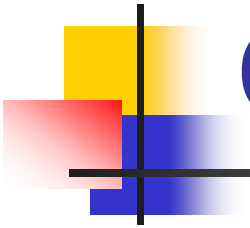
Network Coding for the Internet

Philip A. Chou*, Yunnan Wu†, and Kamal Jain*
*Microsoft Research & †Princeton University

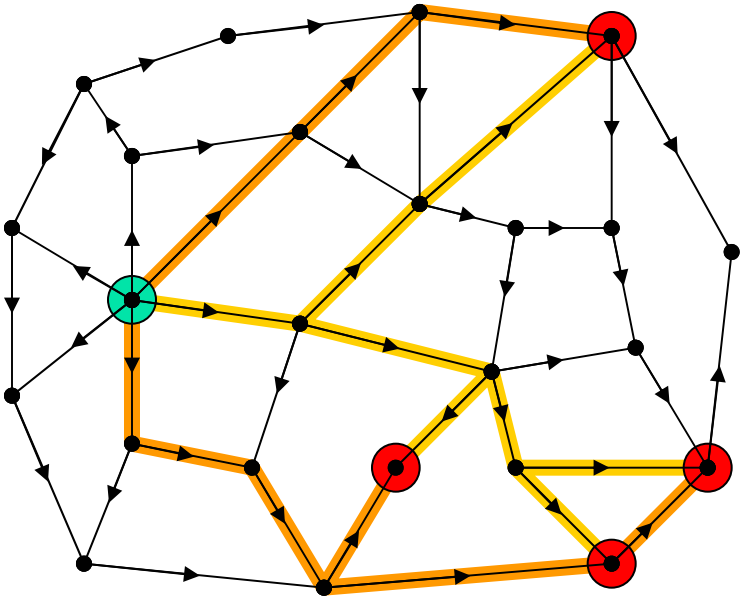
Communication Theory Workshop, May 6-8, 2004
and EPFL, May 10, 2004



Outline

- Introduction to Network Coding
- Practical Network Coding
  - Packetization
  - Buffering
- Internet Applications
  - Live Broadcasting, File Downloading, Messaging, Interactive Communication

Network Coding Introduction

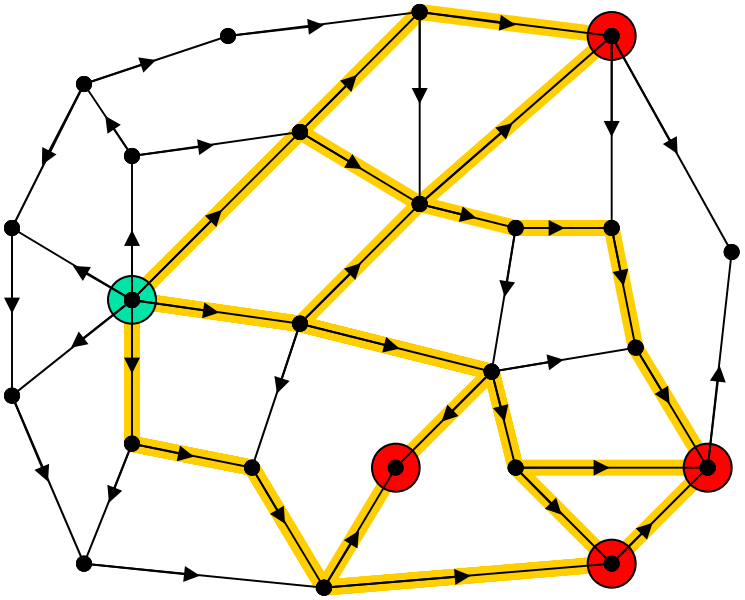


sender s

receiver t in T

- Directed graph with edge capacities
- Sender s, set of receivers T
- Ask: Maximum rate to broadcast info from s to T ?

