codeon

Network Coding Overview

NWRC meeting
March 11, 2013

codeon

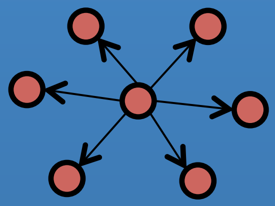
Coding → Network Coding

Coding Today

(all end-to-end)

Classical

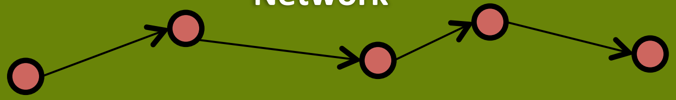
Multicast



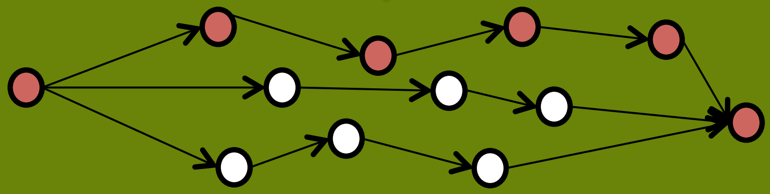
Coding Tomorrow

(using Random Linear Network Coding)

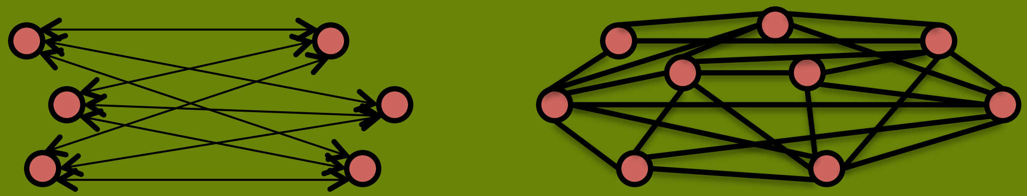
Network



Multipath



Multisource – Multi-destination / Mesh

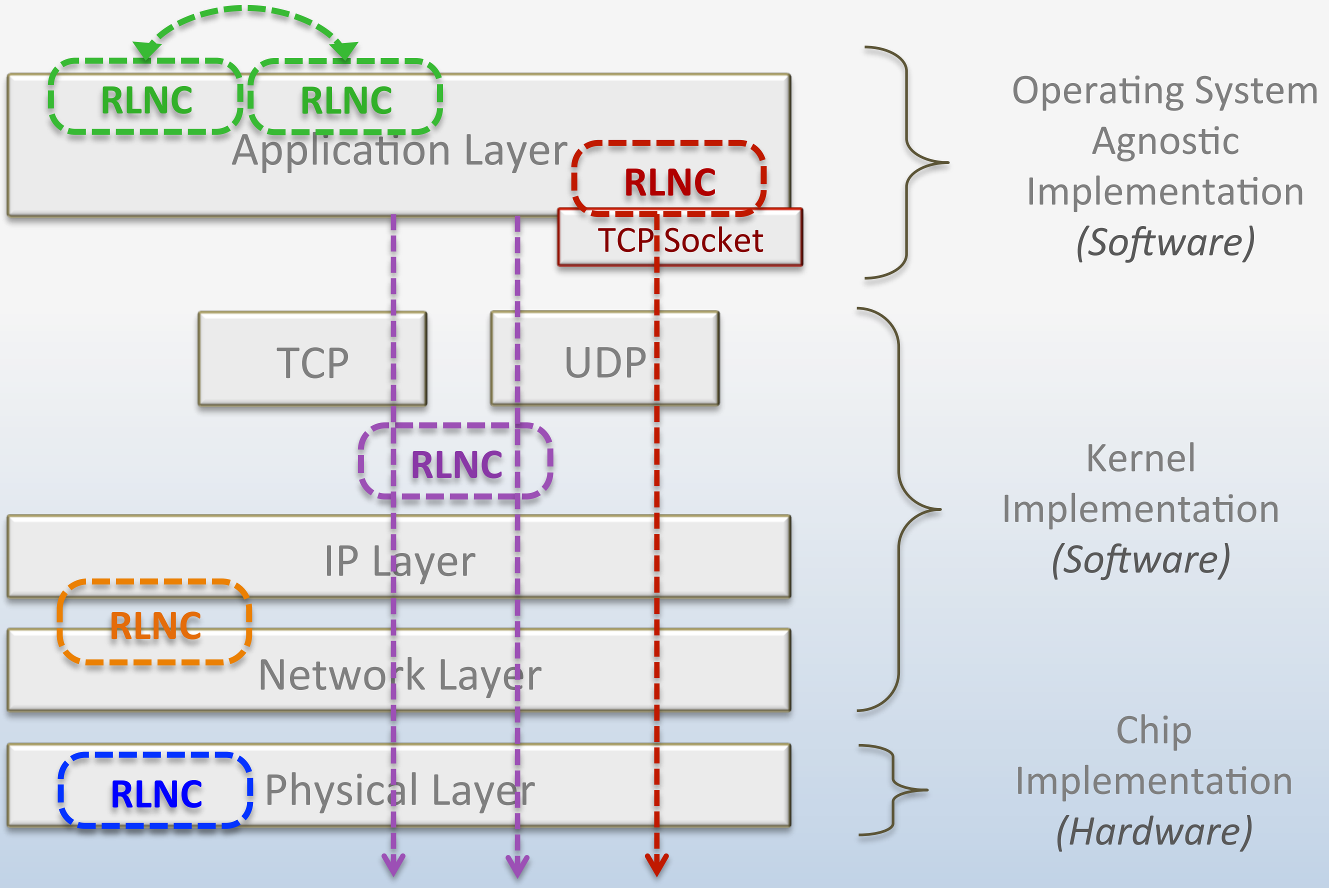


Why RNLC is Different

Code Capabilities	RLNC	Rateless (e.g. Fountain) Codes	Block Codes
Erasure correction	✓	✓	✓
Code is carried within each packet	✓	✗	✗
Completely distributed operation	✓	✗	✗
De-code using unencoded packets	✓	✗	✗
Able to generate valid codes from coded or unencoded packets	✓	✗	✗
Composability without decoding (add incremental redundancy)	✓	✗	✗
Encode data in a sliding window	✓	✗	✗

