TECHNISCHE UNIVERSITÄT DRESDEN

Netzwerkkodierung in Theorie und Praxis

Praktische Anwendungen der Netzwerkkodierung

Professor Dr.-Ing. Dr. h.c. Frank H.P. Fitzek

M.Sc. Juan Cabrera

Deutsche Telekom Chair of Communication Networks (ComNets)

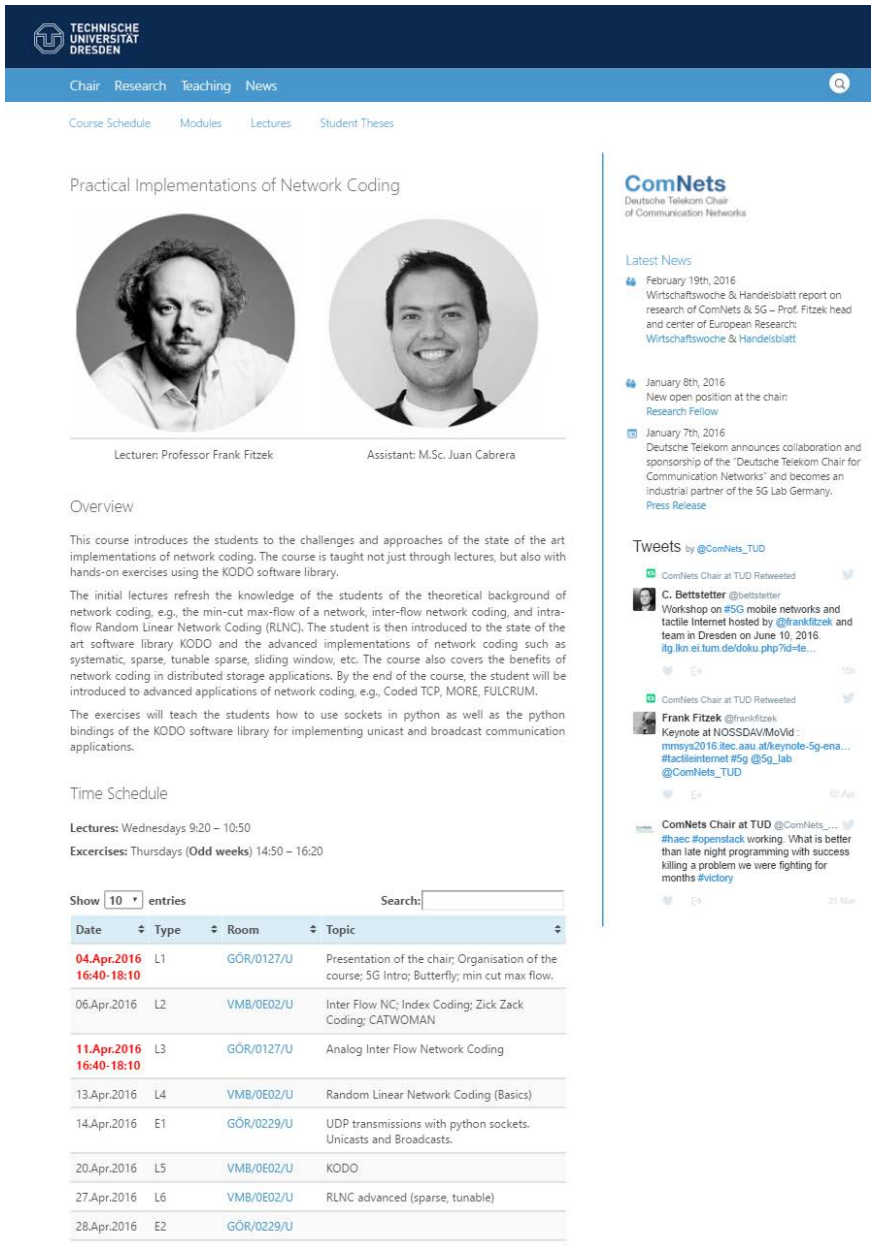
Netzwerkkodierungstheorie

Professor Dr.-Ing. Eduard Jorswieck

Dipl.-Ing. Johannes Richter

Theoretische Nachrichtentechnik

Lecture / Exercise Dates - tinyurl.com/zooafld



- Here all information for the lecture and the exercise can be found.
- Slides
- Links
  - Steinwurf
  - Python
  - KODOMARK (google play)

Please check every week!

TECHNISCHE UNIVERSITÄT DRESDEN

Coded TCP

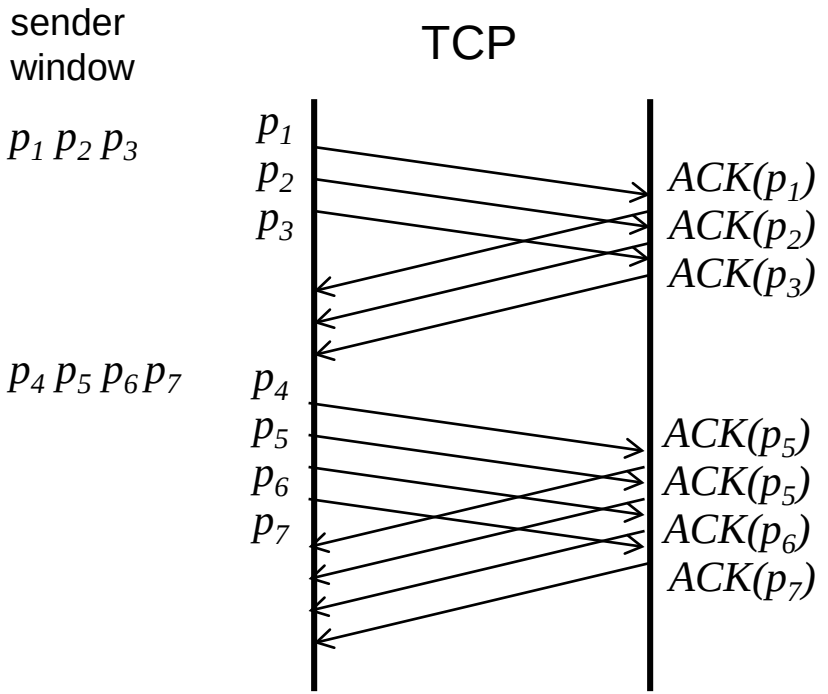
http://arxiv.org/abs/1212.2291 Medard and team

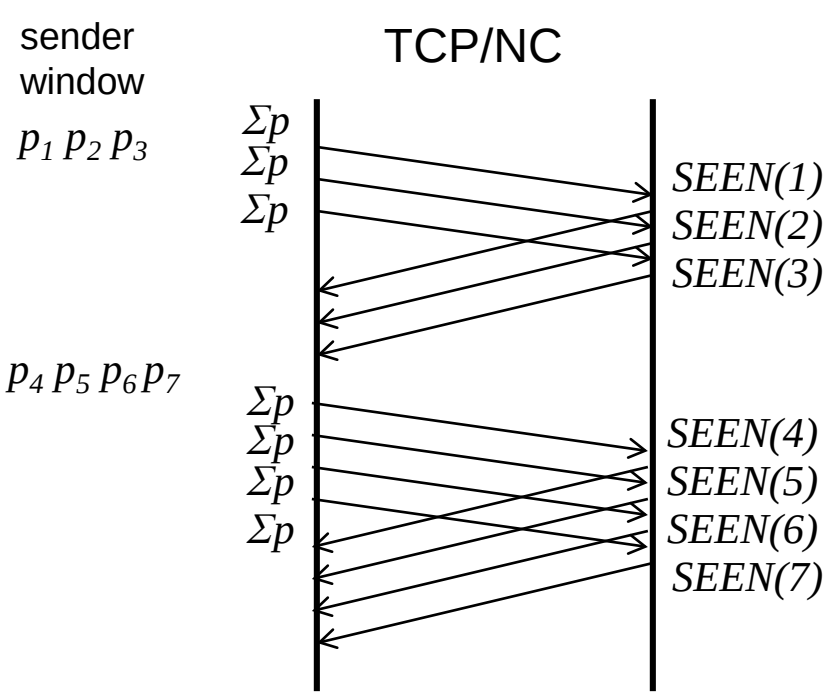
TECHNISCHE UNIVERSITÄT DRESDEN

Coded TCP

- Instead of a single XOR packet, Random Linear Network Coding is used
- XOR of all packets is a “Super Packet” able to heal any loss, but there is only one such packet in GF(2), RLNC with high fields can generate infinitive number of “Super Packets”

Example: No Losses

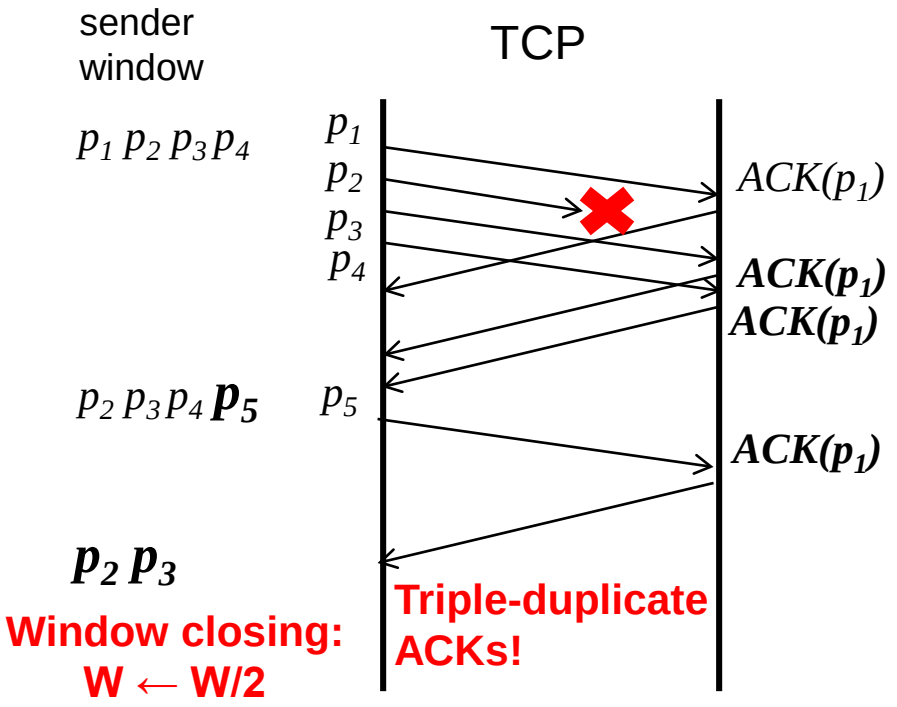


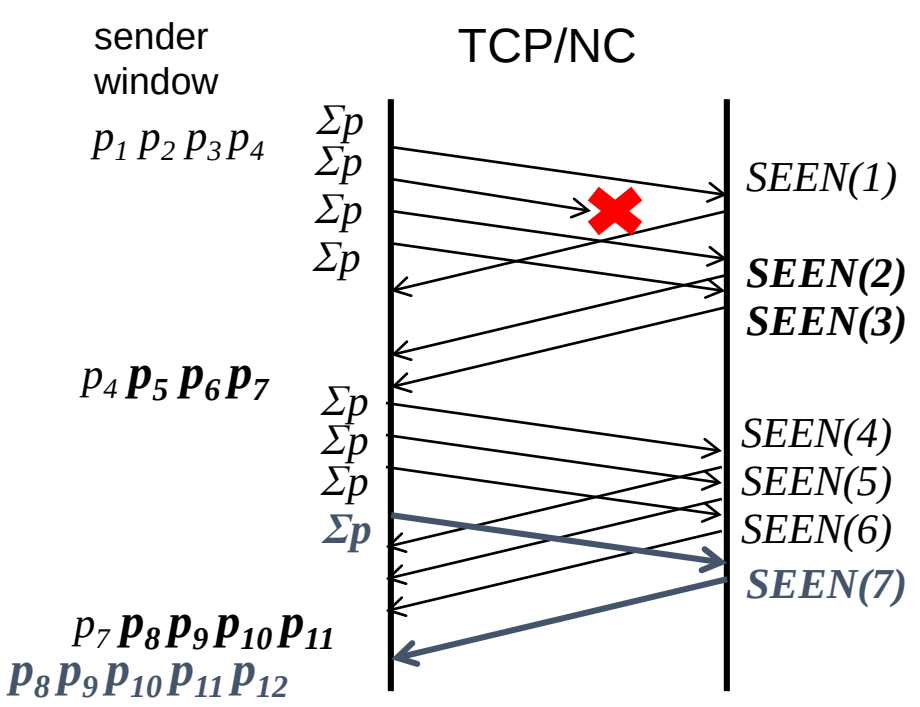


- Increment window by 1
- Sliding window

When no losses, network coding doesn't provide benefits (erasure correction)

Example: Random Losses (Triple-Duplicate ACKs)



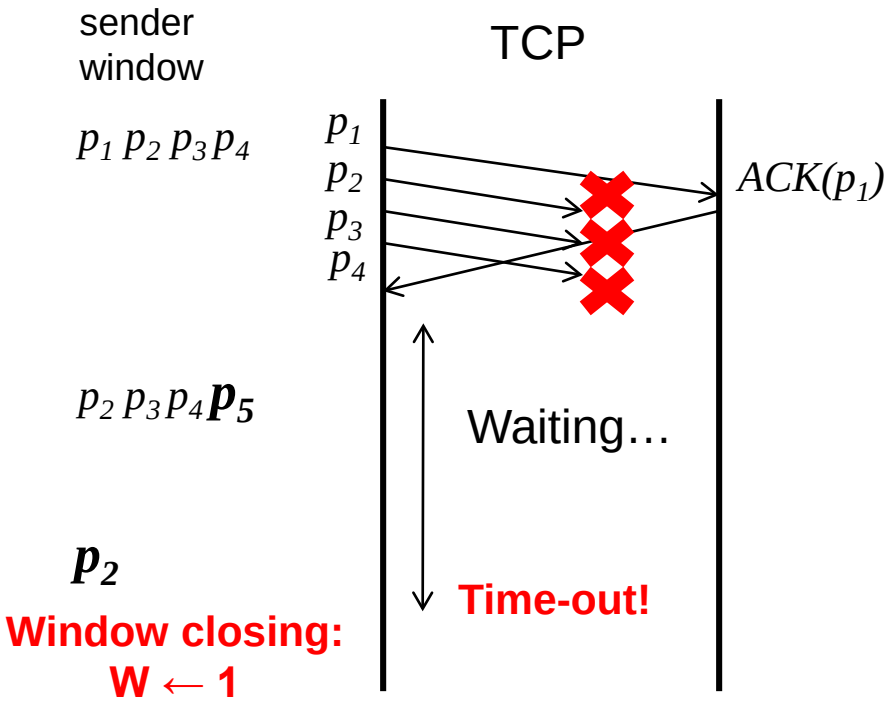


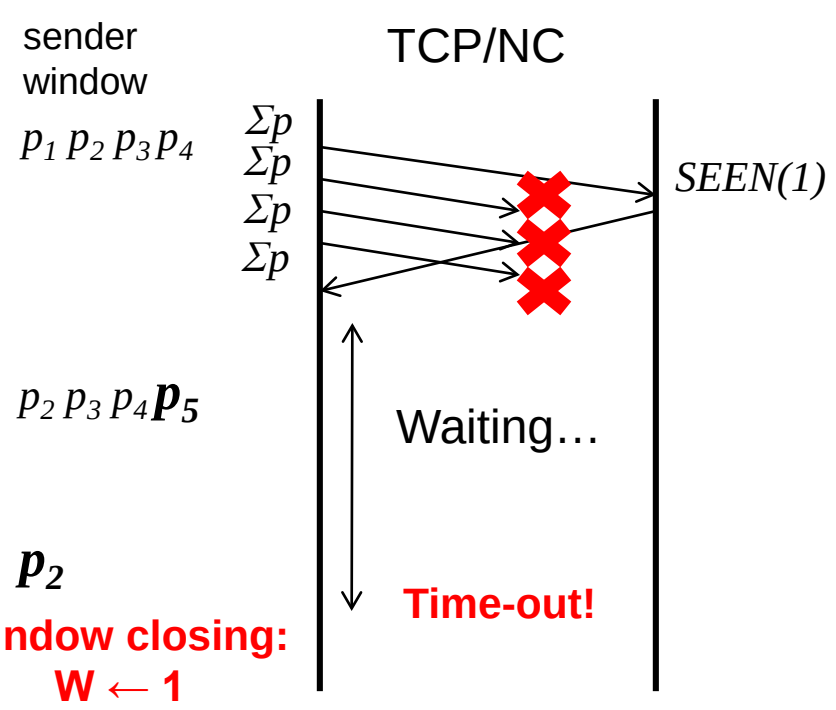
- Can't increment window by 1
- Partial sliding window

Prevents random losses being interpreted as congestion!

There is a lag in the "SEEN" acks: To avoid lag, introduce redundancy!

Example: Congestion Losses (Time-outs)

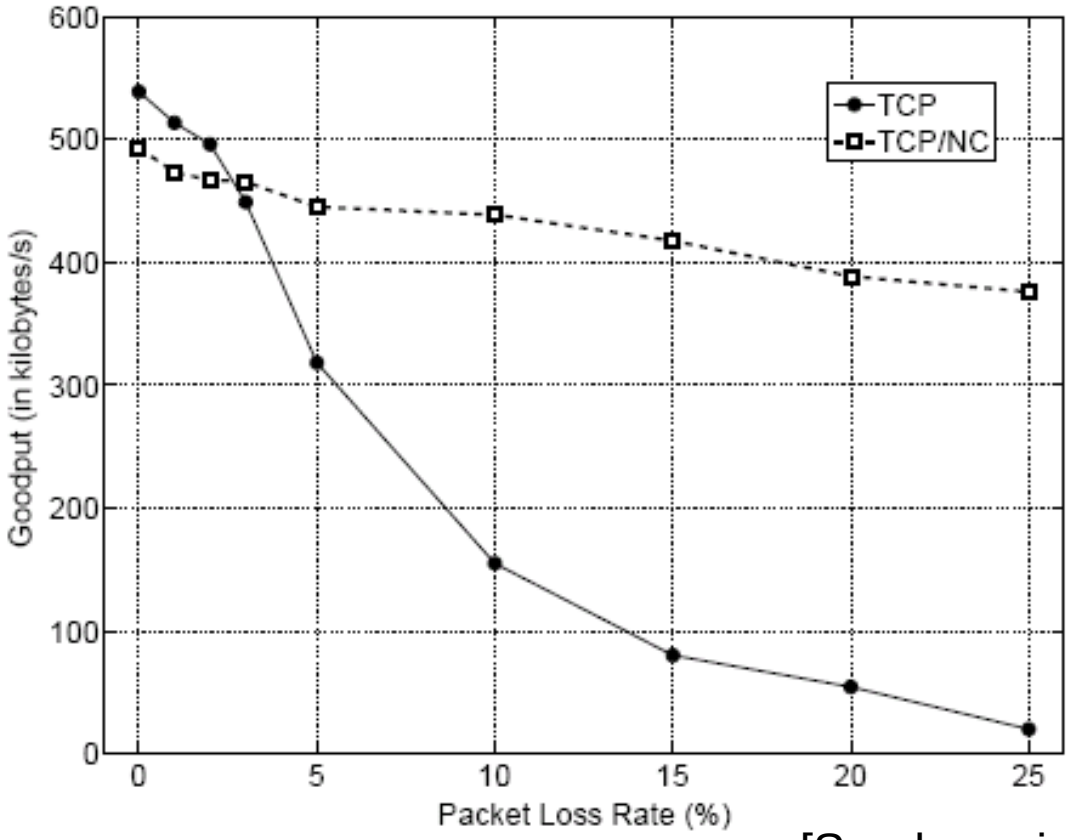




Still allows congestion control while masking random losses!