Maximum Flow

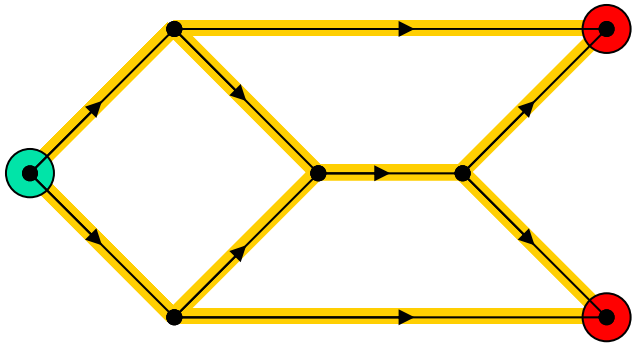


sender s

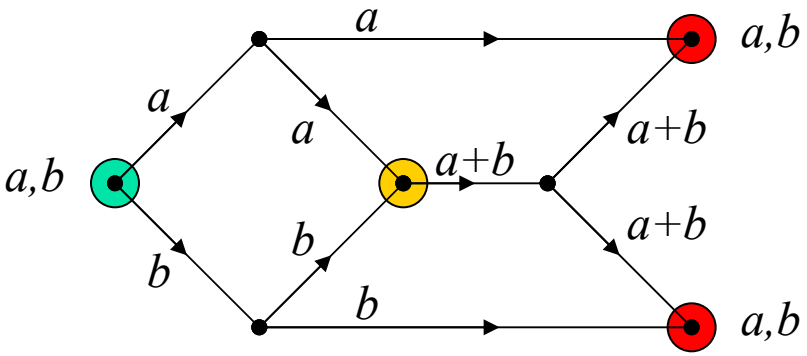
receiver t in T

- Menger (1927) – single receiver
  - Maxflow(s,t) ≤ Mincut(s,t) ≡ h_t achievable
- Edmonds (1972) – all nodes are receivers
  - Maxflow(s,T) ≤ $\min_t h_t \equiv h$ achievable if $T=V$

Network Coding Maximizes Throughput



optimal multicast
throughput = 1



network coding
throughput = 2

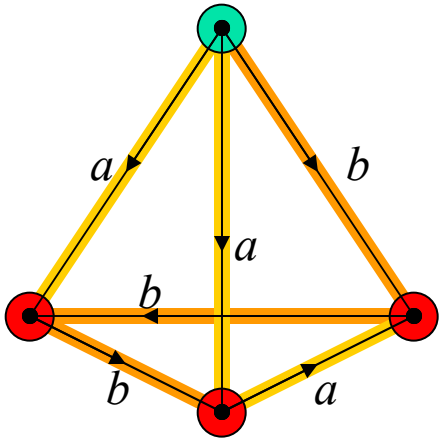
- Alswede, Cai, Li, Yeung (2000)
  - NC always achieves $h = \min_t h_t$
- Li, Yeung, Cai (2003)
- Koetter and Médard (2003)

