fulkerson Max Flow

![Ford-Fulkerson Max Flow Diagram](image-url)
Ford-Fulkerson Max Flow

![Ford-Fulkerson Max Flow Diagram](image)
Ford-Fulkerson Max Flow

![Ford-Fulkerson Max Flow Diagram]
Ford-Fulkerson Max Flow

Diagram showing network flow with capacities on each edge.
Ford-Fulkerson Max Flow

A graph with nodes S, B, C, D, E, F, and T. Edges are marked with capacities and arrows indicate the direction of flow. The capacities are as follows:

- From S to A: 3
- From S to B: 5
- From A to D: 3
- From B to C: 1
- From C to F: 3
- From C to T: 3
- From D to E: 2
- From D to F: 9
- From E to T: 6
- From F to T: 1

The maximum flow is determined by the capacities along the path from S to T.
Ford-Fulkerson Max Flow
Ford-Fulkerson Max Flow

Diagram showing a network with nodes S, A, B, C, D, E, F, and T, with capacities on the edges represented by numbers.
Ford-Fulkerson Max Flow

[Diagram showing a network flow problem with nodes S, A, B, C, D, E, F, and T, with flow values on the edges.]

- Source node S
- Sink node T
- Edges and flow values:
  - S to A: 3
  - A to B: 7
  - B to C: 3
  - C to S: 1
  - C to F: 2
  - F to T: 5
  - D to E: 2
  - E to T: 9
  - E to F: 0
  - F to A: 1
  - F to T: 1
  - A to D: 3
  - D to T: 9
  - B to E: 3
  - C to D: 6
Ford-Fulkerson Max Flow
Ford-Fulkerson Max Flow

Graph representation of the Ford-Fulkerson Max Flow algorithm.
Ford-Fulkerson Max Flow

Capacity = 8

Graph representation of the Ford-Fulkerson Max Flow algorithm with capacities on the edges.
Ford-Fulkerson Max Flow

Capacity = 8

Told you!
Ford-Fulkerson Max Flow

You were lucky!

Capacity = 8
Multi-Cast Example
Multi-Cast Example

Communication Network
Min-cut = 3

Compare Routing with Network Coding
Multi-Cast Example

Compare Routing with Network Coding
Routing: np-hard; not optimal with respect to capacity
Multi-Cast Example

Compare Routing with Network Coding

Network Coding: optimal solution can be found and can be found fast
Multi-Cast Example

Communication Network
Min-cut = 3

Compare Routing with Network Coding
Multi-Cast Example

Communication Network
Min-cut = 3

Compare Routing with Network Coding
The Butterfly++
Network Coding: The Butterfly++

Let’s code all incoming packets .... mmmh

Wrong decision by the node!
Network Coding: The Butterfly++

Better choice! No coding just forwarding!
result = (\text{rank}(A) == 3) ? "decodable" : "better choice next time";
Network Coding: The Butterfly++

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
c_4 \\
c_1 \\
0 \\
\end{pmatrix}
\begin{pmatrix}
c_5 \\
c_2 \\
0 \\
\end{pmatrix}
\begin{pmatrix}
c_6 \\
c_3 \\
1 \\
\end{pmatrix}
\]

result = (rank(A)==3) ? "decodable" : "better choice next time";
Network Coding: The Butterfly++

- What did you send?
- What did you receive?
- What should we send?

Diagram showing the network coding process with nodes labeled 'a', 'b', and 'c'. The diagram illustrates the flow of data and the calculations involved in the network coding process.
Deterministic Network Coding

Deterministic Network Coding refers to a specific method for network code design. I.e. exactly specifying how input data is mapped to output data for all nodes in a network. This is in contrast to Random Network Coding.

Advantages
— Coding coefficients are known and therefore not required to be explicitly communicated.

Drawbacks
— Algorithms often require that the exact and full topology as input. \( \rightarrow \) SDN
— Dynamic networks will require frequent updates, to reflect current state of the network.
Digital Inter-Flow Network Coding: The Basics

Lecture 2
Channel & Source Coding vs. Network Coding
Comparison of Coding Approaches

- **Network Coding**
  - Improves *Network*
  - Reliability, Security, and Efficiency

- **Source Coding**
  - Removes Redundancy for Efficiency

- **Channel Coding**
  - Introduces Redundancy for Reliability

- Introduces Redundancy for Reliability
Comparison of Coding Approaches
Wireless Network Examples
Exploiting the broadcast nature of the wireless medium
It is not about butterflies ...

General Network Topologies and Sub-Nets

- Two way relay (Alice and Bob)
- Chain
- X- Topology
- Cross
- Cross with Overhearing
General Network Topologies and Sub-Nets

- Two way relay (Alice and Bob)
- Chain
- X- Topology
- Cross
- Cross with Overhearing
Alice and Bob – The two way relay
Early VTC paper by Ericsson
Alice and Bob
Alice and Bob: Forwarding
Alice and Bob: Forwarding

A → R → B
Alice and Bob: Forwarding
Alice and Bob: Forwarding

4 time slots for exchange
Alice and Bob: Network Coding
Alice and Bob: Network Coding
Alice and Bob: Network Coding

3 time slots for exchange

Gain = 33%
Two-Way Relay (Alice and Bob) vs. Butterfly
Network Coding: Magic XOR

Alice and Bob: Symbian60

Our starting point
Simple scenario
Seeding of packet a and b is crucial
— Fairness
— Performance
Forms of NC
— XOR in the air (COPE)
Implementation on S60
Lessons learned: It is important to create coding potential!
Alice and Bob: Satellite Communication
The X-Topology
Introducing the concept of overhearing
X Topology with Overhearing

Start

Broadcast
X Topology with Overhearing

Coding @ relay

Decoding at receiver
X Topology with Overhearing
X Topology: SoA

A

B

C

D

R

A

R
X Topology: SoA

A

C

D

B

R
X Topology: SoA
X Topology: SoA

- A
- B
- C
- D
- R

Diagram showing network topology with nodes A, B, C, and D connected in a specific pattern.
X Topology with Overhearing

A

D

R

A

C

B
X Topology with Overhearing
X Topology with Overhearing
X Topology with Overhearing

Diagram of network nodes A, C, D, B with arrows indicating connections and labels A, C.
X Topology with Overhearing

Gain = 33%
The Cross Topology
Cross Topology with Overhearing

Overhearing the activities of the neighboring node
Cross Topology with Overhearing 1/5