TECHNISCHE UNIVERSITÄT DRESDEN

Experimental Results (Reno)



[Sundararajan et al 09]

Analysis and Simulations

- Based on Padhye et al.'s model (correlated losses to model congestion).

$$\mathcal{T}_{tcp} = \min\left(\frac{W_{\max}}{RTT}, \frac{1-p}{p}, \frac{1}{RTT\left(\frac{5}{3} + \sqrt{-\frac{1}{18} + \frac{2}{3}\frac{1-p}{p}}\right) + Q(\mathrm{TO}|E[W])E[\text{duration of TO period}]}\right)$$

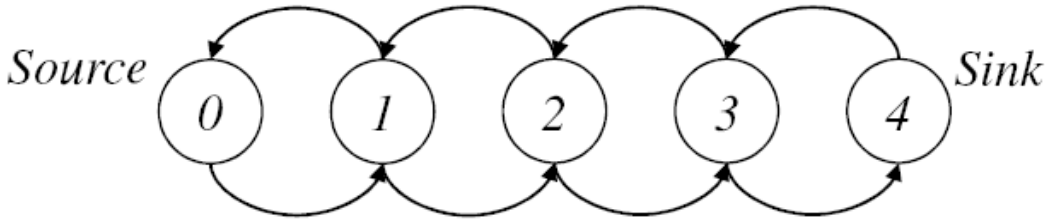
$$\mathcal{T}_{e2e} = \frac{1-p}{nR \cdot SRTT} \cdot f(p),$$

Proportional to *(1-p)*

Does not degrade linearly

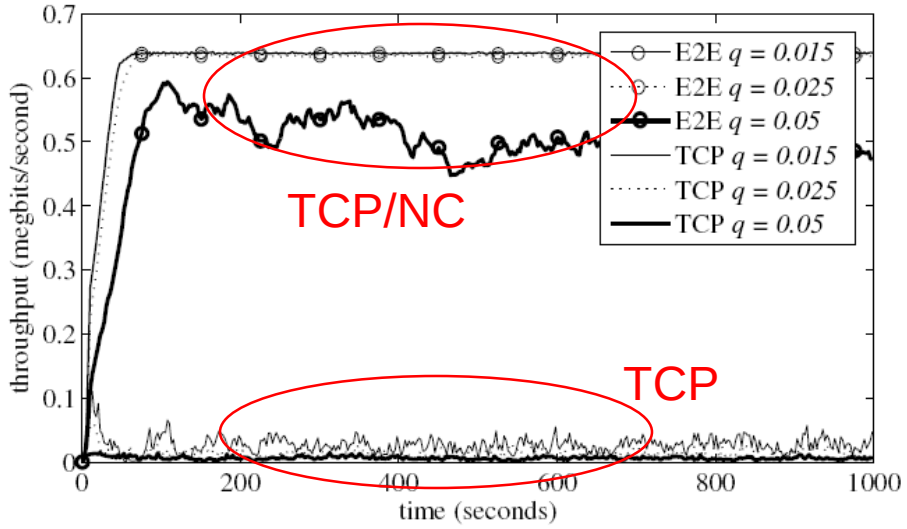
- Simulation using NS-2.

loss rate per link q End-to-end loss rate $1\text{-}p=(1\text{-}q)^4$

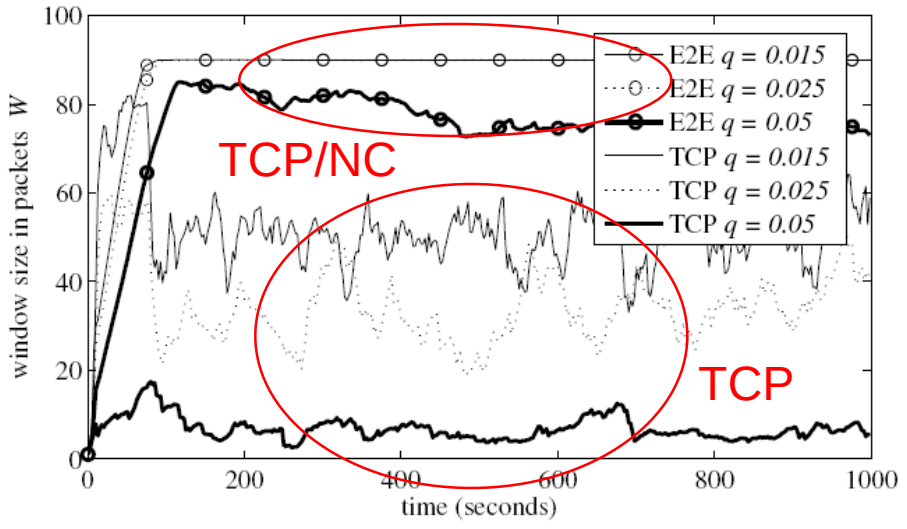


"Modelling TCP Throughput: A Simple Model and its Empirical Validation" by *Padhye*, *Firiou*, Towsley and Kurose (1998)

Throughput Gains Using TCP/NC



(a) Throughput

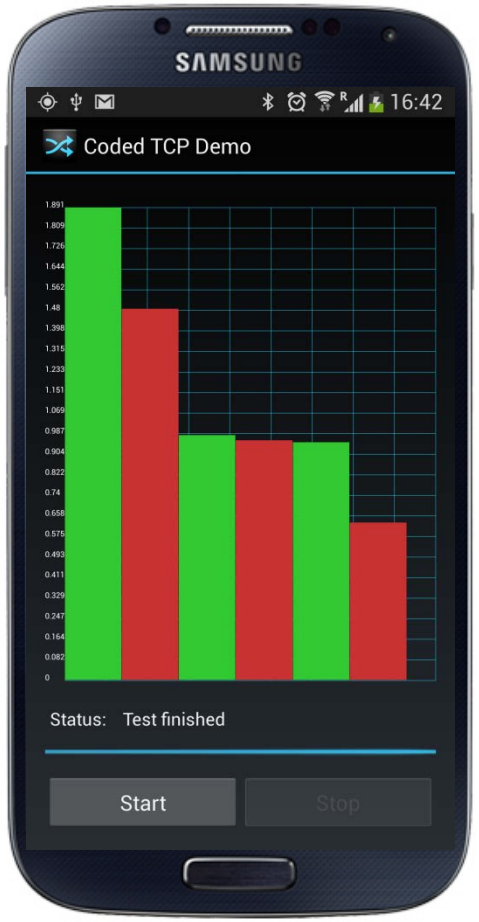


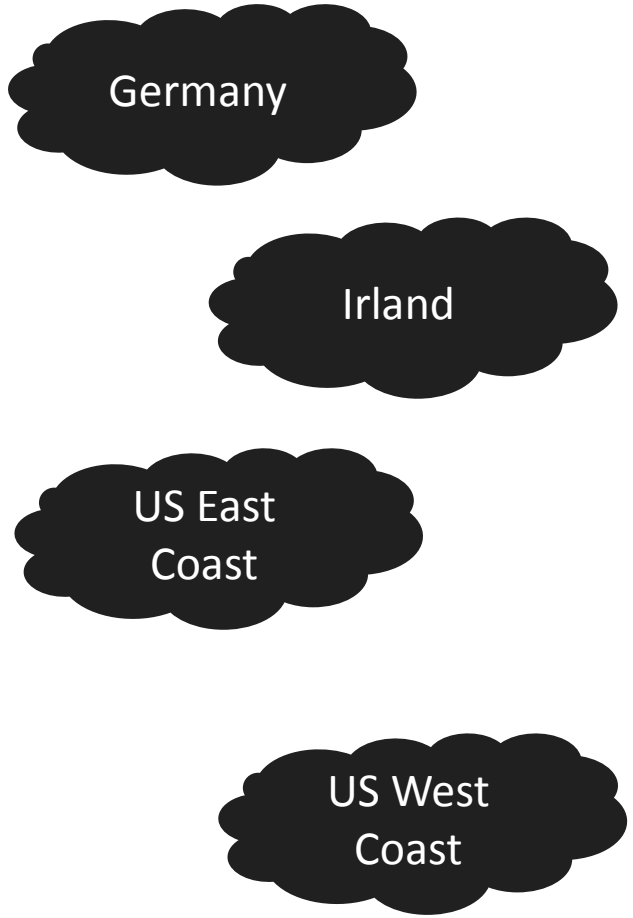
(b) Window size

- TCP/NC is able to grow its throughput and maintain high rate despite losses.

TCP's window size is larger compared to its actual throughput – TCP sender is waiting for ACKs.

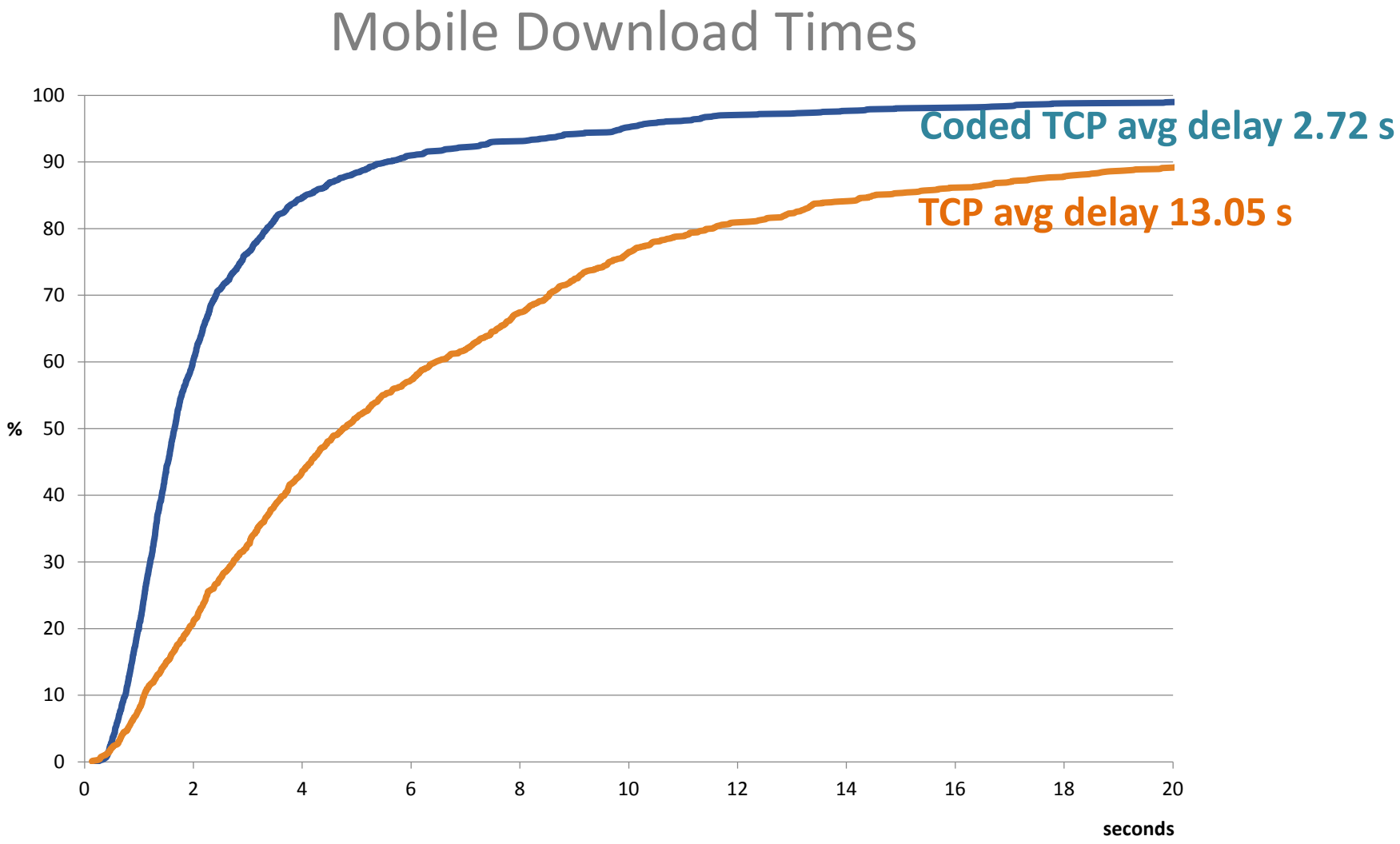
Coded TCP


SAMSUNG
16:42
Coded TCP Demo
Status: Test finished
Start
Stop

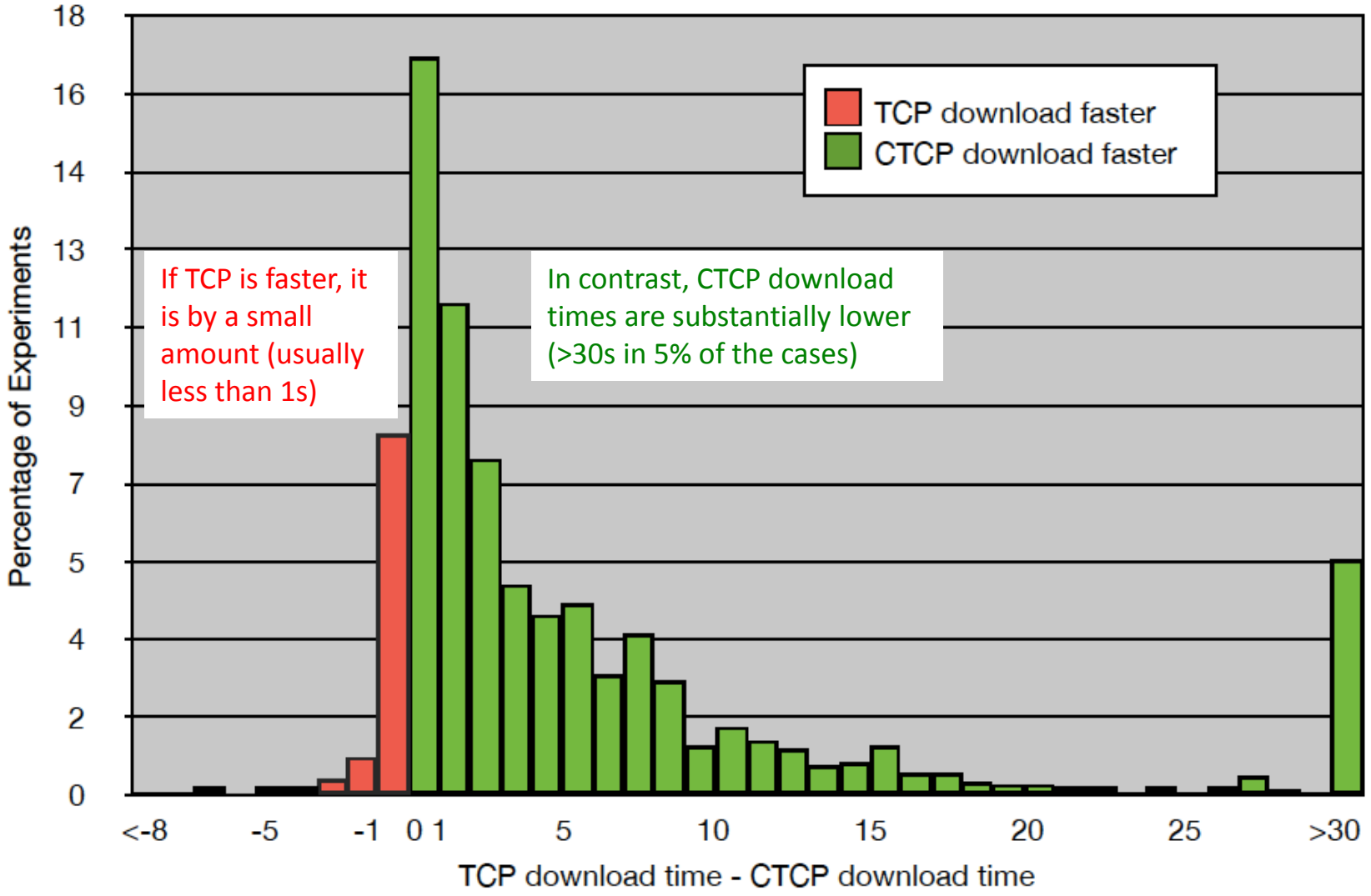

Germany
Irland
US East Coast
US West Coast

TECHNISCHE UNIVERSITÄT DRESDEN

Coded TCP Results



Histogram of CTCP-TCP Data Pairs



- 1354 data pairs

Pacific Island Testbed

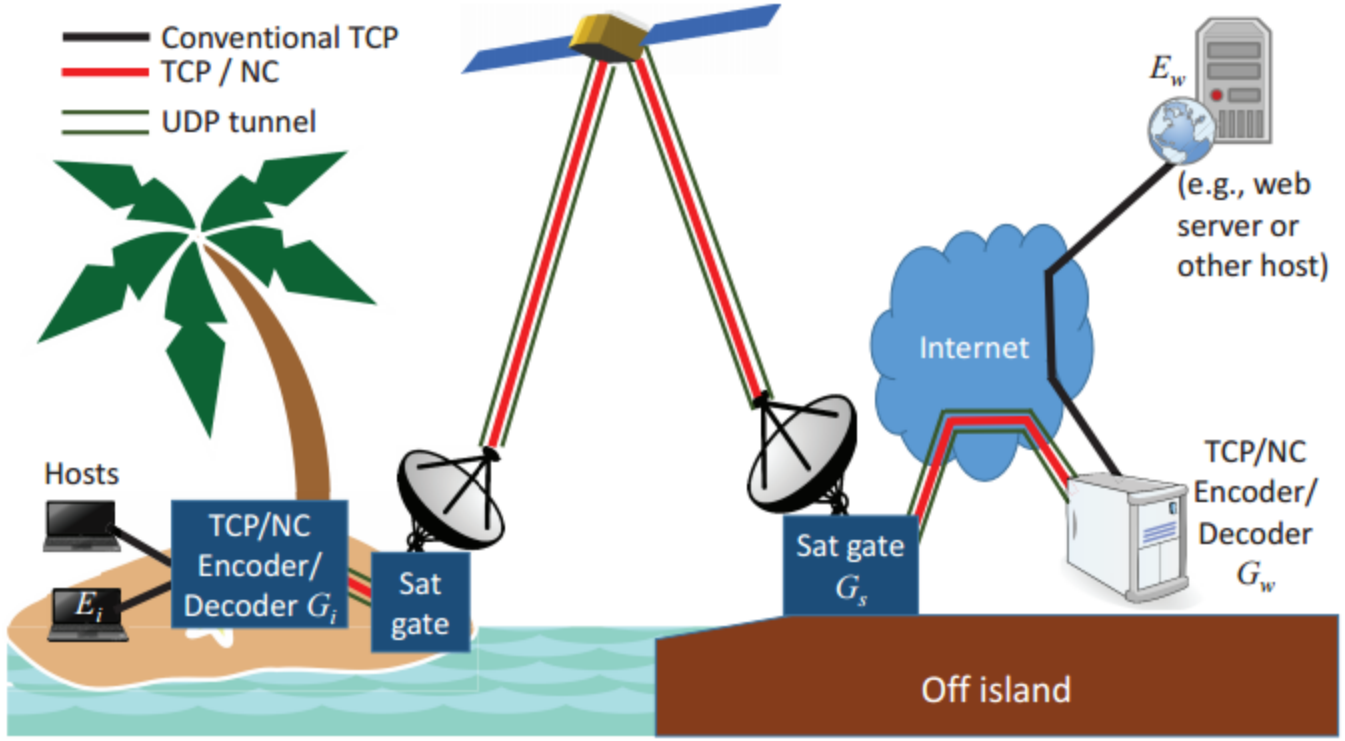
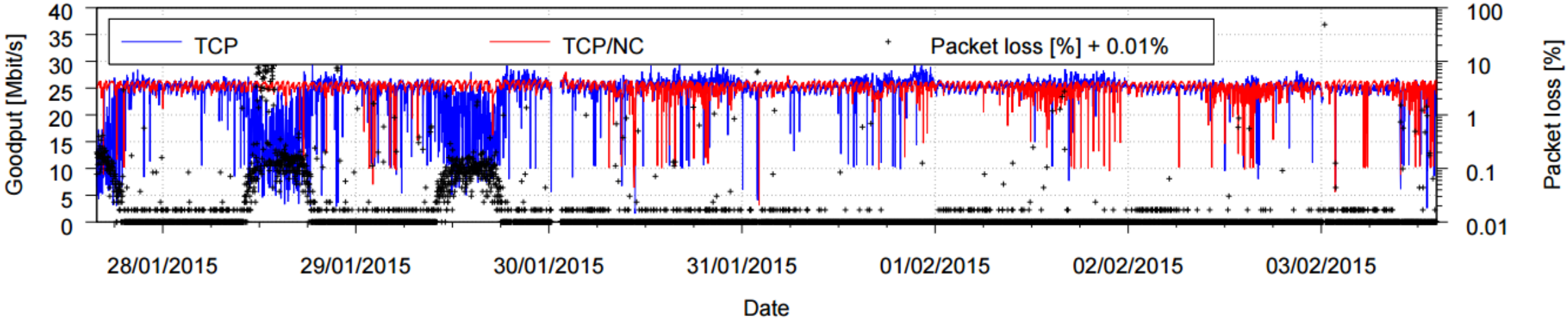
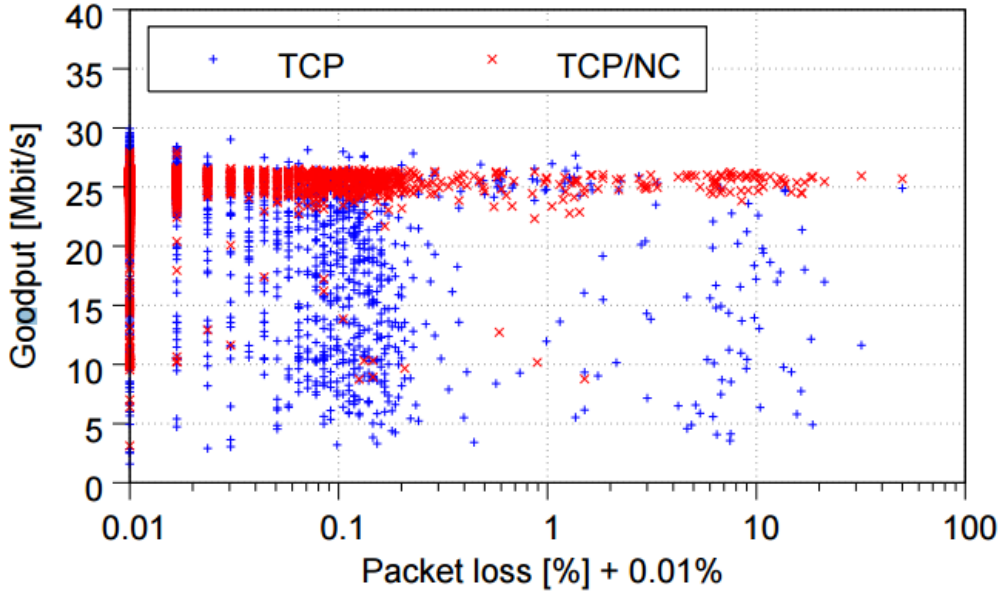