sender

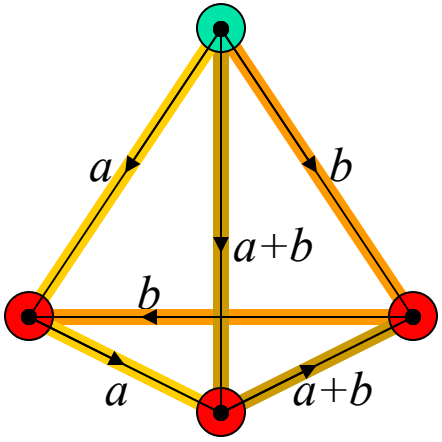
receiver

coding node

Network Coding Minimizes Delay



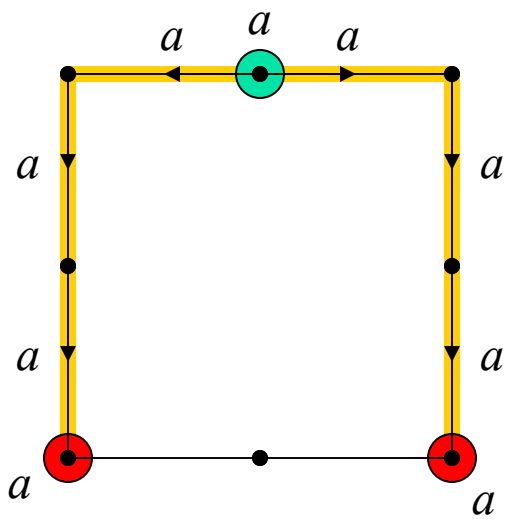
optimal multicast
delay = 3



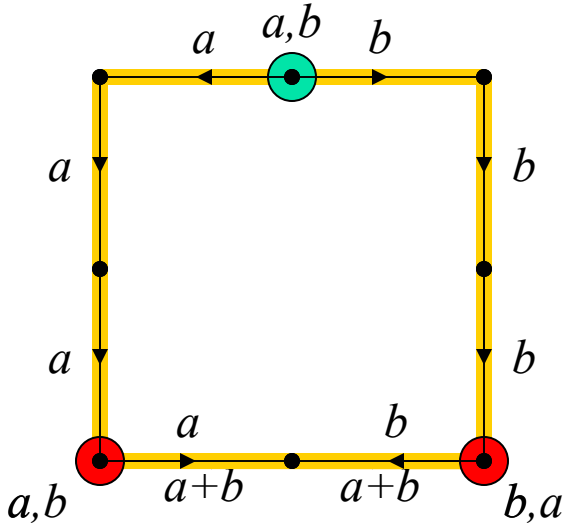
network coding
delay = 2

- Jain and Chou (2004)
  - Reconstruction delay at any node t is no greater than the maximum path length in any flow from s to t.

Network Coding Minimizes Energy (per bit)



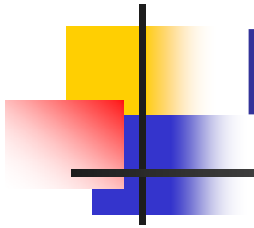
optimal multicast
energy per bit = 5



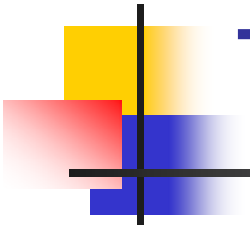
network coding
energy per bit = 4.5

- Wu et al. (2003); Wu, Chou, Kung (2004)
- Lun, Médard, Ho, Koetter (2004)

Network Coding Applicable to Real Networks?



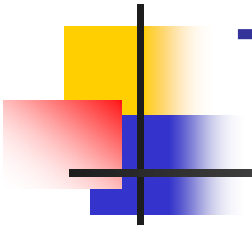
- Internet
  - IP Layer
    - Routers (e.g., ISP)
  - Application Layer
    - Infrastructure (e.g., CDN)
    - Ad hoc (e.g., P2P)
- Wireless
  - Mobile ad hoc multihop wireless networks
  - Sensor networks
  - Stationary wireless (residential) mesh networks



Theory vs. Practice (1/4)

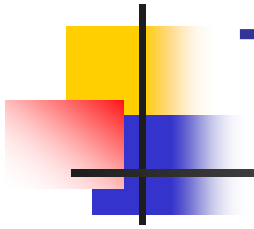
- Theory:
  - Symbols flow synchronously throughout network; edges have integral capacities
- Practice:
  - Information travels asynchronously in packets; packets subject to random delays and losses; edge capacities often unknown, varying as competing communication processes begin/end

Theory vs. Practice (2/4)



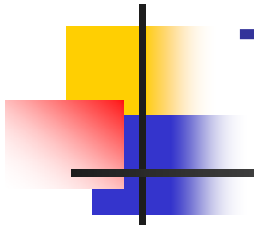
- Theory:
  - Some centralized knowledge of topology to compute capacity or coding functions
- Practice:
  - May be difficult to obtain centralized knowledge, or to arrange its reliable broadcast to nodes across the very communication network being established

Theory vs. Practice (3/4)



- Theory:
  - Can design encoding to withstand failures, but decoders must know failure pattern
- Practice:
  - Difficult to communicate failure pattern reliably to receivers

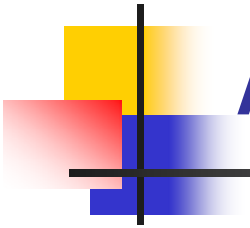
Theory vs. Practice (4/4)



- Theory:
  - Cyclic graphs present difficulties, e.g., capacity only in limit of large blocklength
- Practice:
  - Cycles abound. If $A \rightarrow B$ then $B \rightarrow A$.