Diagram showing a cross topology with nodes A, B, C, and D. Node A is connected to nodes C and R, node R is connected to node D, and node B is isolated. There is an indication of overhearing from node A to node R.
Cross Topology with Overhearing 2/5
Cross Topology with Overhearing 3/5
Cross Topology with Overhearing 4/5
Cross Topology with Overhearing 5/5
Cross Topology with Overhearing 1/2
Cross Topology with Overhearing 2/2

Gain = 60%
Cross Topology with Overhearing 2/2

Gain = 60%

XOR is NOT limited to code only two packets
Cross Topology with Overhearing Problems

Hint: CORE protocol
Inter Flow Network Coding

- No delay
- Low complexity
- Easy integration with commercial solutions
- Some ideas are used in ANALOG network coding

- Planning and book keeping needed
- Certain traffic characteristics are beneficial in mesh networks
- Symmetric traffic allows for more coding potential
Index Coding
Index Coding
Index Coding

In wireless networks
— Broadcast advantage
— Generation of packets (three in this case)
— Packet erasures (losses)
Reliable Multicast: Motivation

- Wireless has the advantage of inherent broadcast
- Wireless is error prone
- Coding versus Retransmissions
- \( N = \) number of packets
- \( J = \) number of users
Reliable Multicast: Motivation

Initial Phase
N=1000; J=10

Recover 1

Recover 2

nodes interested

Initial Phase
N=1000; J=1000

Recover 1

Recover 2

Recover 3

nodes interested

n nodes interested

n nodes interested
### Index Coding

Real devices → broadcast → base station → feedback

#### Information of the devices by the BS

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td></td>
</tr>
</tbody>
</table>

#### Real devices

- $\Sigma_3$
- $\Sigma_3$
- $\Sigma_2$
- $\Sigma_2$
- $\Sigma_2$
- $\Sigma_1$
- $\Sigma_1$

---

Prof. Dr.-Ing. Dr. h.c. Frank H.P. Fitzek  
Practical Implementations of Network Coding Lecture  
Technische Universität Dresden, Deutsche Telekom Chair of Communication Networks  

Slide 117
Index Coding

Initial Phase
N=1000 ; J=10

Recover I
Recover R2

nodes interested

Initial Phase
N=1000 ; J=1000

Recover I
Recover R2
Recover R3

nodes interested

n nodes interested
Broadcast Example

Wireless network with packet losses
Broadcast Example

Wireless network with packet losses

Fewer data packet re-sends
Broadcast Example

Wireless network with packet losses

Same errors as before ...
Broadcast Example: Cooperative Advantage

Single broadcast...

...and peers share missing data
w/o Coding
w Coding
First implementations

Inter-Flow NC on N810 (2008)

- XOR coding on N810 (linux device by NOKIA)
- Remote setup
  - Predefined set
  - Random set
- Constantly exchanging reception updates (who needs what)
- Overall goal is to let all devices have all information
N810 Implementation COPE

https://www.youtube.com/watch?v=VZYLSyZaEO8
Results: N810

Random Starting phase: (duplicates are possible)

Disjoint Starting Phase: (no duplicates on beginning)
Digital Zig Zag Coding

Sagt es ihnen! Zick, zick, zick, nein zack ... die falsche Richtung ... Meister ... meine Bilder ... meine Leinwand ... Zickzack falsch ... sagt es ihnen ... falsch...
Architecture and Example

- Assuming one node broadcasts information to several nodes over error-prone medium
- All nodes are interested in the same content
Architecture and Example

- Assuming one node broadcasts information to several nodes over error-prone medium
- All nodes are interested in the same content
- Assuming we have only two messages A and B with three bits of information each (to make it easy)
Architecture and Example

• Assuming one node broadcasts information to several nodes over error-prone medium
• All nodes are interested in the same content
• Assuming we have only two messages A and B with three bits of information each (to make it easy)
Architecture and Example

- Assuming one node broadcasts information to several nodes over error-prone medium
- All nodes are interested in the same content
- Assuming we have only two messages A and B with three bits of information each (to make it easy)
Decoding 1

Original packet A

Original packet B
**Architecture and Example**

- Assuming one node broadcasts information to several nodes over error-prone medium
- All nodes are interested in the same content
- Assuming we have only two messages A and B with three bits of information each (to make it easy)
- And now? Don’t say repetition coding!
Encoding - XOR

Original packet A

Original packet B

Coded packet $A \oplus B$
Architecture and Example

- Assuming one node broadcasting information to several nodes over error-prone medium
- All nodes are interested in the same content
- Assuming we have only two messages A and B with three bits of information (to make it easy)
- Still not good enough now? Don’t say repetition coding!
Decoding 2

Original packet A

Coded packet $A \oplus B$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Decoding 2

Original packet A

Coded packet A ⊕ B

Simple reading & XOR

\[ \begin{array}{c}
\text{Original packet A} \\
101 \\
\text{Coded packet A ⊕ B} \\
100 \\
\text{Simple reading & XOR} \\
101 \\
001 \\
\end{array} \]
Architecture and Example

• Assuming one node broadcasting information to several nodes over error-prone medium
• All nodes are interested in the same content
• Assuming we have only two messages A and B with three bits of information (to make it easy)
• Still not good enough now? Don’t say repetition coding!
Encoding – Shifted XOR

Legend:
- X: Bit with value x from packet A
- X: Bit with value x from packet B
- X: Coded bit with value x from packet A and packet B

Original packet A: 101
Original packet B*2: 001
Coded packet A⊕B*2: 0111
Decoding 3