Fig. 1. TCP/NC network topology

http://arxiv.org/pdf/1506.01048v1.pdf

Pacific Island Testbed





TECHNISCHE UNIVERSITÄT DRESDEN

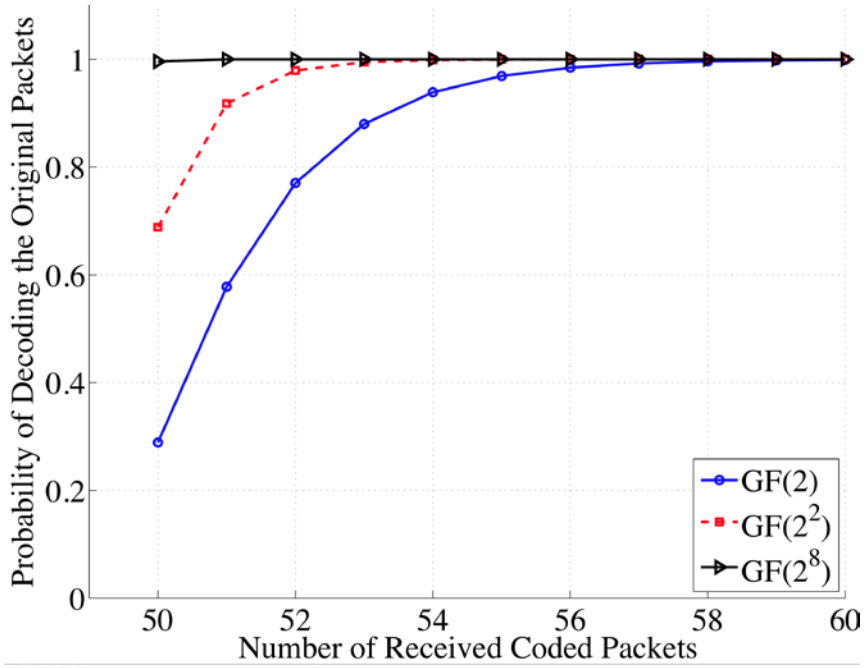
Composite Extension Finite Fields for Low Overhead Network Coding: Telescopic Codes

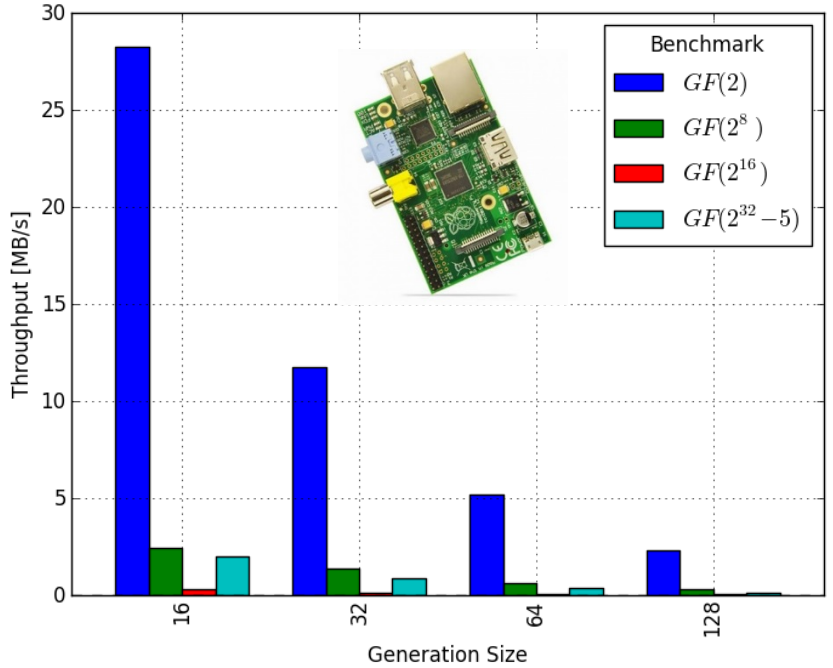
TECHNISCHE UNIVERSITÄT DRESDEN

Introduction

- Network coding is a promising paradigm
  - Store-code-forward instead of store-forward
  - Benefits in throughput, delay, robustness, energy
- Caveats:
  - Computational complexity
  - Overhead per packet, e.g., signaling of coding coefficients

Motivation



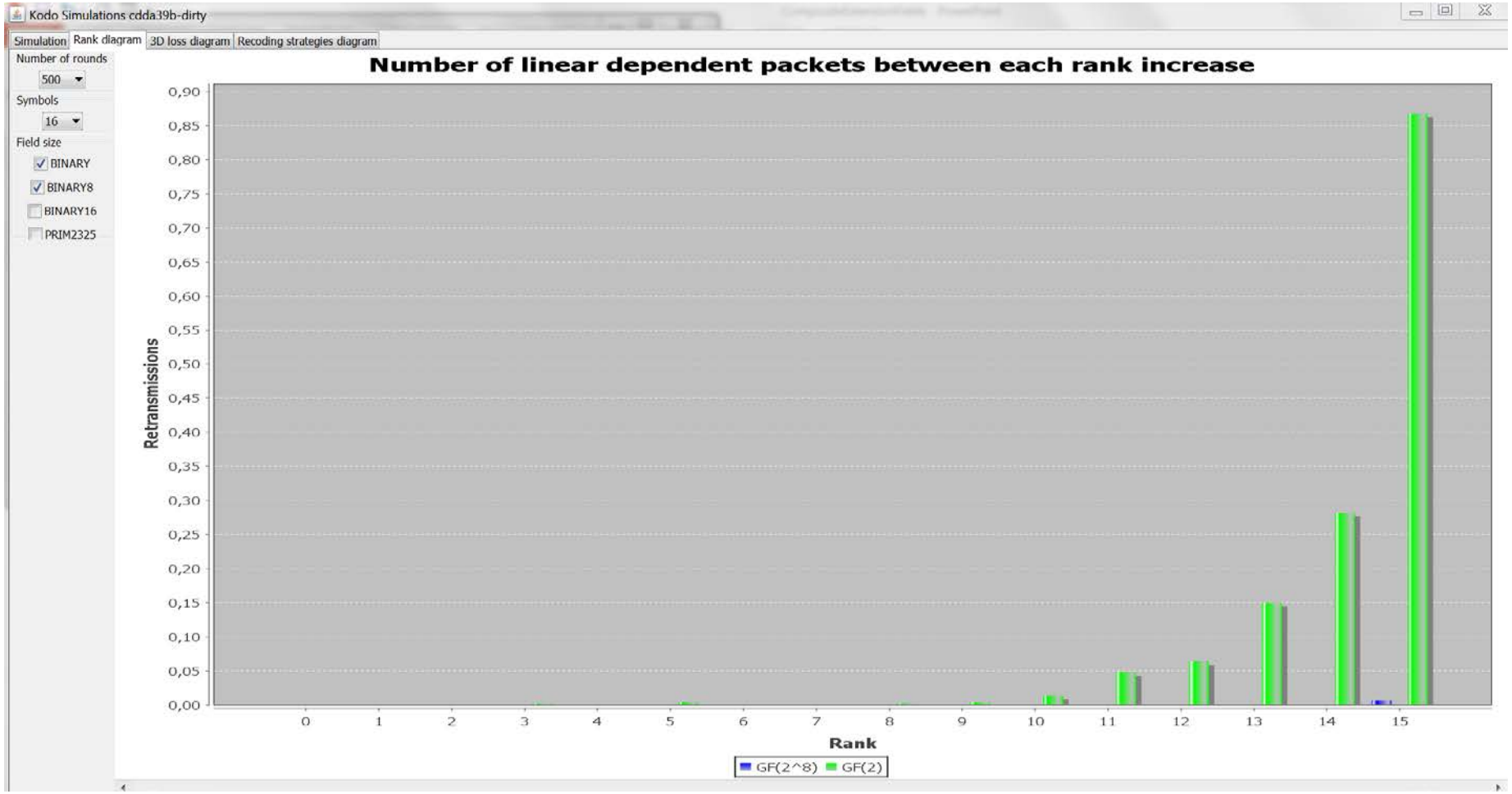


Larger Fields
are more efficient!

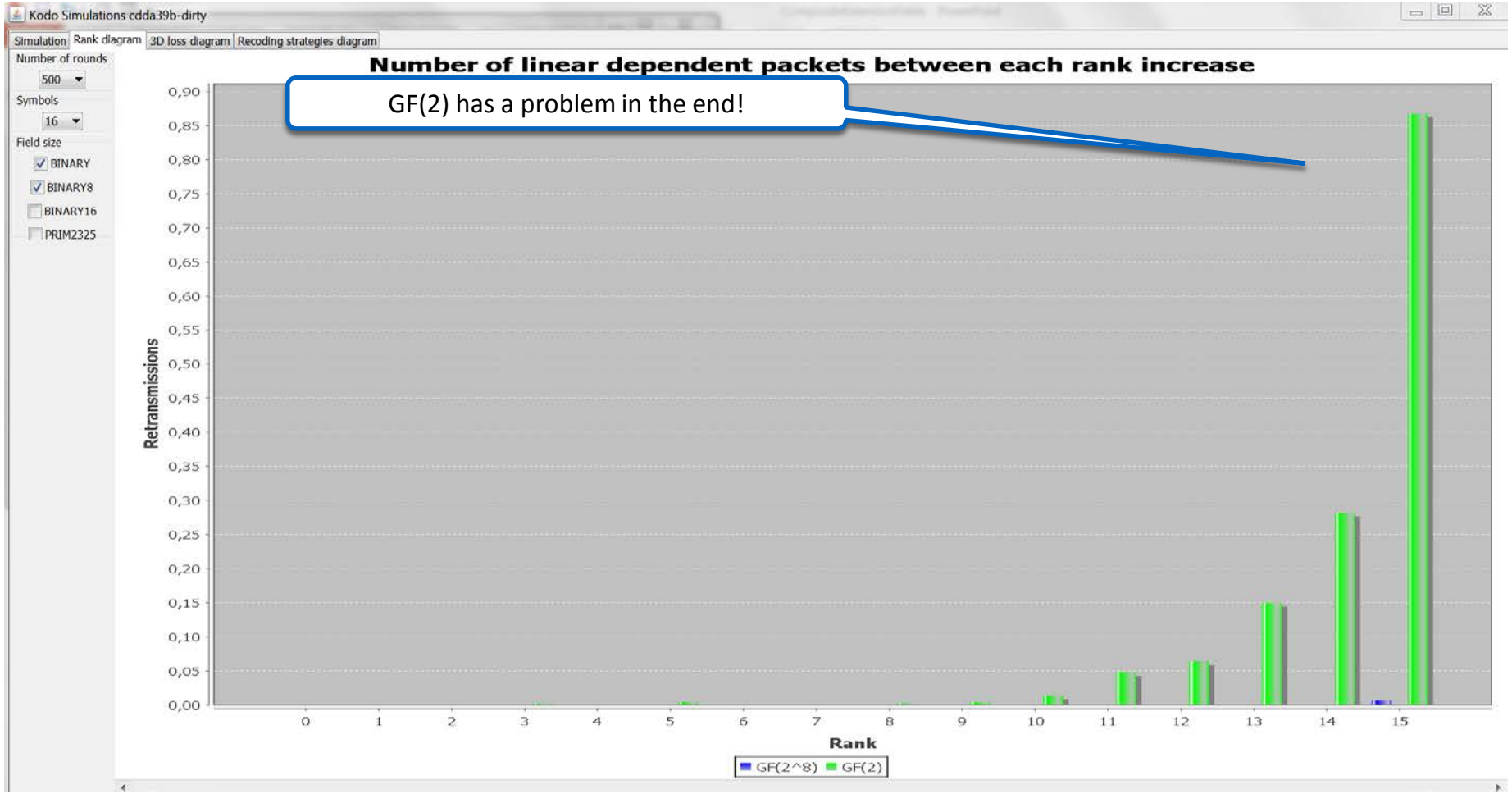
?

Smaller Fields
are faster and need less overhead!

Motivation


Kodo Simulations cdda39b-dirty
Simulation
Rank diagram
3D loss diagram
Recoding strategies diagram
Number of rounds
500
Symbols
16
Field size
BINARY
BINARY8
BINARY16
PRIM2325
Number of linear dependent packets between each rank increase
Retransmissions
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Rank
GF(2^8)
GF(2)

Motivation


Kodo Simulations cdda39b-dirty
Simulation
Rank diagram
3D loss diagram
Recoding strategies diagram
Number of rounds
500
Symbols
16
Field size
BINARY
BINARY8
BINARY16
PRIM2325
Number of linear dependent packets between each rank increase
GF(2) has a problem in the end!
Retransmissions
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Rank
GF(2^8)
GF(2)

TECHNISCHE UNIVERSITÄT DRESDEN

Introduction

- Network coding is a promising paradigm
  - Store-code-forward instead of store-forward
  - Benefits in throughput, delay, robustness, energy
- Caveats:
  - Computational complexity
  - Overhead per packet, e.g., signaling of coding coefficients
- Previous work:
  - Understanding single choice of field size, generation size on processing speed [Paramanathan et al 2013]
  - Techniques that exploit sparsity: tunable sparse network coding, overlapping generations, BATS codes, ...
  - Fulcrum network codes: low overhead, high processing speed, simple recoding, configurable performance [Lucani et al 2014]

TECHNISCHE UNIVERSITÄT DRESDEN

Motivation

- Fulcrum provides support using composite fields:
- GF(2) in the network and any GF(2^k) at source and destinations
- Key: operations performed in GF(2) have an equivalent operation in any GF(2^k)
- Example: P1 + P2 requires a bit by bit XOR in both cases

- Can we expand this idea for more flexibility in code design?
- Typically, GF(2^k) is not compatible with a GF(2^p) if n ≠ p
- Different primitive polynomials → different mapping for product operations

Motivation

- Typically, $GF(2^k)$ is not compatible with a $GF(2^p)$ if $n \neq p$
- Example: $GF(2^2)$ versus $GF(2^4)$

$GF(2^2)$

×	**1**	**2**	**3**
1	1	2	3
2	2	3	1
3	3	1	2

≠

$GF(2^4)$

x	**1**	**2**	**3**	**4**	**5**	**…**	**15**
1	1	2	3	4	5	…	15
2	2	4	6	8	10	…	13
3	3	6	5	12	15	…	2
4	4	8	12	3	7	…	9
5	5	10	15	7	2	…	6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	15	13	2	9	6	…	10

Motivation

- Typically, $GF(2^k)$ is not compatible with a $GF(2^p)$ if $n \neq p$
- Example: $GF(2^2)$ versus $GF(2^4)$

$GF(2^2)$

×	**1**	**2**	**3**
1	1	2	3
2	2	3	1
3	3	1	2

≠

$GF(2^4)$

x	**1**	**2**	**3**	**4**	**5**	…	**15**
1	1	2	3	4	5	…	15
2	2	4	6	8	10	…	13
3	3	6	5	12	15	…	2
4	4	8	12	3	7	…	9
5	5	10	15	7	2	…	6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	15	13	2	9	6	…	10

Actually, we will need even stronger conditions

Idea: Multiple composite extension fields

Multiple Composite Extension Fields

- Use GF(2) to construct $GF(2^2)$, use $GF(2^2)$ to construct $GF(2^{2^2})$

$GF(2^2)$

×	**1**	**2**	**3**
1	1	2	3
2	2	3	1
3	3	1	2

=

$GF(2^{2^2})$

x	**1**	**2**	**3**	**4**	**5**	**…**	**15**
1	1	2	3	4	5	…	15
2	2	3	1	8		…	5
3	3	1	2	12		…	10
4	4	8	12	6		…	1
5	5					…	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	15	5	10	1		…	9

Full compliance: product by

1, 2, 3 in $GF(2^{2^2})$ (4 bits)

Is equivalent to the product on two values of $GF(2^2)$ (2 bits)

Multiple Composite Extension Fields

- Use GF(2) to construct GF(2^2), use GF(2^2) to construct GF(2^{2^2})

Example: 15 x 2 = 5

1111_b x 0010_b in GF(2^{22})

But in GF(2^2) we operate on pairs of bits

11_b x 10_b = 01_b

Thus, on a vector

11_b11_bx 10_b = 01_b01_b

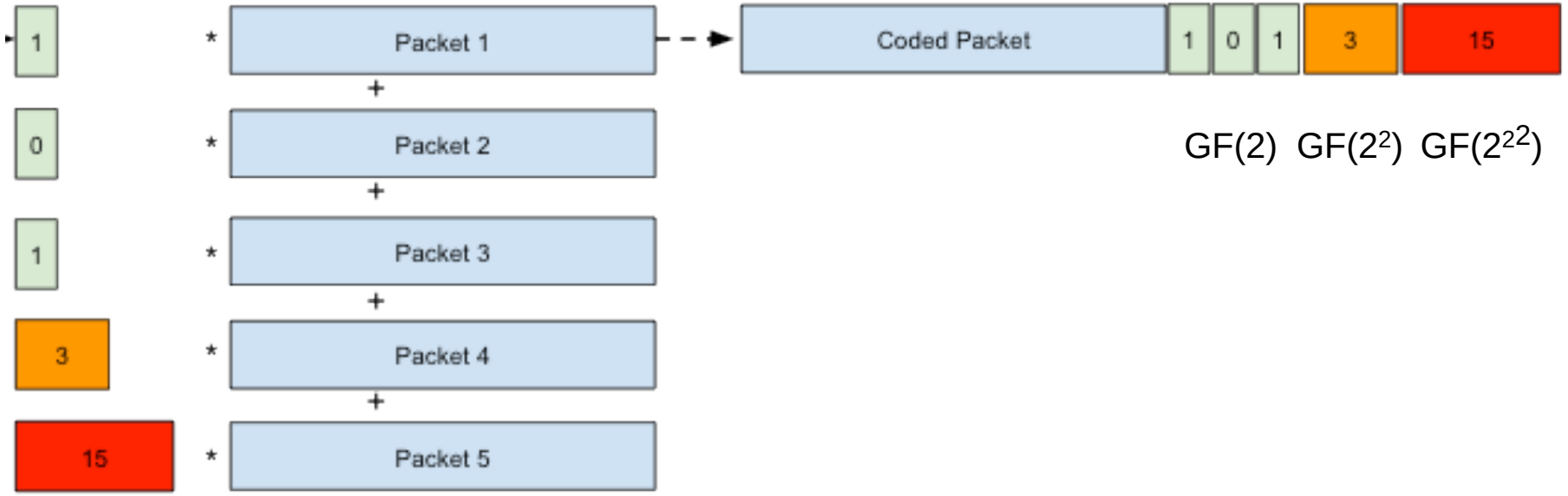
Which results in “5” for the larger field

GF(2^{2^2})

x	1	2	3	4	5	…	15
1	1	2	3	4	5	…	15
2	2	3	1	8		…	**5**
3	3	1	2	12		…	10
4	4	8	12	6		…	1
5	5					…	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	15	5	10	1		…	9

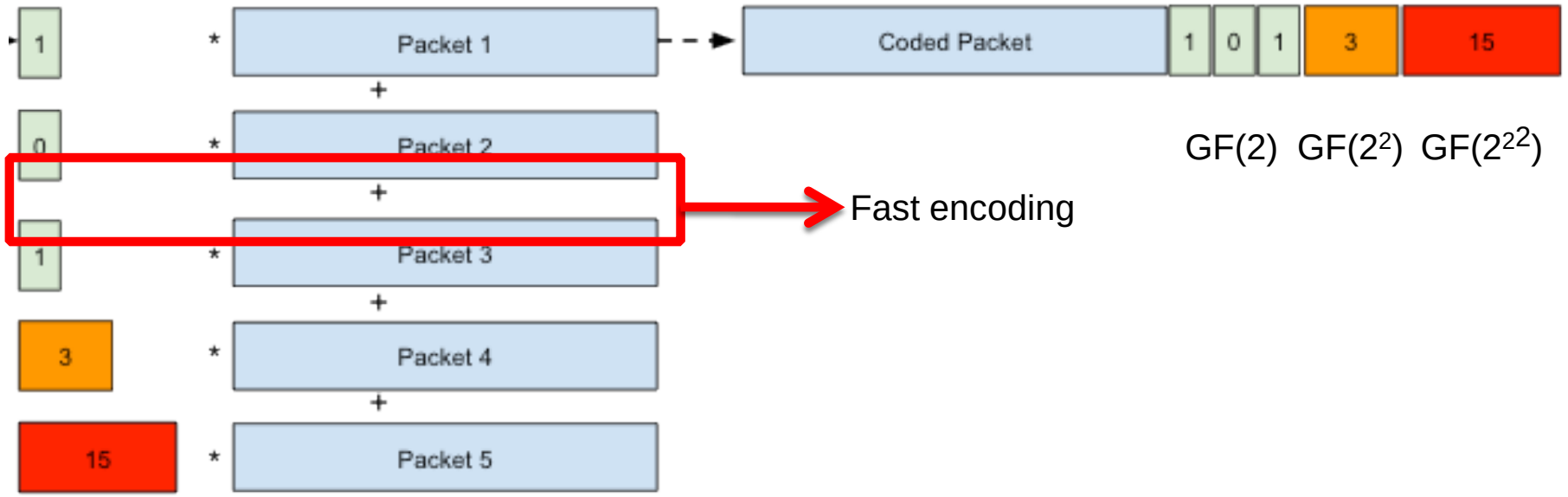
Telescopic Codes

- **Design:**
- Multiple composite extension fields
- Goal: reduce overhead, maintaining high performance, faster encoding/decoding
- Different packets are encoded using different field sizes