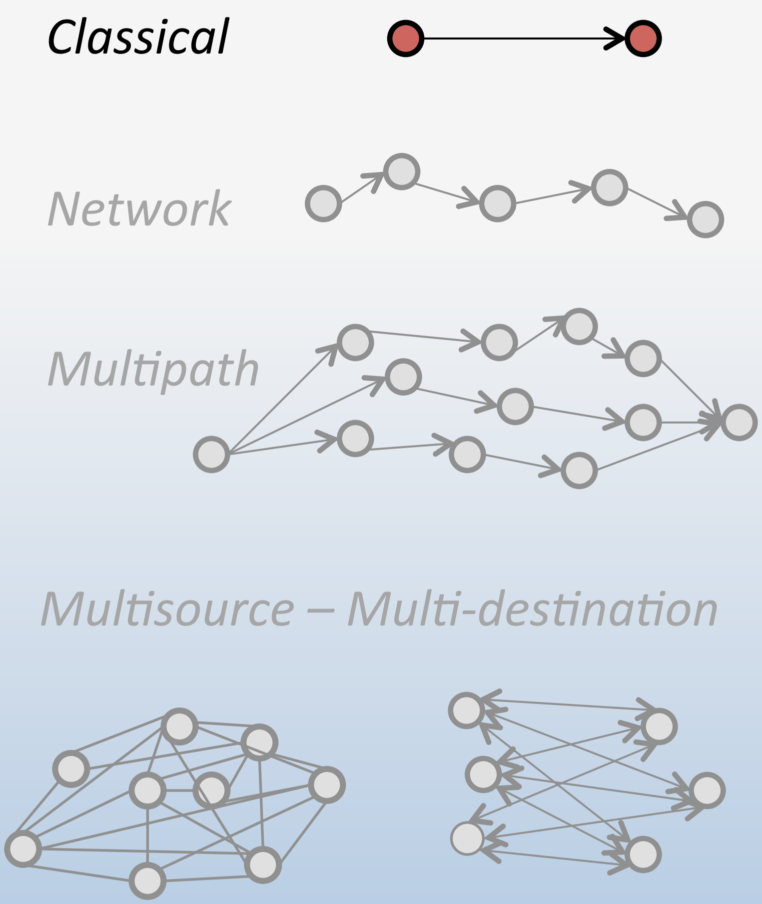
codeon

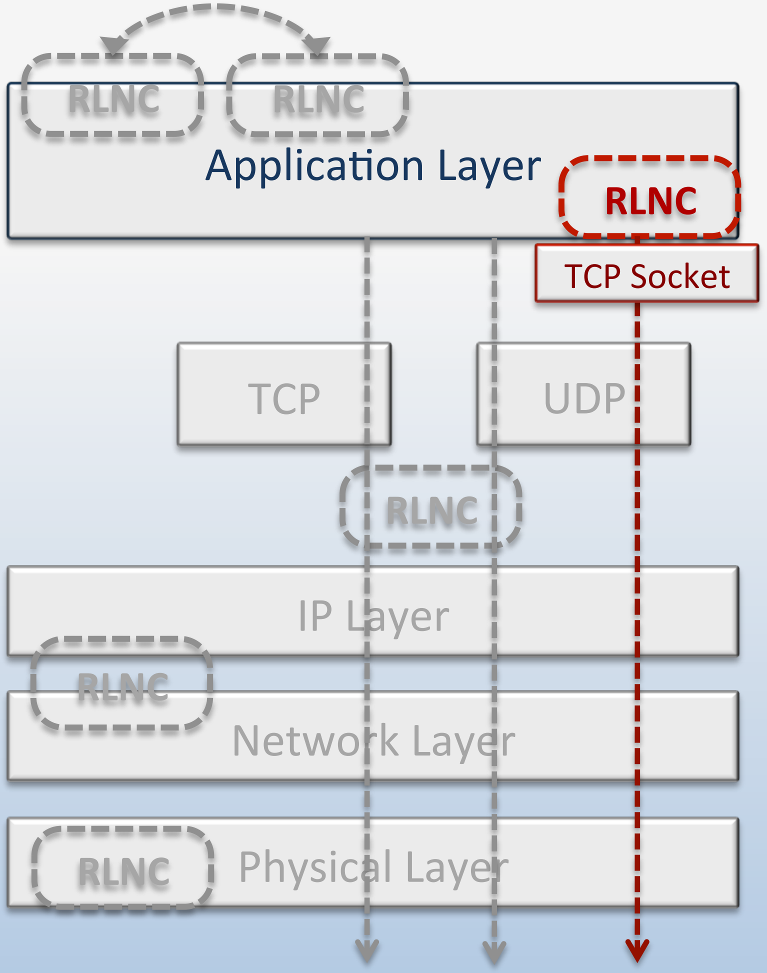
Initial RLNC Stack Implementations



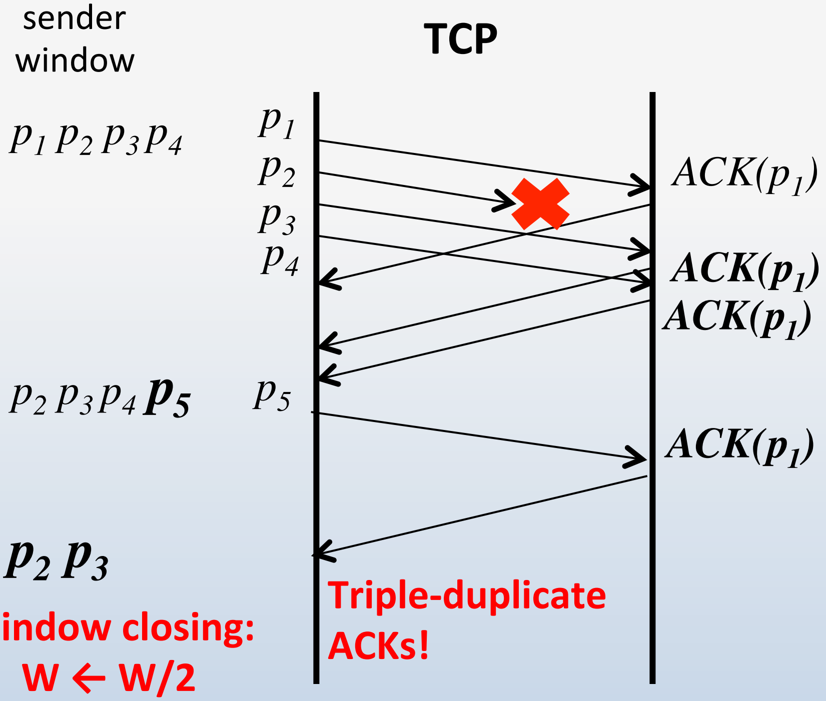
Implementable at any layer as a patch

Application Layer TCP Case Study





TCP Ill-Suited to Random Losses



TCP

sender window

$p_1\, p_2\, p_3\, p_4$

p_1

p_2

p_3

p_4

$ACK(p_1)$

$\boldsymbol{ACK(p_1)}$

$\boldsymbol{ACK(p_1)}$

$p_2\, p_3\, p_4\, \boldsymbol{p_5}$

p_5

$\boldsymbol{ACK(p_1)}$

$\boldsymbol{p_2\, p_3}$

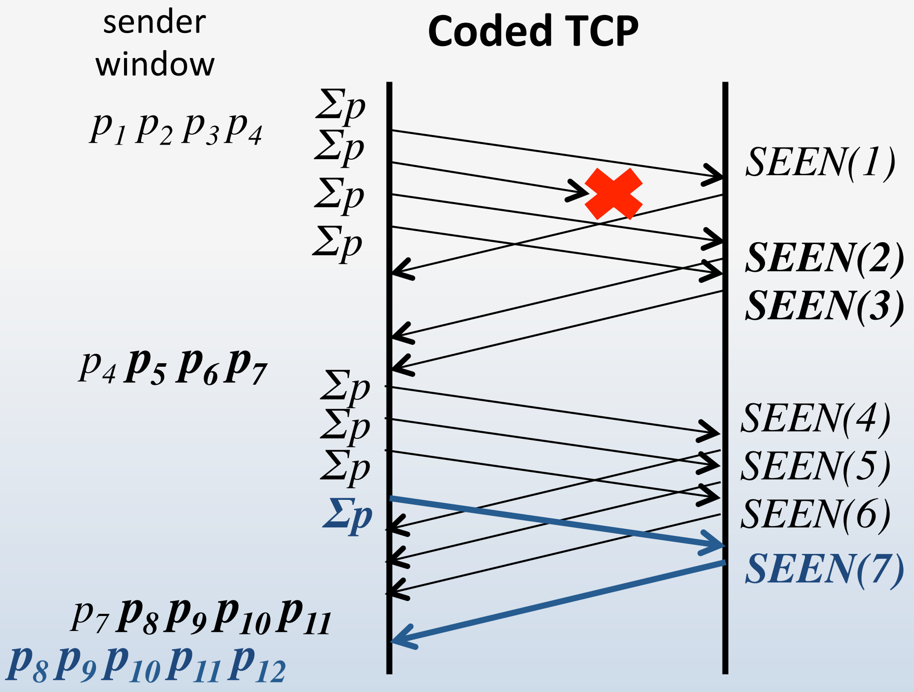
Window closing:
W ← W/2

Triple-duplicate ACKs!

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

codeon

Coded TCP Maintains Throughput

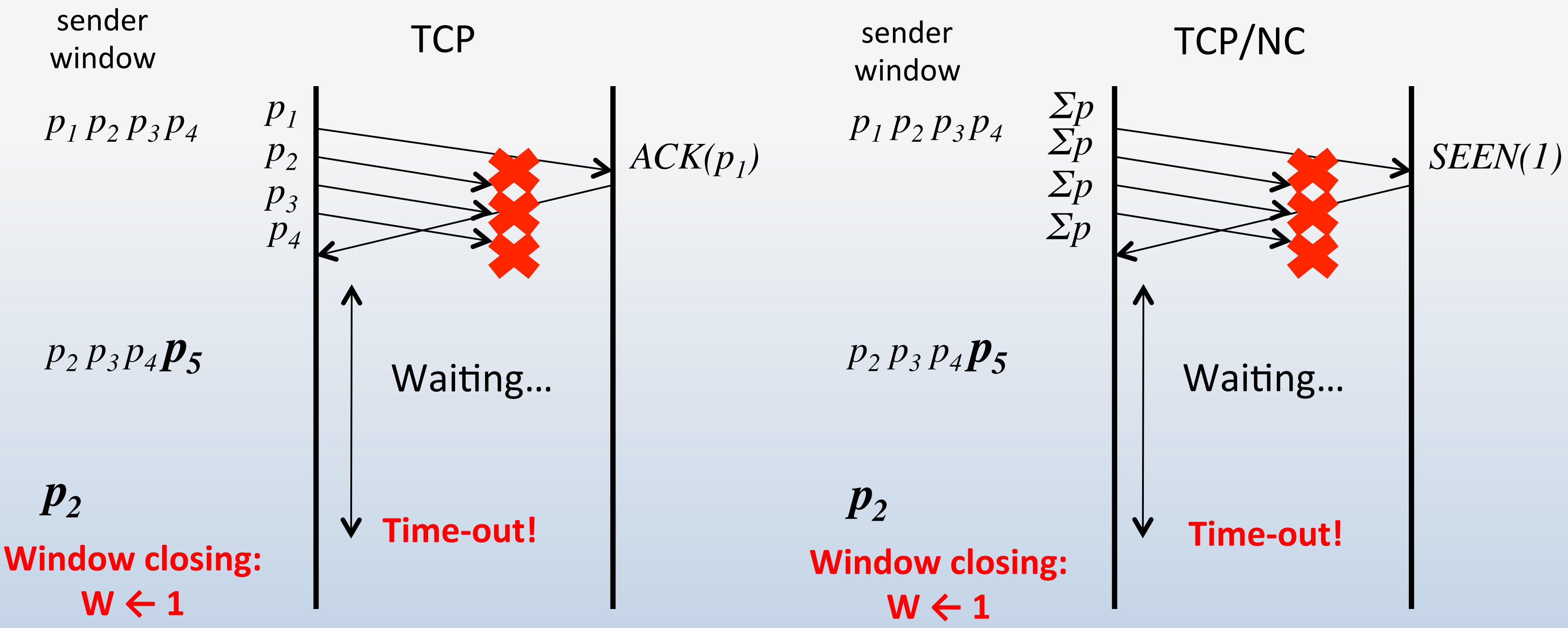


Prevents random losses being interpreted as congestion!

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

Minij Kim, Muriel Médard, João Barros
Modeling network coded TCP throughput: a simple model and its validation
Valuetools, 131-140 (2010)

Performance With Congestion Losses



sender window

TCP

$p_1\, p_2\, p_3\, p_4$

p_1
p_2
p_3
p_4

$ACK(p_1)$

$p_2\, p_3\, p_4\, \mathbf{p_5}$

Waiting…

$\mathbf{p_2}$

Time-out!

Window closing:
W ← 1

sender window

TCP/NC

$p_1\, p_2\, p_3\, p_4$

Σp
Σp
Σp
Σp

$SEEN(1)$

$p_2\, p_3\, p_4\, \mathbf{p_5}$

Waiting…

$\mathbf{p_2}$

Time-out!

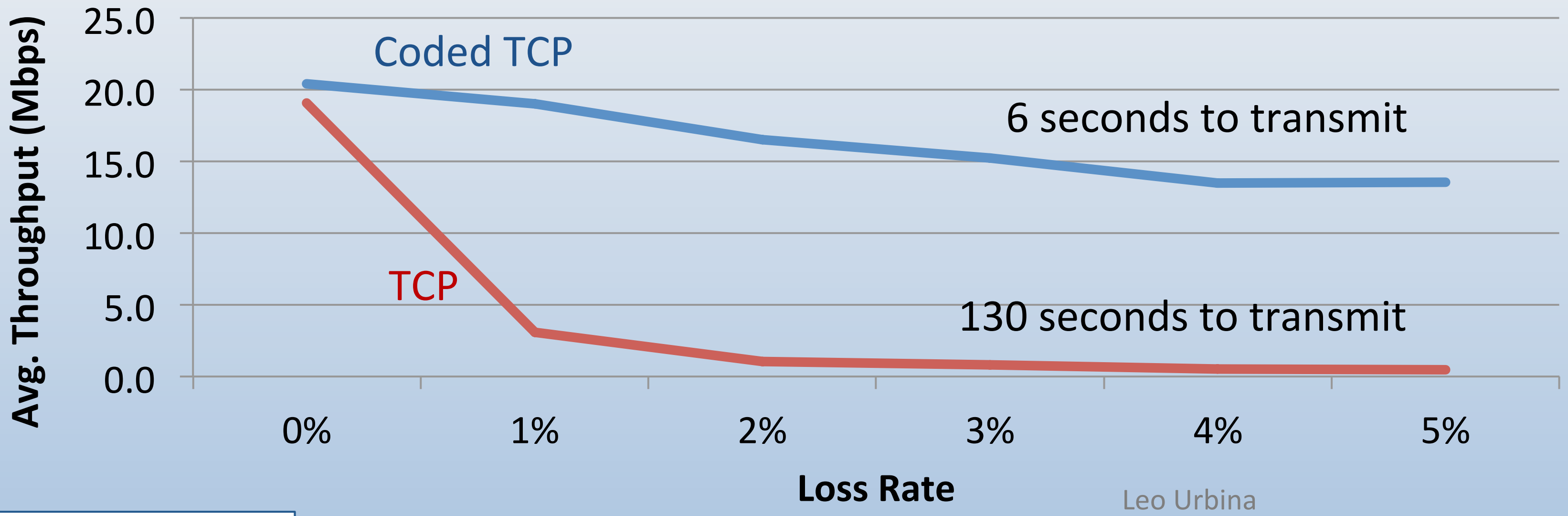
Window closing:
W ← 1

Still allows congestion control while masking random losses!

Experimental Results: Single Path

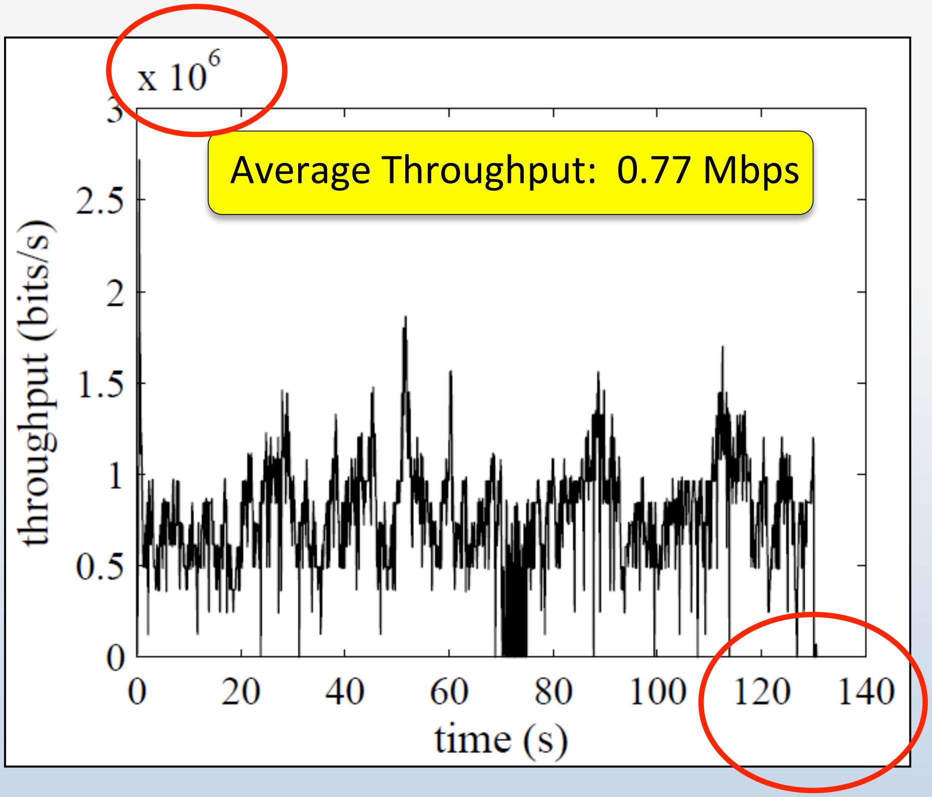
- Server: Amazon EC2 instance in CA
- Client: Desktop at RLE MIT, using WiFi(s)