Coded packet $A \oplus B \times 2$

Original packet B

Original packet A

Original packet B
Decoding 3

Coded packet $A \oplus B \times 2$

Original packet B

Original packet A

Original packet B
Decoding 3

Simple reading
Decoding 3

0 1 1 1 1
0 0 0 1

Restore old shift

0 0 1 1
0 0 1
Decoding 3

\[
\begin{array}{cccc}
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{array}
\]

\[\text{XOR: } 1 \oplus 1\]
Decoding 3

\[
\begin{array}{ccc}
0 & 1 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
\end{array}
\]

\[
\text{XOR: } 1 \oplus 1 = 0
\]
Decoding 3

\[
\begin{array}{cccc}
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 \\
\end{array}
\]

\[
\text{XOR: } 1 \oplus 0
\]
Decoding 3

\[ \begin{array}{cccc}
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{array} \]

\[ XOR: 1 \oplus 0 = 1 \]
Decoding 4

Coded packet $A \oplus B \times 2$

| 0 | 1 | 1 | 1 |

Coded packet $A \oplus B$

| 1 | 0 | 0 |
Decoding 4

Simple reading

0 1 1 1
1 0 0

0 1
1 0
Decoding 4

\[
\begin{array}{c|c|c}
0 & 1 & 1 \\
\hline
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

\[\text{XOR: } 0 \oplus 1\]
### Decoding 4

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**XOR:** $0 \oplus 1$
Decoding 4

```
0 1 1 1
1 0 0
0 1 1
XOR: 1 ⊕ 0
```
Decoding 4

XOR: $1 \oplus 0 = 1$
Decoding 4

XOR: $1 \oplus 1$
Decoding 4

XOR: \(1 \oplus 1 = 0\)
Decoding 4

\[
\begin{array}{cccc}
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
\end{array}
\]

XOR: \(1 \oplus 1\)
Decoding 4

\[
\begin{array}{cccc}
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
\end{array}
\]

\[\text{XOR: } 1 \oplus 1 = 0\]
Architecture and Example

• Assuming one node broadcasting information to several nodes over error-prone medium
• All nodes are interested in the same content
• Assuming we have only two messages A and B with three bits of information (to make it easy)
• Still not good enough now? Don’t say repetition coding!

XOR coded
Overhead consideration

• For packets with $B$ bits and $S$ shift bits, the overhead $O$ is simply $S/B$.
• IP packets of size 1500 byte and 2 bits shift $\rightarrow$ overhead is marginal.
• How to code multiple packets with bit shifting?
• How many combinations can be achieved with $G$ packets and $S$ shifts?
• What else would be needed?
Encoding – Shifted XOR

Different direction

More shift
Zig Zag Summary

• Very simple extension to the XOR world
• Overhead due to shifting (in our example from 3 to 4 bits)
• Very low complexity due to simple XOR operations
• Same can be achieved with Reed-Solomon and Random Linear Network Coding without overhead but more complexity
• Maybe interesting for IoT device (simple sensor networks)
Digital Inter-Flow Network Coding: The Medium Access
COPE

Network Coding in Wireless Networks

Core contribution by Katti et. al. „XOR in the Air“ applying XOR coding to WIFI enabled meshed networks.

Figure 7—Node locations for one floor of the testbed.

Figure 12—COPE can provide a several-fold (3-4x) increase in the throughput of wireless Ad hoc networks. Results are for UDP flows with randomly picked source-destination pairs, Poisson arrivals, and heavy-tail size distribution.
Network Coding in Wireless Networks

COPE characteristics:
- Desktops
- Click software (user space)
- SRCR as routing algorithm
- Ad hoc mode
- RTS/CTS disabled
- Netgear hardware chips
- 802.11 a/g
CATWOMAN


CATWOMAN (2011)
CATWOMAN (2011)

- Multihop network based on BATMAN routing (draft RFC)
- Implementation of network coding on real WiFi access points
- Multihop
- Part of Linux Kernel 3.10
CATWOMAN: Scenarios under Investigation
CATWOMAN: Testbed
CATWOMAN: Testbed Placement

- Bob
- Dave
- Relay
- Charlie
- Alice

Placed on Ground Floor
Placed on Third Floor

45 meters
CATWOMAN Topologies

Alice and Bob

Extended A&B

X

Cross
CATWOMAN Metric

- Throughput [bit/s]
  - Bandwidth used
- Losses [%]
- Energy [J]
  - Activity level of send, receive, idle
  - CPU load
- Delay [s]
Model for Alice and Bob (symmetric traffic)
Assumption

- Alice and Bob have no direct connection
- Same amount of traffic is generated by Alice and Bob
- Medium Access Control (MAC) is based on IEEE802.11, i.e. CSMA/CA
Load 1: w/o NC the relay sends twice as much as Alice and Bob. No impact on the throughput.
IEEE802.11 Alice & Bob Throughput Model

Load 2:
- w/o NC the relay sends twice as much as Alice and Bob and channel capacity is reached. Still no impact on the throughput.
IEEE802.11 Alice & Bob Throughput Model

Load 3: w/o NC Alice and Bob are „stealing“ the capacity from the relay. 802.11 fairness destroys the performance of the system.
IEEE802.11 Alice & Bob Throughput Model

Load 3:
\( w/o \) NC Alice and Bob are "stealing" the capacity from the relay. 802.11 fairness destroys the performance of the system.

33% 33% 33%
IEEE802.11 Alice & Bob Throughput Model

Load 1:
- w NC the relay sends the same amount of data as Alice or Bob.
IEEE802.11 Alice & Bob Throughput Model

Load 2:

w NC channel capacity is NOT reached

![Graph showing expected throughput and coding gain with and without network coding]
IEEE802.11 Alice & Bob Throughput Model

Load 3:
When NC Alice, Bob and the relay live in perfect harmony, each of the entities requests one third of the capacity.
Load 4:
With NC Alice, Bob and the relay live in perfect harmony now the channel capacity is reach and therefore the throughput remains constant.
Alice and Bob: Activity Chart

4 send
4 receive
rest idle

3 send
4 receive
rest idle
Send Activity Model

Send Activity With Network Coding

Send Activity Without Network Coding

Activity vs. Load Factor [l]

Alice or Bob
Relay
Receive Activity Model

![Graphs showing receive activity with and without network coding](image-url)
Idle Activity Model

Idle Activity With Network Coding

$\text{Activity}$ vs $\text{Load Factor \left[ i \right]}$

- Alice or Bob
- Relay

Idle Activity Without Network Coding

$\text{Activity}$ vs $\text{Load Factor \left[ i \right]}$

- Alice or Bob
- Relay
## Activity Model

<table>
<thead>
<tr>
<th>Phase</th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load ([l])</td>
<td>0-0.5</td>
<td>0.5-0.6</td>
<td>0.6-1.0</td>
</tr>
</tbody>
</table>

### Send \([\alpha_s]\)

<table>
<thead>
<tr>
<th>WoNC</th>
<th>A&amp;B</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>(l)</td>
<td>(1 - l)</td>
<td>(\frac{1}{3})</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NC</th>
<th>A&amp;B</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>(\frac{1}{2}l)</td>
<td>(\frac{1}{2}l)</td>
<td>(\frac{1}{3})</td>
<td></td>
</tr>
</tbody>
</table>