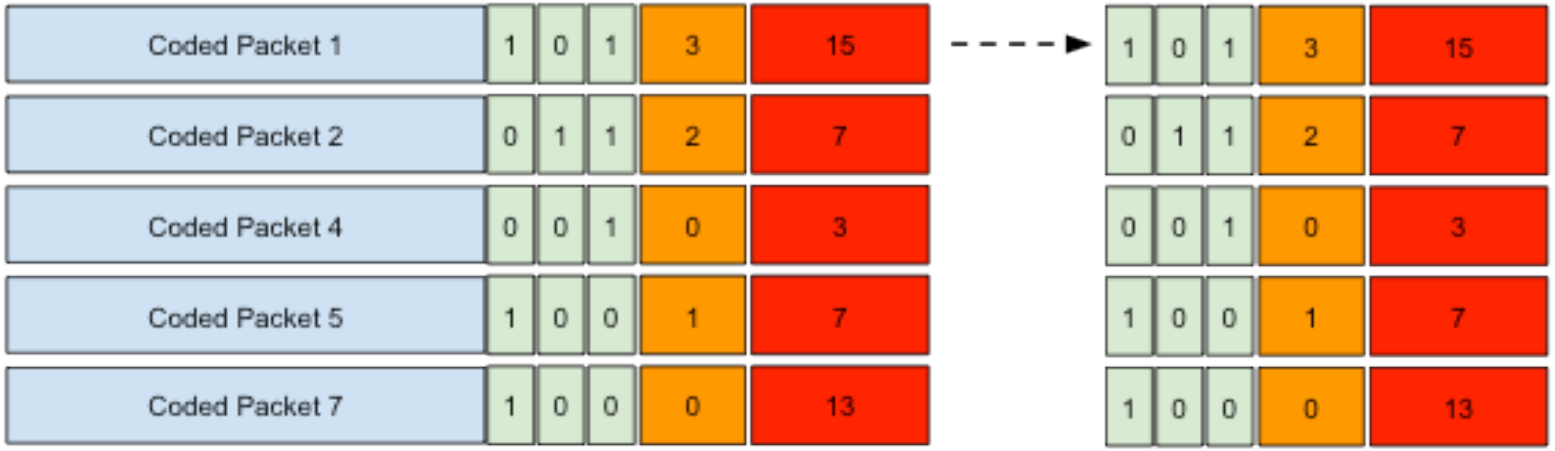


Telescopic Codes

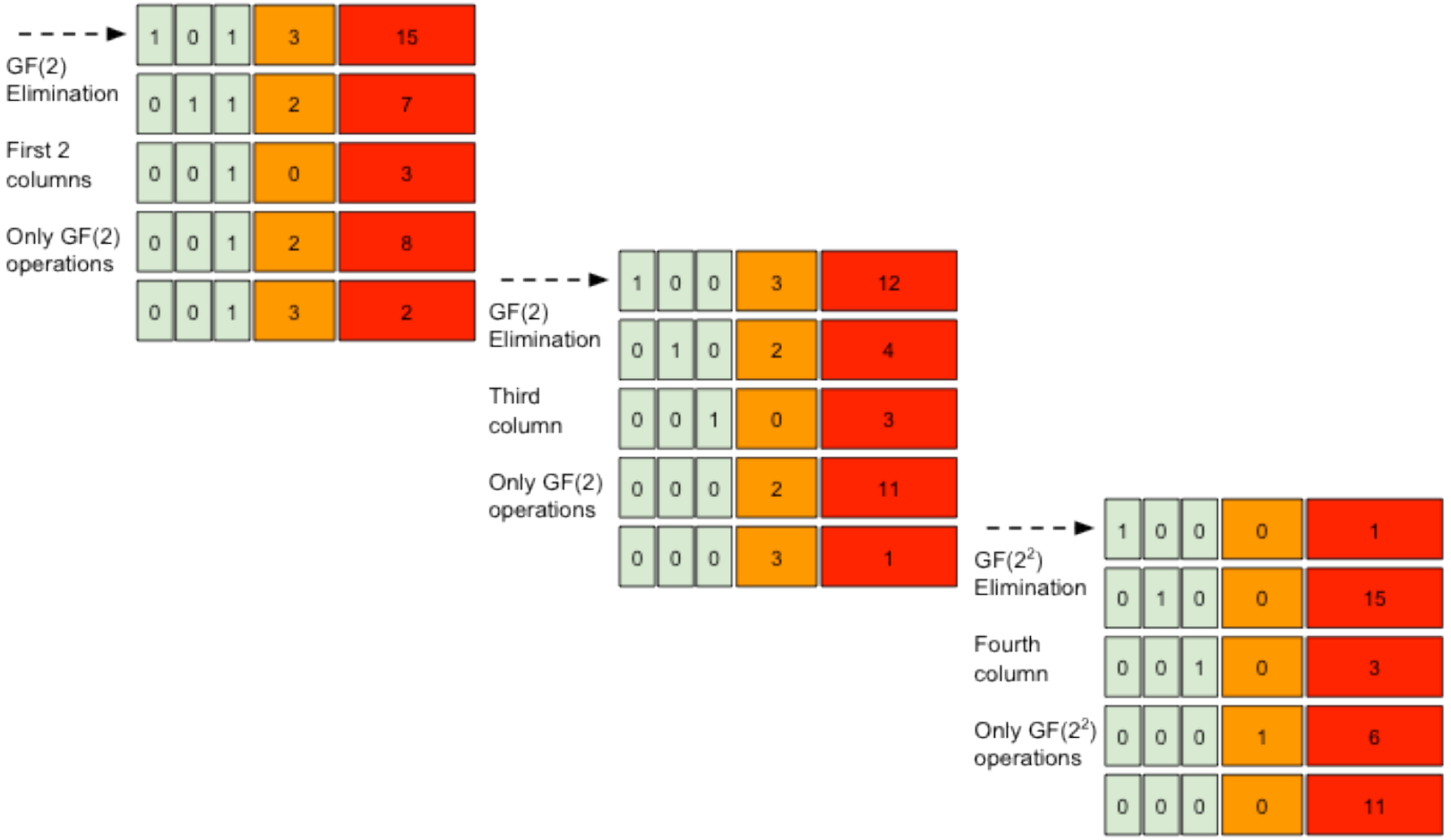
- **Design:**
- Multiple composite extension fields
- Goal: reduce overhead, maintaining high performance, faster encoding/decoding
- Different packets are encoded using different field sizes



Telescopic Codes: Decoder

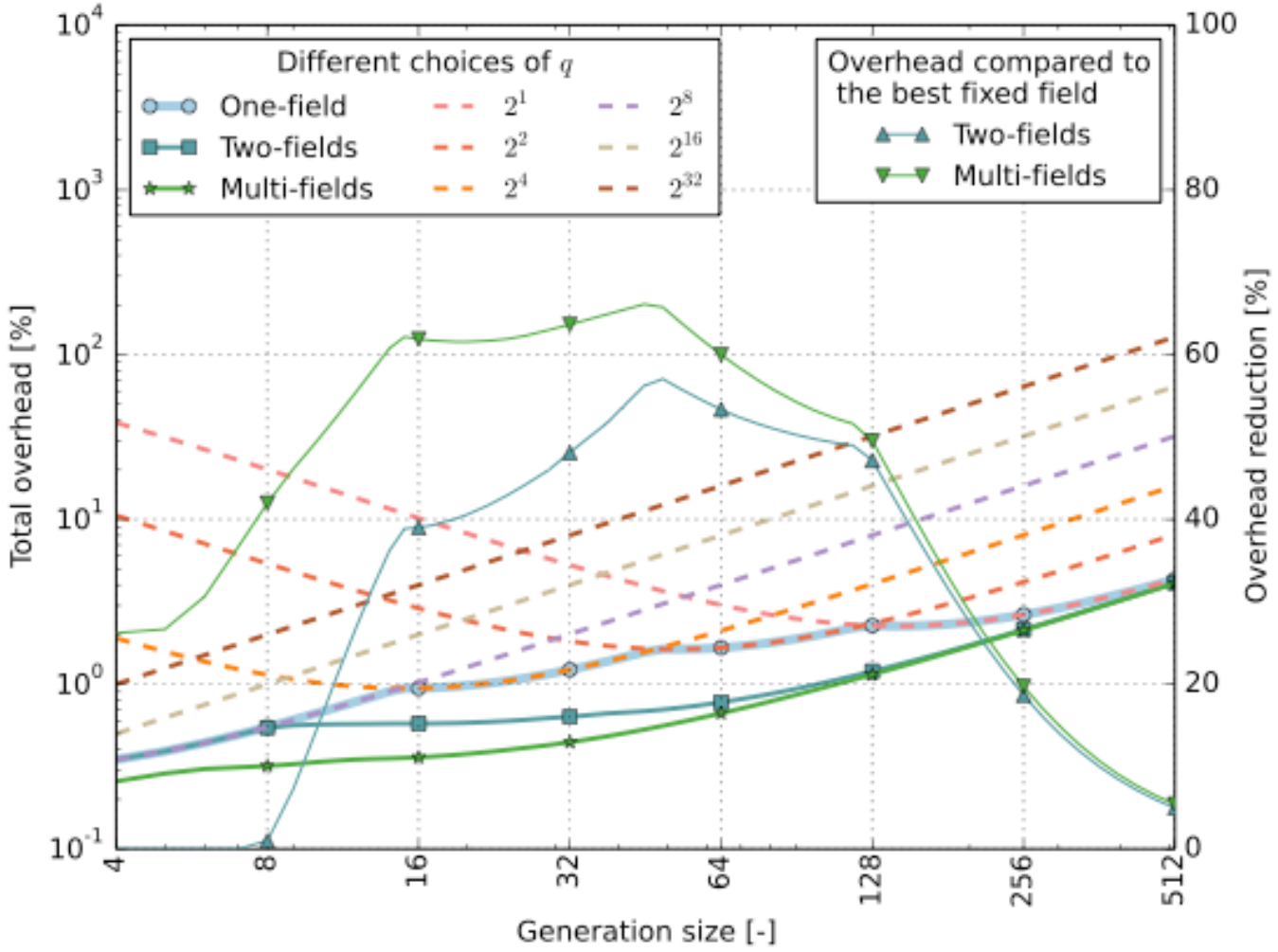


Telescopic Codes: Decoder



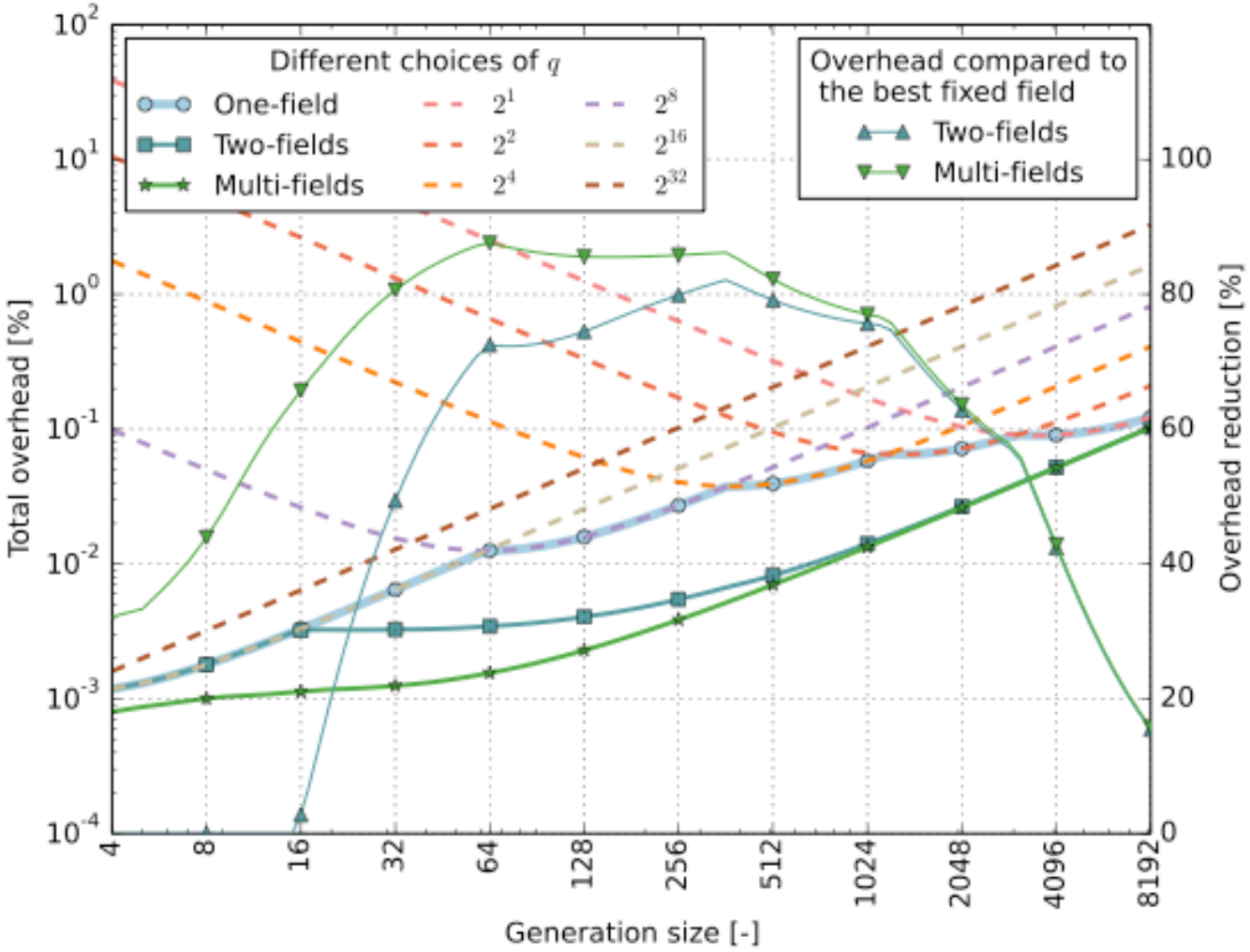
TECHNISCHE UNIVERSITÄT DRESDEN

Results (1600B pkts)



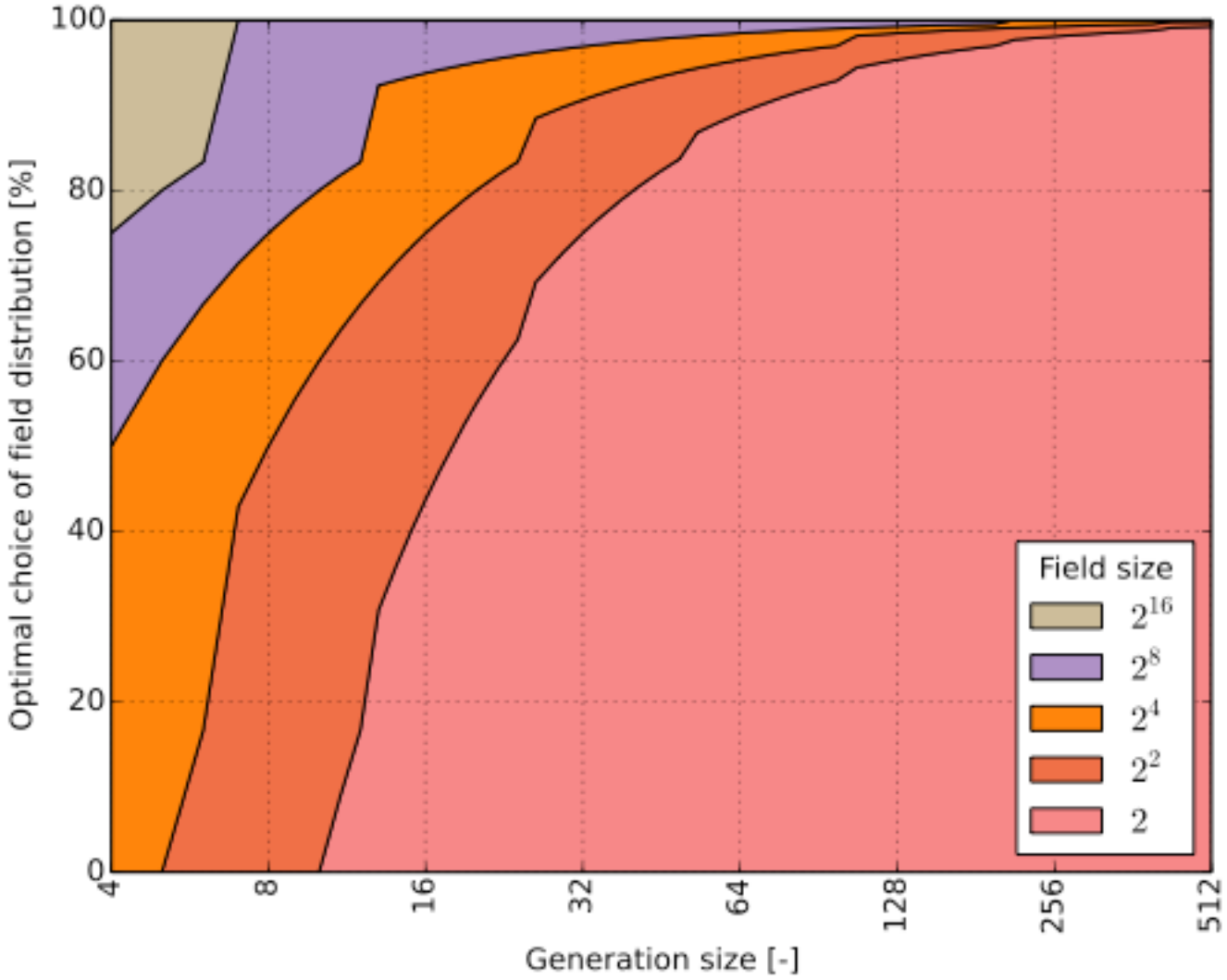
TECHNISCHE UNIVERSITÄT DRESDEN

Results (1MB pkts)



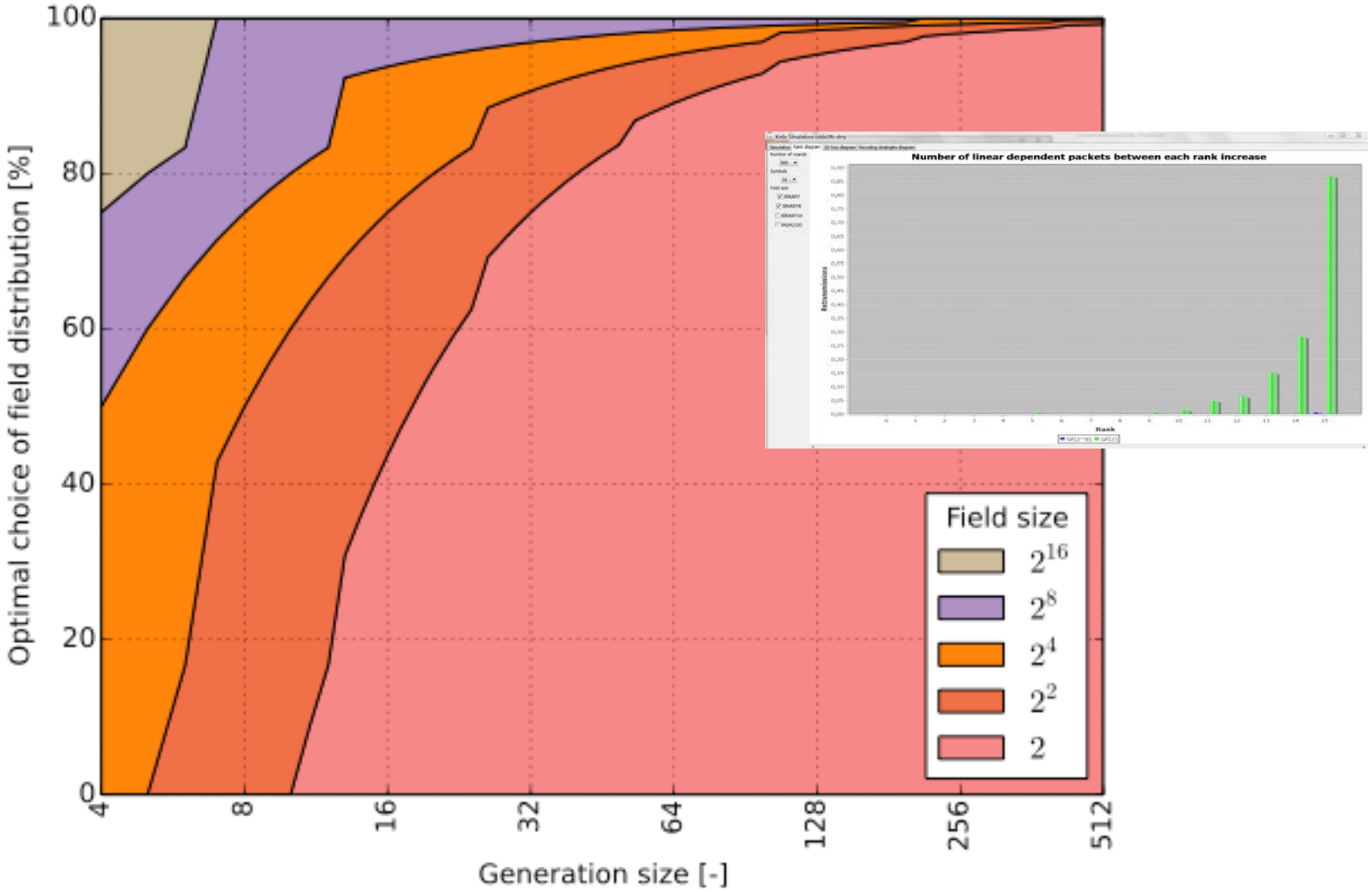
TECHNISCHE UNIVERSITÄT DRESDEN

Results



TECHNISCHE UNIVERSITÄT DRESDEN

Results



Conclusions

- Simple design of composite extension fields
- Proposed telescopic codes
  - Reduce total overhead
  - Maintain high decoding probability
  - Potential for processing at high speed (simpler encoder/decoder)
- Future work
  - Applications in other codes
  - New code designs
  - Performance evaluation in real systems