Loss rate	0%	1%	2%	3%	4%	5%
TCP (Mbps)	19.04	3.07	1.07	0.82	0.53	0.47
CTCP (Mbps)	20.40	18.99	16.52	15.20	13.46	13.53

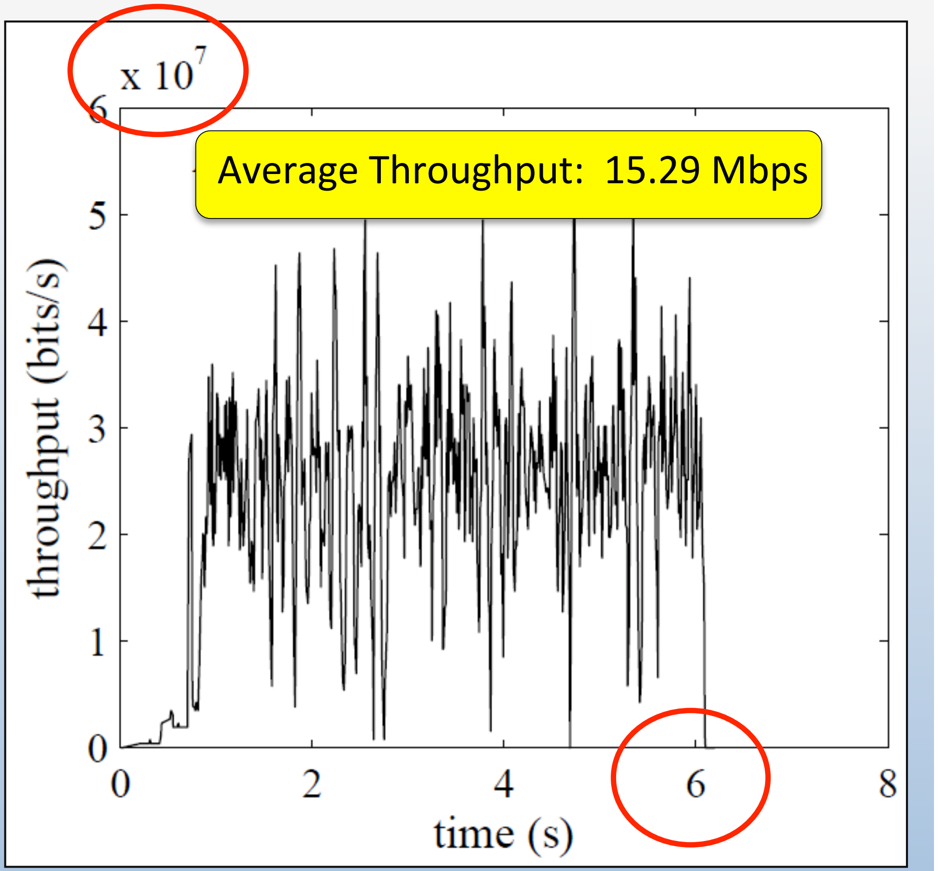


Leo Urbina
Applying Network Coding to TCP
MIT M.Eng. Thesis 2012

Experimental Results: Single Path



Average Throughput: 0.77 Mbps



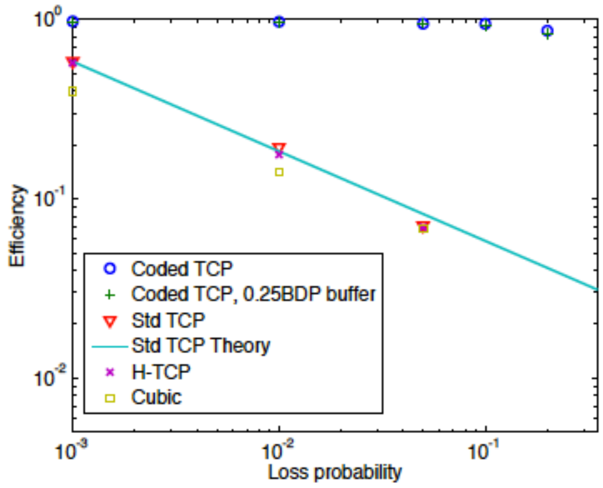
Average Throughput: 15.29 Mbps

Example: 2% loss

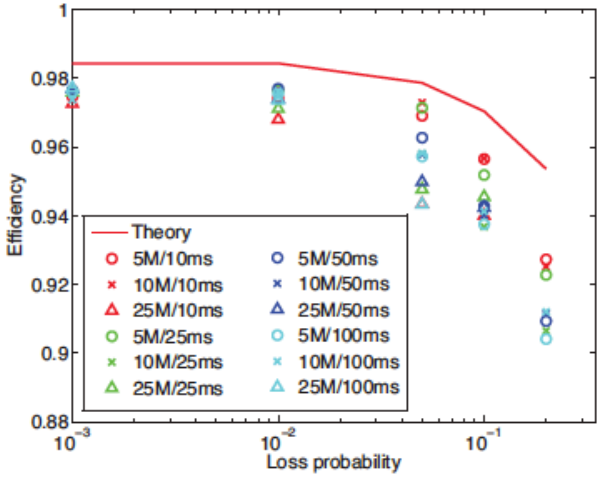
codeon

Testbed Measurements

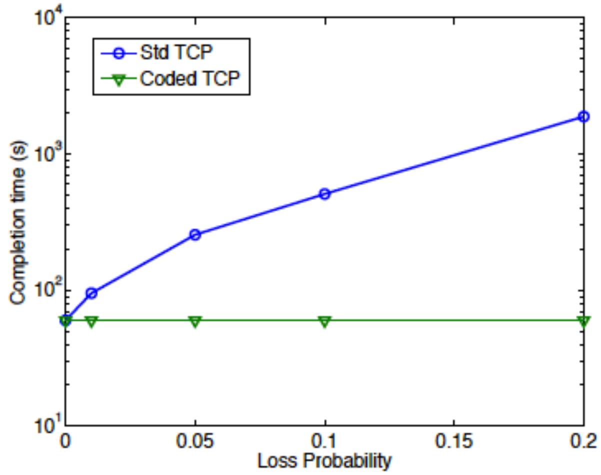
60 s video
Full
download



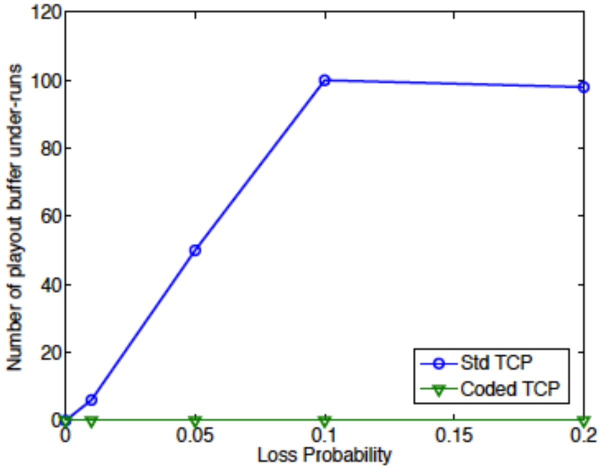
(a) Link 25Mbps, RTT 20ms



(b) Coded TCP



(a) Completion Time

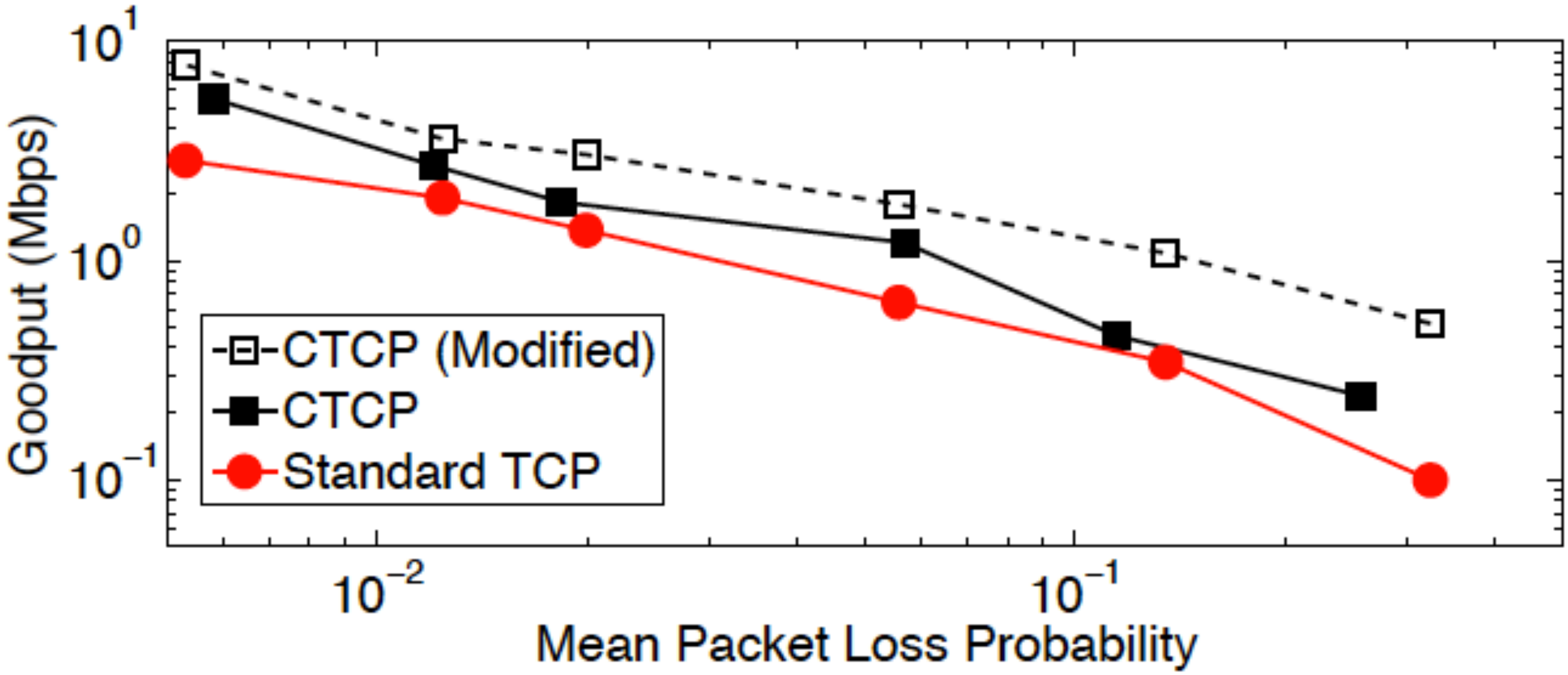


(b) Buffer Under-runs

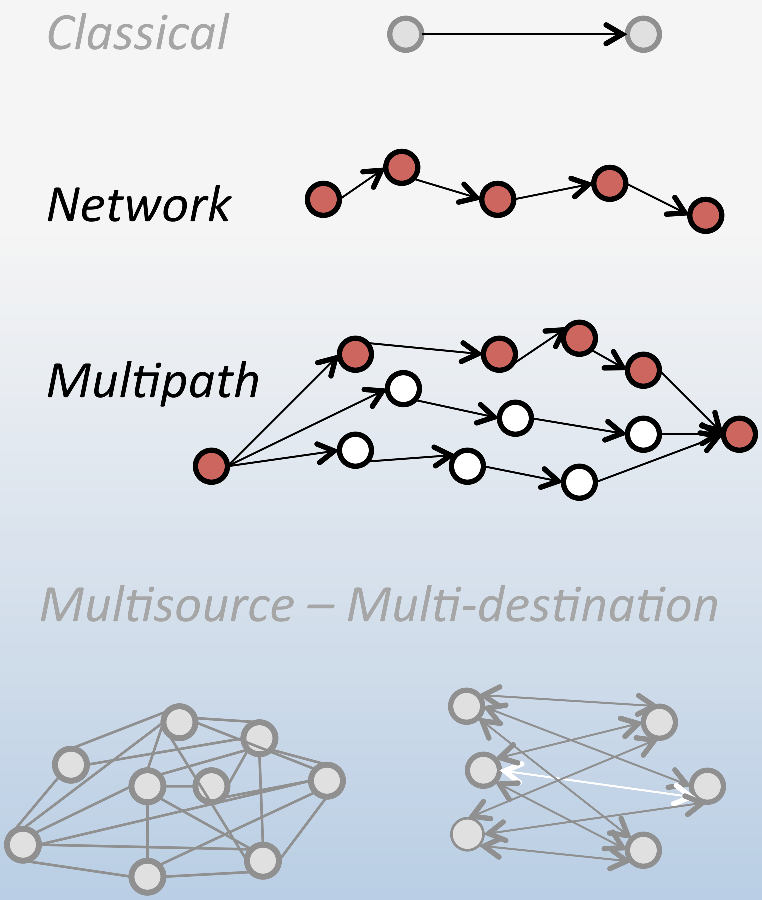
60 s video
Progressive
download

Minji Kim, Jason Cloud, Ali ParandehGheibi, Leonardo Urbina, Kerim Fouli, Douglas Leith, Muriel Médard
Network Coded TCP (CTCP) arXiv: 1212.2291v2

WiMax Experiments



Coded TCP Network Improvements

