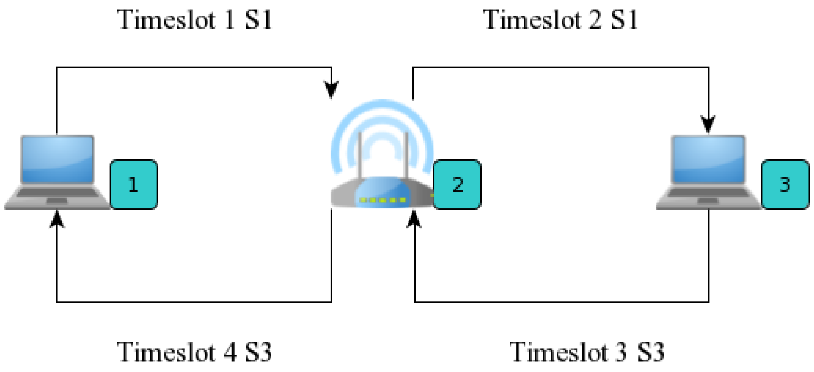
Network Coding

Carleton
UNIVERSITY
Canada's Capital University

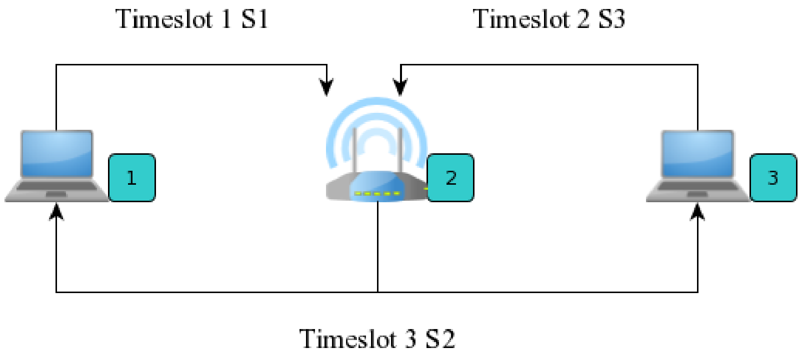
Key Idea: Move Beyond Store-and-Forward

- Originally: communication networks were circuit-switched
- Internet (1960s): break the circuit, develop network based on the idea of packet-switching
  - Increased robustness
  - Better use of resources (multiplex gains) as circuits are not always used 100%
- Network coding: get away from packets, send information “about” the packets
  - Receiver can still receive original data
  - Allows for a range of (potential) improvements: increased throughput, increased robustness, new applications, reduced energy consumption
  - May come with some disadvantages:
    - Security
    - QoS

Simple Network Coding Example: XOR



Packet Forwarding: 4 transmissions to propagate one packet from 1 and 3 to all nodes



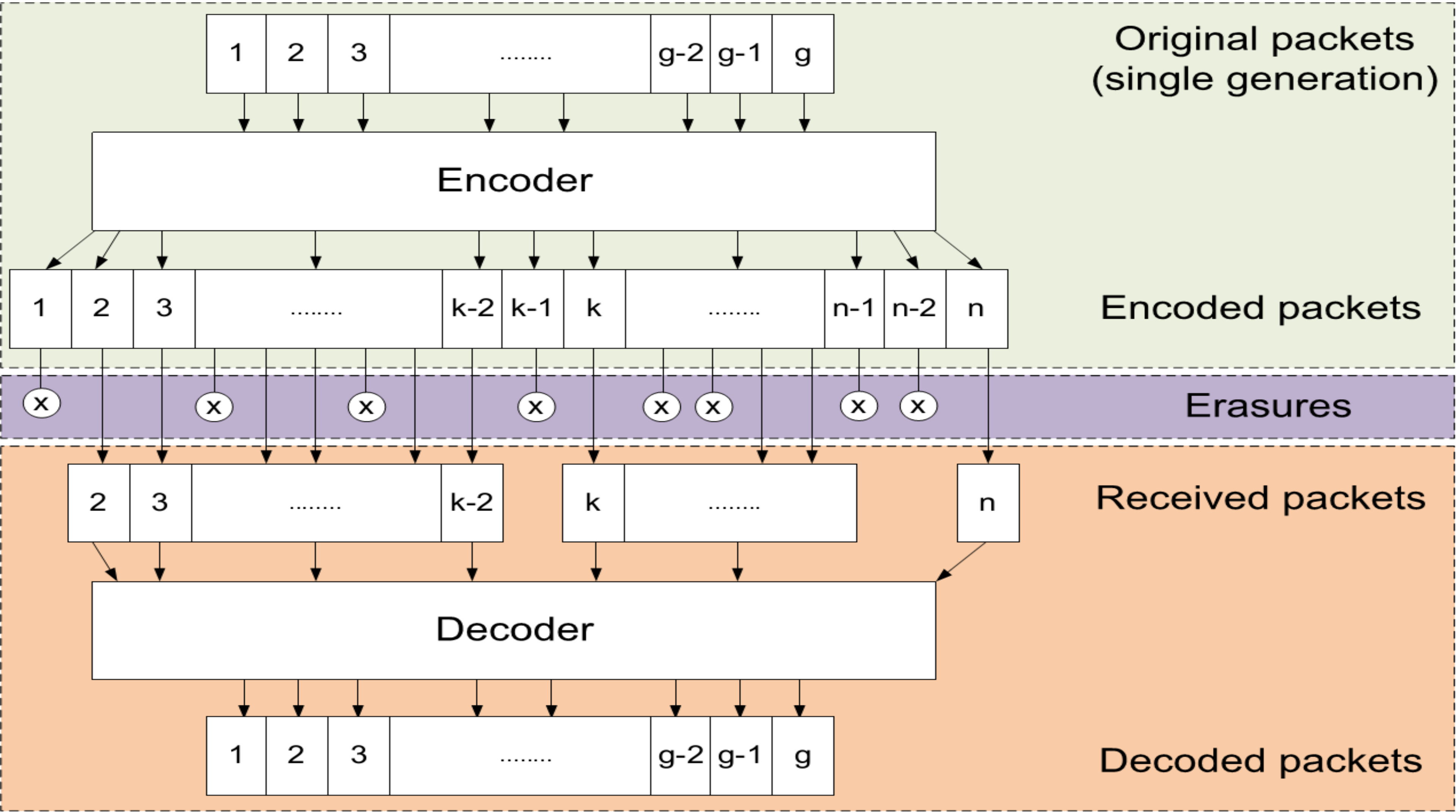
$$S_2 = S_1 \oplus S_3$$

$$S_1 \oplus S_2 = S_1 \oplus (S_1 \oplus S_3) = S_3$$

$$S_3 \oplus S_2 = S_3 \oplus (S_1 \oplus S_3) = S_1$$

Network Coding: 3 transmissions to propagate one packet from 1 and 3 to all nodes
(Node 2 combines S1 and S3 via XOR)

More General Network Coding Framework



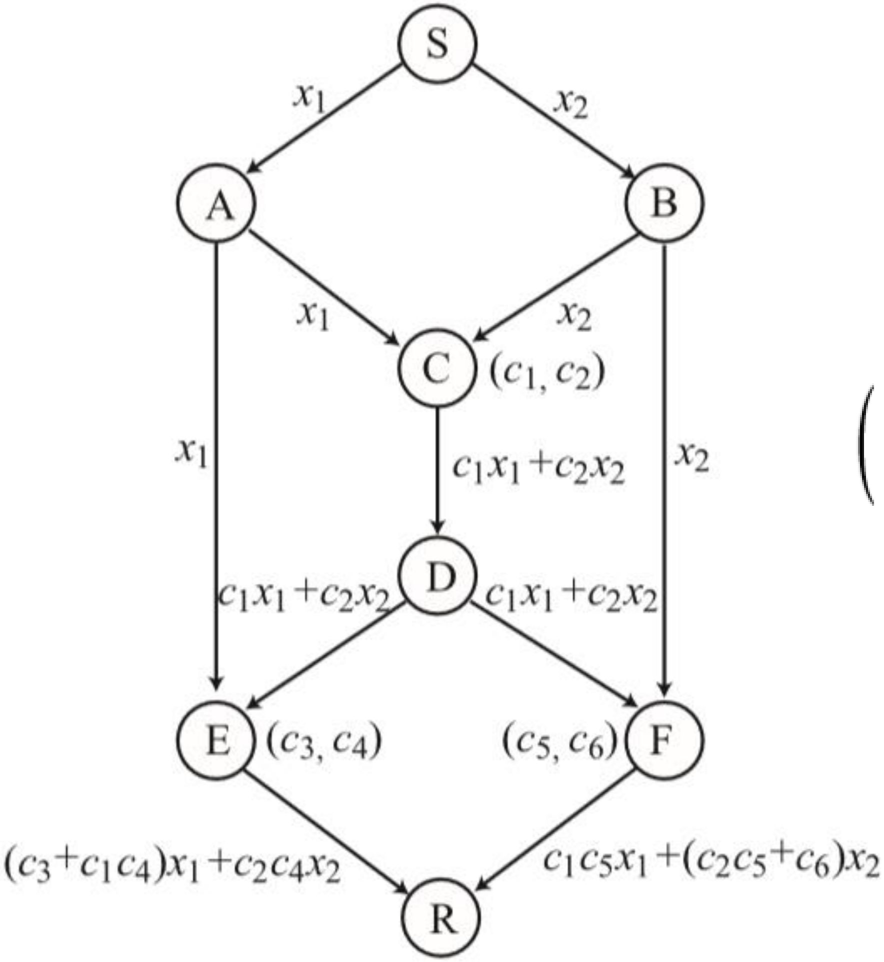
Survey of Network Coding and its Applications

- Survey paper with lots of interesting thoughts on possible applications
- Key Sections:
    - Code Design
    - Throughput/Capacity Enhancement Techniques
    - Robustness Enhancements
    - Network Tomography
    - Security

Network Coding: Code Design

- Given a network topology and a scenario (number of sources, number of receivers, do all sources send the same info, do all receivers want the same info), design a code that optimizes operation:
  - Single source, multiple receivers: codes can achieve max capacity (min-cut max-flow problem)
  - Less clear for other scenarios
- Coding gains (improvement of coding over packet forwarding) depend on
  - Network topology
  - Are links directed/undirected
- For many interesting scenarios:
  - Linear network coding sufficient
  - Implementation: random linear network coding

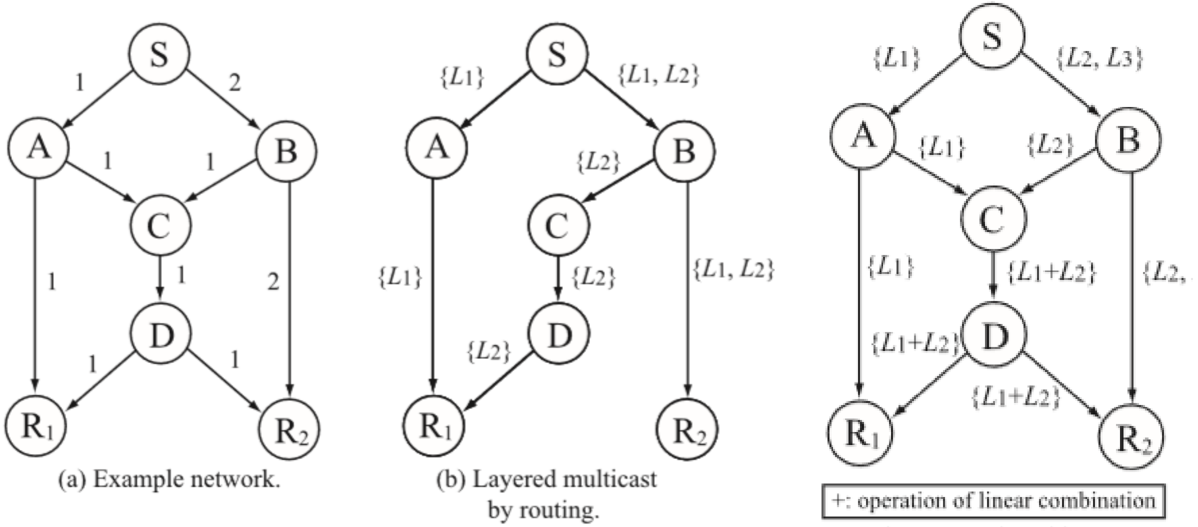
Linear Network Coding Example



$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} c_3 + c_1c_4 & c_2c_4 \\ c_1c_5 & c_2c_5 + c_6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Throughput/Capacity Enhancement Techniques

- Original motivation for network coding
- Examples in survey paper: disk storage, content distribution, layered multicast
- Usually coding is better (or at least not worse than) routing: coding gain



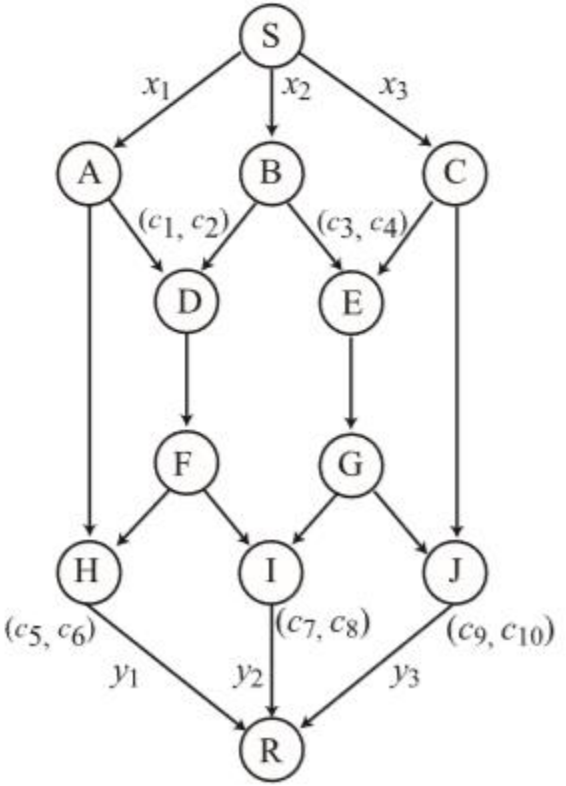
(a) Example network.