### Receive \([\alpha_r]\)

<table>
<thead>
<tr>
<th>WoNC</th>
<th>A&amp;B</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{6})</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>(l)</td>
<td>(\frac{1}{2}l)</td>
<td>(\frac{2}{3})</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NC</th>
<th>A&amp;B</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{2}l)</th>
<th>(\frac{1}{3})</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>(l)</td>
<td>(l)</td>
<td>(\frac{2}{3})</td>
<td></td>
</tr>
</tbody>
</table>

### Expected Throughput, Alice and Bob Topology

- **Without Network Coding**
- **With Network Coding**

### Coding Gain

- **Total Offered Load**
- **Coding Gain**
## Activity Model

<table>
<thead>
<tr>
<th>Phase</th>
<th>Send [$\alpha_s$]</th>
<th>Receive [$\alpha_r$]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WoNC</strong></td>
<td><strong>A&amp;B</strong></td>
<td><strong>R</strong></td>
</tr>
<tr>
<td>Load [$l$]</td>
<td>0.0-0.5</td>
<td>0.5-0.6</td>
</tr>
<tr>
<td>Activity</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{2}l$</td>
</tr>
<tr>
<td><strong>NC</strong></td>
<td><strong>A&amp;B</strong></td>
<td><strong>R</strong></td>
</tr>
<tr>
<td>Activity</td>
<td>$\frac{1}{2}l$</td>
<td>$l$</td>
</tr>
</tbody>
</table>

### Graph

Send Activity With Network Coding

- Activity vs. Load Factor [$l$]
- Three regions: Alice or Bob, Relay

**Graph Details**
- Axes: Activity, Load Factor [$l$]
- Colors: Activity levels in different load factors

---

**Source:**
- Prof. Dr.-Ing. Dr. h.c. Frank H.P. Fitzek
- Practical Implementations of Network Coding Lecture
- Technische Universität Dresden, Deutsche Telekom Chair of Communication Networks
### Activity Model

<table>
<thead>
<tr>
<th>Send $[\alpha_s]$</th>
<th>Phase</th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>WoNC</td>
<td>A&amp;B</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{3}$</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>$l$</td>
<td>$1 - l$</td>
<td>$\frac{1}{3}$</td>
</tr>
<tr>
<td>NC</td>
<td>A&amp;B</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{3}$</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{3}$</td>
</tr>
<tr>
<td>Receive $[\alpha_r]$</td>
<td>WoNC</td>
<td>A&amp;B</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1-l}{2}$</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>$l$</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{6}$</td>
</tr>
<tr>
<td></td>
<td>A&amp;B</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{3}$</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>$l$</td>
<td>$\frac{1}{2}l$</td>
<td>$\frac{1}{3}$</td>
</tr>
</tbody>
</table>

### Send Activity Without Network Coding

- Alice or Bob
- Relay

**Graph**: Load Factor $[l]$ vs. Activity

---

**TECHNISCHE UNIVERSITÄT DRESDEN**

Prof. Dr.-Ing. Dr. h.c. Frank H.P. Fitzek
Practical Implementations of Network Coding Lecture
Technische Universität Dresden, Deutsche Telekom Chair of Communication Networks
Power Model

In order to derive total power we sum up the product of the activity level \( a \) and power level \( P \) of the individual states.

\[
P_{total} = P_r \cdot a_r + P_s \cdot a_s + P_i \cdot a_i
\]
Power Model

Out of our measurements for WHITEBOX we derive:

— Total power level sending: \( P_{\text{send}} = 3.48 \text{ W} \)
— Total power level receiving: \( P_{\text{receive}} = 3.24 \text{ W} \)
— Total power level idle: \( P_{\text{idle}} = 2.94 \text{ W} \)

Later we assume a maximum capacity of the channel of 4.9 Mbit/s
Power Rate Model

![Graph showing system power vs. offered load with and without network coding]
Energy Model

\[
\text{Energy} = \text{Power} \times \text{Time}
\]

\[
\text{Energy per bit} = \frac{\text{Power}}{\text{Throughput}} = \frac{\text{Joule}}{\text{bit}}
\]

- power
- throughput
- energy/bit
Energy per bit = \( \frac{\text{Power}}{\text{Throughput}} = \frac{\text{Joule}}{\text{bit}} \)
CATWOMAN Measurement Results
CATWOMAN: Results

Aggregated Throughput vs. Total Offered Load

Throughput [kb/s]

0 1000 2000 3000 4000 5000 6000

0 500 1000 1500 2000 2500 3000 3500

Without Network Coding
With Network Coding

Total Offered Load [kbit/s]

Coding Gain

0 1.0 1.2 1.4 1.6 1.8 2.0

0 1000 2000 3000 4000 5000 6000
Analytical results

Expected Throughput, Alice and Bob Topology

Measurement results

Aggregated Throughput vs. Total Offered Load

Throughput [kb/s]

Coding Gain

Total Offered Load [kbit/s]
Discussion

Results fit nicely
But
— The coding gain is with 1.6 lower than expected with 2.0
— The throughput of NC is not stable after reaching its maximum
Why?
CATWOMAN: Results

Alice

Bob

Throughput vs. Total Offered Load for Alice

Throughput vs. Total Offered Load for Bob
CATWOMAN: Results

**Coding Gain vs. Link Share**

\[ g_C = \frac{N}{N \cdot x + N((1-x) - x)} = \frac{1}{1-x}, \quad 0 \leq x \leq 0.5. \]
CATWOMAN: Results
First Measurement Result

- **System Energy**
  - Energy (J) vs. Offered Load (Kbit/s)
  - Comparison: With Network Coding vs. Without Network Coding

- **System Throughput**
  - Throughput (Kbit/s) vs. Offered Load (Kbit/s)
  - Comparison: Without Network Coding vs. With Network Coding

- **System Energy Per Bit**
  - Energy [μJ/bit] vs. Offered Load (Kbit/s)
  - Comparison: With Network Coding vs. Without Network Coding

- power
- throughput
- energy/bit
Comparison: Energy

![System Energy Graph](image)