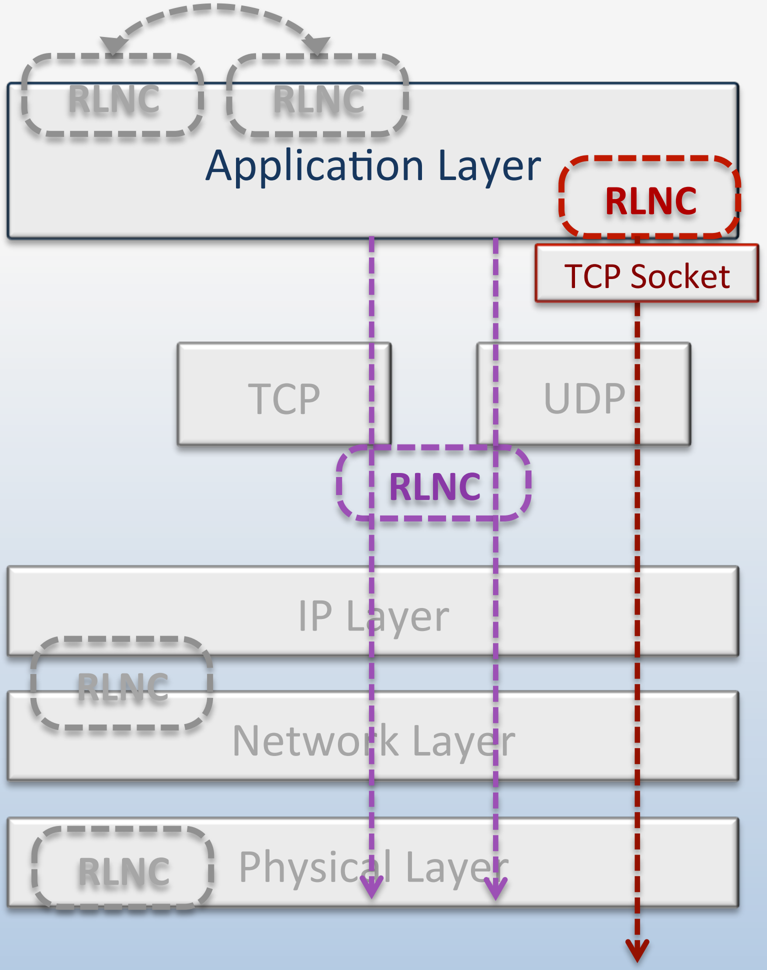


Classical

Network

Multipath

Multisource – Multi-destination



RLNC

RLNC

Application Layer

RLNC

TCP Socket

TCP

UDP

RLNC

IP Layer

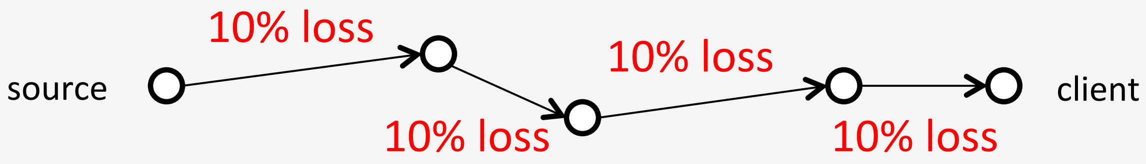
RLNC

Network Layer

RLNC

Physical Layer

Coded TCP (CTCP) Over a Network

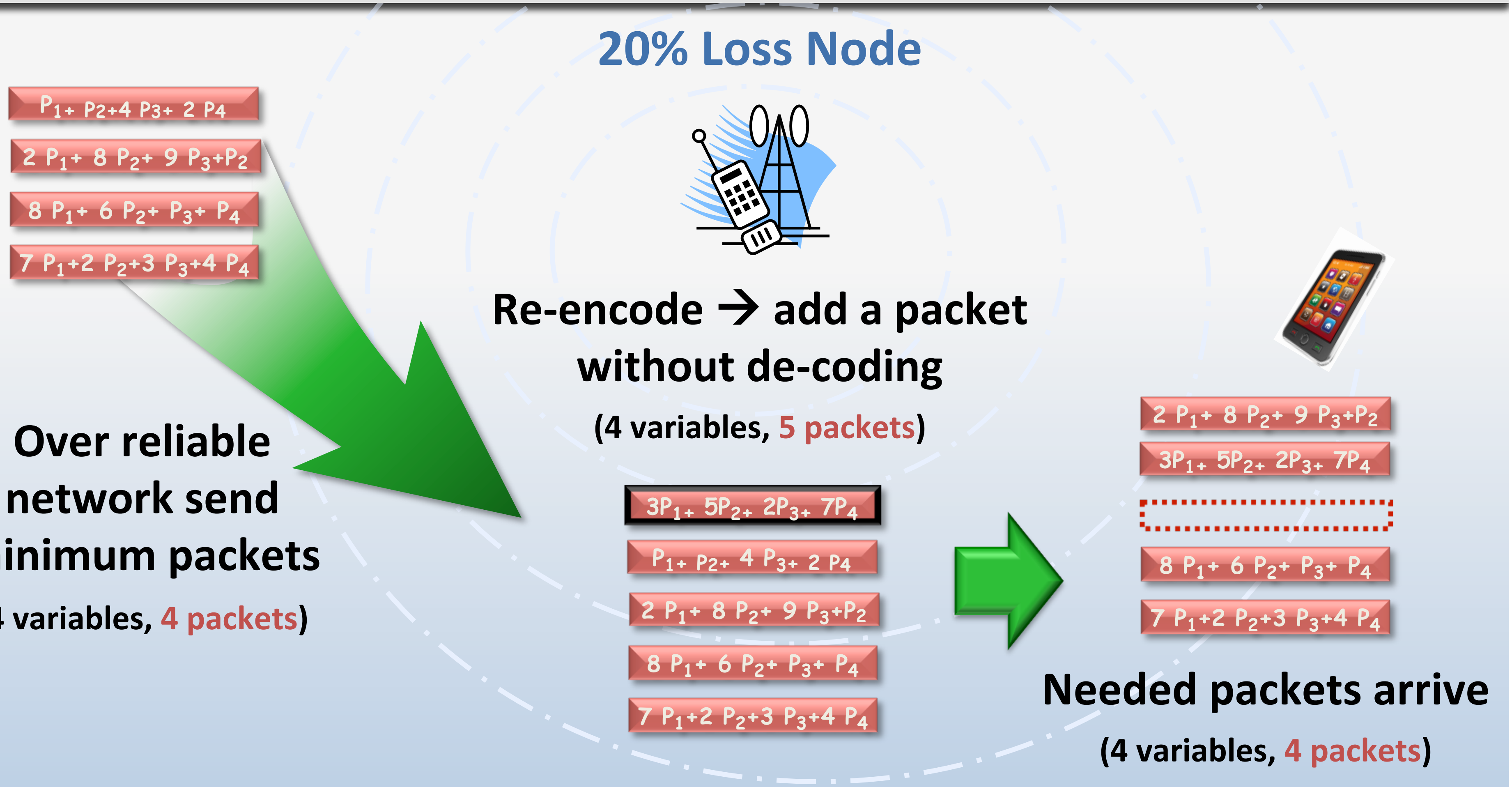


Longer paths lead to higher losses, resulting in poor performance

	RLNC (using Coded TCP)	**End-to-End Coding** (including Fountain Codes)
Best Possible Throughput Rate	0.9	0.66

TCP's performance degrades super-linearly with end-to-end loss rate → benefits of Coded TCP are even greater in this scenario

Composability: Redundancy When Needed



20% Loss Node

$P_1 + P_2 + 4 P_3 + 2 P_4$

$2 P_1 + 8 P_2 + 9 P_3 + P_2$

$8 P_1 + 6 P_2 + P_3 + P_4$

$7 P_1 + 2 P_2 + 3 P_3 + 4 P_4$

Over reliable network send minimum packets

(4 variables, 4 packets)

Re-encode → add a packet without de-coding

(4 variables, 5 packets)