(b) Layered multicast by routing.

(d) Layered multicast with inter-layer network coding.

Robustness Enhancement

- Network Coding can play role similar to Forward Error Correction
  - Add redundancy, allows for packet losses to be recovered
  - Different from FEC: can add redundancy in the middle of the network (where links are known to be lossy), not only end-to-end



No packets lost: decoding matrix has full rank

$$C = \begin{pmatrix} c_5 + c_1c_6 & c_2c_6 & 0 \\ c_1c_7 & c_2c_7 + c_3c_8 & c_4c_8 \\ 0 & c_3c_9 & c_4c_9 + c_{10} \end{pmatrix}.$$

Packet loss on link F-I: decoding matrix still full rank

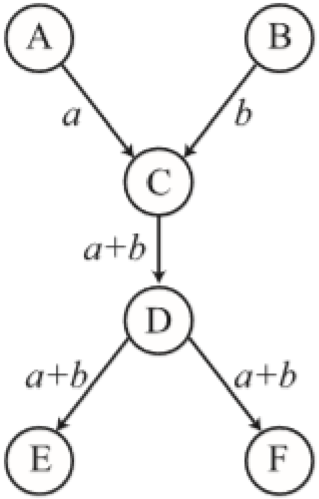
$$C = \begin{pmatrix} c_5 + c_1c_6 & c_2c_6 & 0 \\ 0 & c_3c_8 & c_4c_8 \\ 0 & c_3c_9 & c_4c_9 + c_{10} \end{pmatrix}.$$

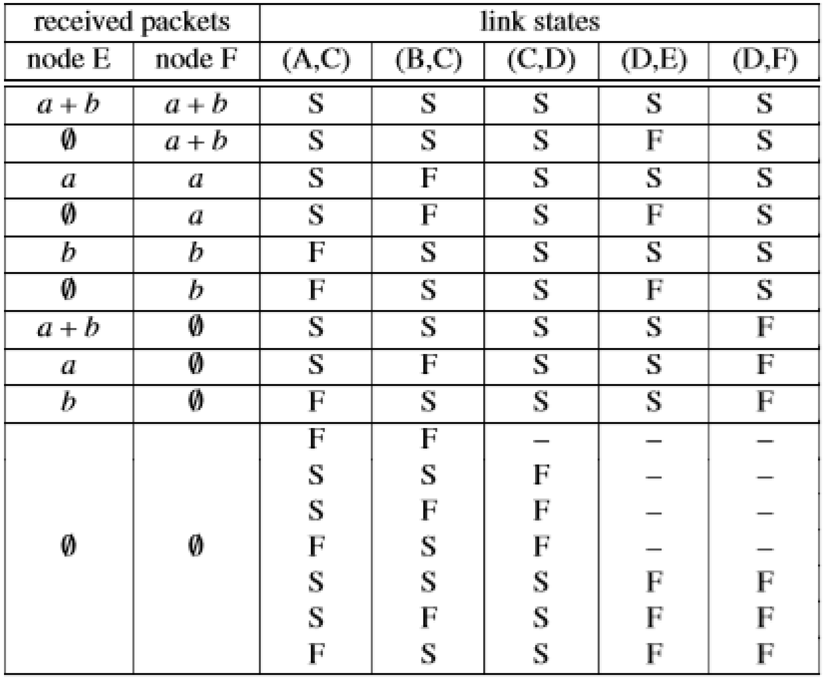
Packets lost on A-H and F-H: decoding matrix not fill rank

$$C = \begin{pmatrix} 0 & 0 & 0 \\ c_1c_7 & c_2c_7 + c_3c_8 & c_4c_8 \\ 0 & c_3c_9 & c_4c_9 + c_{10} \end{pmatrix}.$$

Network Tomography

- Idea: generate knowledge about network from packets and coding coefficients received
  - Deduce link loss rates: monitor what oackets are received over time (including their coding coefficients) and deduce link failure rates
  - Deduce network topology (more complex)





received packets		link states				
node E	node F	(A,C)	(B,C)	(C,D)	(D,E)	(D,F)
$a+b$	$a+b$	S	S	S	S	S
$\emptyset$	$a+b$	S	S	S	F	S
a	a	S	F	S	S	S
$\emptyset$	a	S	F	S	F	S
b	b	F	S	S	S	S
$\emptyset$	b	F	S	S	F	S
$a+b$	$\emptyset$	S	S	S	S	F
a	$\emptyset$	S	F	S	S	F
b	$\emptyset$	F	S	S	S	F
$\emptyset$	$\emptyset$	F	F	–	–	–
		S	S	F	–	–
		S	F	F	–	–
		F	S	F	–	–
		S	S	S	F	F
		S	F	S	F	F
		F	S	S	F	F

Security

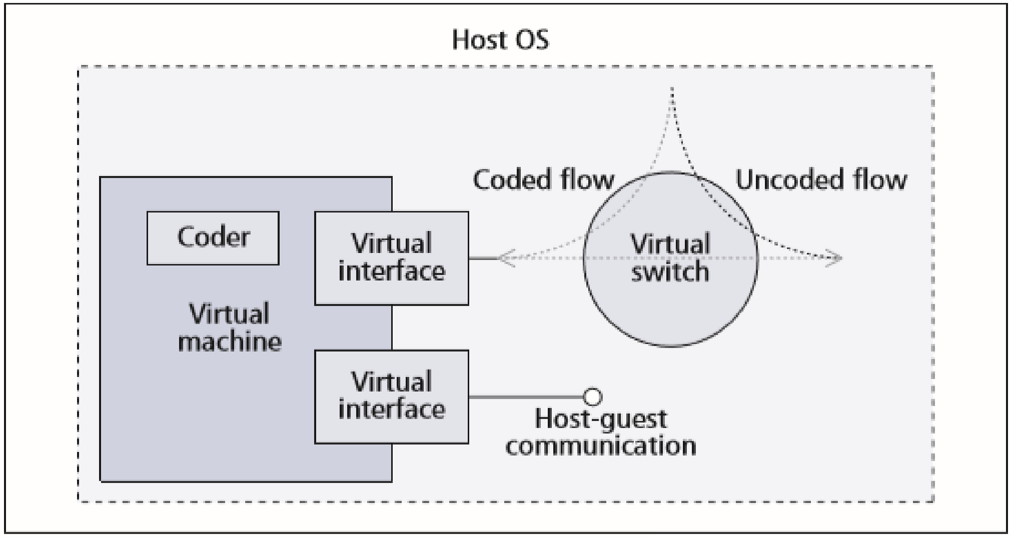
- Problem: if attached injects single (bogus) coded packet into the network, this will pollute all decoded packets
- Defence:
  - Detect packet injection
    - In essence: have packets signed
    - Typically, Message Authentication Codes (MACs) such as MD5 do not work, as they get scrambled by packet encoding
    - Special type of hash functions: homomorphic hash functions
      - Still valid signatures even after linear packet combinations
  - Correct for packet injections
    - In essence: apply FEC on native packets, allows do deal with a certain number of corrupt packets
- One good property of network coding: can code packets at source, eavesdropper in network never sees packets “in the clear”
  - Though if close to receiver, may receive enough coded packets to decode

One Application of Network Coding: SDN

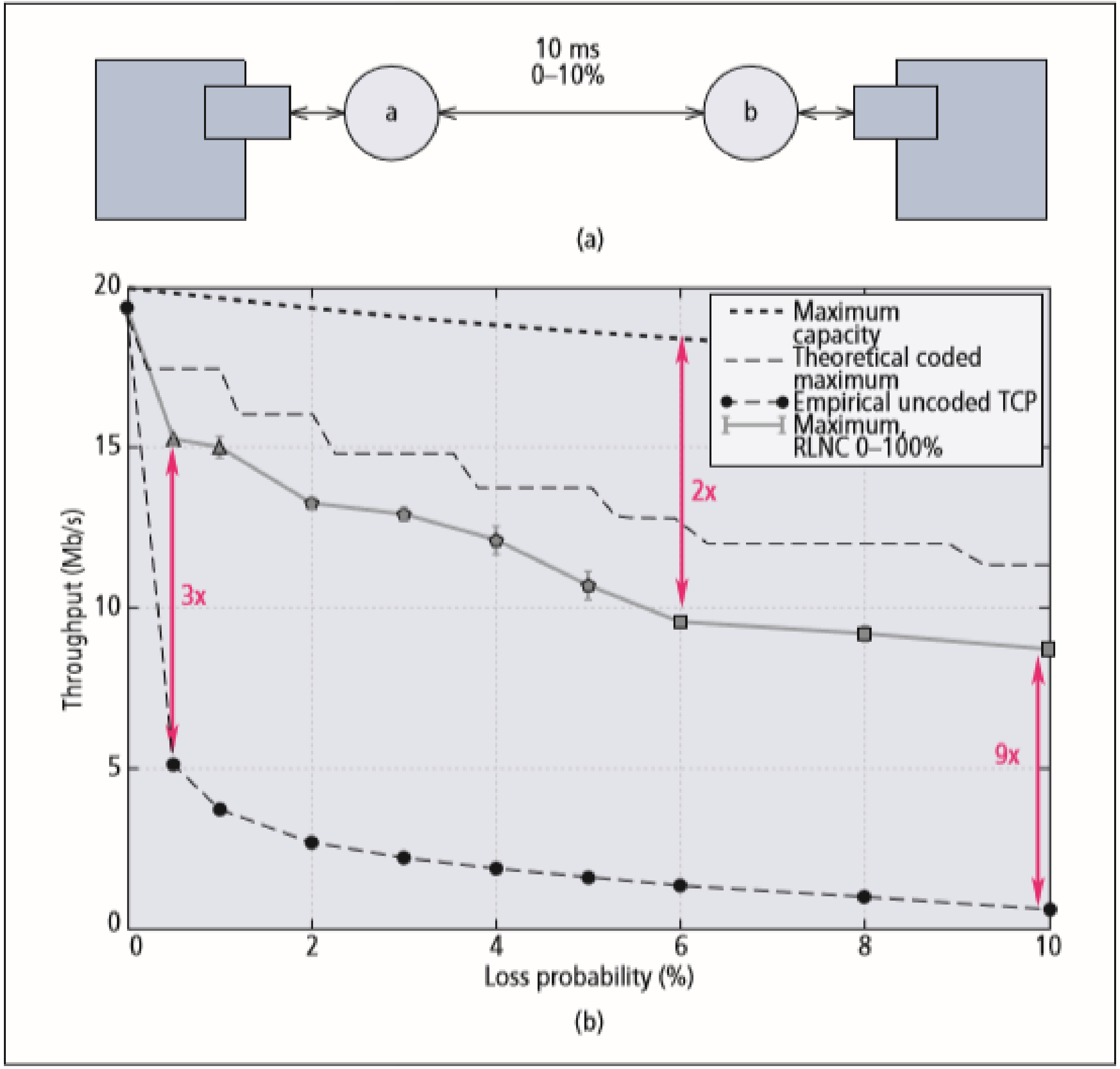
- Network Coded Software Defined Networking: Enabling 5G Transmission and Storage Networks
  - Published September 2015
  - Idea: 5G networks will be fundamentally different from current networks, to support more devices, more services
    - One avenue: SDN and NFV
    - Their claim: add network coding
  - SDN: central controller manages resources, allows to make flexible resource allocations
  - Network Coding: more efficient use of resources, but we need to know when and where to add coding to the network
  - ➔ SDN controller controls how network coding gets applied to data streams
- Examples in paper: all essentially about robustness
  - TCP suffers from packet loss (see 2nd seminar)
  - Network coding can provide increased robustness (i.e., prevent such losses)