TECHNISCHE UNIVERSITÄT DRESDEN

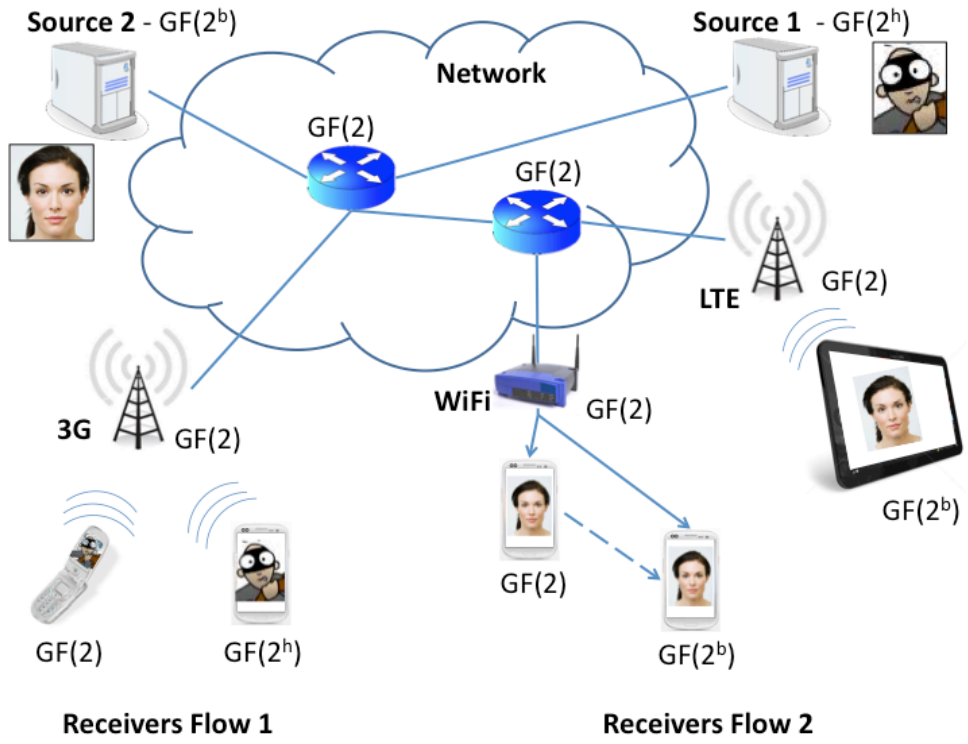
Fulcrum Network Codes

Daniel E. Lucani, Morten V. Pedersen, Janus Heide, Frank H. P. Fitzek

Aalborg University
Steinwurf ApS

General Ideas

- Fluid allocation of complexity
- End devices agree on desired performance:
  - Independent from network
  - Chosen according to application requirements
- Network devices need only support a simple subset of functions
- Reduces overhead
  - Roughly 1 bit per coding coefficient
- Key: code concatenation with different field sizes

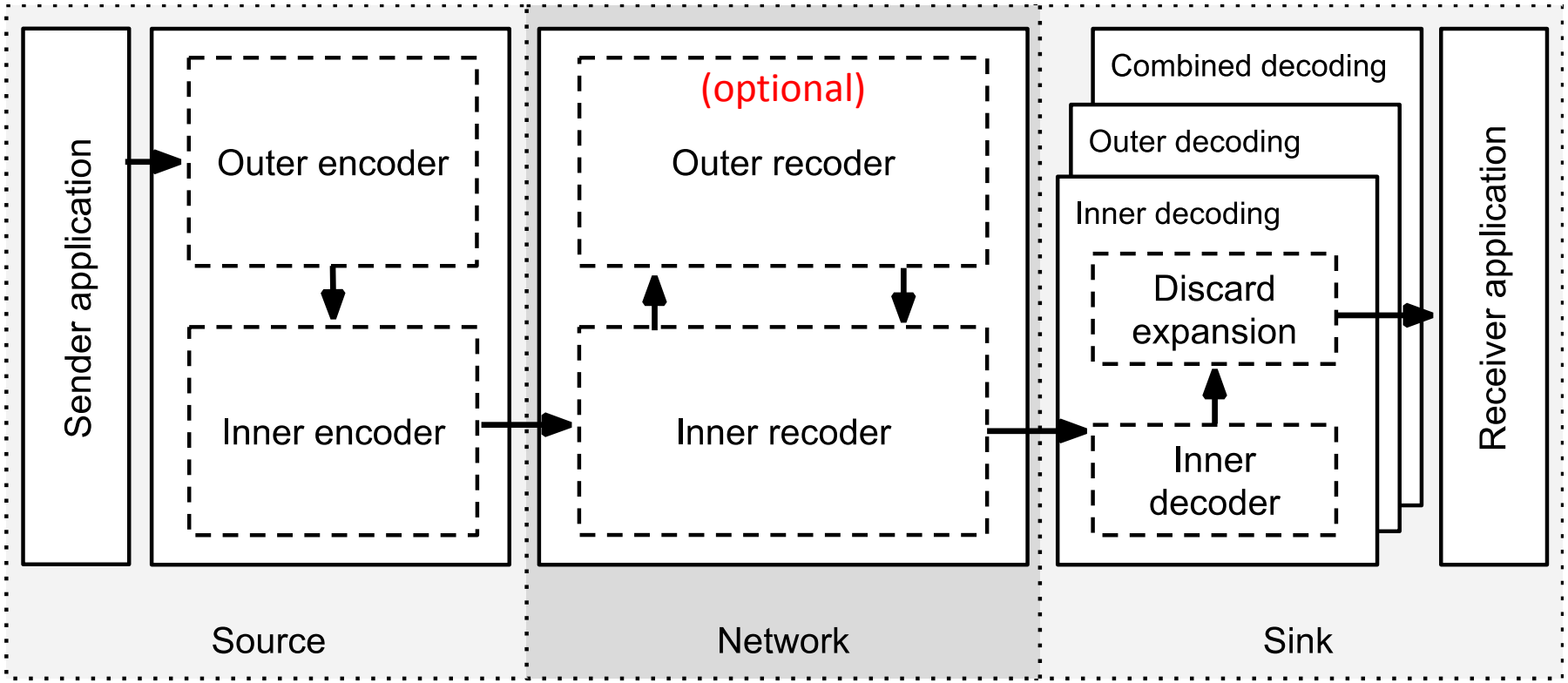


TECHNISCHE UNIVERSITÄT DRESDEN

Benefits

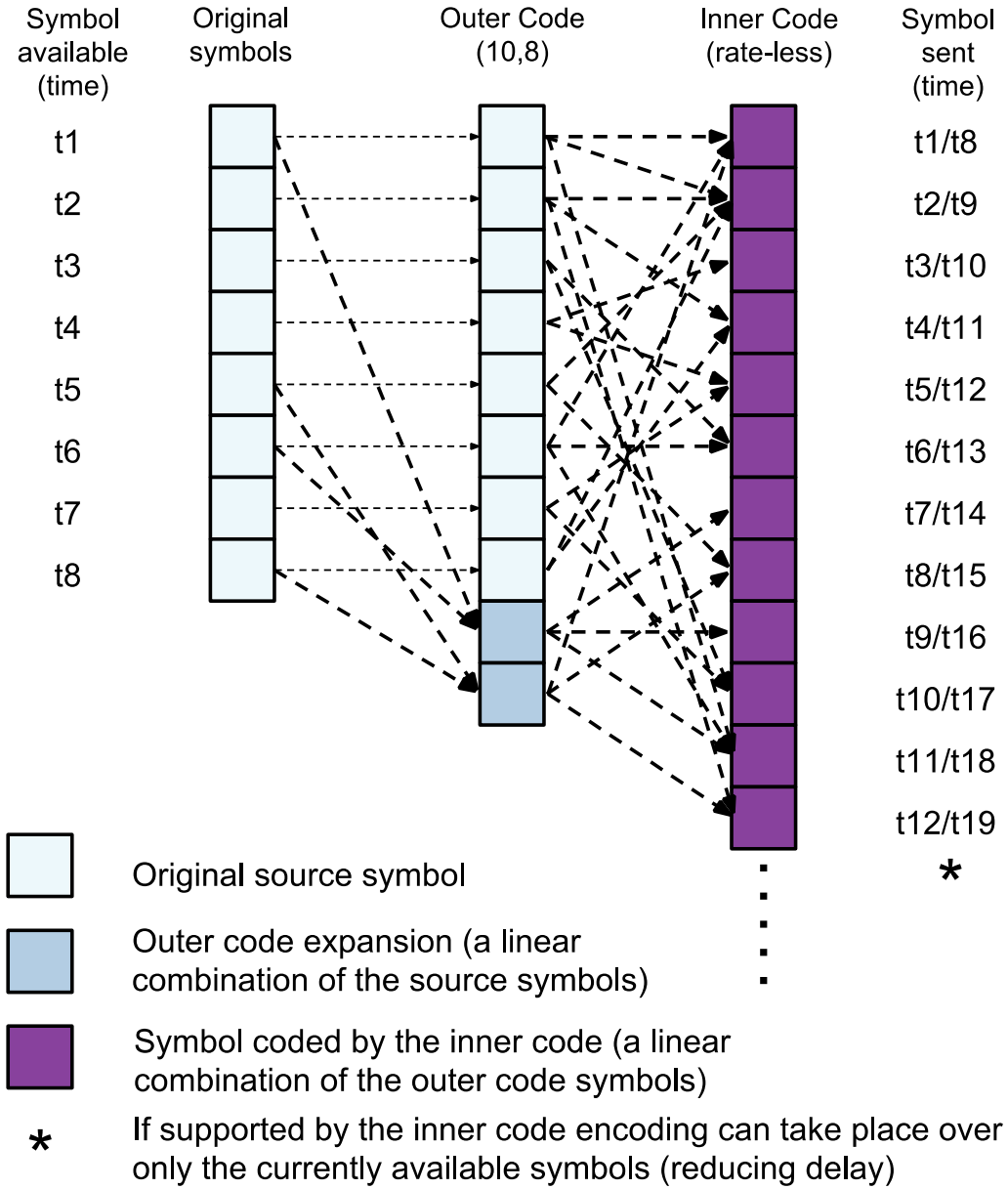
- Simple is green, compatible, deployable
- Supports heterogeneous receivers
- Adaptive performance
- Practical recoding
- Spreads complexity to stronger devices
- Security: simple support
- New designs can be supported: backwards compatible

General Structure



- Inner code: RLNC, Sparse RLNC, Perpetual, … GF(2)
- Outer code: (systematic) RLNC, Reed-Solomon, … GF(2^h)

Encoder

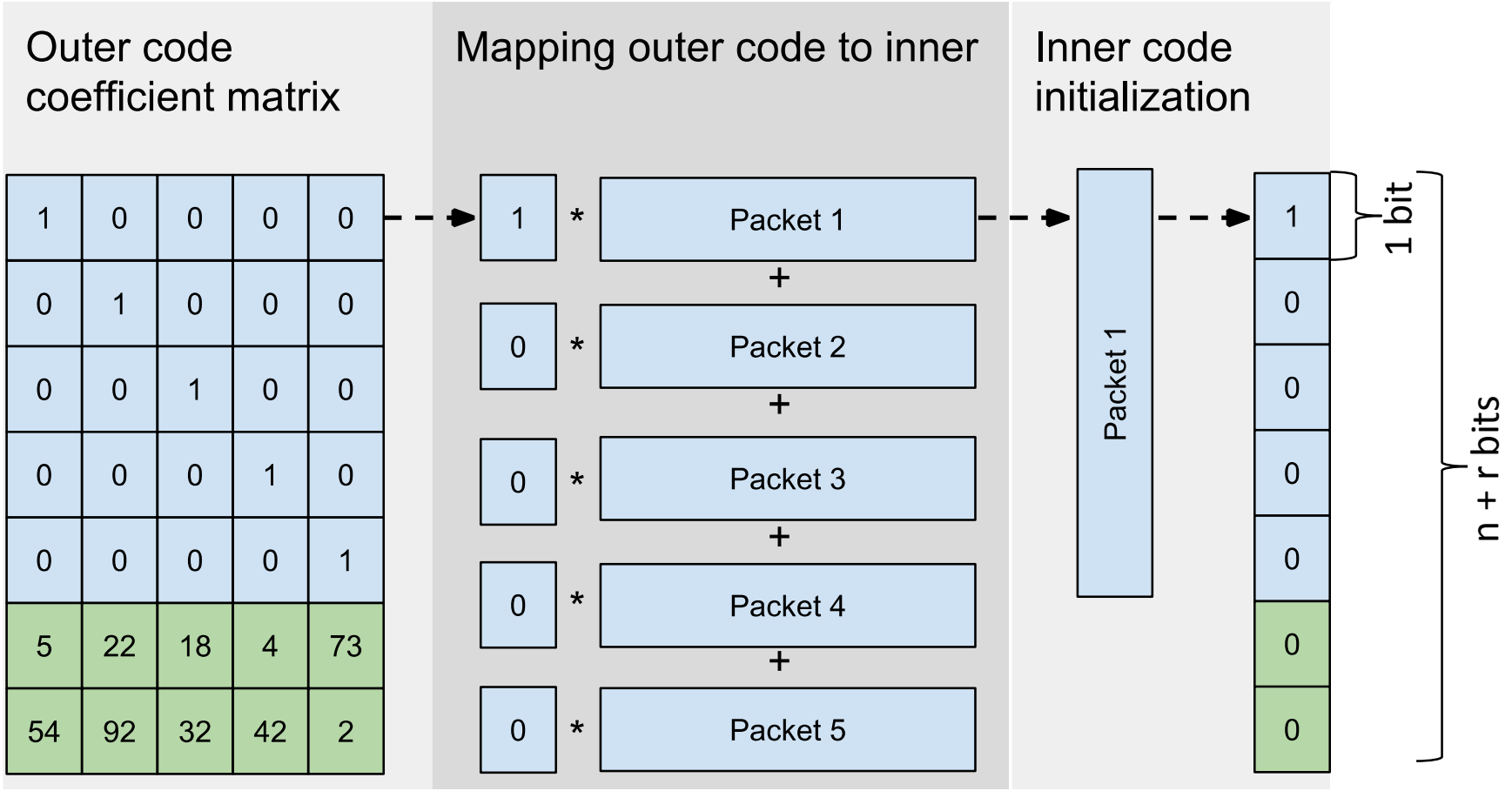


Since inner code is created in GF(2) with pre-coded packets, then:

n+r coefficients per inner coded packet

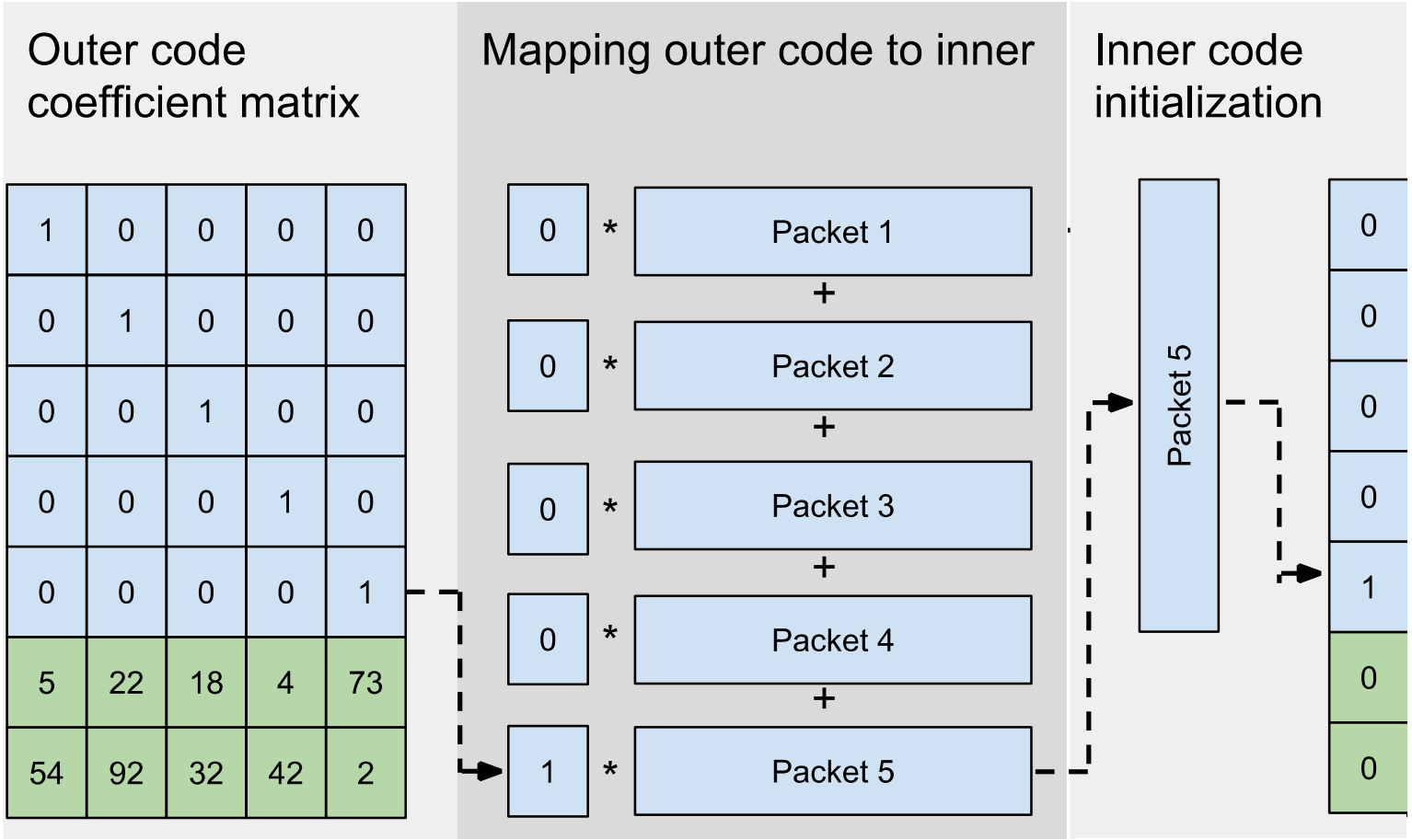
This means: 1 + r/n bits per coefficient per original packet

Encoder


Outer code coefficient matrix
Mapping outer code to inner
Inner code initialization
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
5 22 18 4 73
54 92 32 42 2
1 * Packet 1
+
0 * Packet 2
+
0 * Packet 3
+
0 * Packet 4
+
0 * Packet 5
Packet 1
1
0
0
0
0
0
0
1 bit
n + r bits