$3P_1 + 5P_2 + 2P_3 + 7P_4$

$P_1 + P_2 + 4 P_3 + 2 P_4$

$2 P_1 + 8 P_2 + 9 P_3 + P_2$

$8 P_1 + 6 P_2 + P_3 + P_4$

$7 P_1 + 2 P_2 + 3 P_3 + 4 P_4$

$2 P_1 + 8 P_2 + 9 P_3 + P_2$

$3P_1 + 5P_2 + 2P_3 + 7P_4$

$8 P_1 + 6 P_2 + P_3 + P_4$

$7 P_1 + 2 P_2 + 3 P_3 + 4 P_4$

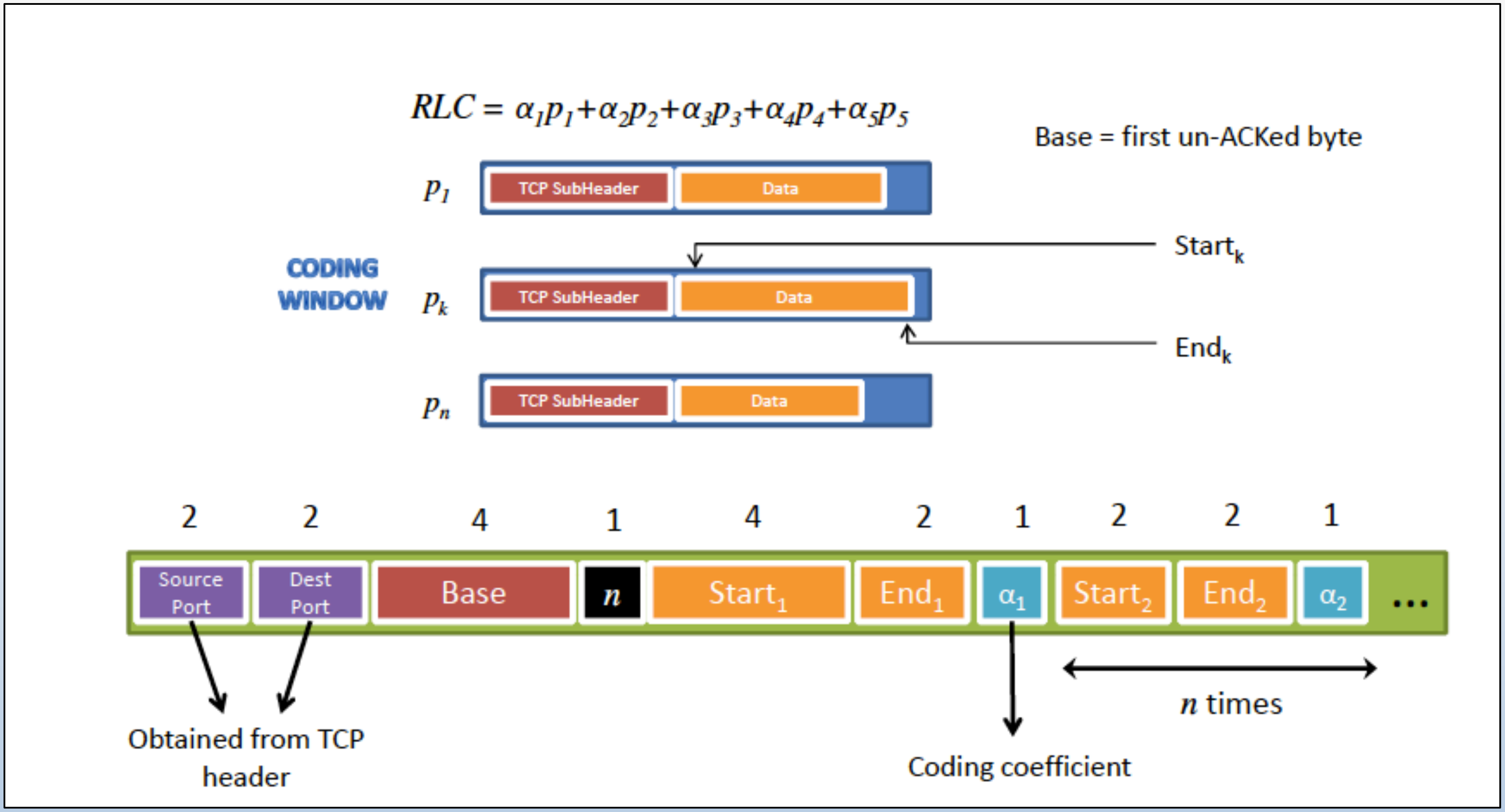
Needed packets arrive

(4 variables, 4 packets)

RLNC allows optimized packet sends at any node

codeon

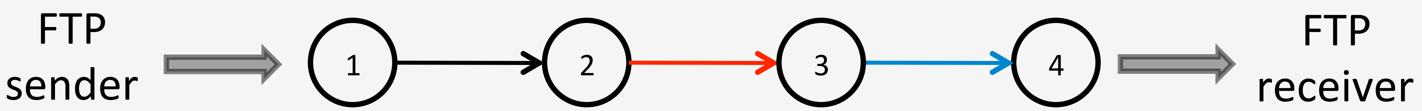
Coding Coefficients Carried within Packet



$$RLC = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 + \alpha_4 p_4 + \alpha_5 p_5$$

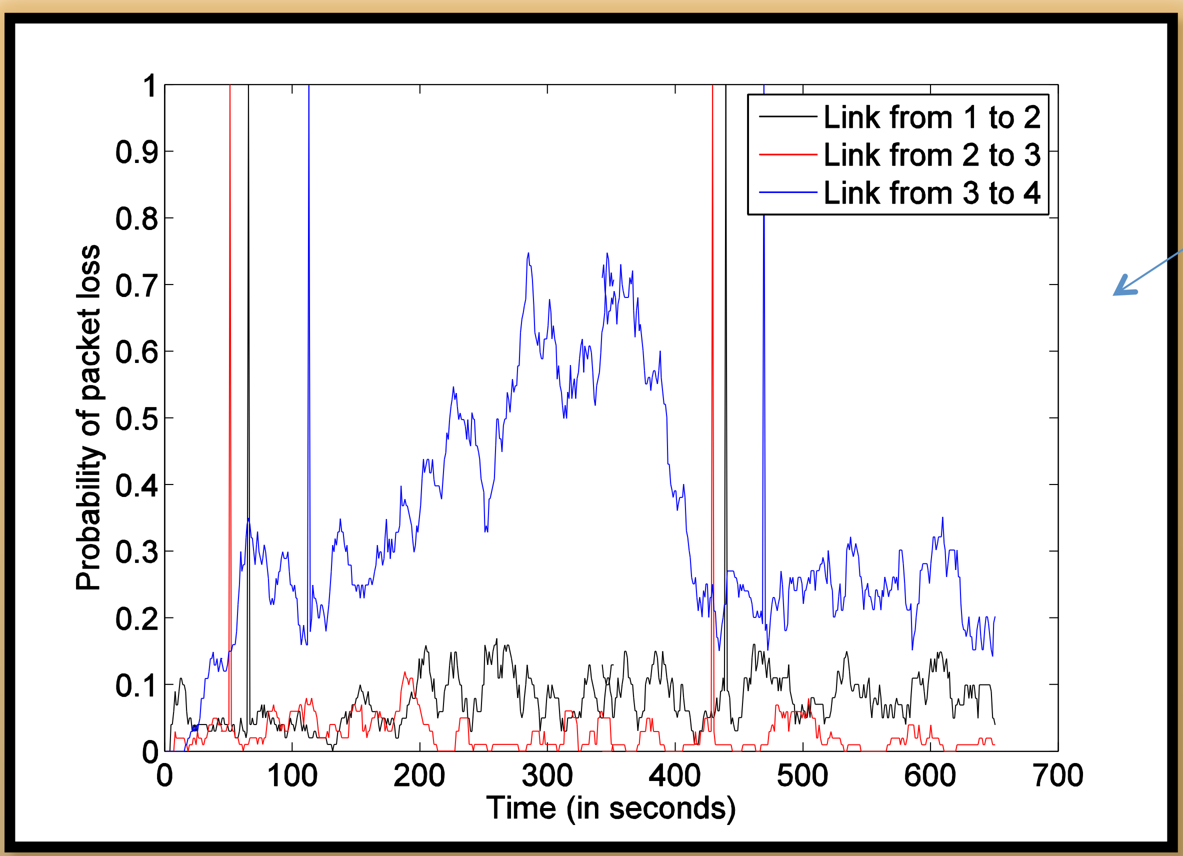
Jay Kumar Sundararajan, Devavrat Shah, Muriel Médard, Szymon Jakubczak, Michael Mitzenmacher, João Barros, Network Coding Meets TCP: Theory and Implementation. Proceedings of the IEEE 99 (3): 490-512 (2011)

Multihop Wireless Network



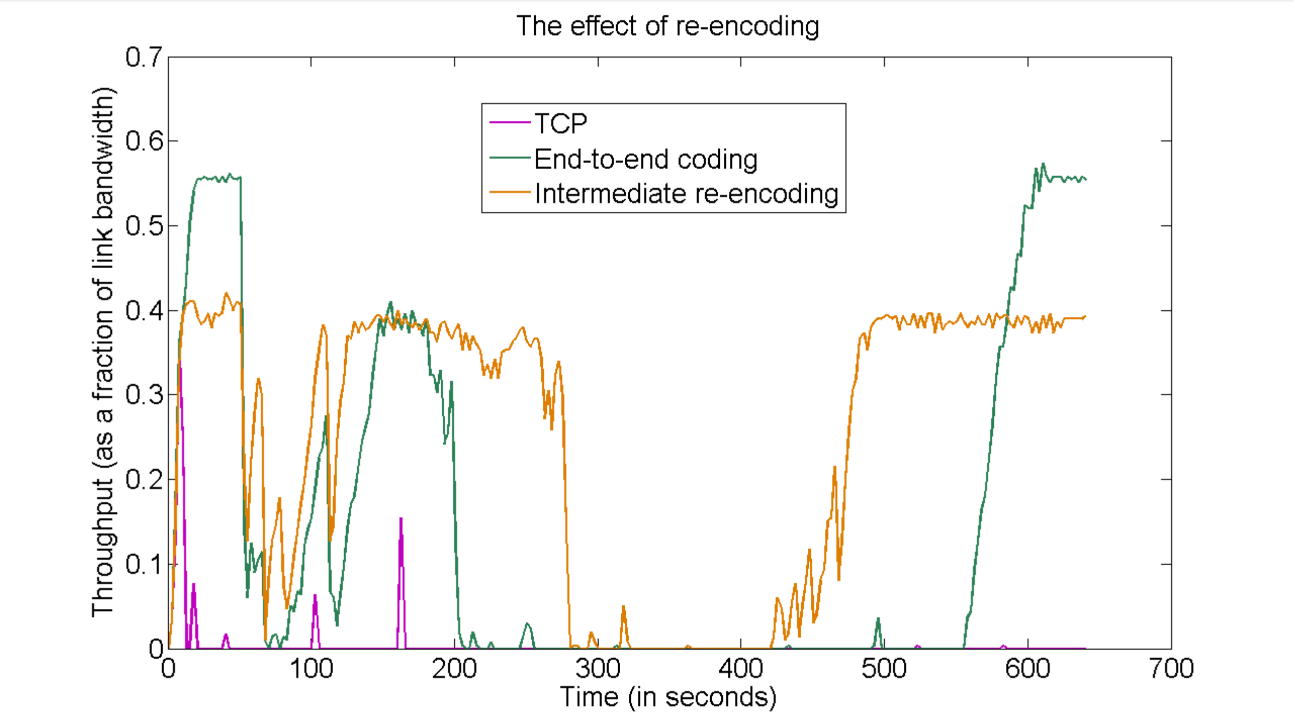
FTP sender

FTP receiver



From given data

Performance Comparison

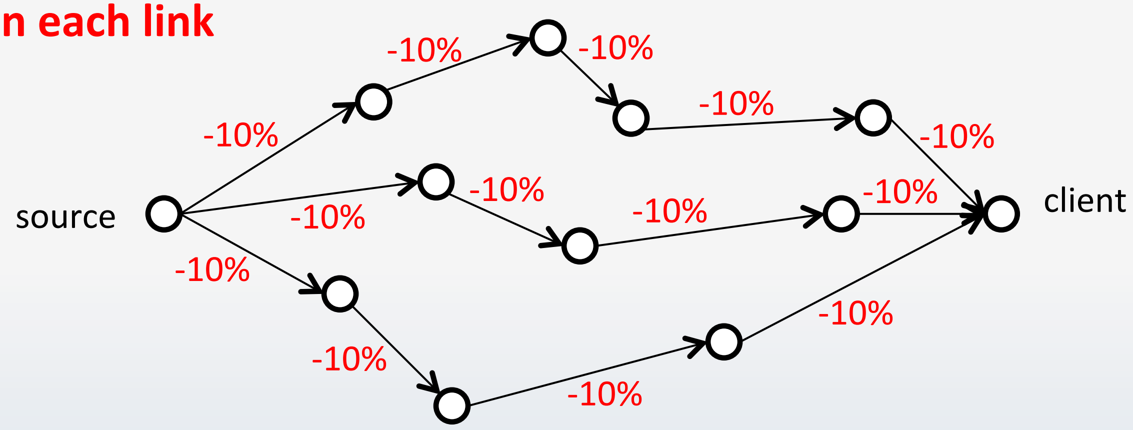


TCP	End-to-end coding	Re-encoding at node 3 only
0.0042 Mbps	0.1420 Mbps	0.2448 Mbps

Time average throughput (over 641 seconds)
(assuming each link has a bandwidth of 1 Mbps in the absence of erasures)

CTCP versus Multipath TCP Using Routing

10% loss on each link



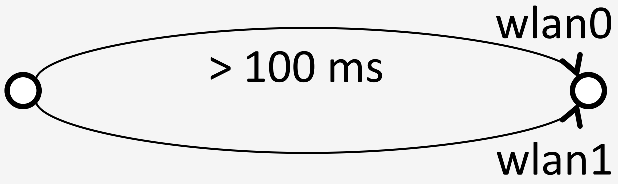
MPTCP may potentially provide multi-path communication BUT:

- Difficult and complex scheduling at the source needed
- Round-robin scheduling is inefficient

CTCP provides multi-path communication throughput of 2.7 without complex scheduling at the source

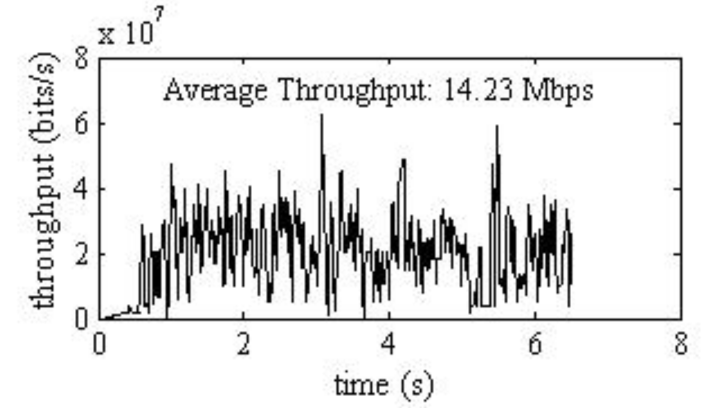
Experimental Results: Multiple Path

- Server: Amazon EC2 instance in CA
- Client: Desktop at RLE MIT, using WiFi(s)
- Limited each path to < 8-9 Mbps

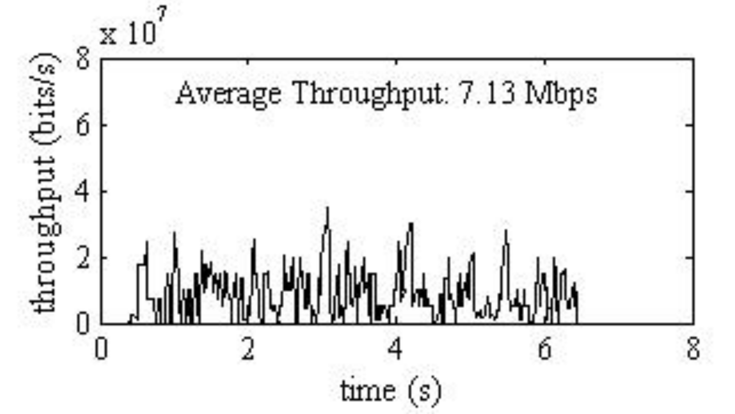


Loss rate	0%	1%	2%	3%	4%	5%
wlan0 (Mbps)	7.13	6.08	6.43	6.25	5.16	5.08
wlan1 (Mbps)	7.11	5.92	6.53	5.90	5.01	4.58
CTCP (Mbps)	14.23	12.03	13.08	11.88	10.10	9.34

Example:
0% loss

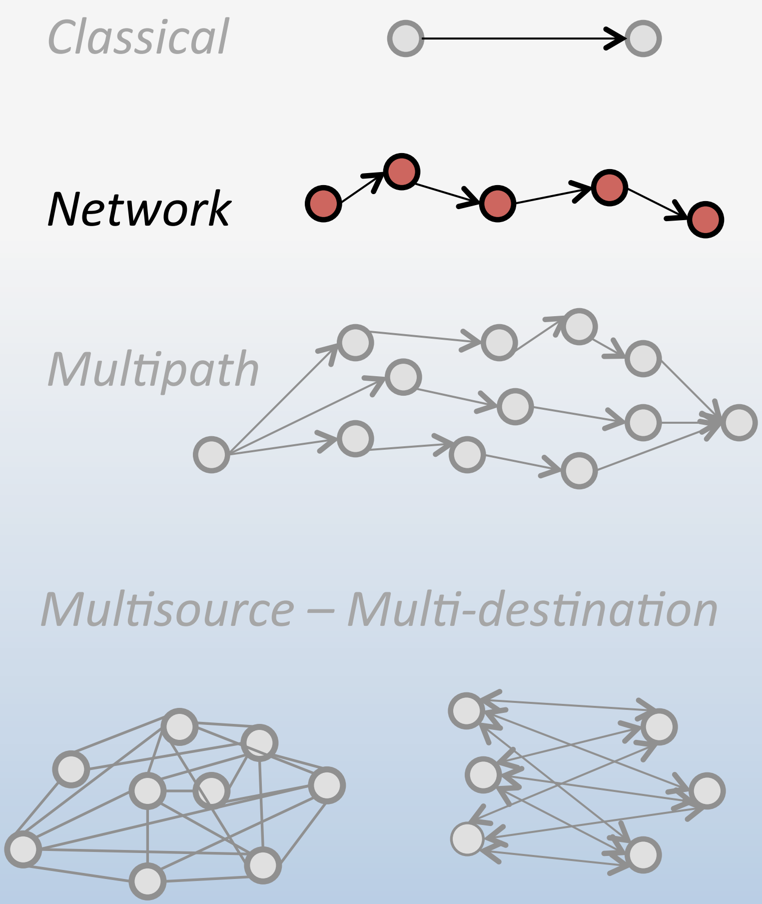


Combined throughput at Client



Throughput on wlan0

Application Layer Base Station Case Study

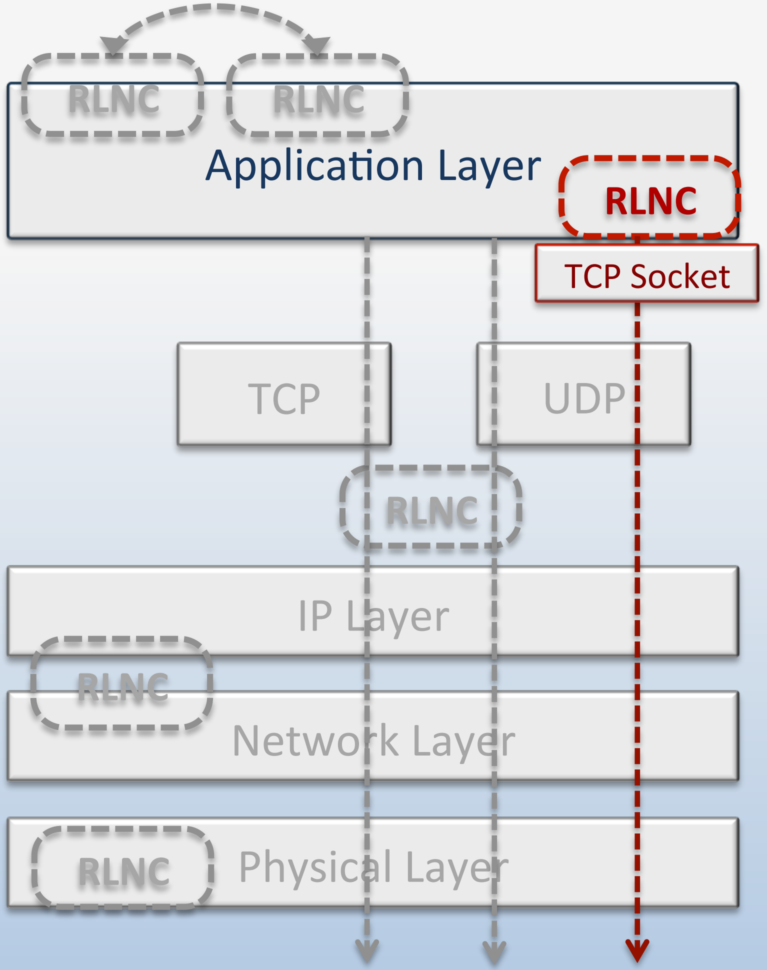


Classical

Network

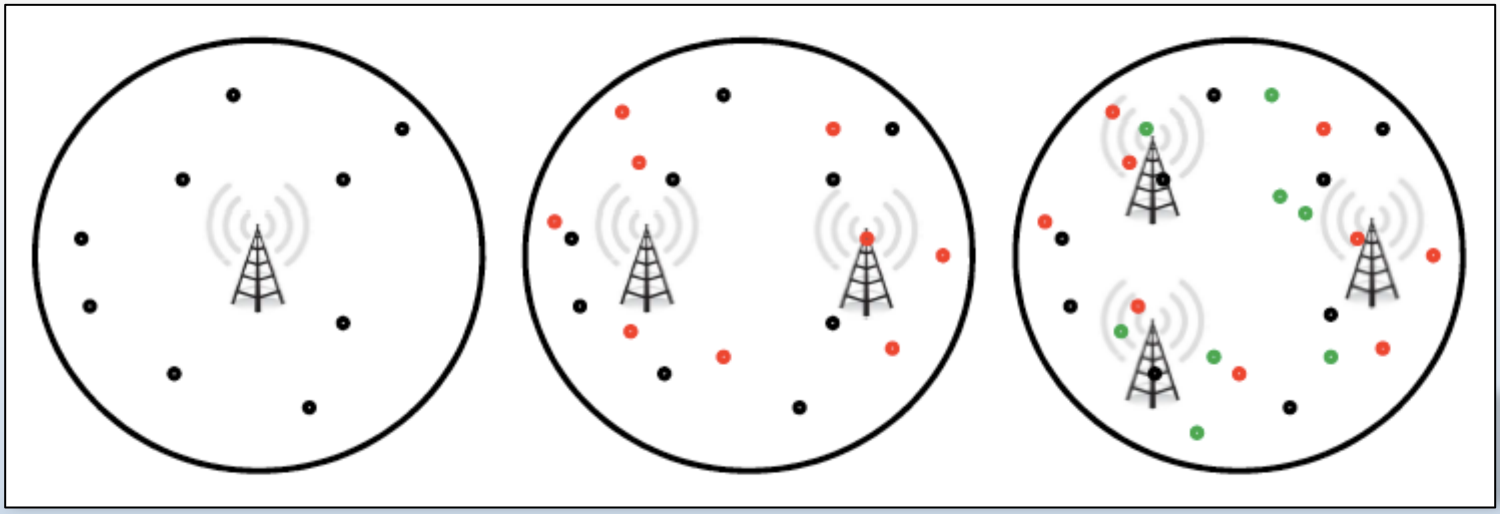
Multipath

Multisource – Multi-destination



Value of Goodput with User Growth

Problem Statement: As # of users grows, additional base stations are added for throughput, not coverage



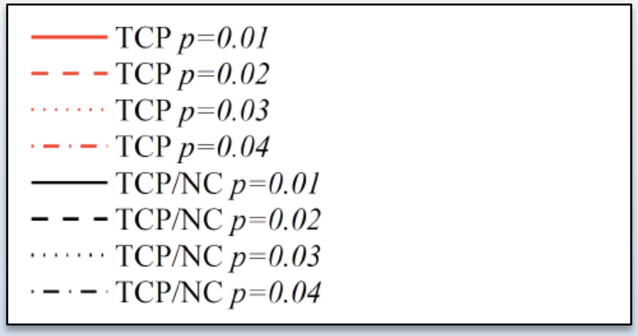
MinJi Kim Thierry Klein Emina Soljanin João Barros Muriel Médard
Modeling Network Coded TCP: Analysis of Throughput and Energy Cost
http://arxiv.org/abs/1208.3212 2012 CoRR abs/1208.3212 db/journals/corr/corr1208.html#abs-1208-3212