TCP Performance Improvements

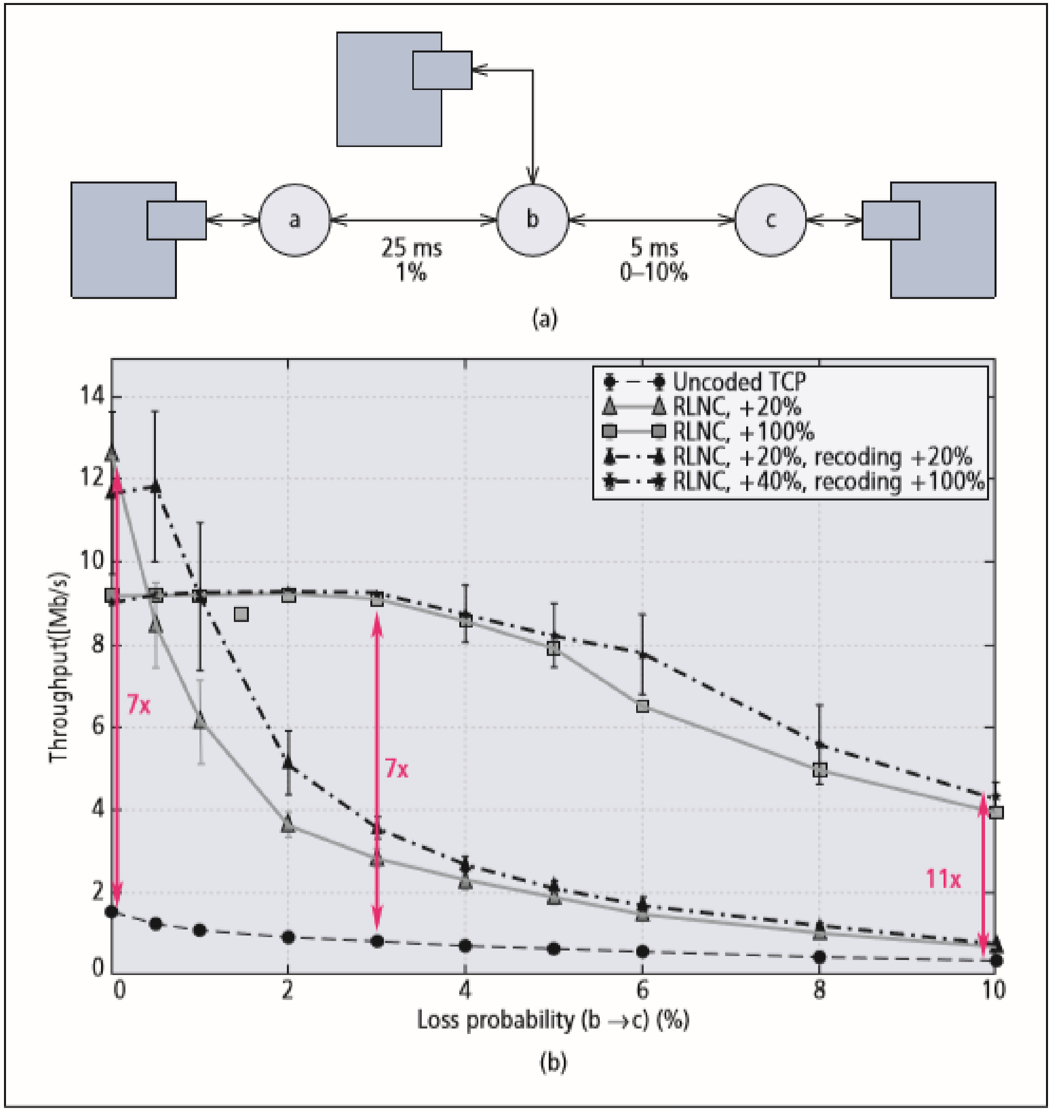
- Set of experiments to show improvements in TCP throughput
  - Single hop
  - Multi-hop
  - Multi-hop and multi-flow
- Common part of the experimental setup:
  - one ore more SDN-capable forwarding devices (Open vSwitch) paired with a network coder, implemented within a VM
  - SDN controller decides whether and how to code based on link characteristics



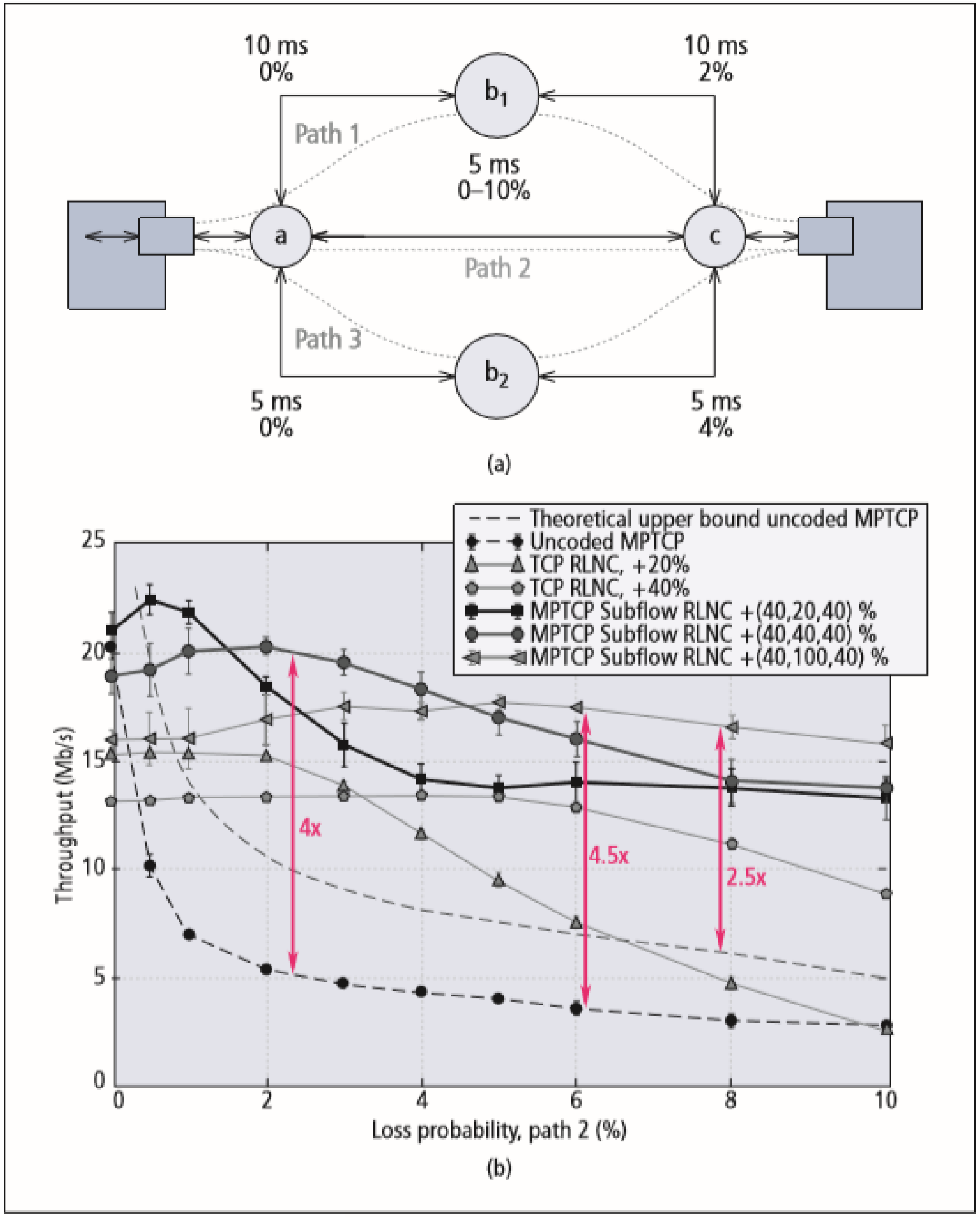
Single-Hop TCP



Multi-Hop TCP



Multi-Hop
Multi-Flow TCP



Own Work: Efficient Broadcasts in Wireless Networks

(part of this I presented last year during my visit here)

Carleton University

Canada's Capital University

Motivation

- Broadcasting: one (or multiple) sources send information to all nodes in a network
  - Extreme case of multicast, can be used to implement multicast as well
- Used for:
  - Control information propagation
    - Link state updates in routing protocols
  - Applications
    - All-informed updates (military, first responders)
- Multihop wireless networks:
  - Topology changes dynamically
  - Bandwidth limited → Flooding is not very attractive
- Goal: broadcast data to all nodes with MINIMUM number of packet transmissions at the MAC/PHY layer

Motivation

- Two steps to a complete solution
  - What is minimum number of packet transmissions required
  - What (distributed) protocols come close to achieving this optimal value
- "Traditionally", efficient broadcast protocols based on routing/packet forwarding
  - Lots of proposals, research for 15+ years in the context of multihop wireless networks
  - Key challenge: determine which nodes get to retransmit a packet (flooding: all nodes retransmit → high costs, many redundant packet transmissions)
  - Assure high PDR (Packet Delivery Ratio) even in the face of topology changes
  - IETF standardizing SMF (Simplified Multicast Forwarding) as efficient broadcast protocol: RFC 6621, May 2012
- Network Coding shown to increase throughput for multicast, would NC result in an efficient broadcast protocol (better than SMF)?

Lower Bounds: Network Coding

- Lower Bound can be formulated as a integer linear optimization problem:

$$\min \sum X_i$$

Subject to:

$$\forall i, d: \; F_{i,\bar{i}}(d) \leq X_i$$

$$\forall\, i, d: F_{i,\bar{i}}(d) - \sum_{j \in N(i)} F_{\bar{j},i}(d) = \begin{cases} N \; for \; i = 0 \\ -N \; for \; i = d \\ 0 \; otherwise \end{cases}$$

$$\forall\, i, d: \sum_{j \in N(i)} F_{\bar{i},j}(d) - F_{i,\bar{i}}(d) = 0$$

$$\forall\, i: X_i \geq 0, X_i \; is \; integer$$

X_i: Packet transmissions by node i

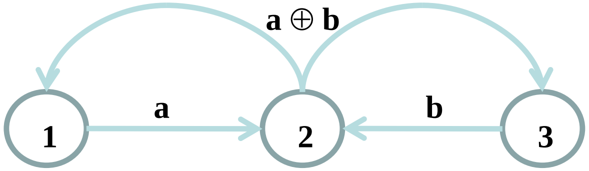
$F_{x,y}$(d): Flow of packets over link x,y destined to d

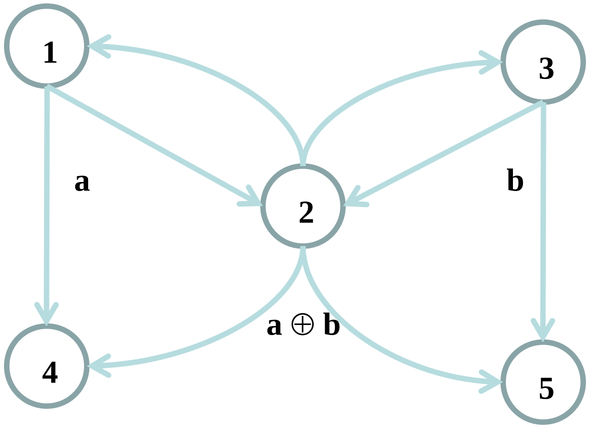
$\bar{i}$: Dummy nodes to model wireless broadcast medium

In ring topology, for example, total costs only half of the packet forwarding (known optimal result)

Lower Bounds: More than One Source Node?