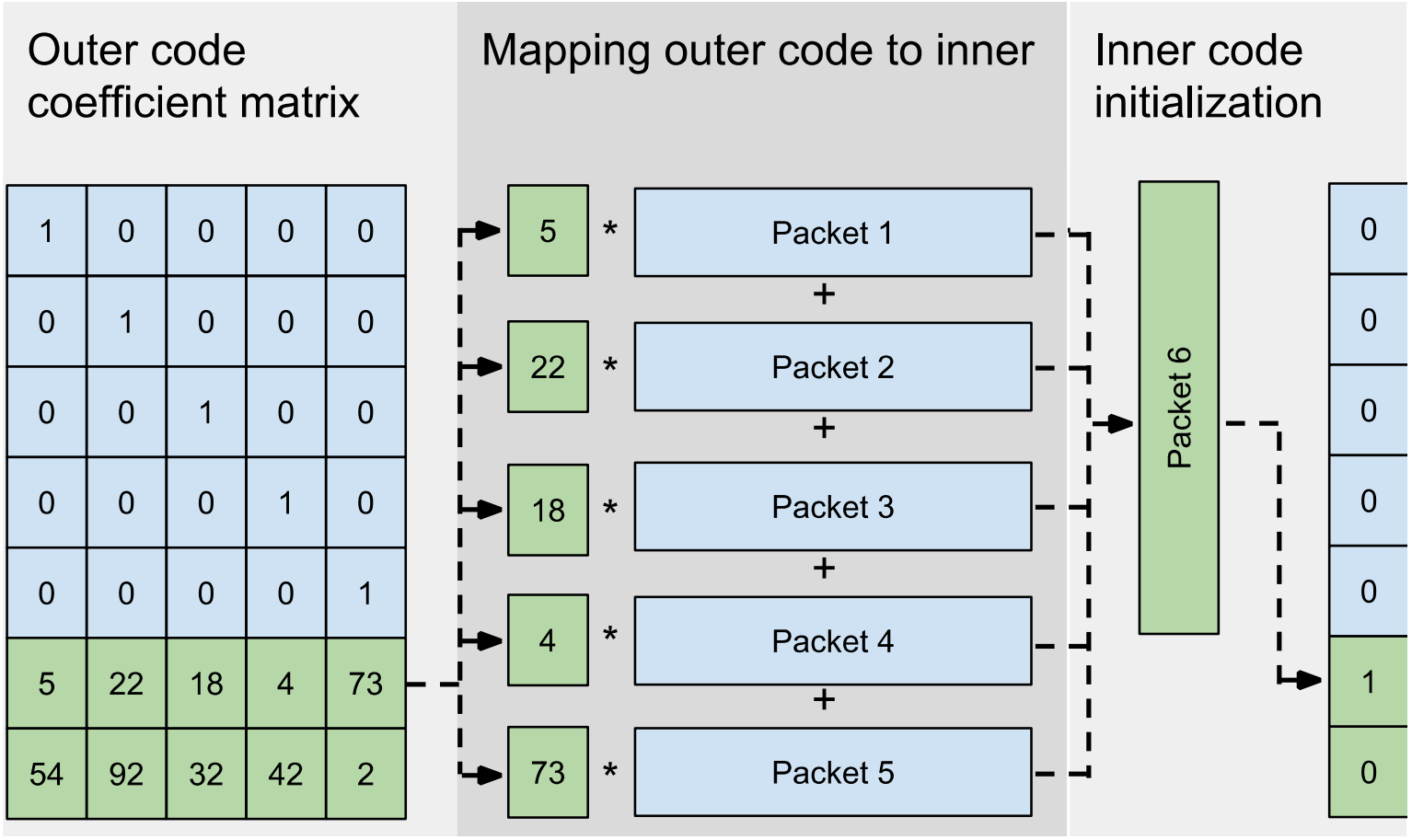
TECHNISCHE UNIVERSITÄT DRESDEN

Encoder



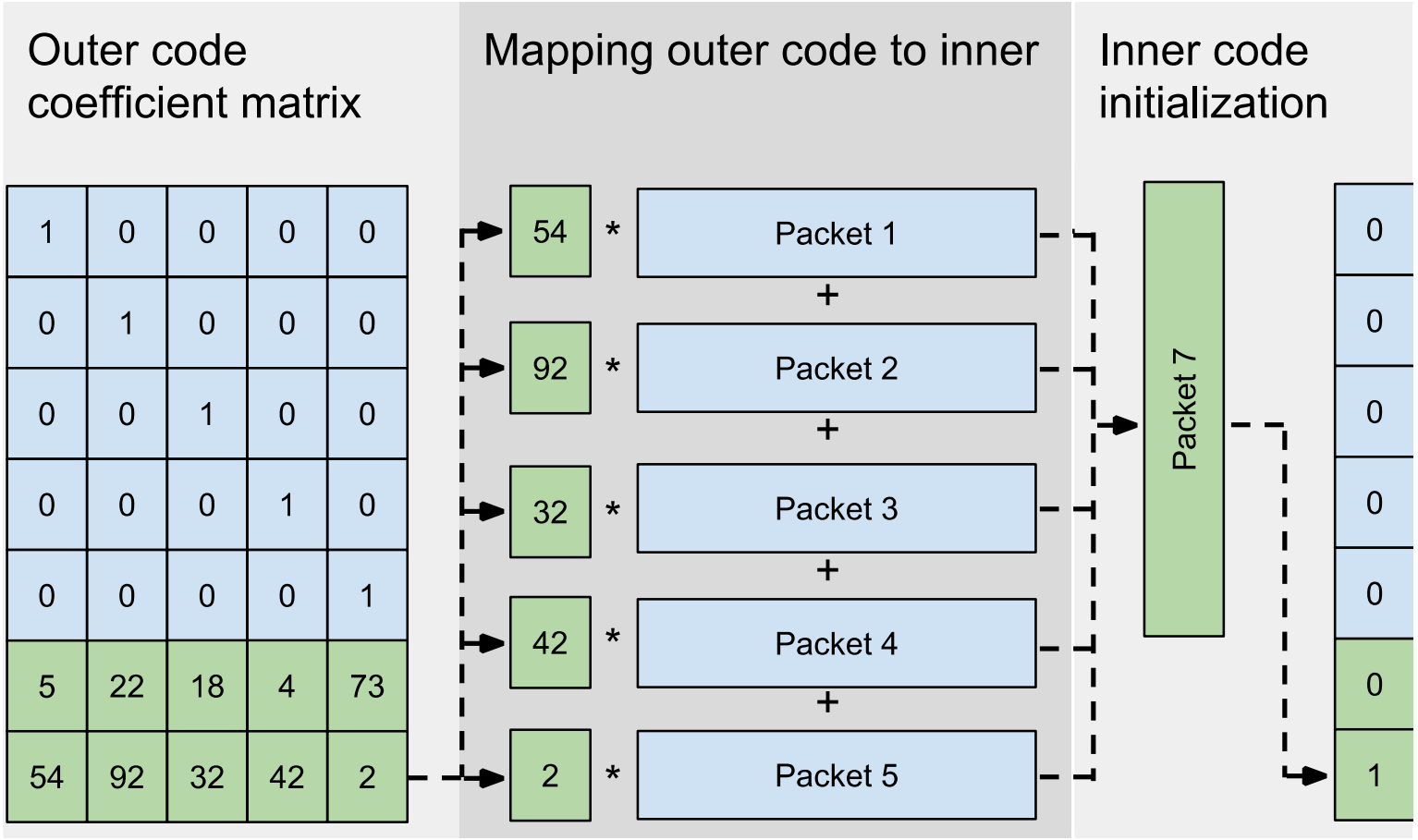
TECHNISCHE UNIVERSITÄT DRESDEN

Encoder



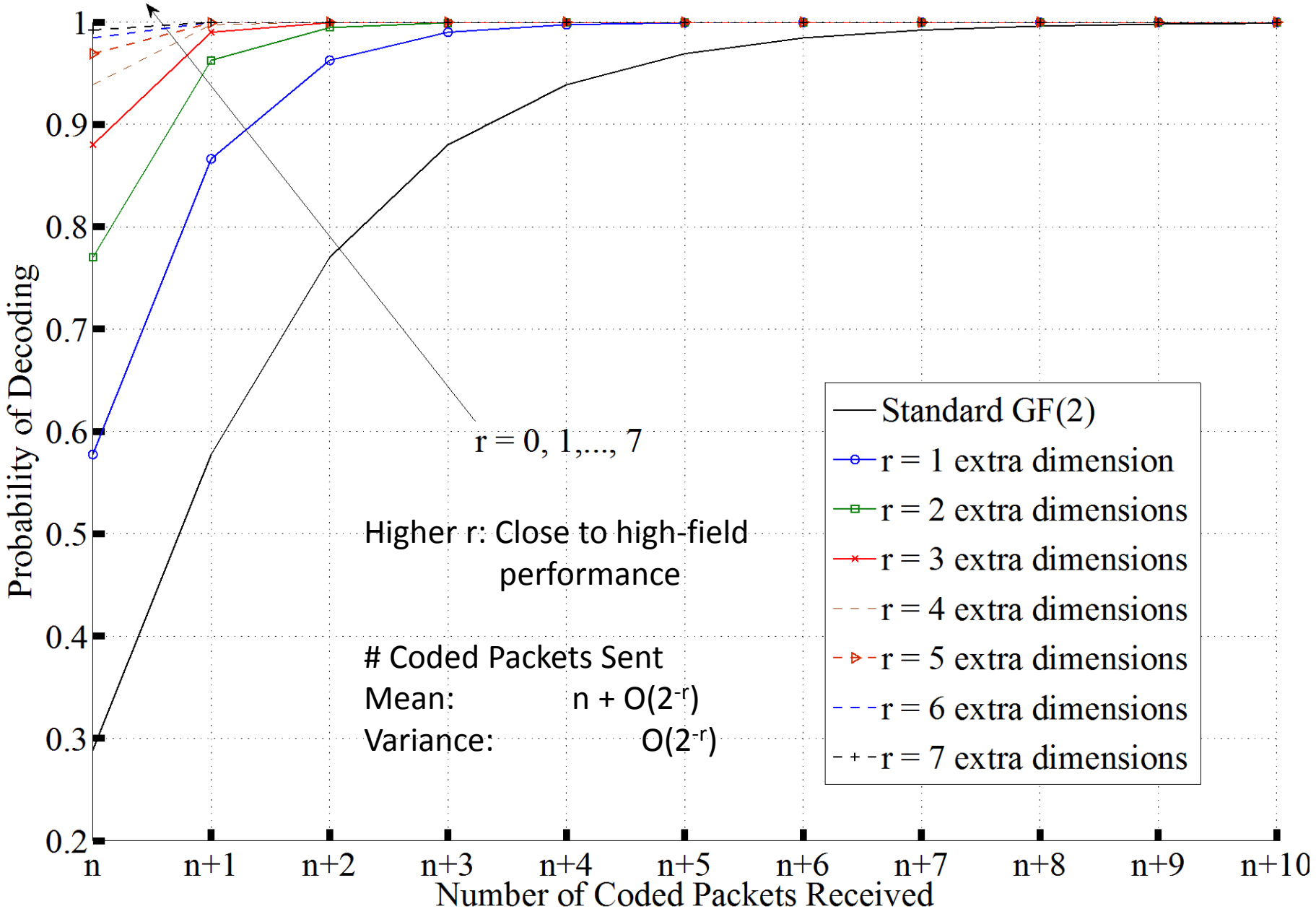
TECHNISCHE UNIVERSITÄT DRESDEN

Encoder



TECHNISCHE UNIVERSITÄT DRESDEN

Received Packets before Decoding



TECHNISCHE UNIVERSITÄT DRESDEN

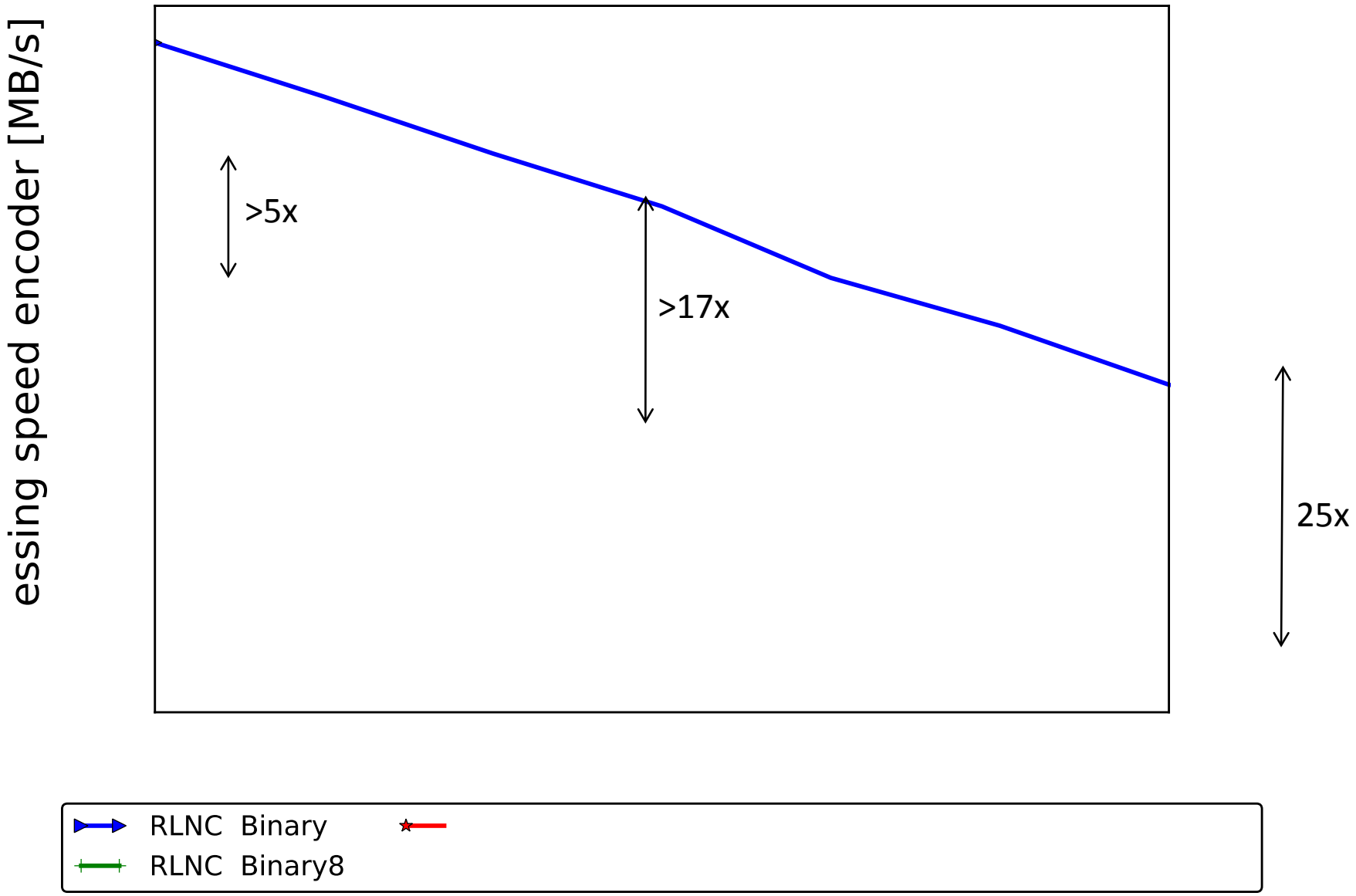
Received Packets before Decoding

Code		Decoding after receiving (coded packets)			
		n	**n + 1**	**n + 2**	**n + 3**
Fulcrum	r = 4	93.87%	99.75%	99.99%	99.9997%
	r = 7	99.22%	99.996%	99.99998%	99.9999992%
	r = 10	99.90%	99.9999%	99.99999996%	99.9999999998%
RaptorQ*		99%	99.99%	99.9999%	

* Qualcomm. (2013, Dec.) Raptorq - the superior fec technology
Available: http://www.qualcomm.com/media/documents/raptorq-data-sheet

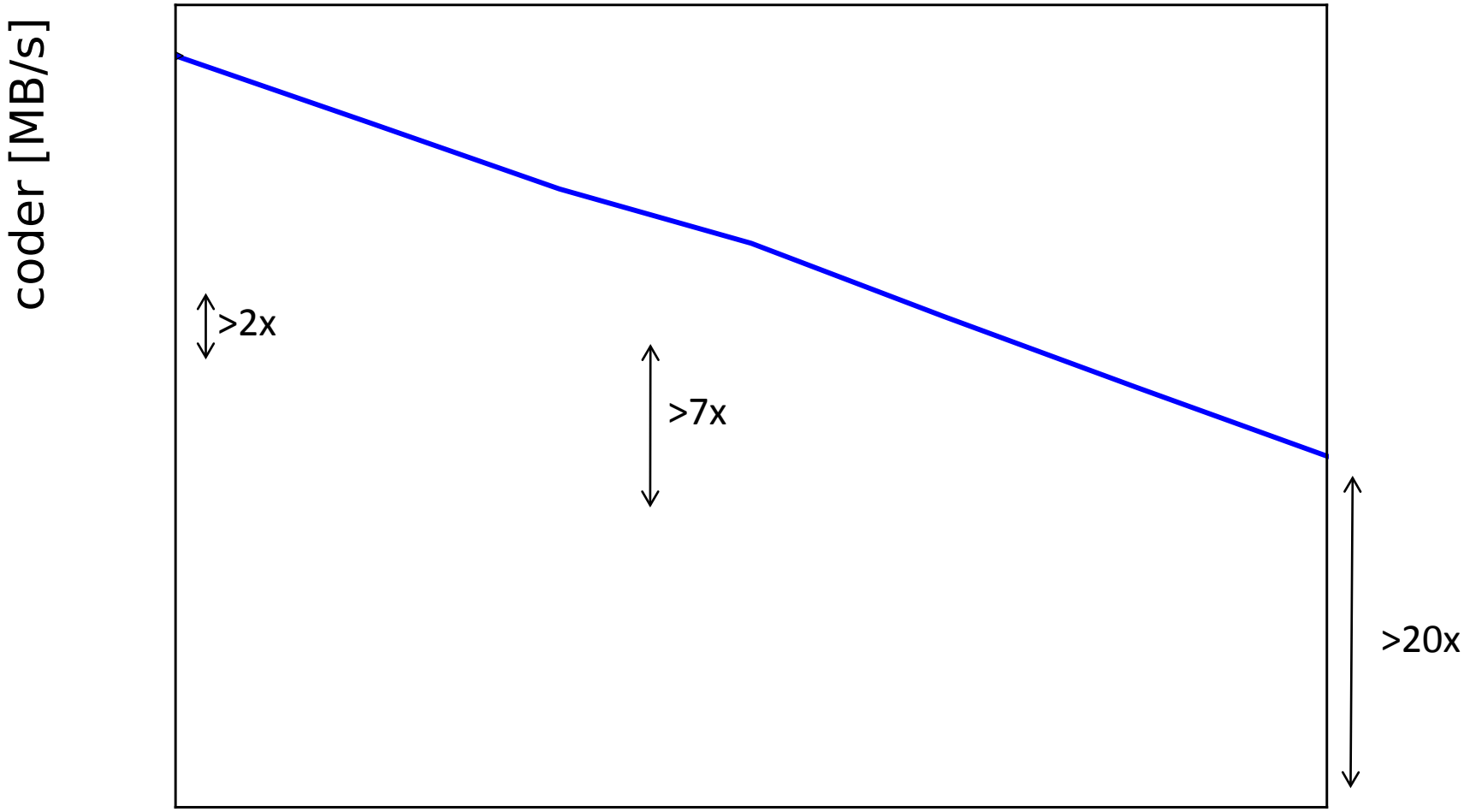
TECHNISCHE UNIVERSITÄT DRESDEN

Performance Results: Encoder



TECHNISCHE UNIVERSITÄT DRESDEN

Performance Results: Decoder

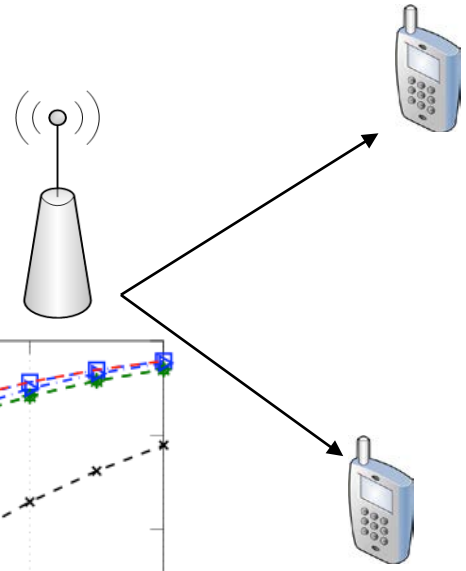


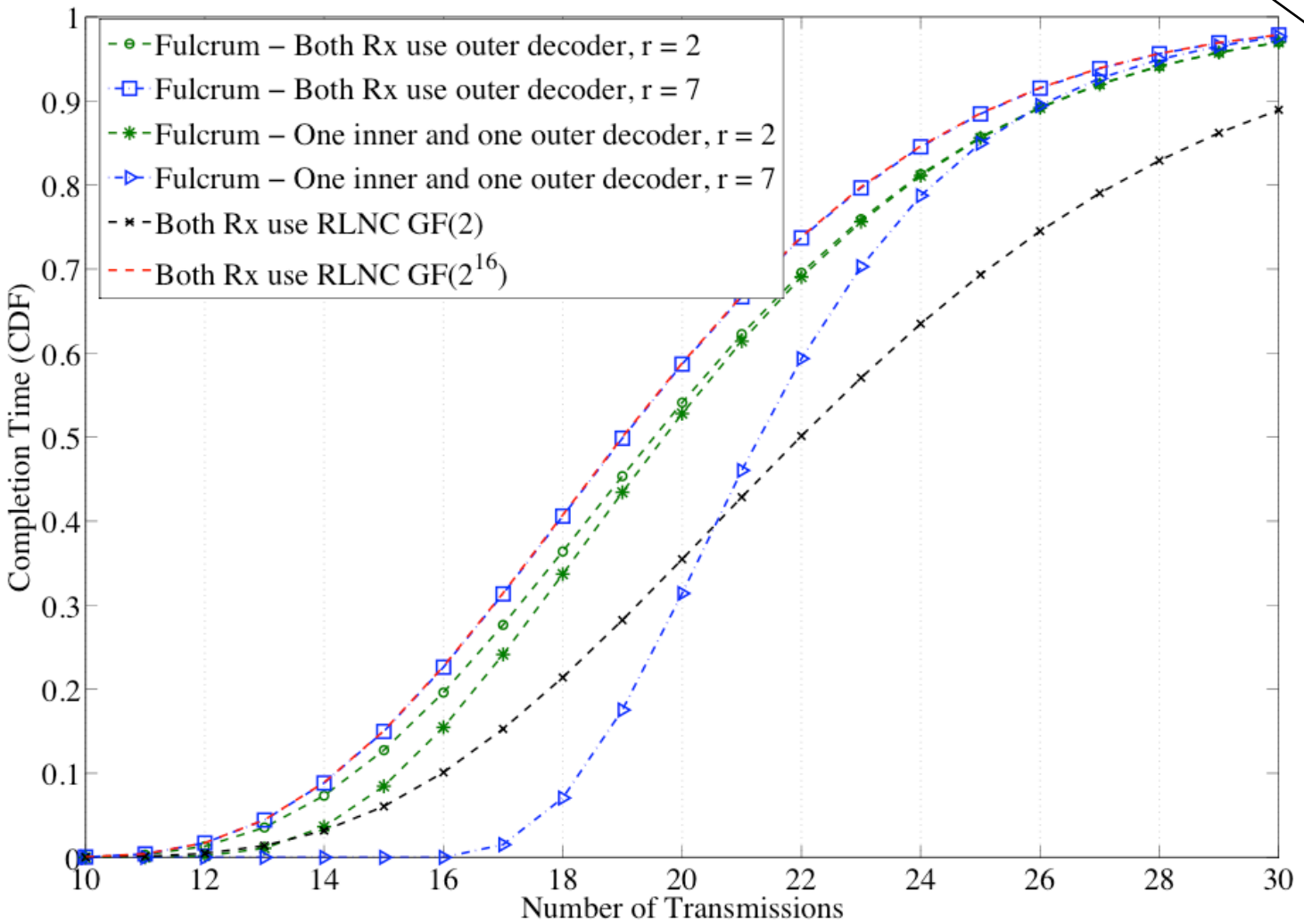
TECHNISCHE UNIVERSITÄT DRESDEN

Advantages

- Low overhead:
  - 1 + r/n ~ 1 bit per coefficient per packet → like GF(2)
  - Total transmitted packets: n + O(2^{-r}) → like higher fields
- Processing speed (complexity) compared to GF(2^8):
  - Encoder 5x to 25x faster
  - Decoder 2x to >20x faster
- Supports heterogeneous receivers
- Allows a fluid allocation of complexity
- Simple security support
- Network can implement a bare minimum:
  - Just XOR packets!