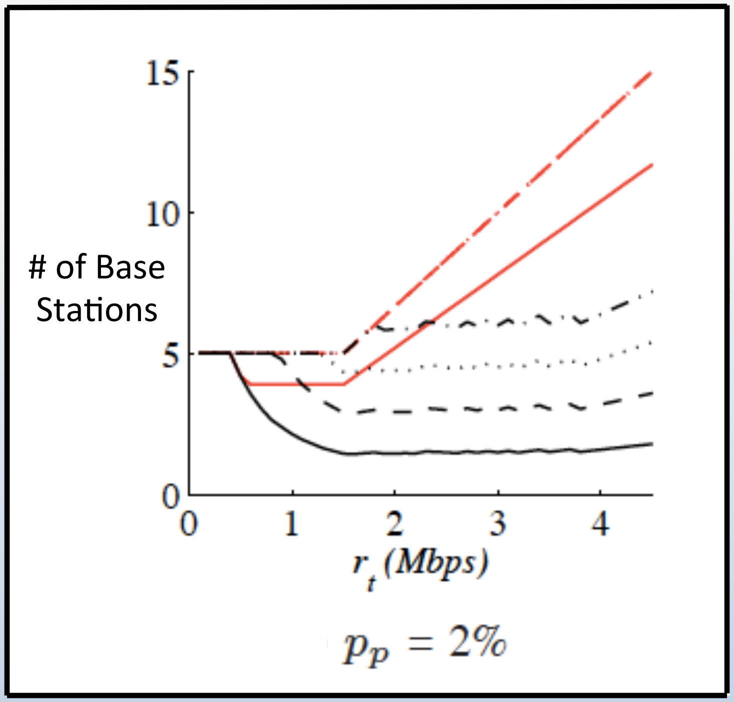
RLNC improved goodput = fewer Base Stations

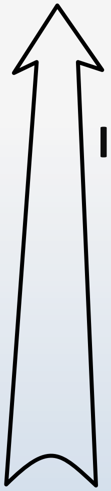
RLNC Expands Base Station Reach

Joint research with MIT and Alcatel-Lucent

5.08 MB



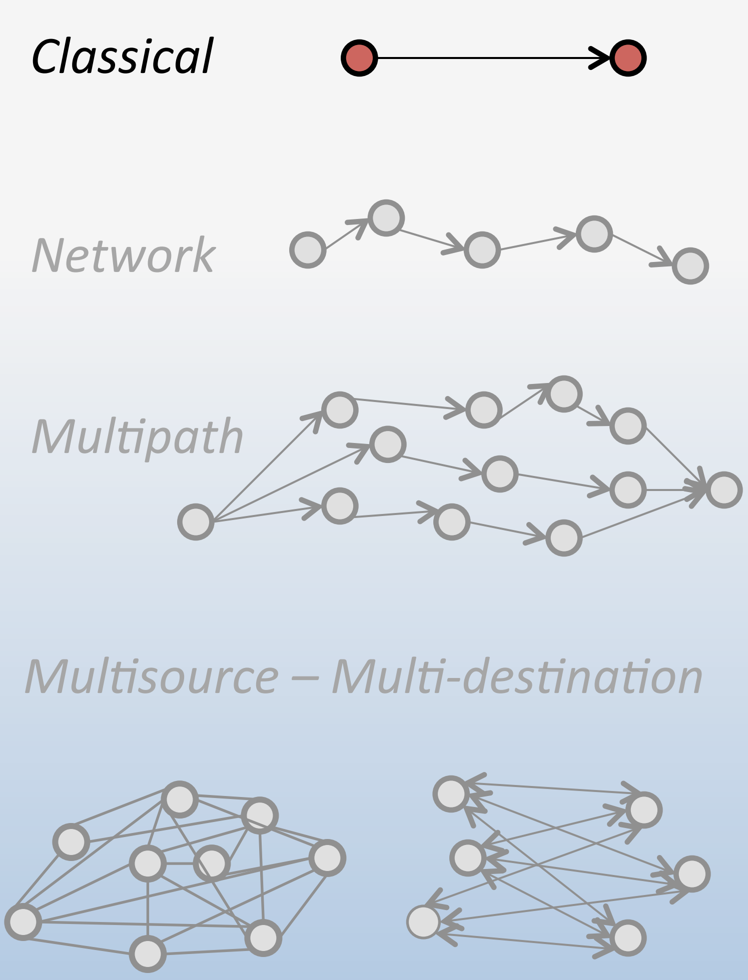




Increasing User Activity

With RLNC operators lower both OpEx and CapEx

Link Layer Implementation Case Study

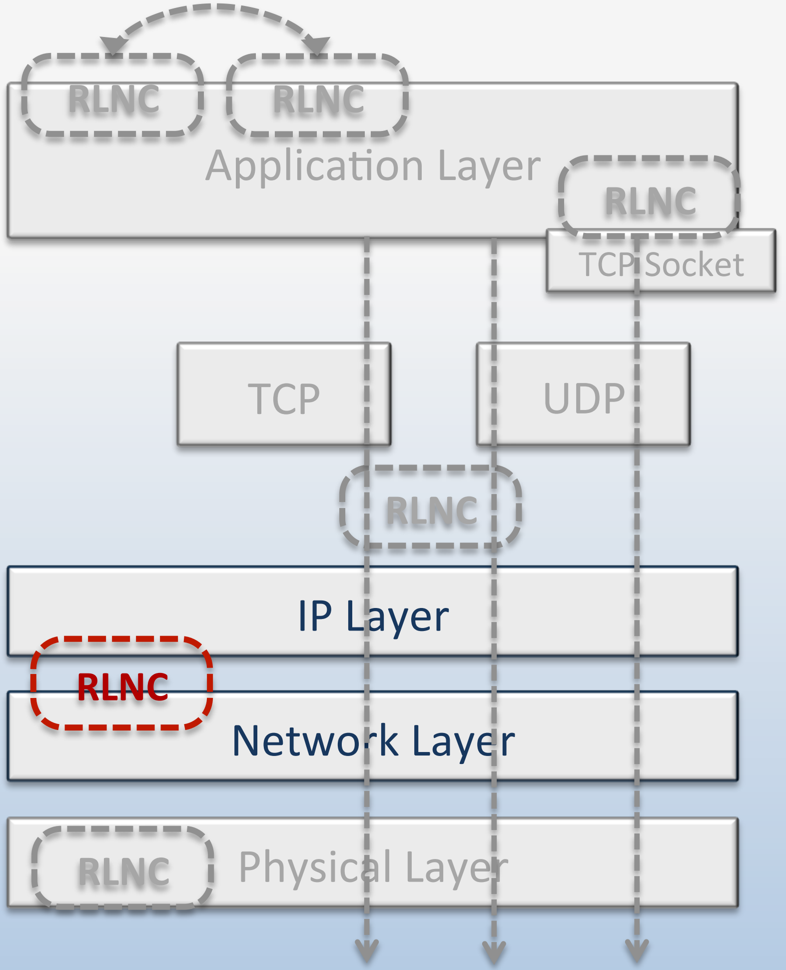


Classical

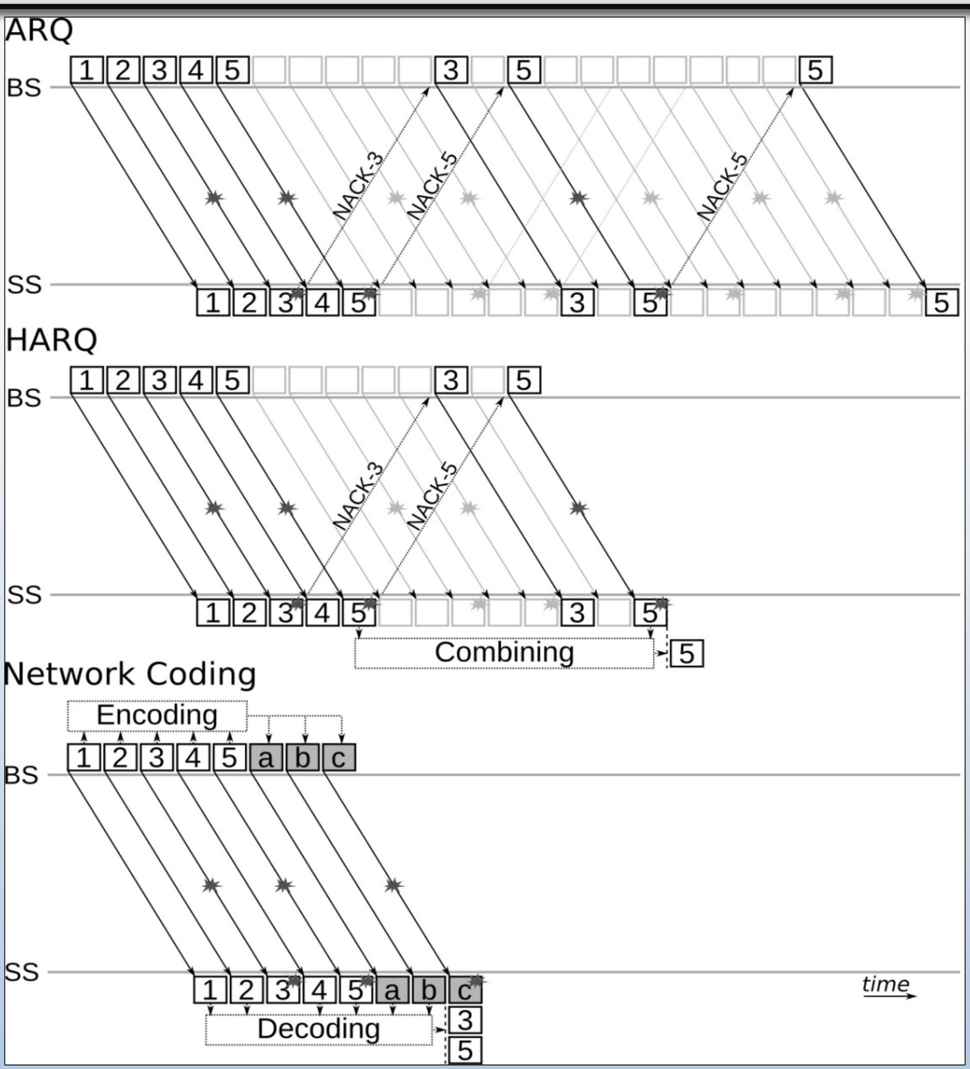
Network

Multipath

Multisource – Multi-destination



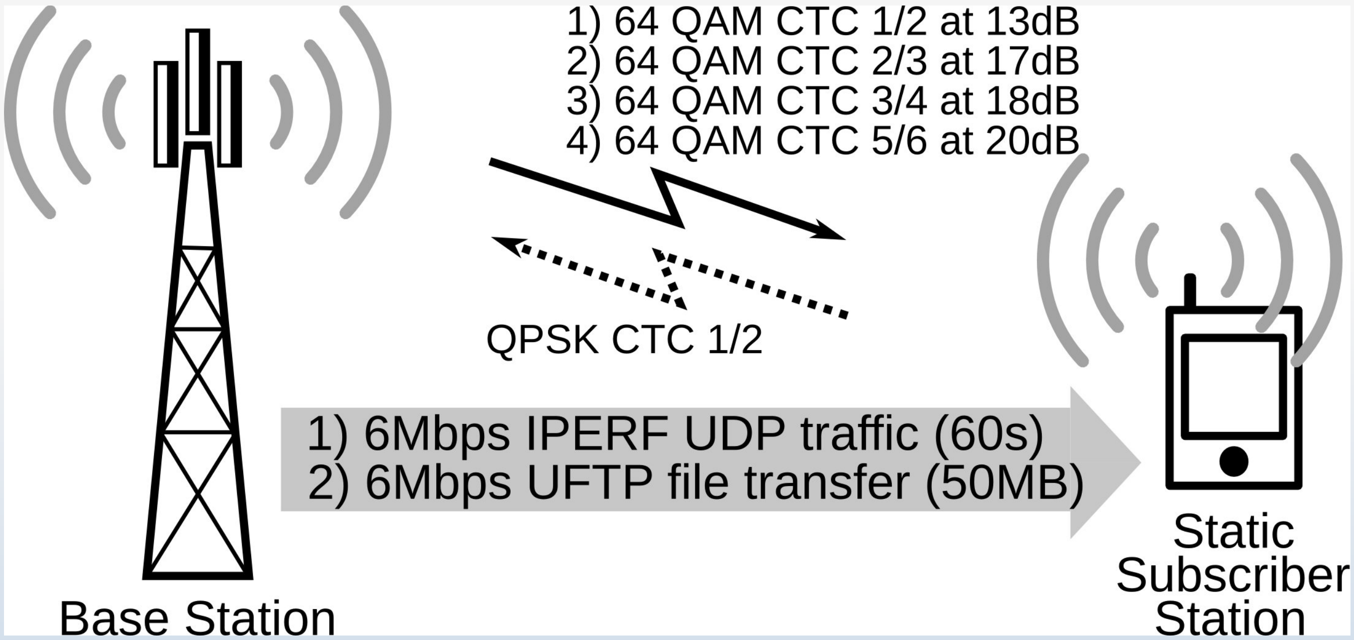
Network Coding vs. ARQ and HARQ



- Scenario: 5-packet block transfer from BS to SS
- Downlink: fixed 40% packet error pattern (every 3rd and 5th packet)
- Uplink: feedback NACKs not subject to loss
- ARQ: repeated transmissions create RTT feedback loops
- HARQ: feedback reduced by combining corrupted packet versions
- Network Coding: a-priori systematic coding with added redundancy of 3/5
- Clear **delay, throughput, and energy gains**
  - No feedback loop
  - Redundancy cost amortized over block