**System Energy**

- **Energy [J]**
  - With Network Coding
  - Without Network Coding

**System Energy Model**

- **Energy [J]**
  - With Network Coding
  - Without Network Coding

**Offered Load [Kbit/s]**

- 0
- 1000
- 2000
- 3000
- 4000
- 5000

---

**Technische Universität Dresden, Deutsche Telekom Chair of Communication Networks**
Comparison: Throughput

- System Throughput
  - Without Network Coding
  - With Network Coding

- System Throughput Model
  - Without Network Coding
  - With Network Coding
Comparison: Energy Per Bit

![Graph showing comparison of energy per bit with and without network coding.](graph.png)
Model for Alice and Bob (asymmetric traffic)

Model for asymmetric traffic (vs. symmetric traffic): SEND

- w/o NC
- w NC

Asymmetric with link share of 60%(A)/40%(B)
Model for asymmetric traffic (vs. symmetric traffic): RECEIVE

- w/o NC
  - Receive Activity Without Network Coding
  - Activity vs. Total Offered Load
  - Line colors: Alice, Bob, Relay

- w NC
  - Receive Activity With Network Coding
  - Activity vs. Total Offered Load
  - Line colors: Alice, Bob, Relay

Asymmetric with link share of 60% (A)/40% (B)
Model for asymmetric traffic (vs. symmetric traffic): IDLE

- **w/o NC**
  - Asymmetric with link share of 60% (A)/40% (B)

- **w NC**
  - Asymmetric with link share of 60% (A)/40% (B)
Model for THE CROSS (symmetric traffic)
Cross Forwarding

Whatever goes into the relay has to be forwarded.

What to do if there is not enough capacity?
XOR Network Coding

- Each out node sends to the relay
- And for each pair the relay sends out one coded packet
XOR Network Coding with overhearing

- Each outer node sends a packet to the relay
- Each outer node will overhear two packets from neighboring nodes
- Relay sends out one full coded packet
Cross Throughput

![Diagram showing the relationship between throughput and total load for different strategies: overhearing, coding, and pure relaying.](image)
Cross Throughput
The Cross (MAC)

Hardware
16bit PIC24 microprocessor
nRF905 transceiver (433 MHz, 50 kbps)

Software
MAC-Protocol: A CSMA/CA design
Network Coding: A simple XOR design [COPE06]

Capability
Easy access to the software
Full control of both HW and SW
The Cross (MAC)

Expected vs. Measurement

Throughput vs. Total Offered Load [kbit/s]
Analog Network Coding
Overview

• So far network coding was done in the packet domain
• Network coding can be applied in any ISO/OSI layer
• Analog and physical network coding is a special case at the physical layer (lowest ISO/OSI layer) coding “symbols”
• Coding symbols is nothing else than a superposition of signals. Analog network coding breaks with the paradigm to separate signals in time and force “collision” to achieve higher coding gains
Analog Network Coding for Wireless Networks

Conventional relaying
4 time slots
3 sinks

Use of network coding
3 time slots
3 sinks

Use of analog network coding
2 time slots
2 sinks
Analog Network Coding for Wireless Networks

Analog network coding seems to be more efficient than digital network coding.

For the two way relay it reduces the number of necessary transmissions to two (three for digital network coding and four for store and forward).

Advantages
- Throughput
- Energy
- Security (role of the relay differs)
Physical Layer Network Coding

Presented by [Zhang et al 2006]

First, simple example: no fading
Let us look at bandpass signals

\[ r_3(t) = s_1(t) + s_2(t) + n(t) \]
\[ = [a_1 \cos(\omega t) + b_1 \sin(\omega t)] + [a_2 \cos(\omega t) + b_2 \sin(\omega t)] + n(t) \]
\[ = (a_1 + a_2) \cos(\omega t) + (b_1 + b_2) \sin(\omega t) + n(t) \]

How to generate \( s_3(t) \)?
Physical Layer Network Coding

How to generate $s_3(t)$?
Amplify and forward?
Decode and forward?
Initial approach: decode and forward

**Example with BPSK:** say $b_i = 0, \ a_i \in \{-1,1\} \ i = 1,2,3$

$$r_3(t) = s_1(t) + s_2(t) + n(t) = (a_1 + a_2)\cos(\omega t) + n(t)$$

Note that there are 3 possible values of $a_1 + a_2$:
- "-2" and "2" correspond to $a_1 = a_2$
- "0" corresponds to $a_1 = -a_2$
Physical Layer Network Coding

Example with BPSK:

Let us generate $s_3(t)$
(hint: XOR-like operation)

If $a_1 = a_2$ then $a_3 = 1$ (logical "0")
If $a_1 = -a_2$ then $a_3 = -1$ (logical "1")

Alice and Bob receive as standard BPSK modulation

Then, XOR bit by bit with the sent packet
Physical Layer Network Coding

What if we A&F?

1st step – coding in the air

Alice may send two symbols

Bob may send two symbols

Alice

Relay

Bob

BPSK Example
Physical Layer Network Coding

What if we A&F?

1st step – coding in the air

Relay may receive three symbols

Alice  Relay  Bob
Physical Layer Network Coding

1st step – coding in the air – e.g. 1/1

Alice  Relay  Bob
Physical Layer Network Coding

1st step – coding in the air – e.g. 0/0
Physical Layer Network Coding

1st step – coding in the air – e.g. 1/0
Physical Layer Network Coding

1st step – coding in the air – e.g. 0/1
Physical Layer Network Coding

2nd step - relay

Alice  Relay  Bob

50%  50%  50%
Physical Layer Network Coding

2nd step - relay

Alice  Relay  Bob
Physical Layer Network Coding

3rd step decoding

Alice

Alice

Alice
Physical Layer Network Coding

3rd step decoding

Alice

Alice

Alice
Analog Network Coding

What were our assumptions so far?
• No fading → there is amplitude + phase distortion
• Perfect sync
• Perfect detection of a collision
• Perfect knowledge of packet used for decoding at Alice and Bob
• The “right” packets interfere (MAC / Network impact)

How to make it practical?
[Katti et al 2007] Analog network coding
[Gollakota et al 2008] ZigZag decoding
(different problem, similar intuition)
Analog Network Coding

Key intuition: exploit asynchrony [Katti et al 2007]
Analog Network Coding

Pilot sequence: channel estimation ID of sender + destination + sequence number of the packet: active session and to determine which packet was used.

Construct „header“ and „footer“ for each packet.