Our work addresses practical network coding in real networks

- Packets subject to random loss and delay
- Edges have variable capacities due to congestion and other cross traffic
- Node & link failures, additions, & deletions are common (as in P2P, Ad hoc networks)
- Cycles are everywhere
- Broadcast capacity may be unknown
- No centralized knowledge of graph topology or encoder/decoder functions
- Simple technology, applicable in practice

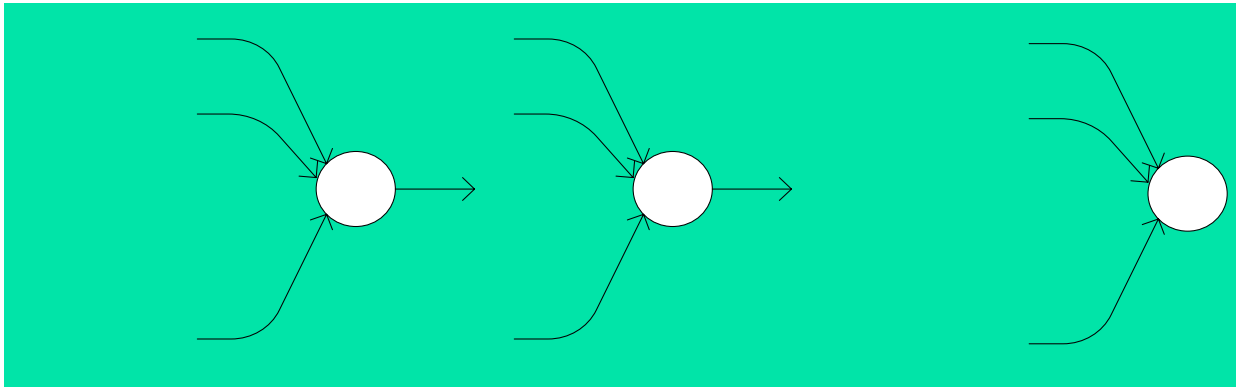


Approach

- Packet Format
  - Removes need for centralized knowledge of graph topology and encoding/decoding functions
- Buffer Model
  - Allows asynchronous packets arrivals & departures with arbitrarily varying rates, delay, loss

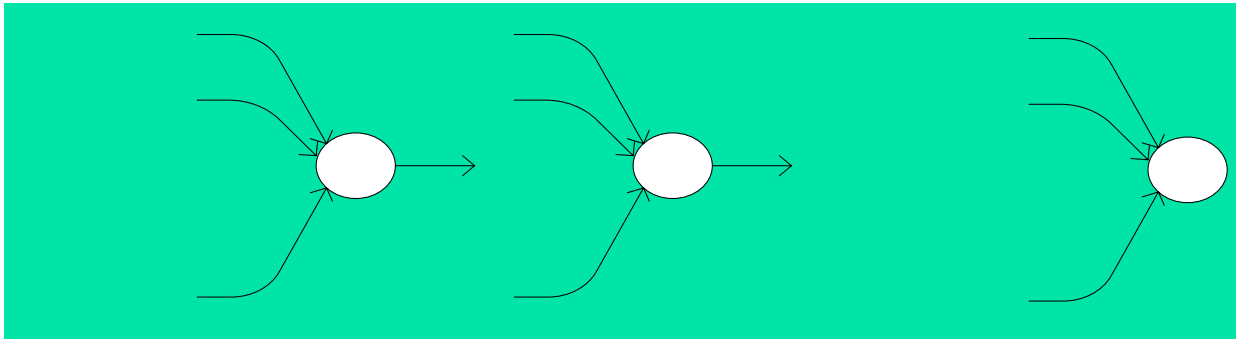
Standard Framework

- Graph (V,E) having unit capacity edges
- Sender s in V, set of receivers $T=\{t,\ldots\}$ in V
- Broadcast capacity $h = \min_t \text{Maxflow}(s,t)$



- $y(e) = \sum_{e'} m_e(e')\, y(e')$
- $\mathbf{m}(e) = [m_e(e')]_{e'}$ is *local encoding vector*