- Forwarding solutions: for each of K sources, use MCDS
  - Total cost is K times cost of single source
- Network Coding: benefits from coding packets belonging to different sources





Lower Bounds: Network Coding with K Sources

- Expand linear program to allow for
  - Multiple sources (each with flows to all destinations)
  - Packets belonging to different sources can be coded together

$$\min \sum X_i$$

M: "meta"-source

Subject to:

$$\forall s, i, d: F^{s}_{i,\bar{i}}(d) \leq X_i$$

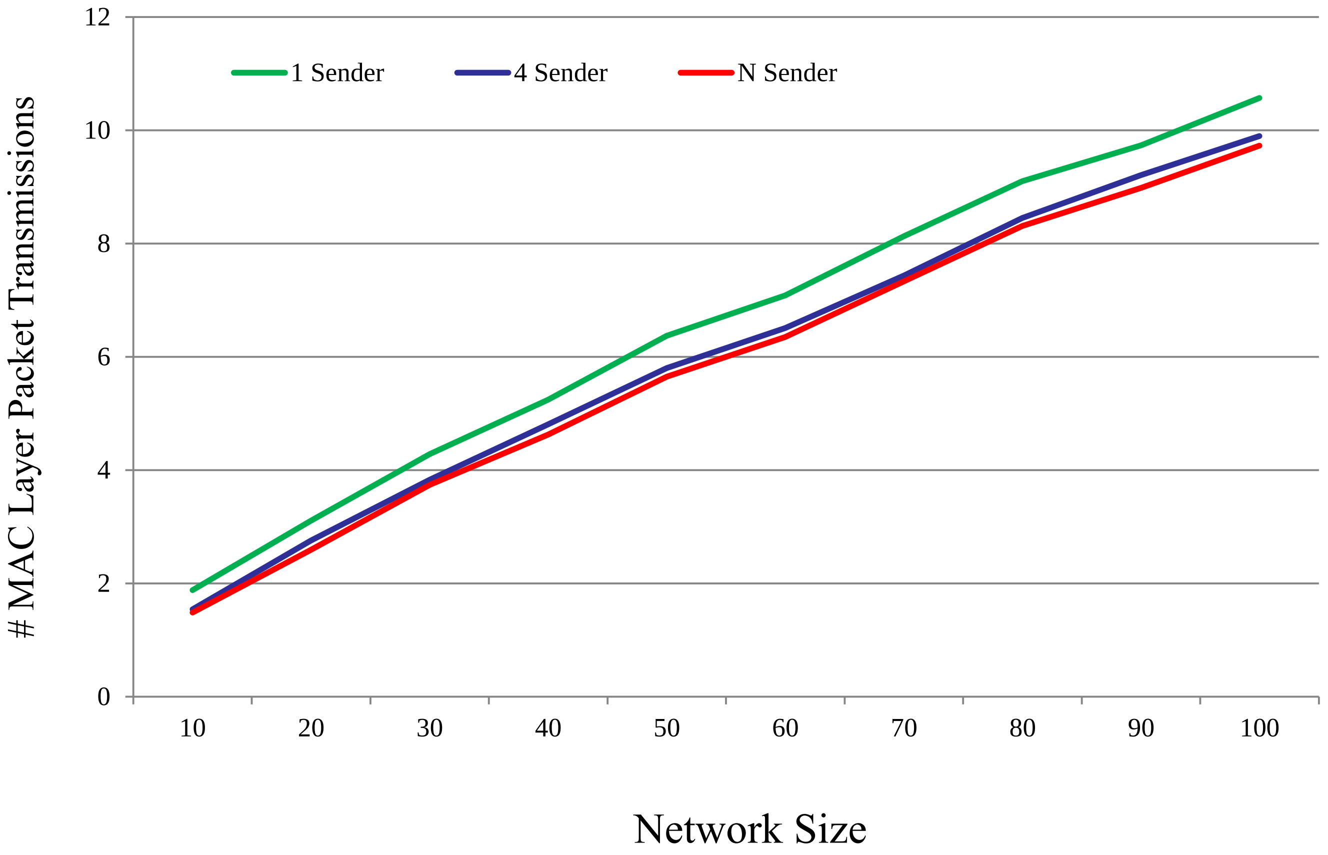
$$\forall\, s, i, d: F^{s}_{i,\bar{i}}(d) - \sum_{\mathrm{j} \in \mathrm{N(i)} \cup \{\mathrm{M}\}} F^{s}_{j,i}(d) = \begin{cases} N \; for \; i = s \\ -N \; for \; i = d \\ 0 \; otherwise \end{cases}$$

$$\forall\, s, i, d: \sum_{j \in N(i)} F^{s}_{i,j}(d) - F^{s}_{i,\bar{i}}(d) = 0$$

$$\forall\, i: X_i \geq 0, X_i \; is \; integer$$

$$\forall\, s, d: F^{s}_{M,s}(d) = N$$

Lower Bounds: Network Coding with K Sources



Random Linear Network Coding

- Source/Intermediate nodes linearly combine packets
  - Coefficients randomly chosen from a field
  - Combined packet(s) are rebroadcast
  - One challenge/question: how many packets does a node need to rebroadcast
- Receiver has to decode packets, solving a system of linear equations
  - Math is typically expressed in terms of matrix operations
  - To decode n native packets, need at least n coded packets
  - As coefficients are chosen randomly, whp n is sufficient
- To control matrix size/coding complexity/memory requirements, only packets belonging to the same generation can be coded together
  - Typical values are 4 or 8
  - Generation size impacts coding efficiency (larger is usually better) and coding latency (larger is usually worse)

ARLNCCF Features

- Supports coding packets from different sources
  - May increase generation size, as sources independently add packets into a generation
- Generation size controlled by “Generation Distance”
  - In essence, use generation that is either created locally or, failing that, close by, avoid using generations that originated from a node more then a threshold distance (in hop counts) away from current source
- Controls broadcast rate based on network density
  - Need to ensure that, collectively, a node receives N packets (where N is generation size), through broadcasts from ALL its neighbors

$$N_T(i) = \left\lceil \textit{generation size} / \textit{Min}\ (NrN_n(m), \textit{for all } n) \right\rceil$$

- Generation Timeout: set dynamically based on data rate

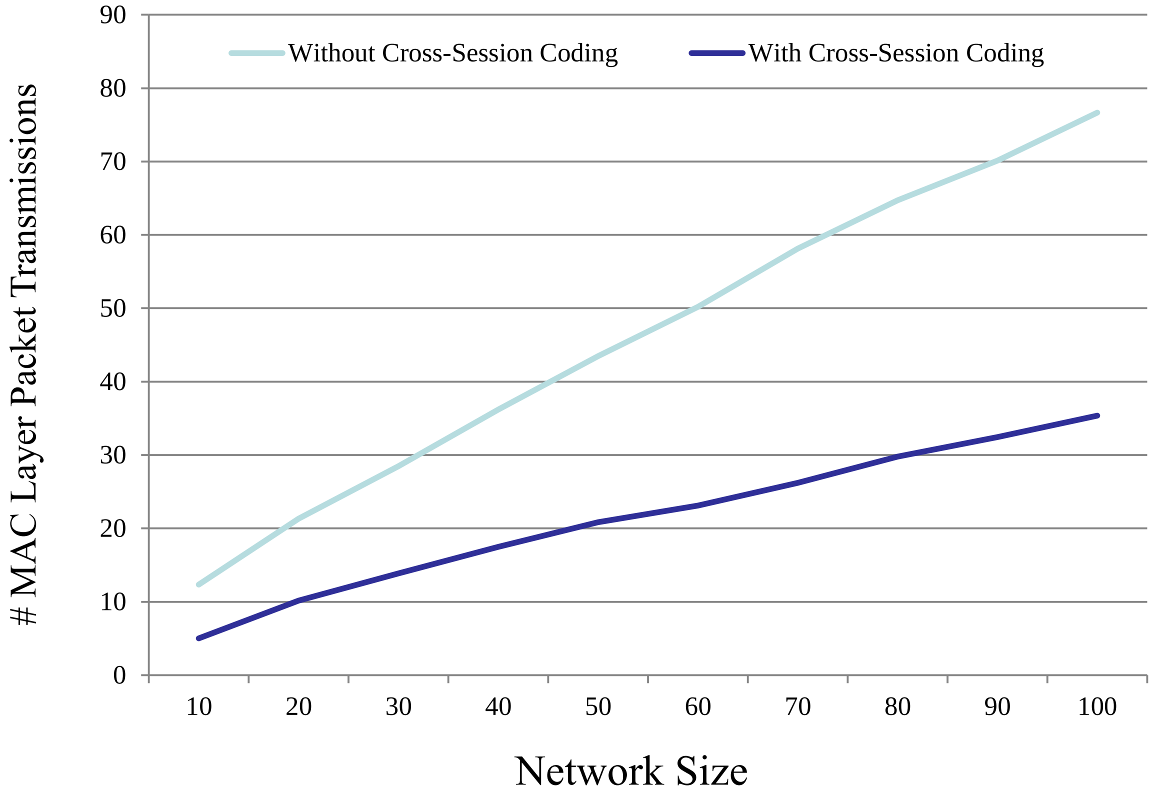
$$T = \textit{Generation Size} / \textit{Data rate (packets per second)}$$

- Supports early decoding

ARLNCCF Evaluation

- Implemented in NS2
- Range of scenarios:
  - Single source
  - Multiple sources
    - Each source generates just one packet
    - Each source generates a full generation of data packets
- Range of metrics:
  - PDR
  - Number of Generations
  - Size of Generations
- Ensure close to 100% PDR
- As discussed elsewhere, performance competitive to superior, relative to protocols based on packet forwarding (SMF, etc.)

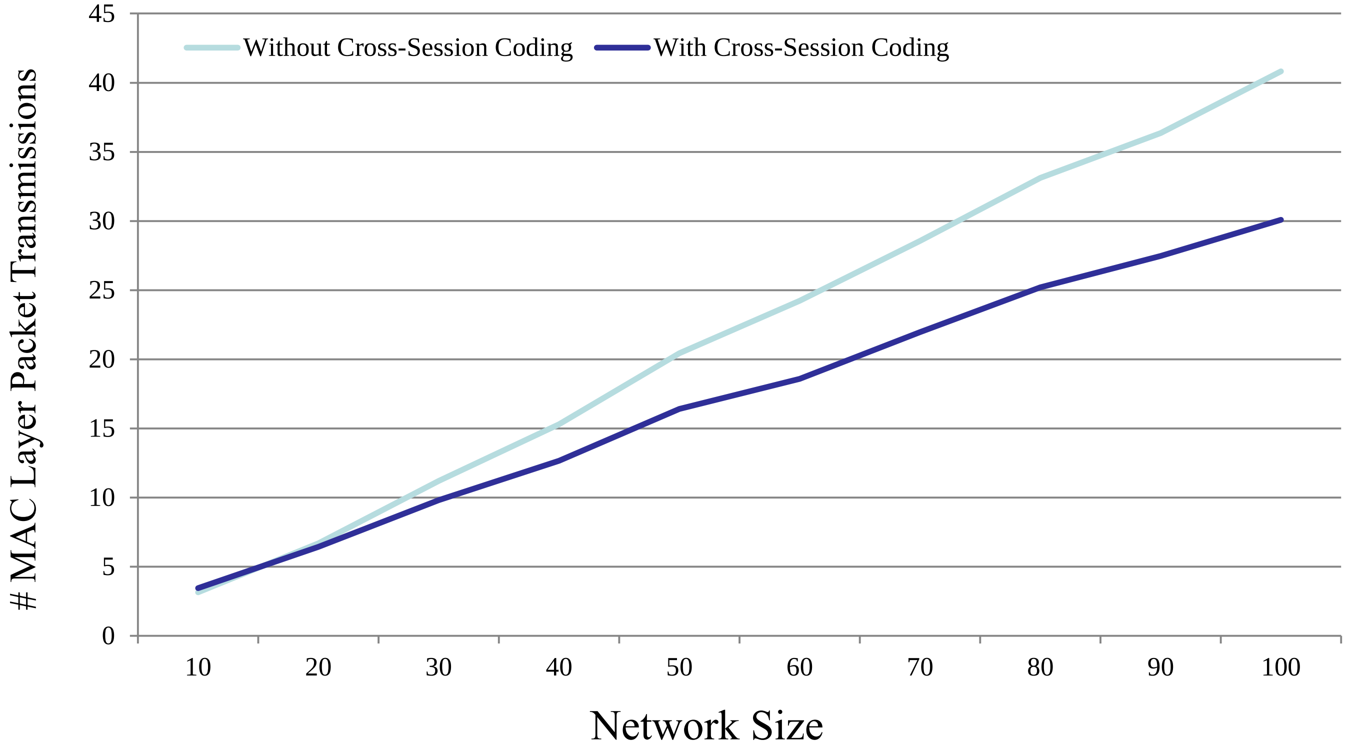
ARLNCCF Results I: Benefits of Cross-Session Coding