Heterogeneous Users



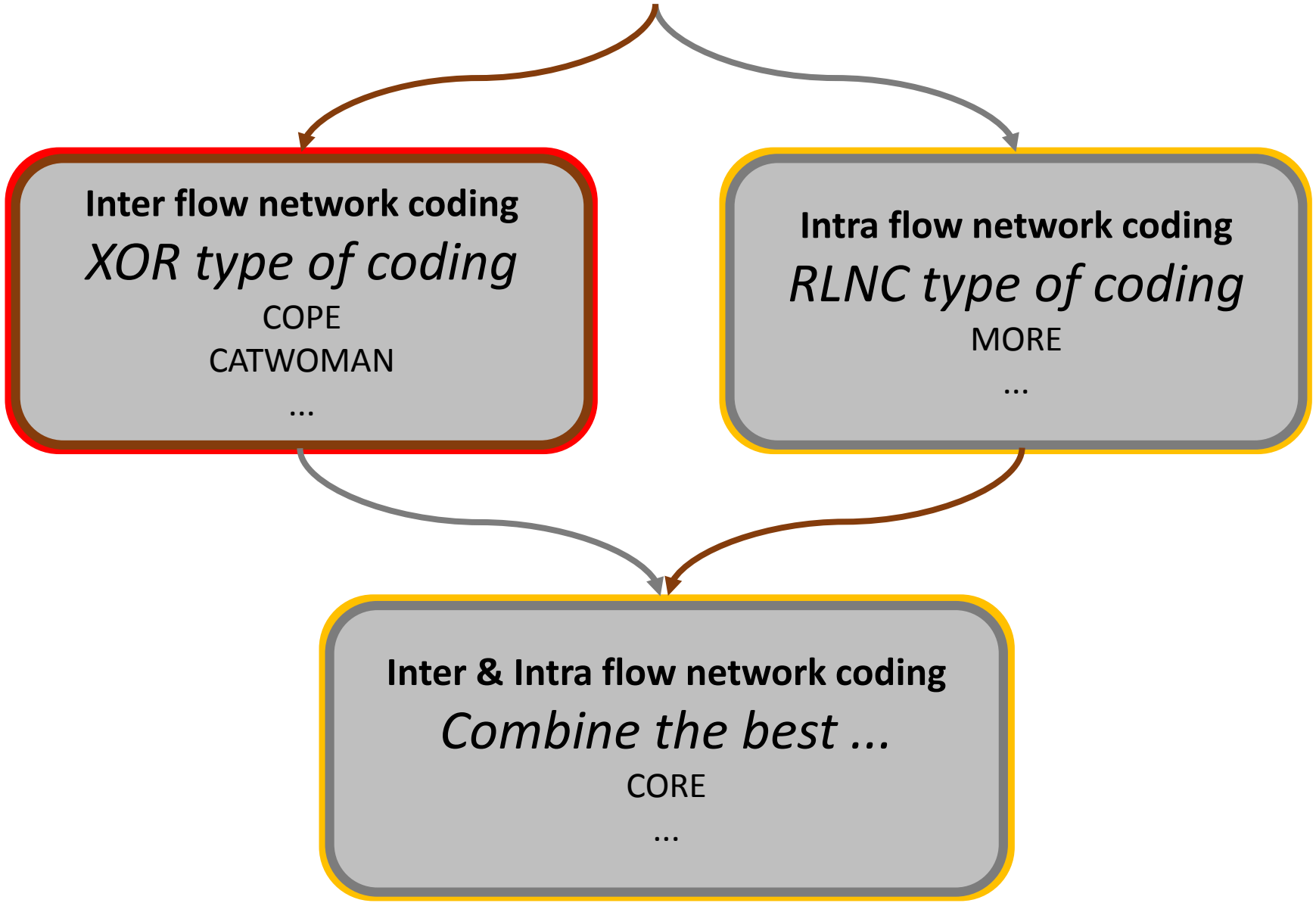


TECHNISCHE UNIVERSITÄT DRESDEN

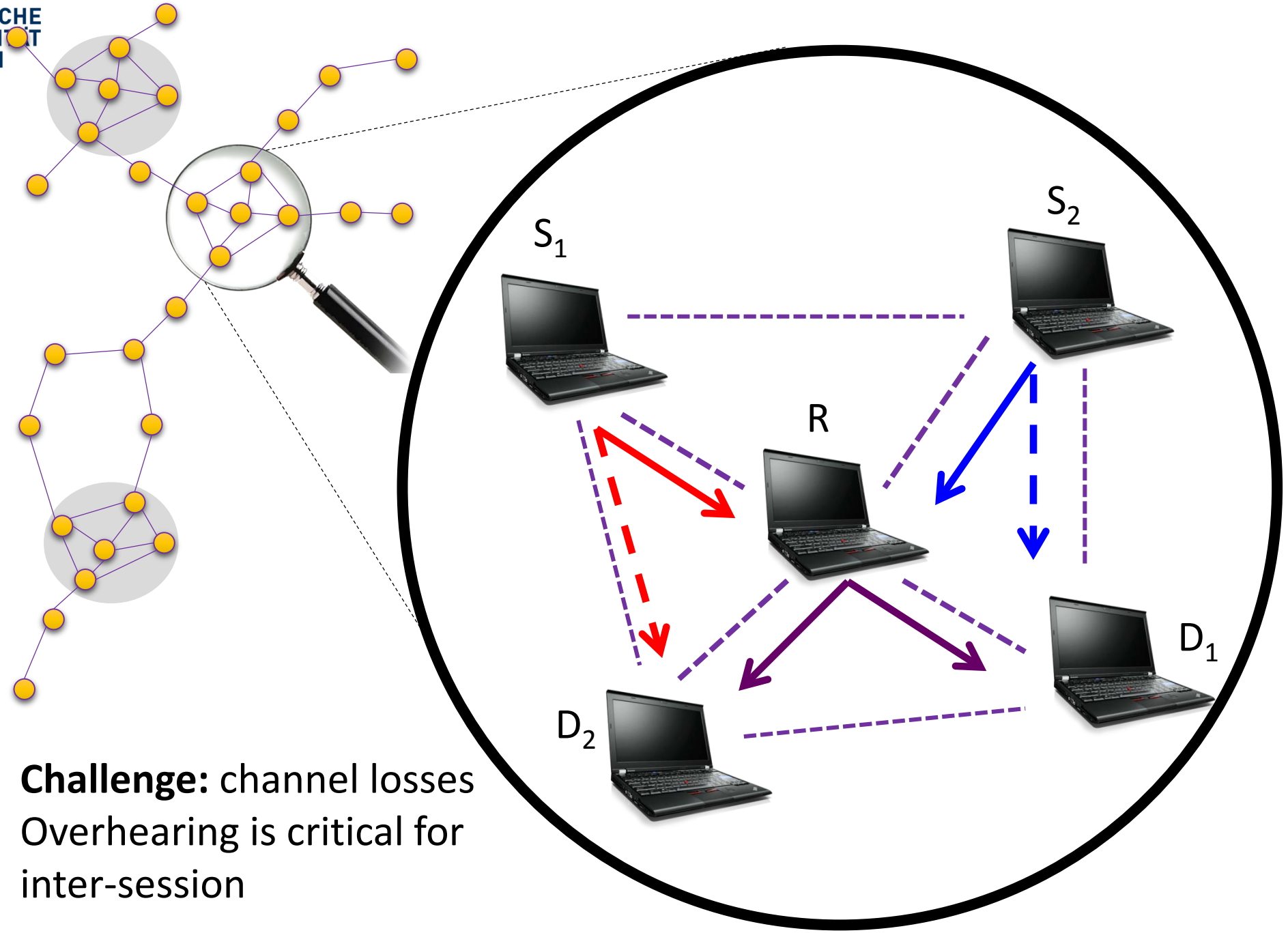
Combination of Inter and Intra Flow Network Coding

TECHNISCHE UNIVERSITÄT DRESDEN

What is Network Coding?