No overlap

Alice

Bob

No overlap
Analog Network Coding
ZigZag Decoding

Draws from the same intuition as the above problem

Difference:

• More general setting
• A node can use it to recover several interfering signals (no knowledge required on its end)
• We need to receive $n$ collisions of $n$ packets to recover

Where is it useful?

• Hidden terminal problem
• In high SNR, to boost overall data rate from multiple sources to a single receiver [ParandehGheibi et al 2010]
ZigZag Decoding: Basic Idea

Again: asynchrony
Chunk 1 of bits from user A from 1st collision is decoded successfully
Thus, can subtract it from 2nd collision to decode Chunk 2 of bits of user B
Once Chunk 2 is free, can use to free Chunk 3, and so on
ZigZag Decoding: Single Hop Analysis

Work in [ParandehGheibi et al 2010]

Diagram showing ZigZag Decoding with single hop analysis.
ZigZag Decoding: Single Hop Analysis

Work in [ParandehGheibi et al 2010]

First result: Mean time to deliver one packet each

With zigzag:

\[
\frac{n}{1 - p^n} \leq E[T_D] \leq \sum_{i=1}^{n} \frac{1}{1 - p^{n-i+1}}
\]

Perfect scheduler (no collisions):

\[
E[T_D] = \frac{n}{1 - p}
\]

\[p = \frac{1}{2}, \ n = 3\]

ZZ: 4+

10/21

PS: 6
ZigZag Decoding: Single Hop Analysis

Work in [ParandehGheibi et al 2010]

Second result: Stable throughput increases

Coded Access
Multiple Access SoA

Random access was the key to get access to the wireless channel

In time slotted systems
  — Idle slots: nobody was transmitting
  — One transmission: successful access to the resources
  — More than one transmission: Collision
Coded Access
Our Analog Network Coding Testbed

GNU Radio, from design to deployment
• With GNU Radio you can simulate, prototype, and deploy, all from the same workflow

Active Community

Free Software
• GNU Radio is free software. That means you have the liberty to use it and modify it as you wish
• [https://www.gnuradio.org/](https://www.gnuradio.org/)
ANC for cross topologies
Cross Topology ANC without Overhearing

A

C

R

B

D
Cross Topology ANC without Overhearing – Slot 1

Diagram:
- Nodes: A, B, C, D, R
- Connections:
  - A to R
  - R to A
  - R to B
  - B to R
  - C to A
  - D to A

Legend:
- Node A
- Node B
- Node C
- Node D
- Node R
Cross Topology ANC without Overhearing – Slot 2
Cross Topology ANC without Overhearing – Slot 3

Diagram showing nodes A, B, C, and D in a cross topology network setup.
Cross Topology ANC without Overhearing – Slot 4

4 Time Slot for full exchange
Cross Topology ANC with Overhearing

Overhearing the activities of the neighboring node
Cross Topology ANC with Overhearing – Slot 1
Cross Topology ANC with Overhearing – Slot 2
Cross Topology ANC with Overhearing – Slot 3

3 Time Slot for full exchange
Cross Topology ANC with Overhearing Duplex

Overhearing the activities of the neighboring node
Cross Topology ANC with Overhearing Duplex – Slot 1
Cross Topology ANC with Overhearing Duplex – Slot 2

2 Time Slot for full exchange
Throughput for the Cross Topology

![Diagram showing throughput for different network topologies and load conditions.]

- ANC duplex
- ANC w OH
- ANC w/o OH
- DNC w OH
- DNC w/o OH
- Forwarding
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Andreas Müller & Timo Lothspeich
Robert Bosch GmbH, Germany

Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Andreas Müller & Timo Lothspeich
Robert Bosch GmbH, Germany
Inter-Flow Network Coding on CAN Bus

**Outline**

1. Motivation
2. Fundamental Idea
3. Implementation Aspects
4. Security Considerations
5. Conclusion

*Courtesy of BOSCH*
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Motivation

- Current trends (e.g., Cloud/Internet connectivity) lead to novel & serious security threats
- Today’s CAN networks are often hardly secured
- Cryptographic methods may help (e.g., message / entity authentication)

**But:** Distribution of cryptographic keys between devices as a major challenge

Our Idea: Novel approach for completely **automated & secure** key distribution of very **low complexity** for CAN networks (“plug-and-secure”)

- Potential building block and enabler for future secure CAN networks
- Especially suitable against software-based & remote attack scenarios
- Basic idea: Exploit special properties of CAN bus (dominant / recessive bits)

© BOSCH - CRUZH - Andreas Mauser | 25 October 2015 | © Robert Bosch GmbH 2015. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Major Benefits

- Low complexity & low cost
- Simplicity / Ease-of-Use
- Easy & scalable re-keying
- Easy key individualization
- Confidentiality, Authenticity, Integrity
- ProS for CAN, Crypto
- General security enabler

Seamless integration in CAN ecosystem, simple add-on to existing CAN controllers sufficient
Approach may be readily extended to other bus systems, such as LIN², IPO, etc.

Plug-and-Secure Communication for CAN

BOSCH
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Implementation Aspects
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Bit Stuffing

After 5 equal bits in a frame, a stuff bit has to be introduced

Alice

Bob

\( S_{Alice} = 10101010 \)

Bit stuffing requirement satisfied

\( S_{Bob} = 01010101 \)

\( S_{eff} = 00000000 \)

Bit stuffing requirement violated in superimposed sequence on bus

Insert stuffing bits on-the-fly based on effective bit sequence on the bus
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Bit Stuffing

⚠️ After 5 equal bits in a frame, a stuff bit has to be introduced

Alice

Bob

$S_{Alice} = 10101$

$S_{Bob} = 01010$

$S_{eff} = 00001$

1 1

1 1

Insert stuff bit

Insert stuff bit on-the-fly based on effective bit sequence on the bus
Inter-Flow Network Coding on CAN Bus

*Courtesy of BOSCH*

**Plug-and-Secure Communication for CAN**

**Bit Stuffing**

After 5 equal bits in a frame, a stuff bit has to be introduced.