Each node has a data packet to transmit, all transmissions within 1 second

Huge saving in terms of MAC Transmissions

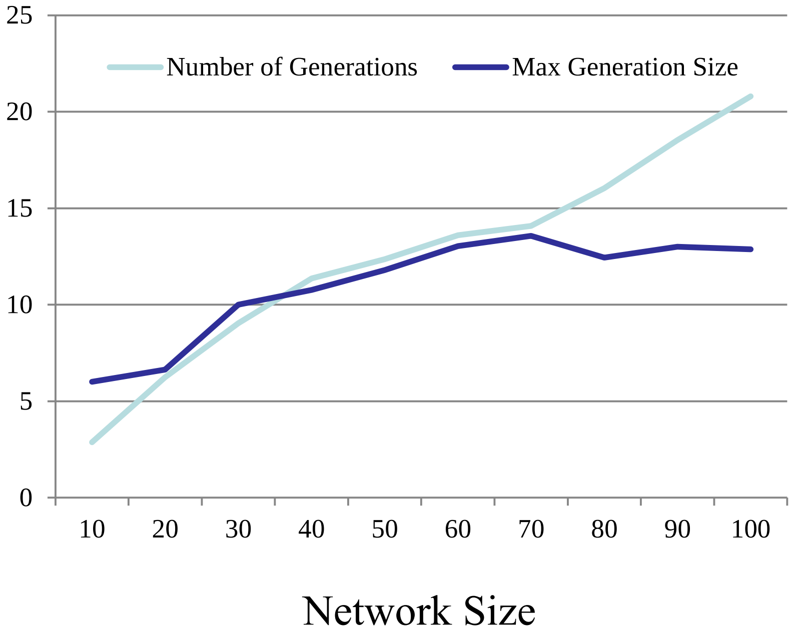
ARLNCCF Results I: Benefits of Cross-Session Coding



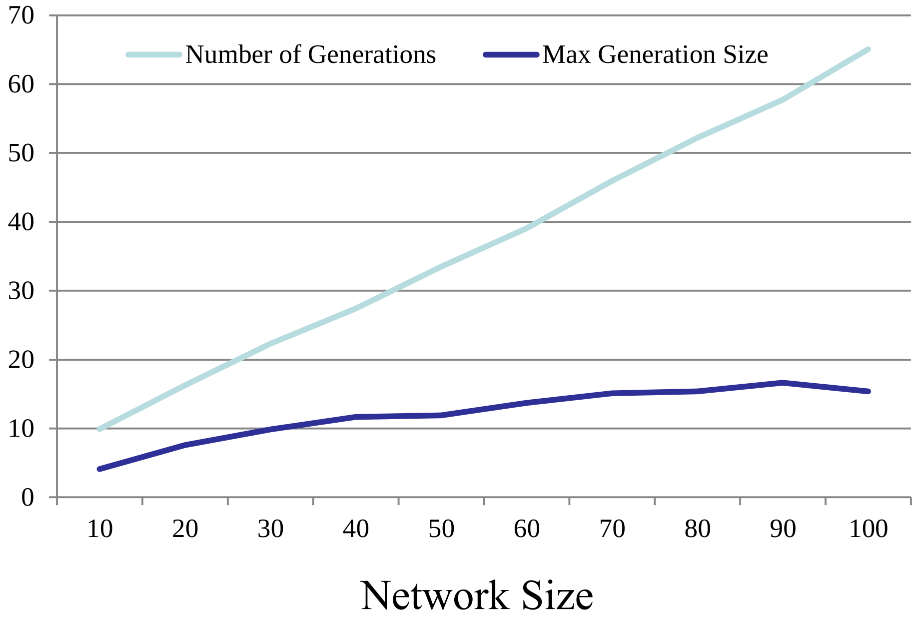
Each node has a generation worth of data packet to transmit, all transmissions within 1 second

Less relative gain, as nodes can efficiently fill local generation

ARLNCCF Results II: Cross-Session Coding Costs



1 Packet per Source



4 Packets per Source

Generation Management becomes crucial

Conclusions

- In Theory:
  - NC outperforms Packet Forwarding for Broadcasting in Multihop Wireless Networks
  - Cross-Session Coding has some impact on NC Efficiency
- In Reality:
  - No proposed packet forwarding protocol close to lower bound as network size increases
    - Partial view of network makes it more and more difficult to locally make “optimal” decision
    - Are there other/better ways?
  - NC protocol good/competitive to SMF, but not necessarily better
  - Cross-Session Coding has potentially HUGE impact on protocol efficiency
- Some Future Work:
  - Further improve ARLNCCF (generation mgmt, forwarding factor, etc.)
  - Add support for overlapping generations to increase robustness in face of packet loss

Own Work: Joint MAC Layer Scheduling and Network Coding for Wireless Networks

Cross-Layer Approach, with Prof. Banihashemi and Dr. Niati

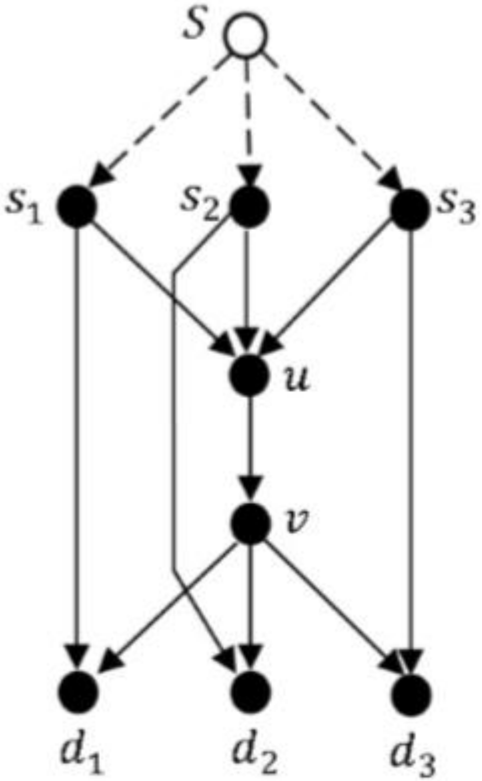
Research Motivation

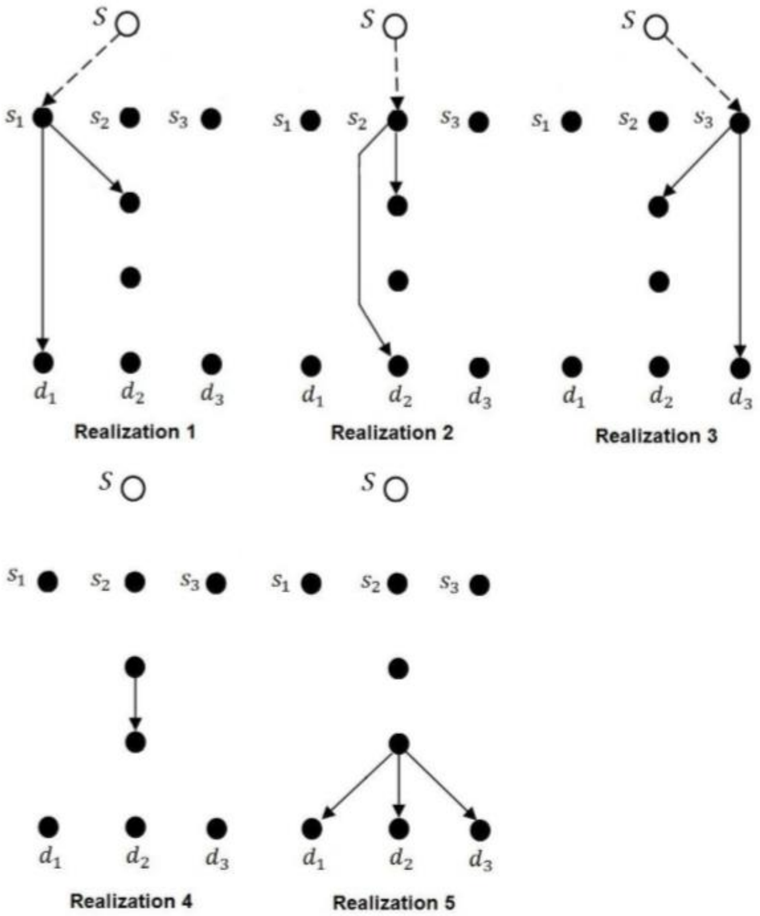
- Joint Network Coding and MAC Scheduling
  - Three steps
    - Interference-free Scheduling
      - Complexity: NP-hard ➔ Heuristics
      - Network realizations
      - Timesharing
    - Joint Optimization Problem
      - MAC scheduling and network coding
        - Linear
        - Non-linear, mixed integer
    - Network Code Design
      - For equal timeshares only

Research Contributions

- Formulating a joint linear optimization problem
  - Schedule-specific flows
- Unequal scheduling timeshares and network code design requirements for wireless networks
  - Preserving the broadcast property in code design
- Performance comparison of physical and protocol interference models
- Capacity-bundling Scheduling
- Objective functions: throughput, energy

MAC Heuristic: Finding Network Realizations (set of non-conflicting transmitters)





Some Basic Comments

- Need an interference model
  - In previous example, used the “protocol” model: transmitters up to two hops away will interfere with a node’s transmission
  - Protocol model captures how RTS/CTS mechanism in 802.11works
- Once we have all (or a large subset of possible) realizations, we need to determine:
  - How often they should be used (could be 0)
  - How the nodes code the information received in each instance
    - Prior work proposed a scheme by designing code for equivalent wired network and translated these codes back to the wireless network
- To answer this, need some objective (what are we trying to achieve). Used two objectives in this work:
  - Maximize throughput
  - Given a target throughput (that is feasible), what is the most energy-efficient way to achieve that throughput

Network Model

- Multihop wireless network: Directed Acyclic Graph (DAG)

$$G = (V, E)$$

- Multicast scenario
  - Independent sources
  - Destinations

$$S = \{s_i\} \subset V$$

$$D = \{d_i\} \subset V$$

Optimization Problem

- Input: realizations $N^f = \{N_1^f, N_2^f, \ldots, N_M^f\}$, $N_m^f = (V_m^f, E_m^f)$
- Objective function: Throughput
- Variables: link flows & scheduling time fractions
- Scheduling time fractions:

$$\tau_m \le 1 \quad m = \{1, 2, \ldots, M\}, \qquad \sum_{m=1}^{M} \tau_m = 1$$

- Model the set of all realizations N^f with a wired network

$$N^g = (V^g, E^g) = \left(\bigcup_{m=1}^{M} V_m^f, \bigcup_{m=1}^{M} E_m^f\right)$$

  - For $(i, j) \in V^g$, $C_{i,j} = \sum_{m=1}^{M} \tau_m c_{i,j} \mathbf{I}_{E_m^f}((i, j))$

Linear Formulation

$$\max r$$

Subject to

$$\sum_{m=1}^{M}\Big(\sum_{j:(i,j)\in E_m^f} f_{i,j}^{(m)}(d) - \sum_{i:(i,j)\in E_m^f} f_{j,i}^{(m)}(d)\Big) = \sigma_i$$

(flow conservation constraint)

$$\sigma_i = \begin{cases} r & if \quad i = s \\ -r & if \quad i = d \\ 0 & otherwise \end{cases}$$

$$0 \le f_{i,j}^{(m)}(d) \le f_{i.j}^{(m)}$$

$$0 \le f_{i,j}^{(m)} \le \tau_m c_{i,j} \mathbf{I}_{E_m^f}((i,j))$$

(capacity constraint)

$$\sum_{m=1}^{M} \tau_m = 1$$

(Normalized working cycle)