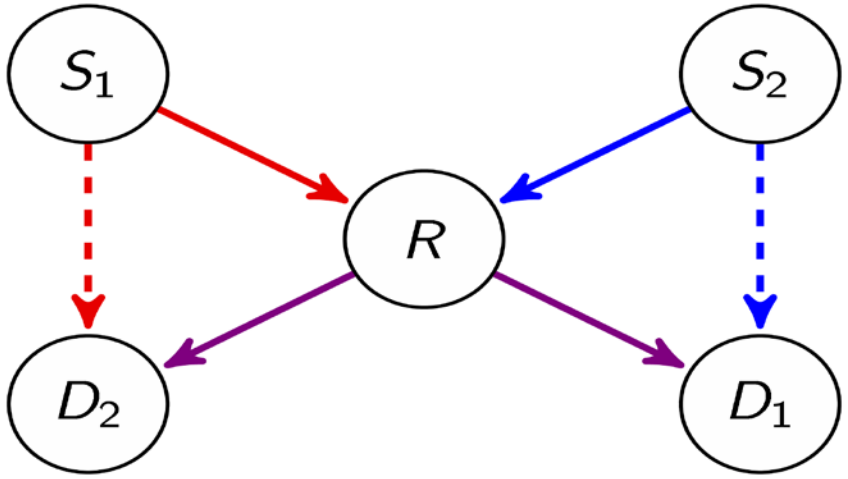
TECHNISCHE UNIVERSITÄT DRESDEN



Challenge: channel losses
Overhearing is critical for inter-session

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

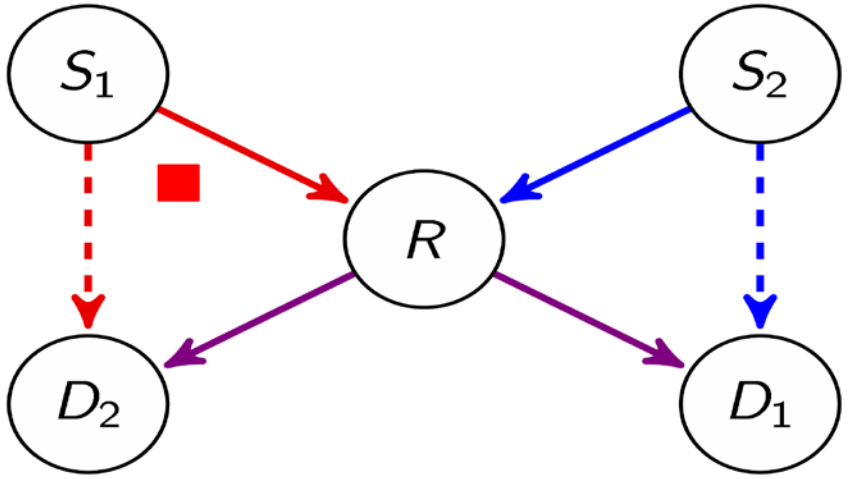


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

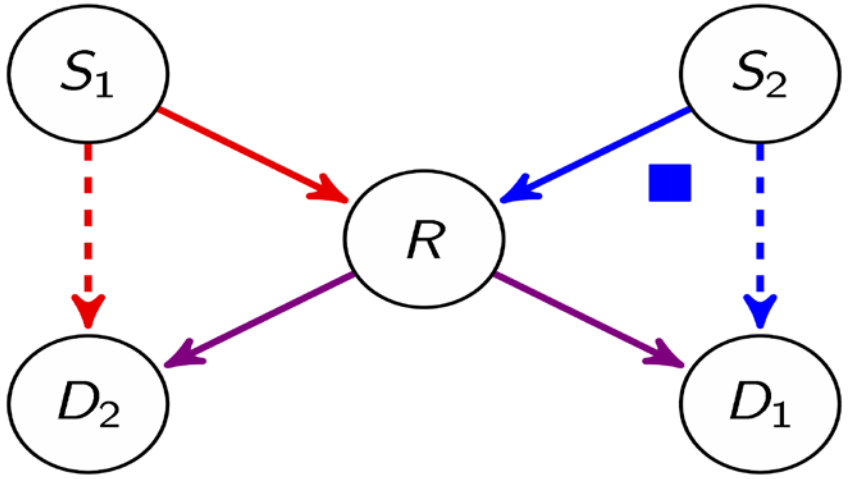


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

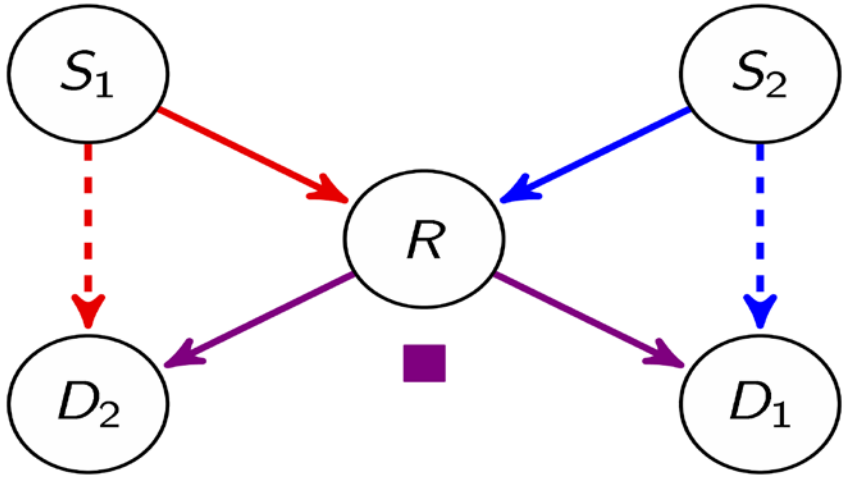


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

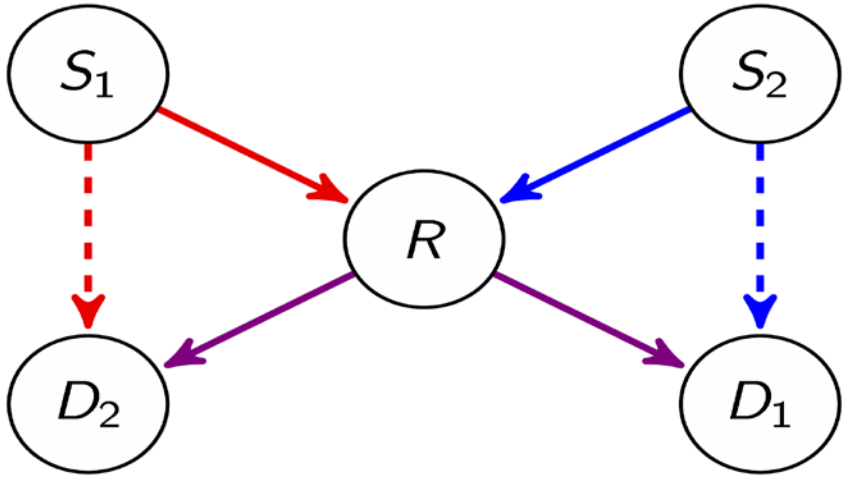


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

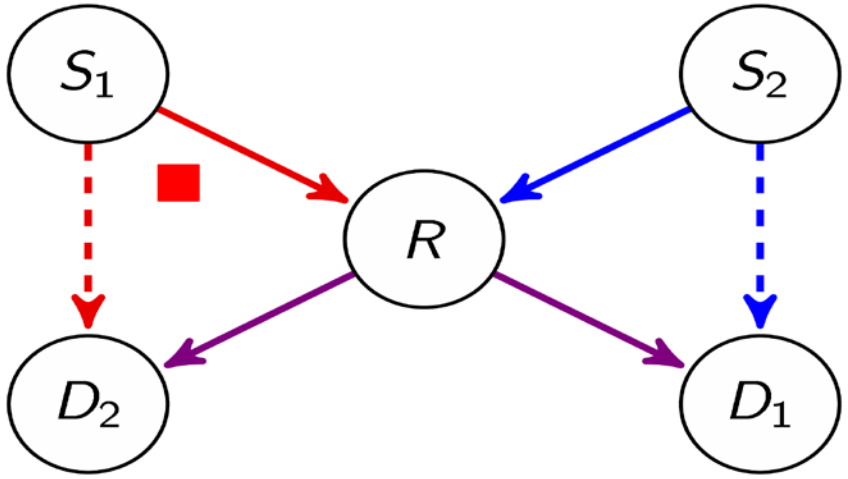


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

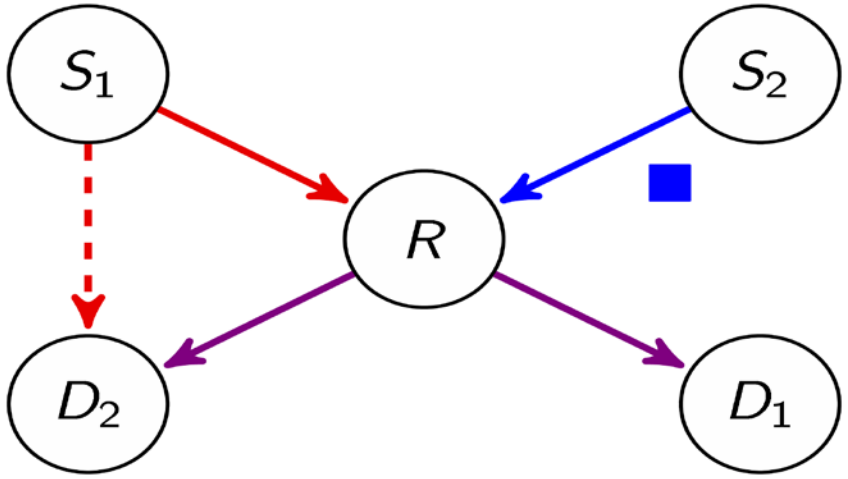


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

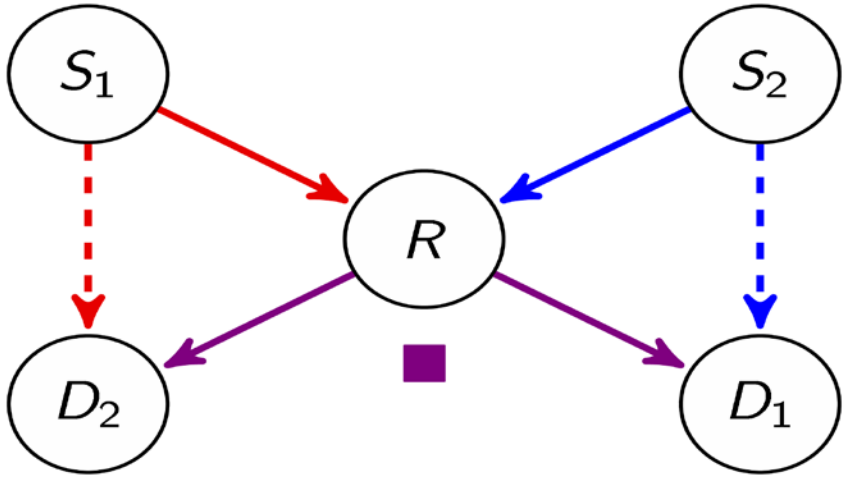


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

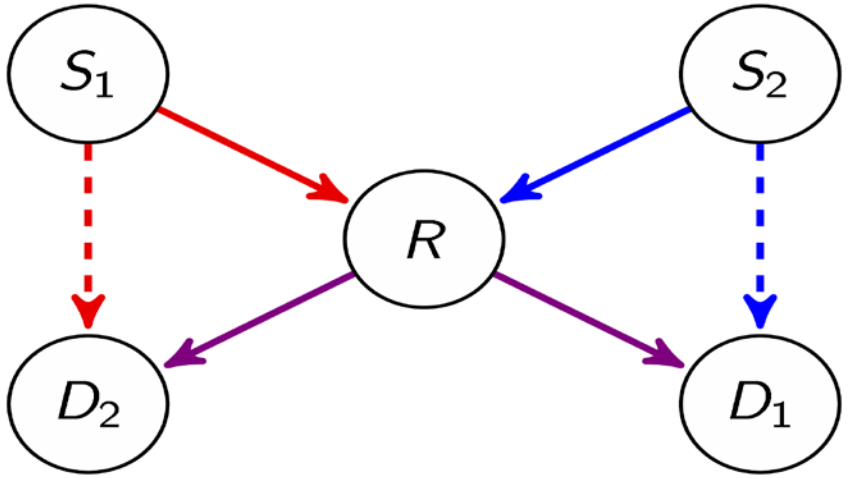


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

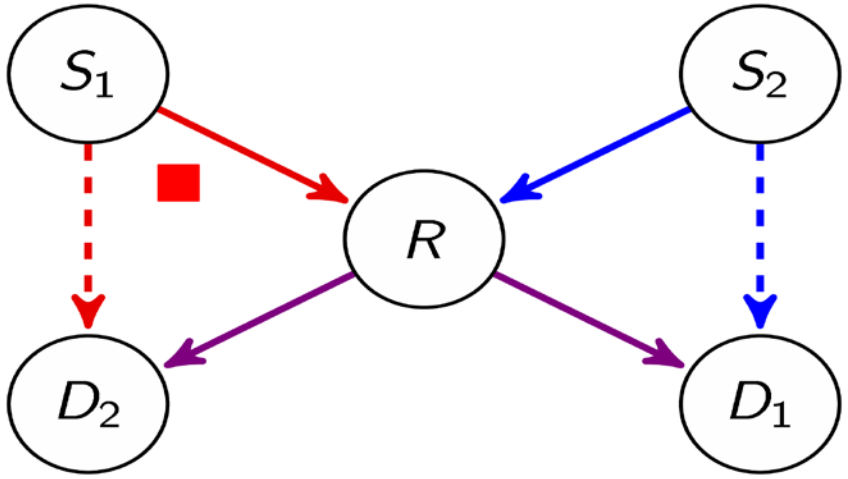


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

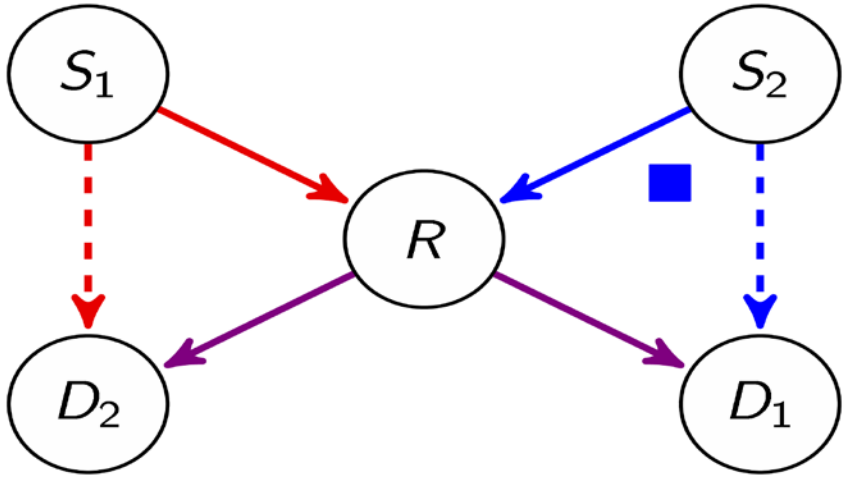


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE



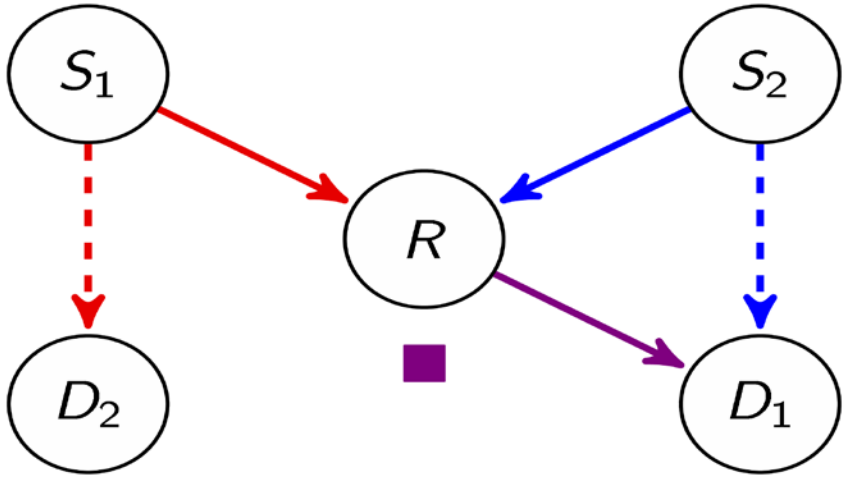
Relay has full rank!
Stop sources …

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE



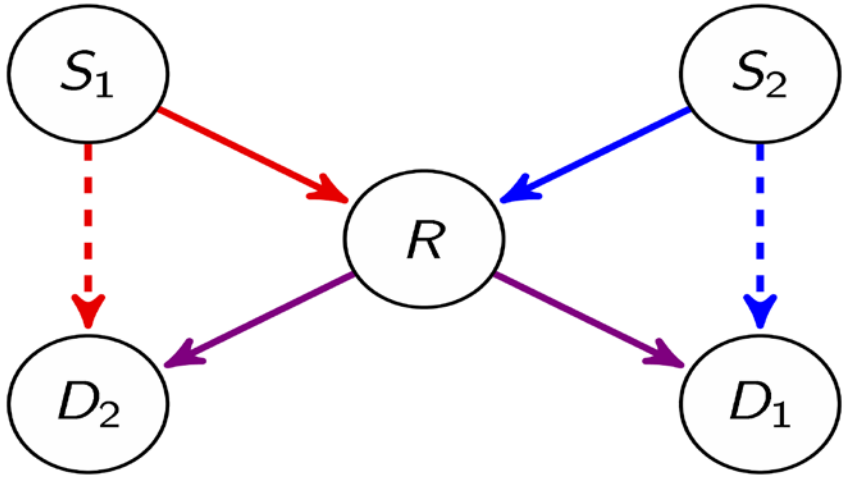
No retransmission

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

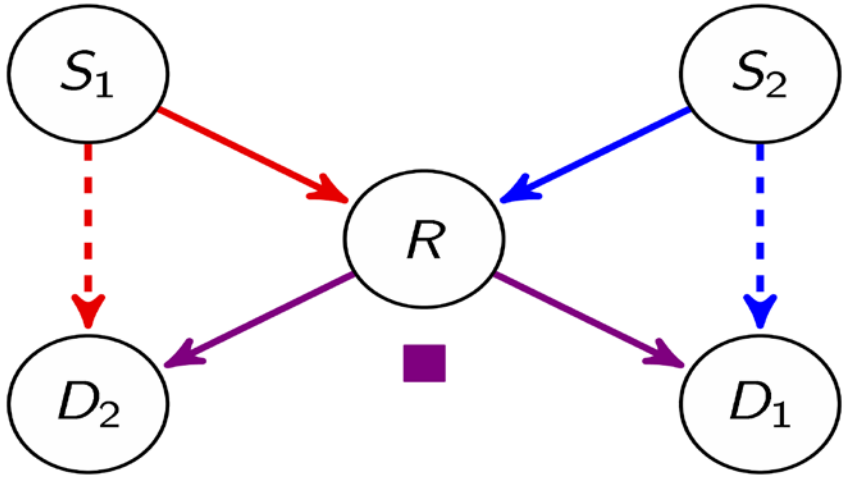


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

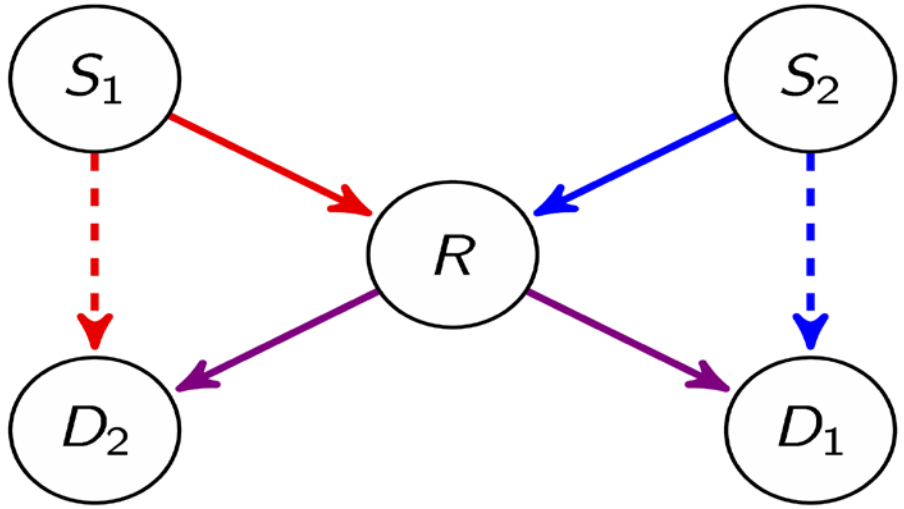


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						
7						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

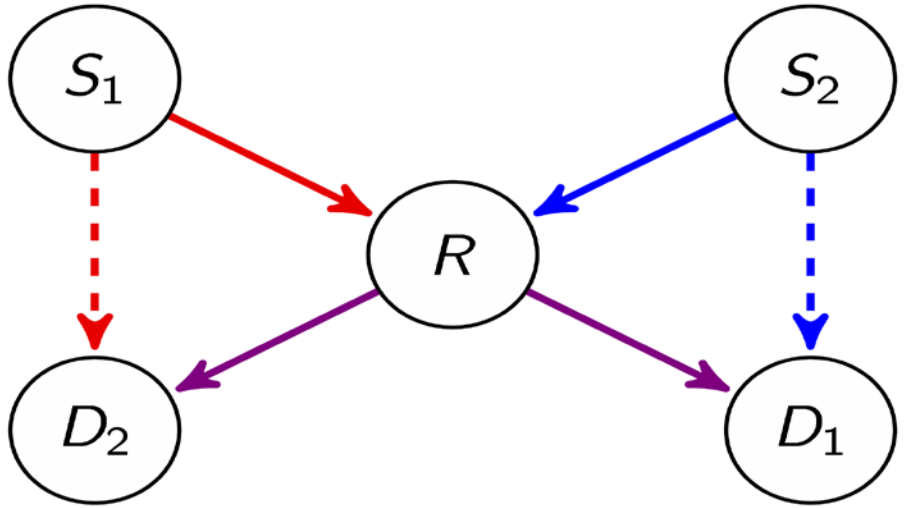


	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						

	P_1	P_2	P_3	P_1	P_2	P_3
1						
2						
3						
4						
5						
6						

TECHNISCHE UNIVERSITÄT DRESDEN

CORE

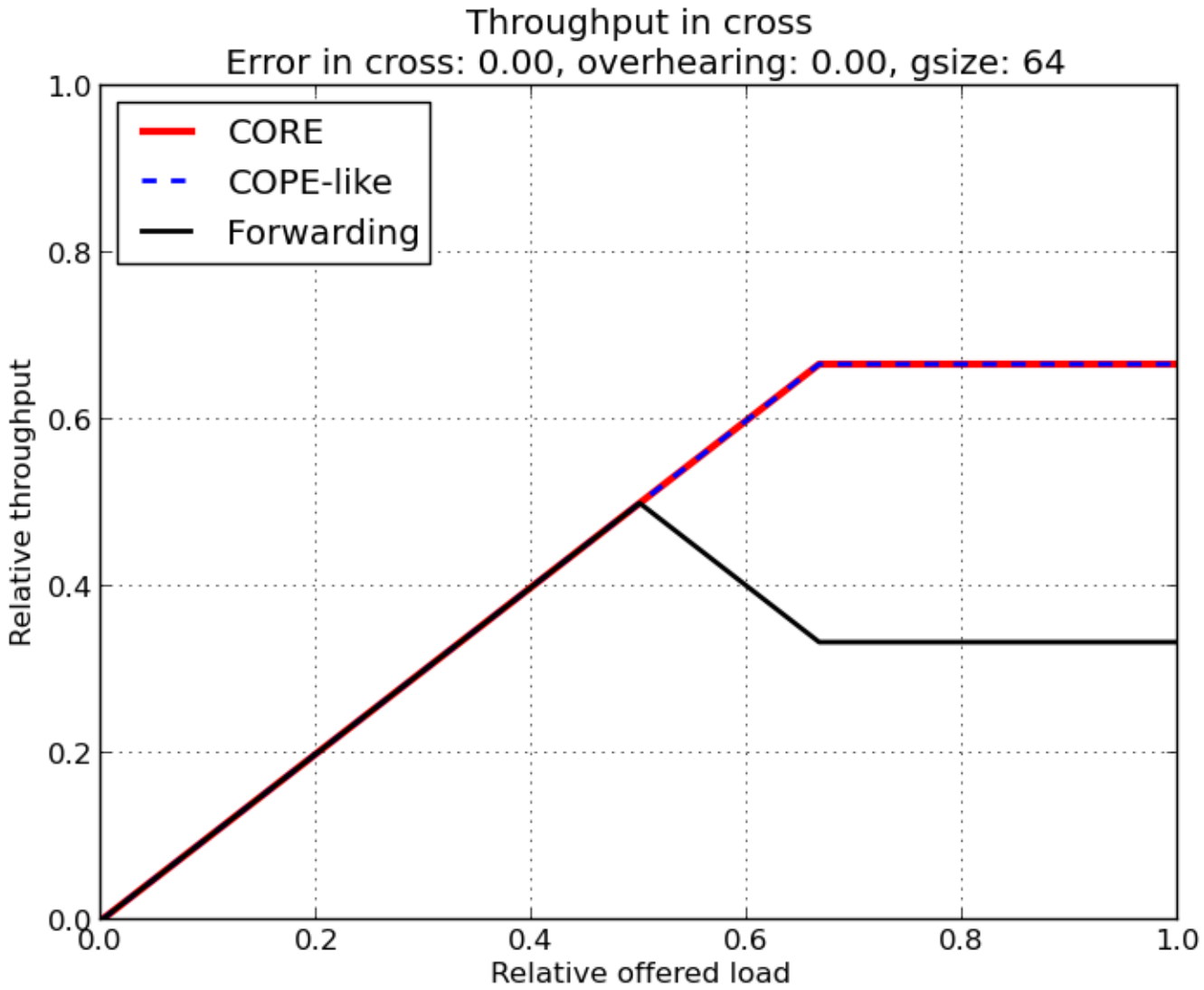


	P_1	P_2	P_3	P_1	P_2	P_3
1	■					
2		■				
3			■			
4				■		
5					■	
6						■

	P_1	P_2	P_3	P_1	P_2	P_3
1	■					
2		■				
3			■			
4				■		
5					■	
6						■

TECHNISCHE UNIVERSITÄT DRESDEN

Throughput vs Offered Load

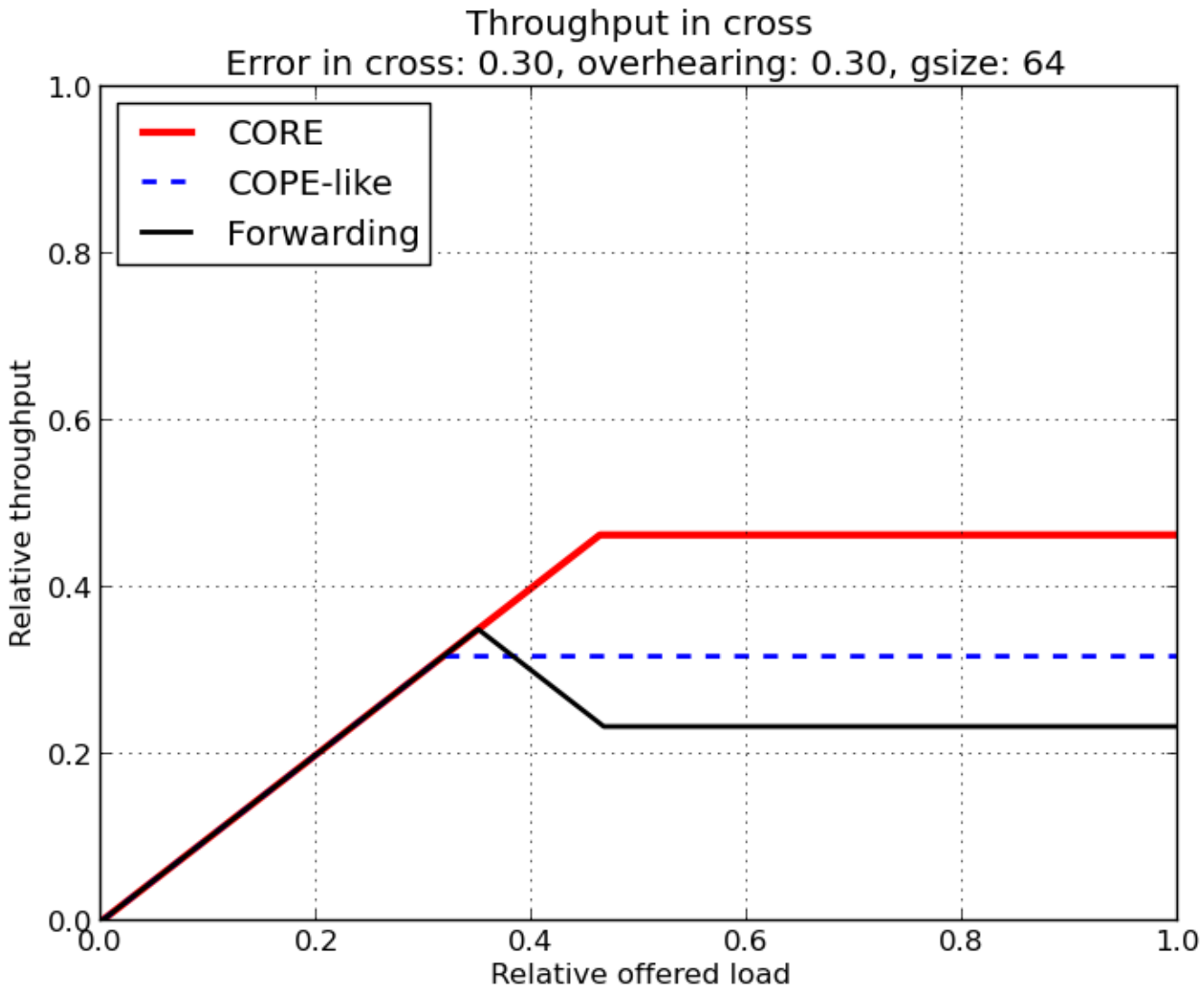


Symmetric losses

Loss prob = 0

TECHNISCHE UNIVERSITÄT DRESDEN

Throughput vs Offered Load

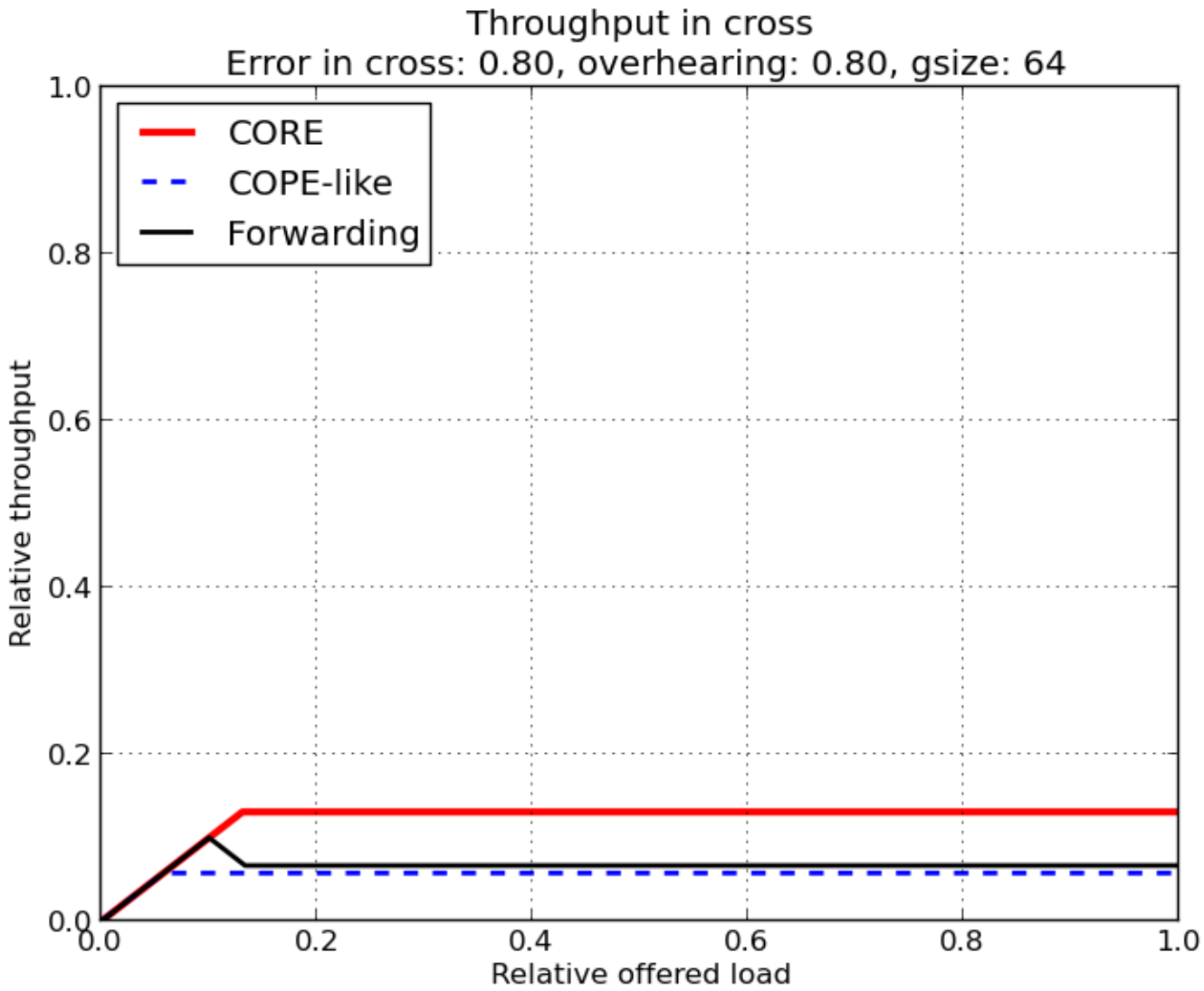


Symmetric losses

Loss prob = 0.3

TECHNISCHE UNIVERSITÄT DRESDEN

Throughput vs Offered Load



Symmetric losses

Loss prob = 0.8

TECHNISCHE UNIVERSITÄT DRESDEN

CORE