![Diagram](image)

$S_{\text{Alice}} \rightarrow = 1010101010$

$S_{\text{Bob}} \rightarrow = 0101010101$

$S_{\text{eff}} = 0000010000$

Bit stuffing requirement satisfied 🎉

Insert stuffing bits on-the-fly based on effective bit sequence on the bus.
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

CRC Check

- Cyclic redundancy check field enables detection of transmission errors
- Superimposed CRC fields → correct CRC for superimposed payloads

Combination of CRCs does generally not yield a valid CRC for combined payload bits

(Bosch)
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

CRC Check

Cyclic redundancy check field enables detection of transmission errors → Superimposed CRC fields ⇒ correct CRC for superimposed payloads

(Random) Payload $P_A$\(\text{CRC (P_A AND P_B)}\)

Valid effective CAN message on the bus

Provides automatic verification that Alice and Bob have received the same superimposed bit sequence for free ©

Technical University of Dresden, Deutsche Telekom Chair of Communication Networks
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Security Considerations
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Remote Attacker Model

Victim of a remote attack

Assumptions

1. Eve is using standard HW with modified (malicious) SW
2. Eve may eavesdrop on all messages exchanged on the CAN bus
3. Eve may inject arbitrary bits on the CAN bus (via the CAN transceiver)

Highly relevant attacker model due to easy scalability of attacks!
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Passive Eavesdropping

<table>
<thead>
<tr>
<th>Transmitted Bits</th>
<th>Observed Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice 01</td>
<td>Bob 01</td>
</tr>
<tr>
<td>Bob 01</td>
<td>Eve 01</td>
</tr>
<tr>
<td>Alice 01</td>
<td>Bob 10</td>
</tr>
<tr>
<td>Bob 10</td>
<td>Eve 00</td>
</tr>
<tr>
<td>Alice 10</td>
<td>Bob 01</td>
</tr>
<tr>
<td>Bob 01</td>
<td>Eve 00</td>
</tr>
<tr>
<td>Alice 10</td>
<td>Bob 10</td>
</tr>
<tr>
<td>Bob 10</td>
<td>Eve 10</td>
</tr>
</tbody>
</table>

Eve has full knowledge of bits transmitted by Alice & Bob, but bits will be discarded anyway
Eve is not able to tell who has transmitted '01' and who has transmitted '10' → secret 😊
Eve has full knowledge of bits transmitted by Alice & Bob, but bits will be discarded anyway

 Transmission of tuples (random bit + inverse bit) → Observation of regenerated bits only (through CAN transceiver)

A passive Eve cannot determine the established secret bits 😊
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Active Attacks (I)

Idea: Actively interfere with key establishment procedure

By superimposing recessive bits

<table>
<thead>
<tr>
<th>Transmitted Bits</th>
<th>Eve</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali</td>
<td>Bob</td>
<td>Eve</td>
</tr>
<tr>
<td>0     0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0     1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1     1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

No different from not transmitting at all → no impact

By superimposing dominant bits

<table>
<thead>
<tr>
<th>Transmitted Bits</th>
<th>Eve</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali</td>
<td>Bob</td>
<td>Eve</td>
</tr>
<tr>
<td>00   00</td>
<td>0    00</td>
<td></td>
</tr>
<tr>
<td>01   10</td>
<td>0    00</td>
<td></td>
</tr>
<tr>
<td>?1  ?0</td>
<td>01   00</td>
<td></td>
</tr>
</tbody>
</table>

Requires a closer look
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Attacker Model w/ Physical Access to CAN Bus

Direct access to the CAN bus with own hardware (e.g., oscilloscope)

Principle Threats

- Direct physical access may enable more sophisticated attacks (e.g., exploitation of timing or attenuation effects)
- Detailed security analysis still ongoing

BUT:

- With physical access, an attacker could compromise a vehicle much easier (e.g., cutting a cable)
- Attacks requiring physical access do not scale; threat w/ physical access has existed ever since
- Countermeasures are possible → e.g., artificial (random) jitter in bit timing
Inter-Flow Network Coding on CAN Bus

*Courtesy of BOSCH*

**Plug-and-Secure Communication for CAN**

**Proof-of-Concept Demonstration**
Inter-Flow Network Coding on CAN Bus

*Courtesy of BOSCH*

**Plug-and-Secure Communication for CAN**

**Proof-of-Concept Demonstrator (I)**

- Alice
- Bob
- Eve

Portable demonstrator including Alice & Bob

Bus analyzer emulating attacker Eve

**ComNets**
Inter-Flow Network Coding on CAN Bus
Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Proof-of-Concept Demonstrator (II)

- Basic idea successfully demonstrated using off-the-shelf (maker) hardware
- 100% standard compliant CAN frames on the bus → full backwards-compatibility

Courtesy of BOSCH
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH

Plug-and-Secure Communication for CAN

Conclusion

1. Security is getting more and more important in many domains
2. Distribution of cryptographic keys as a major challenge
3. PnS\textsuperscript{4} for CAN as an innovative approach of very low complexity
4. Basic idea successfully demonstrated, fully backwards compatible
5. Major strengths: Remote / SW-based attacks, re-keying, wide applicability

Possible Extensions
- Extension to other bus systems with similar PHY rather straightforward

Many Applications
- Numerous applications in automotive, industrial & other CAN networks

Your Input
- Interested?
- Get in touch with us

---

CRUISE - Andreas Maurer | 25 October 2015 | © Robert Bosch GmbH 2015. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Inter-Flow Network Coding on CAN Bus

Courtesy of BOSCH
Summary Inter-Flow Network Coding

• Advantages
  • Easy to understand
  • Huge gains in theory

• Disadvantages
  • Requires signaling/planning
    • Not a huge problem in static networks, e.g. SDN controller
    • Huge problem for dynamic networks, more load and inefficient if not in time

• Real implementation (more than lab test run)
  • CATWOMAN in LINUX Kernel 3.10 onwards
  • BOSCH CAN Bus
Backup
Cautious View

Janus Heide, Morten V. Pedersen, Frank H.P. Fitzek, and Torben Larsen, “Cautious view on network coding - from theory to practice,” Journal of Communications and Networks (JCN), 2009
**XOR operations**

Simple XOR seems to be no problem with respect to computation.

More energy used for the encoding vector to be transmitted than in the actual coding.