Throughput Maximization Solution

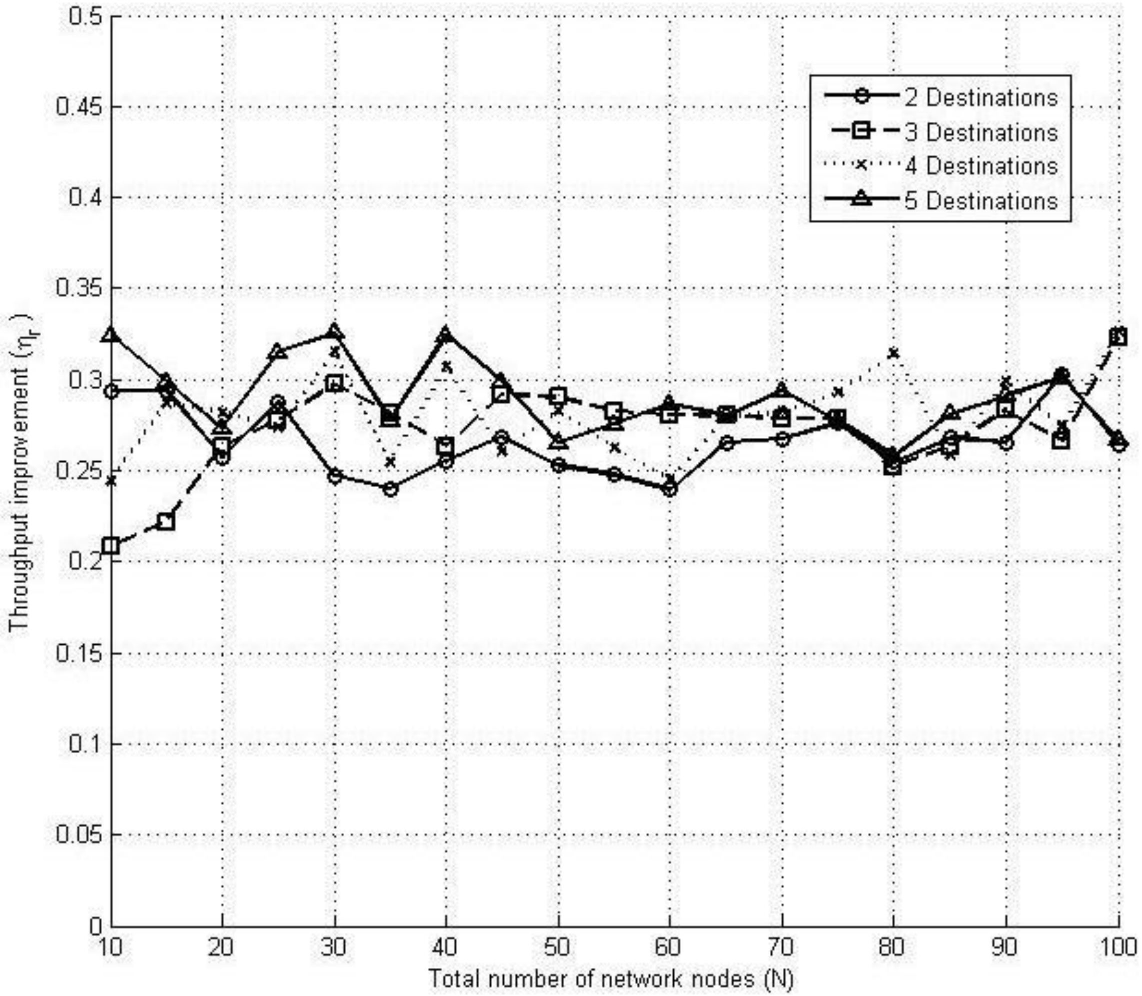
- Optimal solution uses all realizations

$$\tau_1 = \tau_2 = \tau_3 = 1/7, \tau_4 = \tau_5 = 2/7,$$

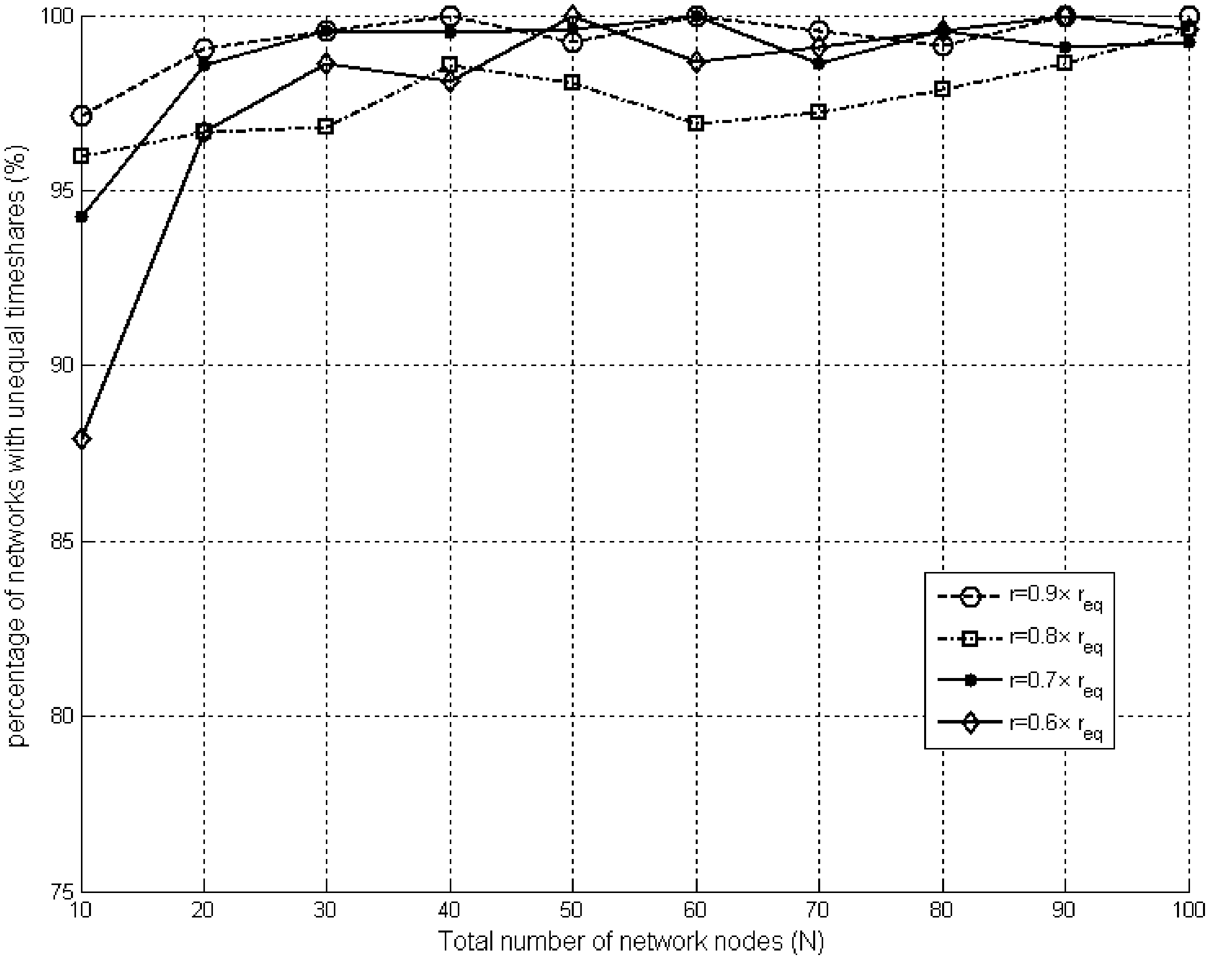
$$r = 3/7 = 0.42 \ symbols/timeslot.$$

- Realizations 4 and 5 used twice as often (twice as long) as the other three
- Linear program also tells us what the achievable max rate is
- Missing: how should intermediate nodes combine/code packets to achieve this rate
  - Related work: proposed a solution, but that worked only when all timeshares were equal
  - Our contribuiton:
    - Fixed the code design approach
    - Demonstrated that considering unequal timeshares is importent

Throughput Improvement When Considering Unequal Timeshares



Percentage of Networks that Require Unequal Timeshares (Objective Function here: minimize energy for a given rate)



Conclusions

- Non-negligible percentage of networks require unequal timeshares
- Appropriately incorporating unequal timeshares in code design can
  - Improve the throughput by 35%
  - Save the energy between 13-30%
- Solution: wireless-aware code construction

Performance Comparison of Physical and Protocol Interference Models

- Physical Model (SINR Model)
  - A transmission is successful according to the physical model if the SINR at the receiving node is higher than a specified threshold
- Protocol Model
  - Transmission range
  - Interference range

Channel Model

- Transceivers use a set of Q embedded MQAM signal constellations, with sizes

$$\{M_1, \ldots, M_Q\}$$

- Constellation size: M → spectral efficiency $\frac{c_{ij}}{B_{ij}} = \log_2 M$

- Spectral efficiency also parameterized by the transmit power and the BER
  - Each M translates to a SINR threshold

Scheduling: Physical vs. Protocol Model

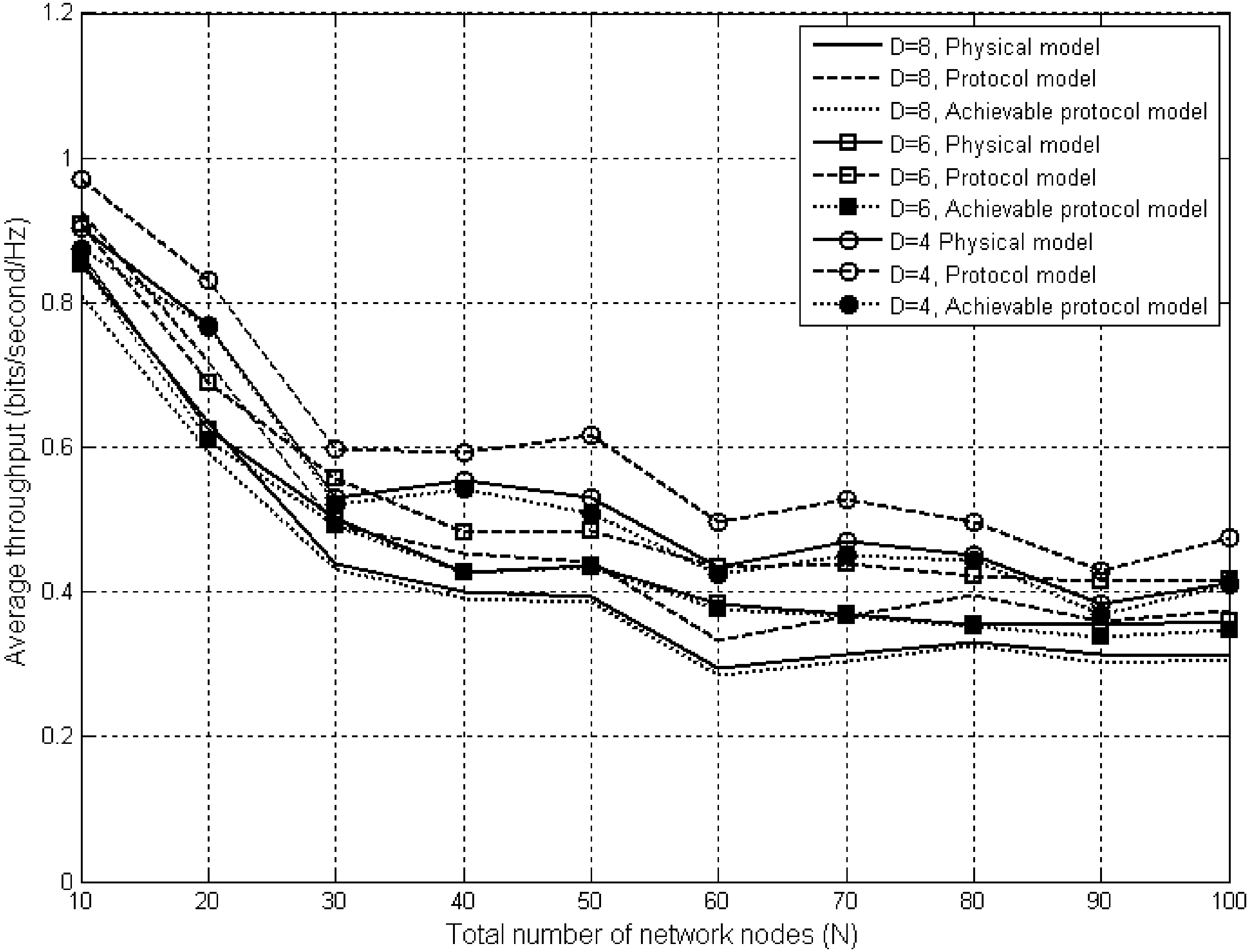
- Physical Model:
  - Minimize the powers in each realization
  - While satisfying the SNIR condition
- Protocol Model:
  - Set the transmission range based on SNR
  - Interference range $R_i' = (1+\Delta)R_i\ ,\quad \Delta > 0$
  - Set transmission powers
    - Minimum:
      - satisfies the SNR threshold
    - Maximum: increases interference, not suitable for energy problems