Global Encoding Vectors



- By induction $y(e) = \sum_{i=1}^{h} g_i(e)\, x_i$
- $\mathbf{g}(e) = [g_1(e),\ldots,g_h(e)]$ is *global encoding vector*
- Receiver t can recover $x_1,\ldots,x_h$ from

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} g_1(e_1) & \cdots & g_h(e_1) \\ \vdots & \ddots & \vdots \\ g_1(e_h) & \cdots & g_h(e_h) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G_t \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix}$$

$$y(e') = x$$

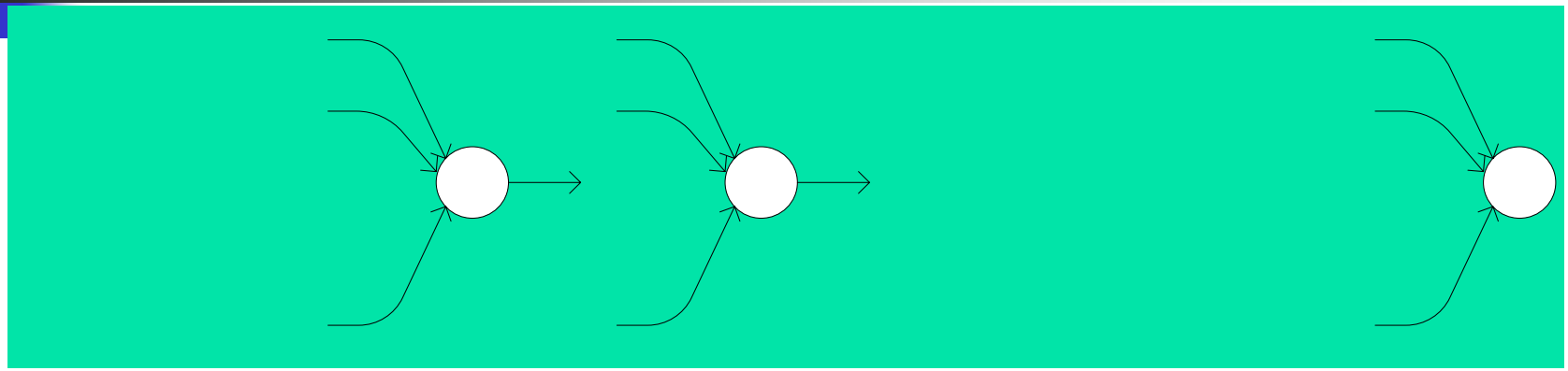
Invertibility of G_t

- G_t will be invertible with high probability if local encoding vectors are random and field size is sufficiently large
  - If field size = 2^{16} and |E| = 2^8 then G_t will be invertible w.p. ≥ $1-2^{-8} = 0.996$

[Ho et al., 2003]

[Jaggi, Sanders, et al., 2003]

Packetization



- Internet: MTU size typically ≈ 1400+ bytes
- $\mathbf{y}(e) = \sum_{e'} m_e(e') \, \mathbf{y}(e') = \sum_{i=1}^{h} g_i(e) \, \mathbf{x}_i$ s.t.

$$\begin{bmatrix} \mathbf{y}(e_1) \\ \vdots \\ \mathbf{y}(e_h) \end{bmatrix} = \begin{bmatrix} y_1(e_1) & y_2(e_1) & \cdots & y_N(e_1) \\ \vdots & \vdots & & \vdots \\ y_1(e_h) & y_2(e_h) & \cdots & y_N(e_h) \end{bmatrix} = G_t \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ \vdots & \vdots & & \vdots \\ x_{h,1} & x_{h,2} & \cdots & x_{h,N} \end{bmatrix}$$

Key Idea

- Include *within each packet* on edge e
  $\mathbf{g}(e) = \sum_{e'} m_e(e')\,\mathbf{g}(e')$; $\mathbf{y}(e) = \sum_{e'} m_e(e')\,\mathbf{y}(e')$
- Can be accomplished by prefixing i th unit vector to i th source vector $\mathbf{x}_i$, $i=1,\ldots,h$

$$\begin{bmatrix} g_1(e_1) & \cdots & g_h(e_1) & y_1(e_1) & y_2(e_1) & \cdots & y_N(e_1) \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ g_1(e_h) & \cdots & g_h(e_h) & y_1(e_h) & y_2(e_h) & \cdots & y_N(e_h) \end{bmatrix} = G_t \begin{bmatrix} 1 & & 0 & x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ & \ddots & & \vdots & \vdots & & \vdots \\ 0 & & 1 & x_{h,h} & x_{h,2} & \cdots & x_{h,N} \end{bmatrix}$$

- Then global encoding vectors needed to invert the code at any receiver can be found in the received packets themselves!