Surat Teerapittayanon, Kerim Fouli, Muriel Médard, Marie-José Montpetit, Xiaomeng Shi, Ivan Seskar, Abhimanyu Gosain
Network Coding as a WIMAX Link Reliability Mechanism
MACOM 2012L 1-12
Surat Teerapittayanon, Kerim Fouli, Muriel Médard, Marie-José Montpetit, Xiaomeng Shi, Ivan Seskar, Abhimanyu Gosain
Network Coding as a WIMAX Link Reliability Mechanism: An Experimental Demonstration
MACOM 2012: 75-78

Experimental Setup



1) 64 QAM CTC 1/2 at 13dB
2) 64 QAM CTC 2/3 at 17dB
3) 64 QAM CTC 3/4 at 18dB
4) 64 QAM CTC 5/6 at 20dB

QPSK CTC 1/2

1) 6Mbps IPERF UDP traffic (60s)
2) 6Mbps UFTP file transfer (50MB)

Base Station

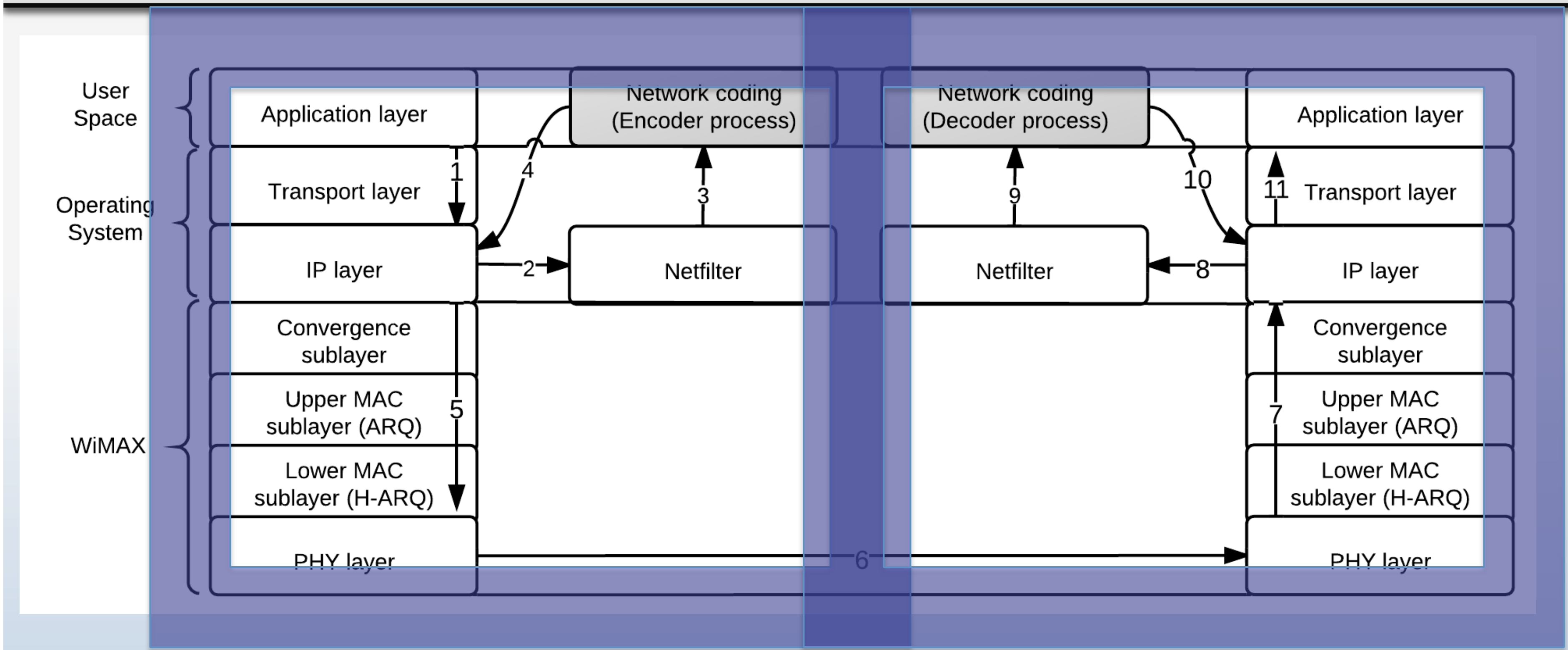
Static Subscriber Station

- Intra-flow NC modules at the Base Station (BS) and Subscriber Station (SS)
- Toggle ARQ, HARQ, and various NC configurations
- IPERF → application-layer throughput / loss
- UFTP (FTP over UDP) → application-layer file-transfer delay

IP-Based Implementation

Remote access of eNode B

Card driver



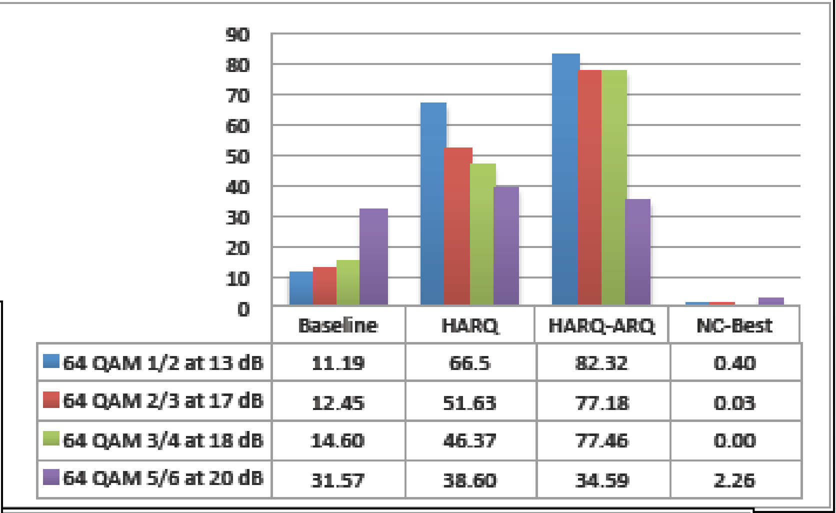
- WiMAX MAC inaccessible → IP-based implementation
- Performance measurements at the application layer (IPERF and UFTP)
- IP layer: *Netfilter* used to intercept packets, route them to encoder/decoder, then re-inject them to IP layer
- PDCP does not need to be involved, although that may be quite doable
- Network coding included at the e-Node B before handing it to the MAC, and ARQ and HARQ bypassed at the MAC – does not require a proxy, but can be used if convenient. Occurs below transport, so not touching TCP per se

codeon

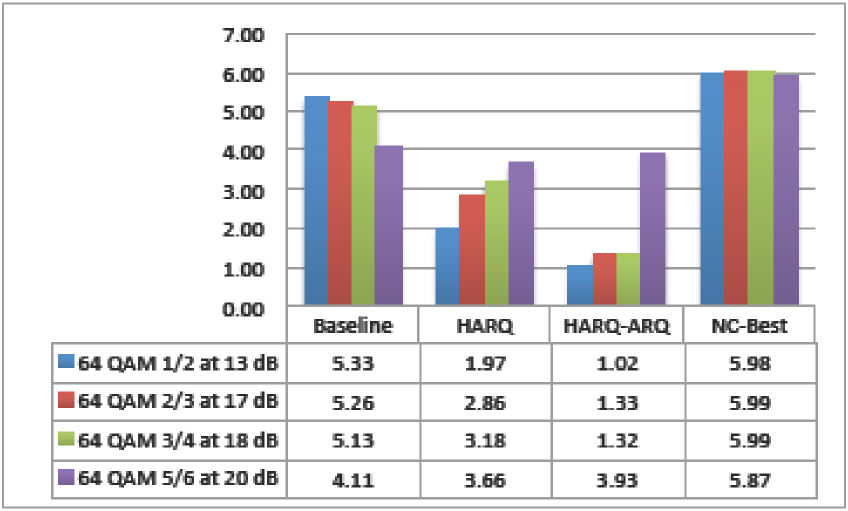
Consistency

- *NC-Best* decreases packet loss **from 11-32% to nearly 0%**
- NC offers up to **5.9x gain in throughput** and **5.5x reduction in file transfer delay**

Loss

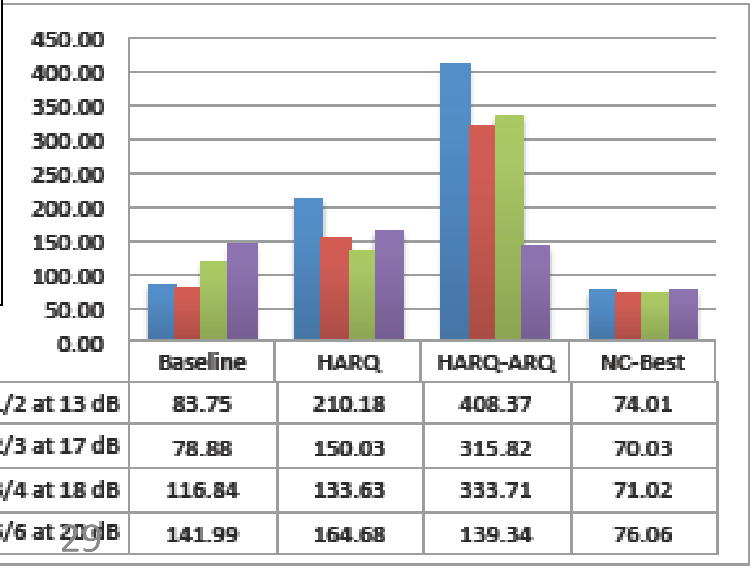


	Baseline	HARQ	HARQ-ARQ	NC-Best
🟦 64 QAM 1/2 at 13 dB	11.19	66.5	82.32	0.40
🟥 64 QAM 2/3 at 17 dB	12.45	51.63	77.18	0.03
🟩 64 QAM 3/4 at 18 dB	14.60	46.37	77.46	0.00
🟪 64 QAM 5/6 at 20 dB	31.57	38.60	34.59	2.26



	Baseline	HARQ	HARQ-ARQ	NC-Best
🟦 64 QAM 1/2 at 13 dB	5.33	1.97	1.02	5.98
🟥 64 QAM 2/3 at 17 dB	5.26	2.86	1.33	5.99
🟩 64 QAM 3/4 at 18 dB	5.13	3.18	1.32	5.99
🟪 64 QAM 5/6 at 20 dB	4.11	3.66	3.93	5.87

Throughput



	Baseline	HARQ	HARQ-ARQ	NC-Best
🟦 64 QAM 1/2 at 13 dB	83.75	210.18	408.37	74.01
🟥 64 QAM 2/3 at 17 dB	78.88	150.03	315.82	70.03
🟩 64 QAM 3/4 at 18 dB	116.84	133.63	333.71	71.02
🟪 64 QAM 5/6 at 20 dB	141.99	164.68	139.34	76.06

File transfer delay

codeon

Looking Forward

- The applications of network coding are varied and changing
- This is indeed the right time for a standardization effort
- Network coding can be implemented at different layers and in different portions of the network
- Standardization should allow the flexibility needed to allow these diverse implementations