Simulations

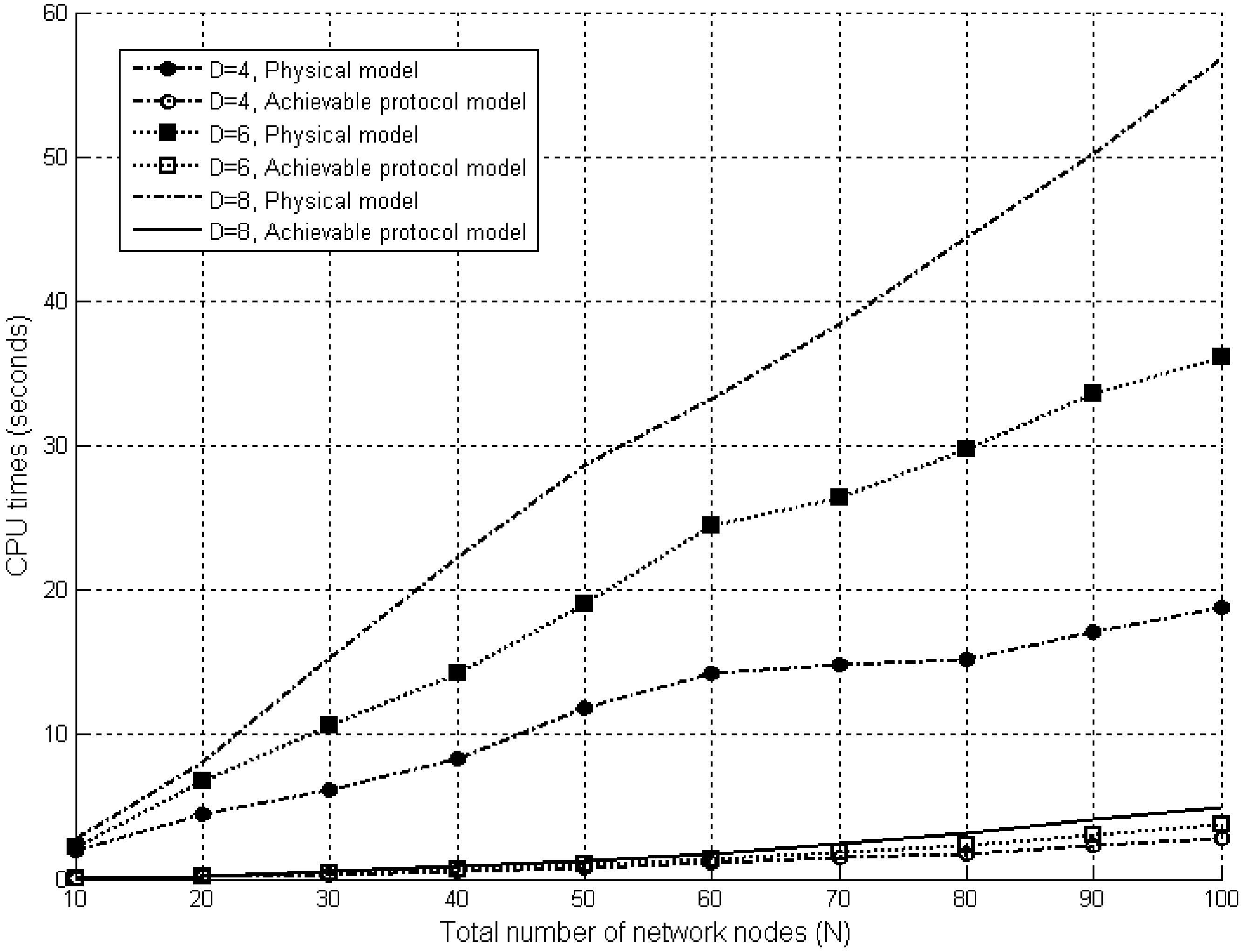
- Fixing the interference range for protocol model (fixing Δ)
- Examining the effect of changing δ
- Comparing the performance of the two models in throughput maximization/energy minimization problems.

- Assumption: MQAM constellation sizes: {2,4,16,64} corr. to BPSK, 4QAM, 16QAM, 64QAM

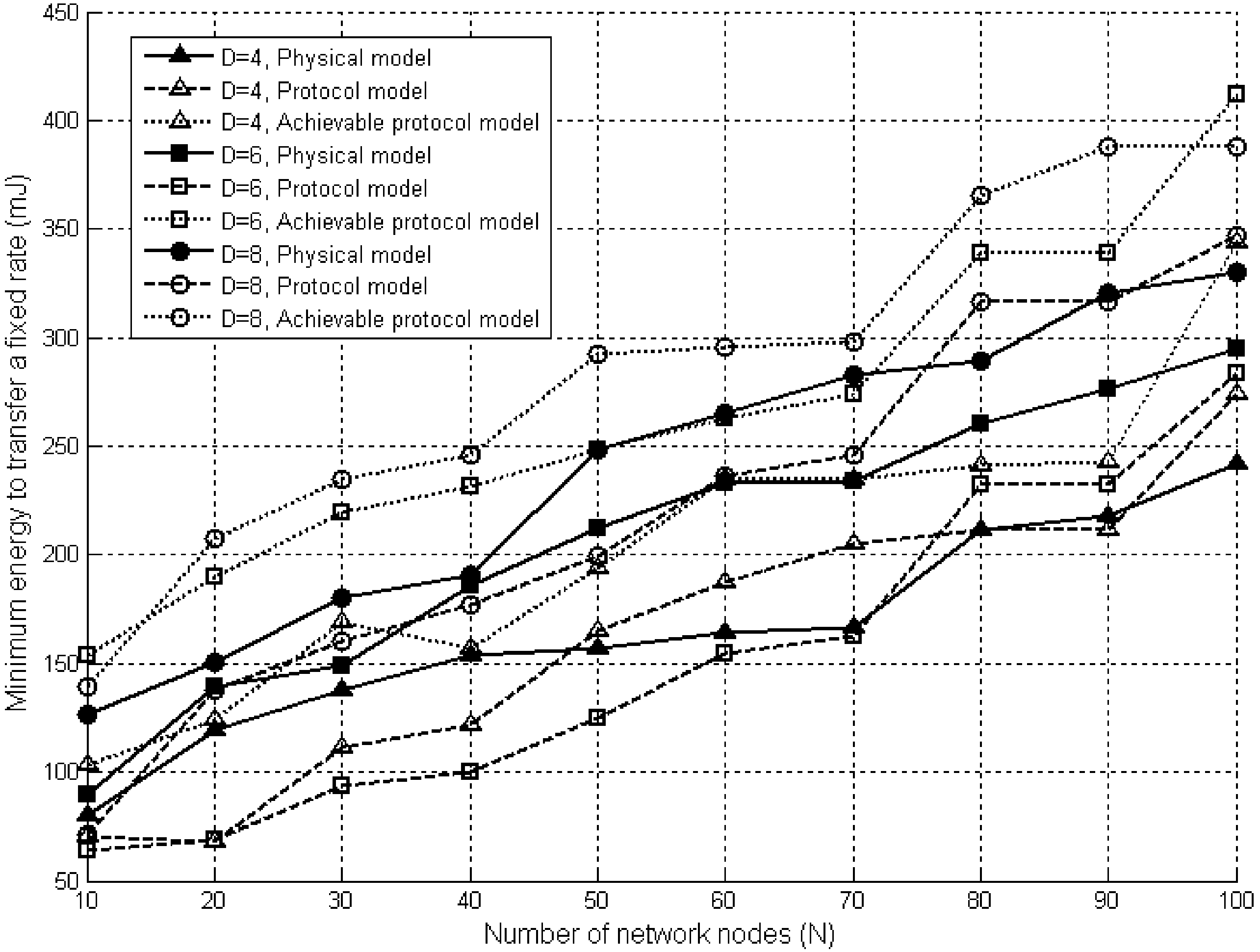
Throughput Maximization Results



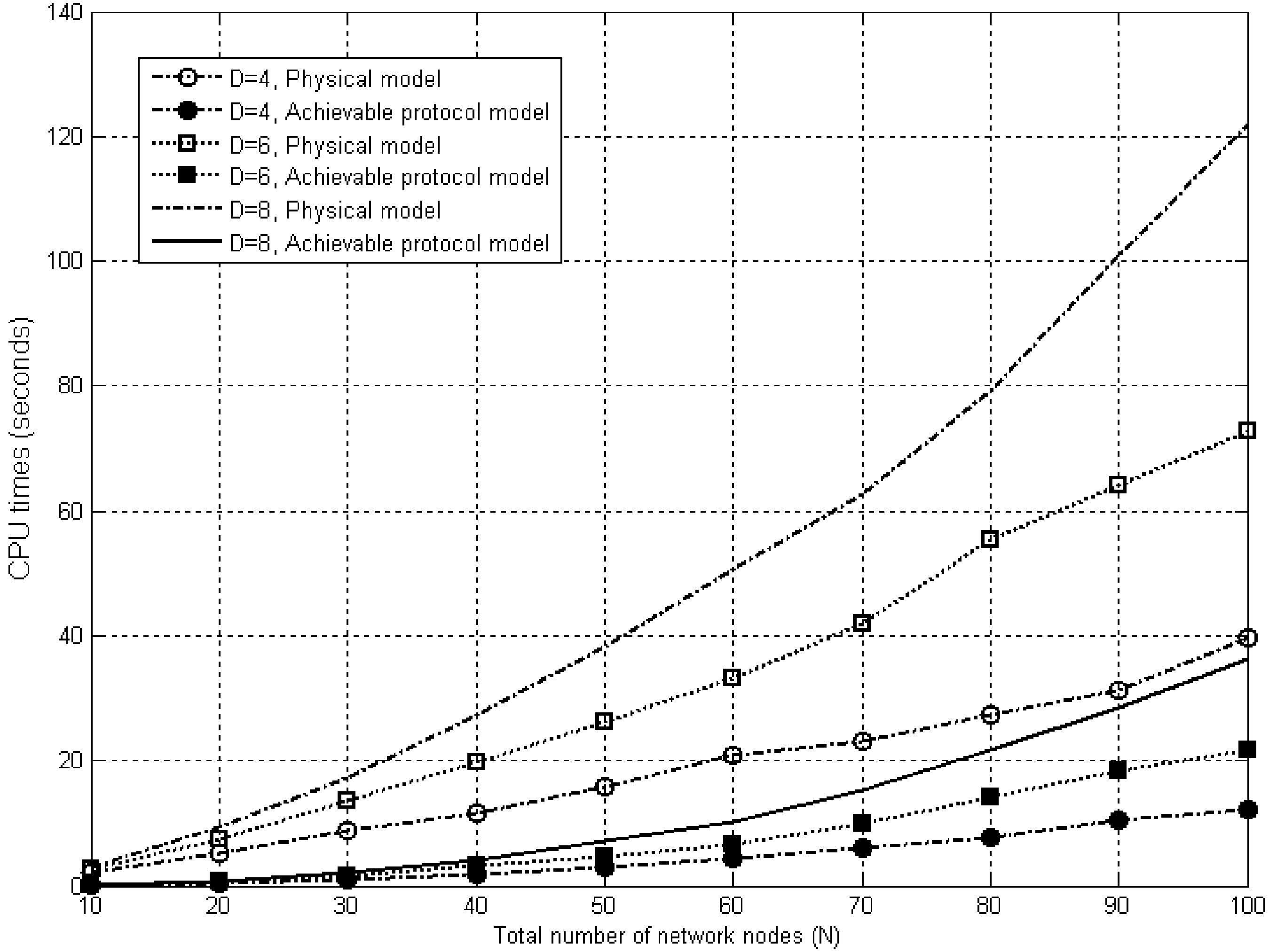
Comparing the CPU Times



Energy Minimization Results



Comparing the CPU Times



Conclusions

- Protocol model can be replaced with the physical model in throughput maximization problems
  - Similar results
  - Much lower computational complexity to determine realizations
- Protocol model is not recommended for energy minimization problems
  - Computational effort lower but results are quite a bit poorer

Capacity-bundling Scheduling

- Wireless scheduling → NP-hard
- Extensive search through all possible scheduling sets
  - High complexity, not scalable
- Identify influential factors (beside the interference) that can bias the scheduling selection towards a better solution for optimization problem
  - Link capacities

Link Capacities: new factor in scheduling wireless transmissions

- Idea: schedule links with same or close capacities together
  - Specially avoiding scheduling a very high capacity with a very low capacity link
    - They have a common scheduling timeshare
    - Results in potentially wasted (unused) capacity for the link with high capacity
- Increases the number of realizations