Cost vs. Benefit

- Cost:
  - Overhead of transmitting h extra symbols per packet; if $h = 50$ and field size $= 2^8$, then overhead ≈ $50/1400 \approx 3\%$
- Benefit:
  - Receivers can decode even if
    - Network topology & encoding functions unknown
    - Nodes & edges added & removed in ad hoc way
    - Packet loss, node & link failures w/ unknown locations
    - Local encoding vectors are time-varying & random

Erasure Protection

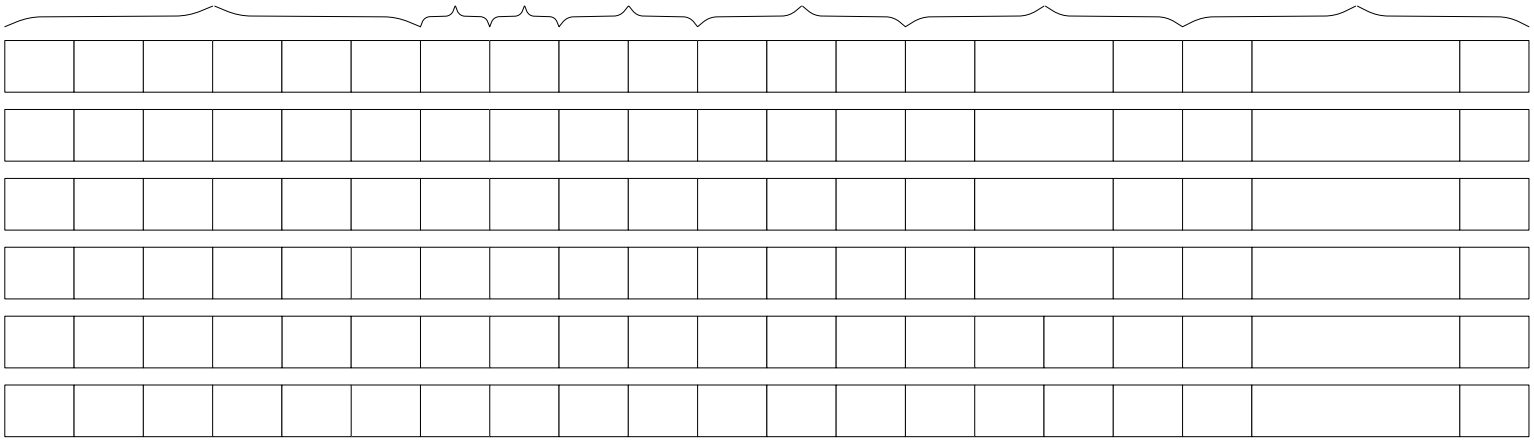
- Removals, failures, losses, poor random encoding may reduce capacity below h

$$\begin{bmatrix} g_1(e_1) & \cdots & g_h(e_1) & y_1(e_1) & y_2(e_1) & \cdots & y_N(e_1) \\ \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ g_1(e_k) & \cdots & g_h(e_k) & y_1(e_k) & y_2(e_k) & \cdots & y_N(e_k) \end{bmatrix} = G_t^{k\times h} \begin{bmatrix} 1 & & 0 & x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ & \ddots & & \vdots & \vdots & & \vdots \\ 0 & & 1 & x_{h,h} & x_{h,2} & \cdots & x_{h,N} \end{bmatrix}$$

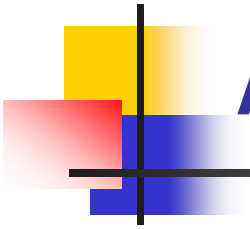
- Basic form of erasure protection: send redundant packets, e.g., last $h-k$ packets of $\mathbf{x}_1,\ldots,\mathbf{x}_h$ are known zero