---

**Power and Energy Levels Used for Network Coding on the Mobile Phone**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>0.593</td>
<td>4,789</td>
<td>25000000</td>
</tr>
<tr>
<td>Coding</td>
<td>0.593</td>
<td>0.191 \cdot 10^{-6}</td>
<td>1</td>
</tr>
<tr>
<td>Idle</td>
<td>0.041</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Wheel Example: Multi-dimension Alice and Bob

Energy Plot

Delay Plot

Scenario: Wheel
• 97 mobile devices in total
• 96 flows (one per cluster)
• Just relaying to exchange

F number of flows
T number of teams

<table>
<thead>
<tr>
<th></th>
<th>sending</th>
<th>receiving</th>
<th>idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single node</td>
<td>1</td>
<td>F/T-1</td>
<td>F*(T-1)/T +T</td>
</tr>
<tr>
<td>Outer nodes</td>
<td>F</td>
<td>(F/T-1)F</td>
<td>(F(T-1)/T +T)F</td>
</tr>
<tr>
<td>Inner node</td>
<td>T</td>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>SUM netcod</td>
<td>F+T</td>
<td>F²/T</td>
<td>(F(T-1)/T +T)F</td>
</tr>
<tr>
<td>SUM relay</td>
<td>2F</td>
<td>2F</td>
<td>2F(F-1)</td>
</tr>
</tbody>
</table>
Wheel Example: Multi-dimension Alice and Bob

Without considering IDLE power value

With considering IDLE power value
Scenario: Wheel++
- 12 mobile devices in total
- Four flows (one per cluster)
- Each device receives cellular input
- Just relaying to exchange

<table>
<thead>
<tr>
<th>Mobile Device Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Slots</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

- **One activity matrix**
  - Sending: 5
  - Receiving: 6
  - Idle: 4

- **All activity matrices**
  - Sending: 20
  - Receiving: 24
  - Idle: 16

- **All idle matrices**
  - Sending: 0
  - Receiving: 0
  - Idle: 180

**SUM**
- Sending: 20
- Receiving: 24
- Idle: 196
Scenario: Wheel++

- 12 mobile devices in total
- Four flows (one per cluster)
- Each device receives cellular input
- Network coding within each cluster

<table>
<thead>
<tr>
<th>mobile device number</th>
<th>sending</th>
<th>receiving</th>
<th>idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>s</td>
<td>i</td>
<td>r</td>
</tr>
<tr>
<td>2</td>
<td>r</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>3</td>
<td>s</td>
<td>i</td>
<td>r</td>
</tr>
<tr>
<td>4</td>
<td>i</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>5</td>
<td>r</td>
<td>s</td>
<td>r</td>
</tr>
<tr>
<td>6</td>
<td>s</td>
<td>i</td>
<td>r</td>
</tr>
<tr>
<td>7</td>
<td>i</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>8</td>
<td>r</td>
<td>s</td>
<td>r</td>
</tr>
<tr>
<td>9</td>
<td>s</td>
<td>i</td>
<td>r</td>
</tr>
<tr>
<td>10</td>
<td>i</td>
<td>r</td>
<td>s</td>
</tr>
<tr>
<td>11</td>
<td>r</td>
<td>s</td>
<td>r</td>
</tr>
<tr>
<td>12</td>
<td>s</td>
<td>i</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time slots</th>
<th>1 2 3 4 5 6 7 8 9 0</th>
<th>1 2 3 4 5 6 7 8 9 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>s  i  r  r</td>
<td>s  i  r  r</td>
</tr>
<tr>
<td>2</td>
<td>r  r  s  s</td>
<td>i  i  i  i</td>
</tr>
<tr>
<td>3</td>
<td>i  s  r  r</td>
<td>i  s  r  r</td>
</tr>
<tr>
<td>4</td>
<td>s  i  r  r</td>
<td>s  i  r  r</td>
</tr>
<tr>
<td>5</td>
<td>i  r  s  s</td>
<td>i  r  s  s</td>
</tr>
<tr>
<td>6</td>
<td>r  s  s  r</td>
<td>i  i  i  i</td>
</tr>
<tr>
<td>7</td>
<td>s  i  r  r</td>
<td>s  i  r  r</td>
</tr>
<tr>
<td>8</td>
<td>i  r  s  s</td>
<td>i  r  s  s</td>
</tr>
<tr>
<td>9</td>
<td>r  s  s  r</td>
<td>i  i  i  i</td>
</tr>
<tr>
<td>10</td>
<td>i  i  i  i</td>
<td>i  i  i  i</td>
</tr>
<tr>
<td>11</td>
<td>r  s  s  r</td>
<td>i  i  i  i</td>
</tr>
<tr>
<td>12</td>
<td>s  i  r  r</td>
<td>s  i  r  r</td>
</tr>
</tbody>
</table>

- idle slot
- receiving slot
- sending unicast slot
- broadcasting slot
- broadcasting coded slot

<table>
<thead>
<tr>
<th></th>
<th>sending</th>
<th>receiving</th>
<th>idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>One activity matrix</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>All activity matrices</td>
<td>16</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>All idle matrices</td>
<td>0</td>
<td>0</td>
<td>144</td>
</tr>
<tr>
<td>SUM</td>
<td>16</td>
<td>24</td>
<td>152</td>
</tr>
</tbody>
</table>


**Scenario: Wheel++**

- 12 mobile devices in total
- Four flows (one per cluster)
- Each device receives cellular input
- Network coding over all cluster
- Device 2 is doing the most work

<table>
<thead>
<tr>
<th>mobile device number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>time slots</td>
<td>i</td>
<td>s</td>
<td>i</td>
<td>r</td>
<td>i</td>
<td>s</td>
<td>i</td>
<td>r</td>
<td>r</td>
<td>i</td>
<td>s</td>
<td>i</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>sending</th>
<th>receiving</th>
<th>idle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phase 1</strong></td>
<td>4</td>
<td>8</td>
<td>36</td>
</tr>
<tr>
<td><strong>Phase 2</strong></td>
<td>8</td>
<td>62</td>
<td>26</td>
</tr>
<tr>
<td><strong>Phase 3</strong></td>
<td>1</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td><strong>SUM</strong></td>
<td>13</td>
<td>78</td>
<td>65</td>
</tr>
</tbody>
</table>
Results

Energy usage Wheel++ 2 clusters

Normalized energy

- Sending
- Receiving
- Idle

Pure relaying
Coding in teams
Pure coding
Energy and Channel Measurements: Point to point

Measurement Campaign for NC
Unicast Results

Energy vs size vs distance

• Small packets have higher energy per bit ratios due to the MAC overhead
• Small increase for packets larger than MTU size

Data rate vs size vs distance

• Larger data rates for small distances
• Small decrease after MTU sizes