Capacity Bundling Scheduling

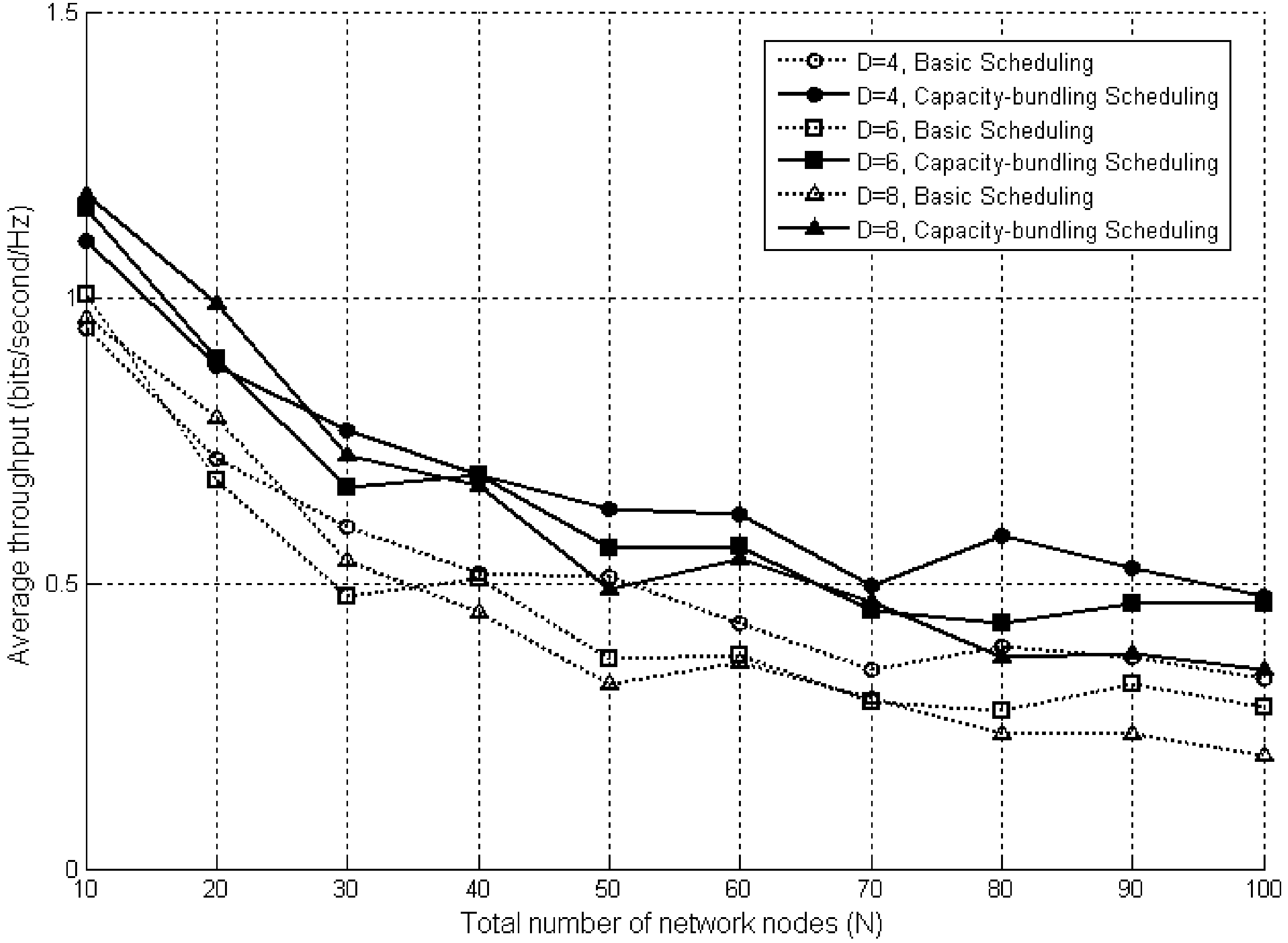
- Basic scheduling: schedule links that satisfy

$$\min \sum_{i \in X} P_i$$

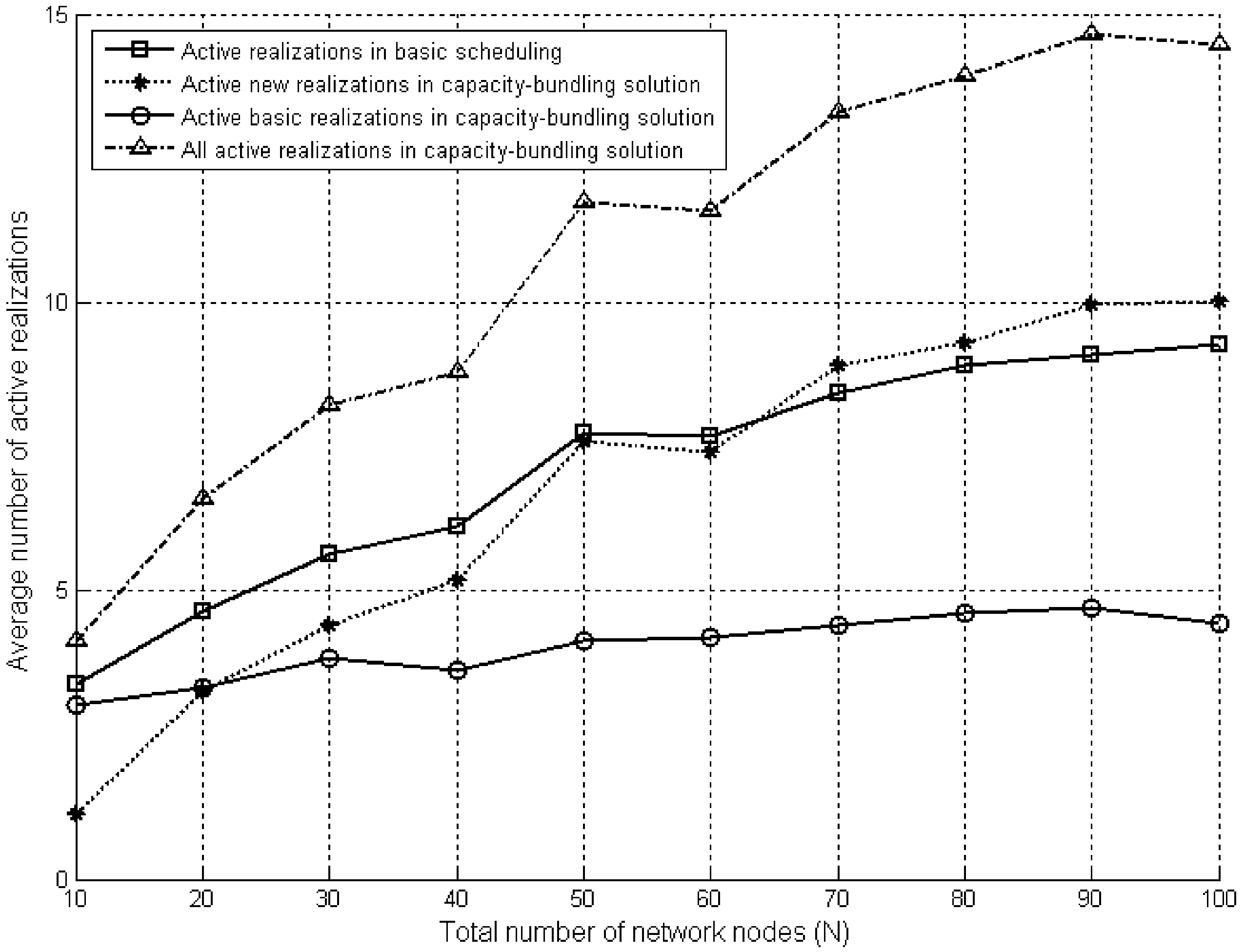
$$s.t. \quad \begin{cases} \gamma_q \le SINR_{ij} \le \gamma_{q+1} & q < Q \\ SINR_{ij} \ge \gamma_Q & otherwise \end{cases}$$

- Capacity-bundling: add realizations to basic scheduling with equal capacities
- $N_{CB}^{f} = N_{B}^{f} \cup N_{equal\_capacity}^{f}$

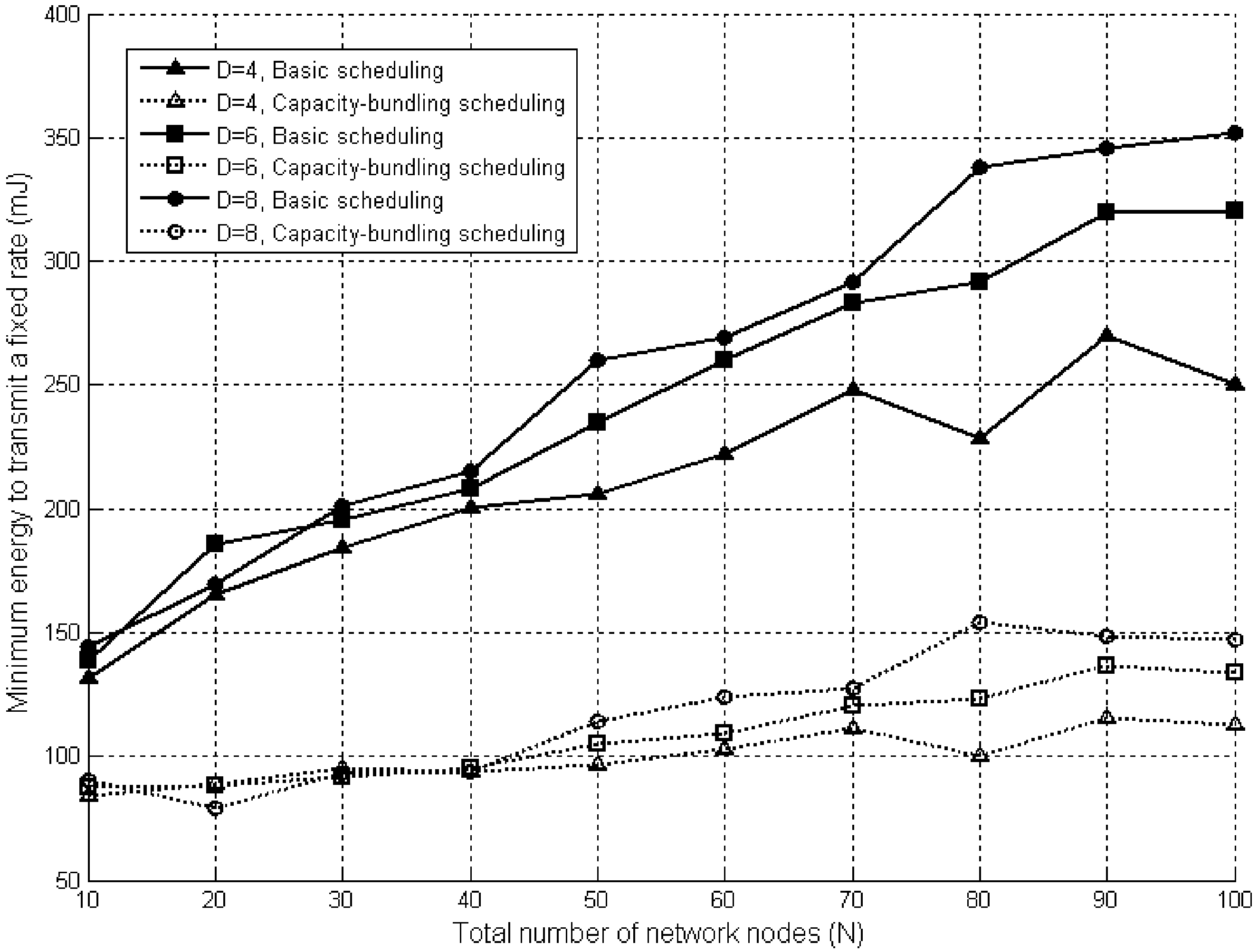
Maximum Throughput



Average Number of Active Realizations



Minimum Energy



Conclusion

- We introduced a new influential factor for scheduling wireless transmissions
  - Interferences (SINR)
  - Link capacities
- Simulation results show improvement in both throughput maximization & energy minimization problems
  - Throughput improvement up to 80%
  - Energy saving up to 55%
  - Underutilized capacity: constant for different network sizes

Network Coding: Conclusion

- New approach to transmitting packets
- Has demonstrable benefits in both wired and wireless networks
- Coding gain typically a constant factor
  - Not too impressive for theorist, but of interest to engineers
- Poses a range of problems as well
  - What is a "flow" (remember, SDN/OpenFlow is flow-based)
  - How to assure QoS to some "flows" but not others?
  - How to manage/monitor in such a network?