Priority Encoding Transmission (Albanese et al., IEEE Trans IT '96)

- More sophisticated form: partition data into layers of importance, vary redundancy by layer
- Received rank $k \rightarrow$ recover k layers
- Exact capacity can be unknown

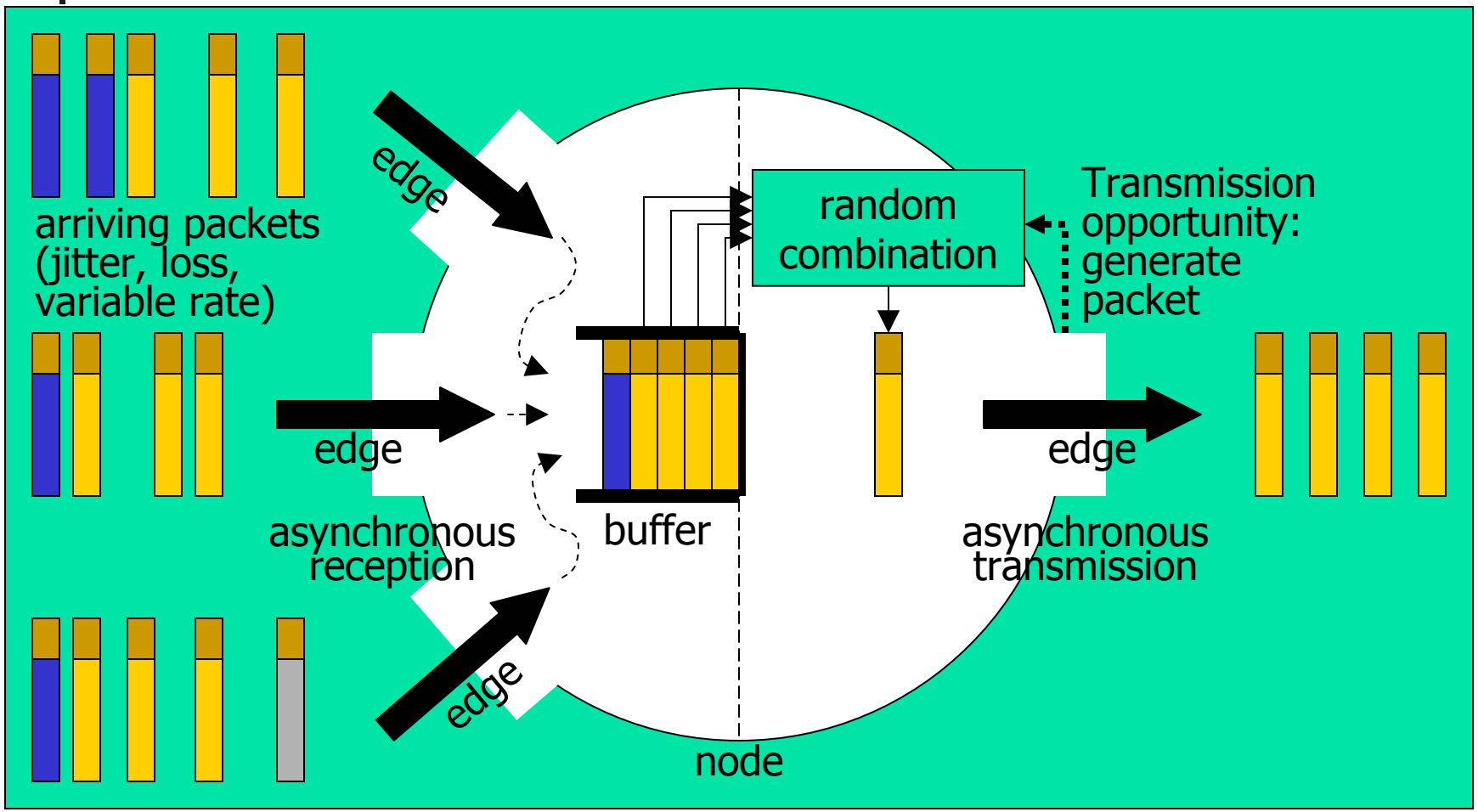


Asynchronous Communication



- In real networks, "unit capacity" edges grouped
  - Packets on real edges carried sequentially
  - Separate edges $\rightarrow$ separate prop & queuing delays
  - Number of packets per unit time on edge varies
    - Loss, congestion, competing traffic, rounding
- Need to synchronize
  - All packets related to same source vectors $\mathbf{x}_1, \ldots, \mathbf{x}_h$ are in same generation; h is generation size
  - All packets in same generation tagged with same generation number; one byte (mod 256) sufficient

Buffering

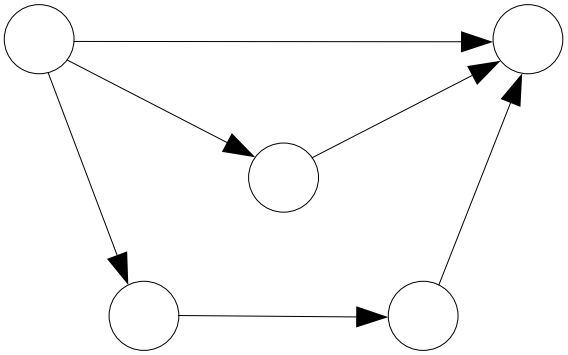

arriving packets
(jitter, loss,
variable rate)
edge
random
combination
Transmission
opportunity:
generate
packet
edge
edge
asynchronous
reception
buffer
asynchronous
transmission
edge
node

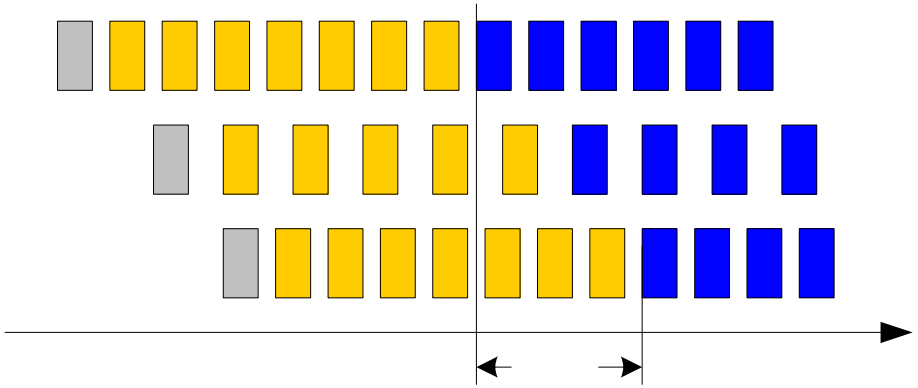
Decoding

- Block decoding:
  - Collect h or more packets, hope to invert G_t
- Earliest decoding (recommended):
  - Perform Gaussian elimination after each packet
    - At every node, detect & discard non-informative packets
  - G_t tends to be lower triangular, so can typically decode $\mathbf{x}_1, \ldots, \mathbf{x}_k$ with fewer more than k packets
  - Much lower decoding delay than block decoding
    - Approximately constant, independent of block length h

Flushing Policy, Delay Spread, and Throughput loss

- Policy: flush when first packet of next generation arrives on any edge
  - Simple, robust, but leads to some throughput loss

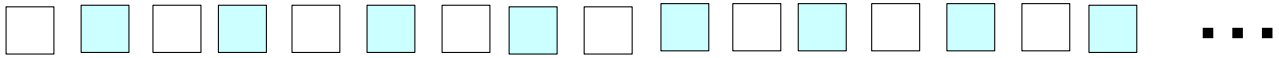




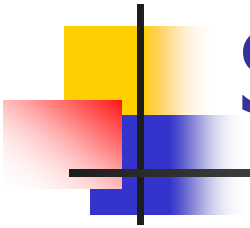
$$\text{throughput loss (\%)} \approx \frac{\text{delay spread (s)}}{\text{generation duration (s)}} = \frac{\text{delay spread (s)} \times \text{sending rate (pkt/s)}}{h \times I}$$

Interleaving

- Decomposes session into several concurrent interleaved sessions with lower sending rates

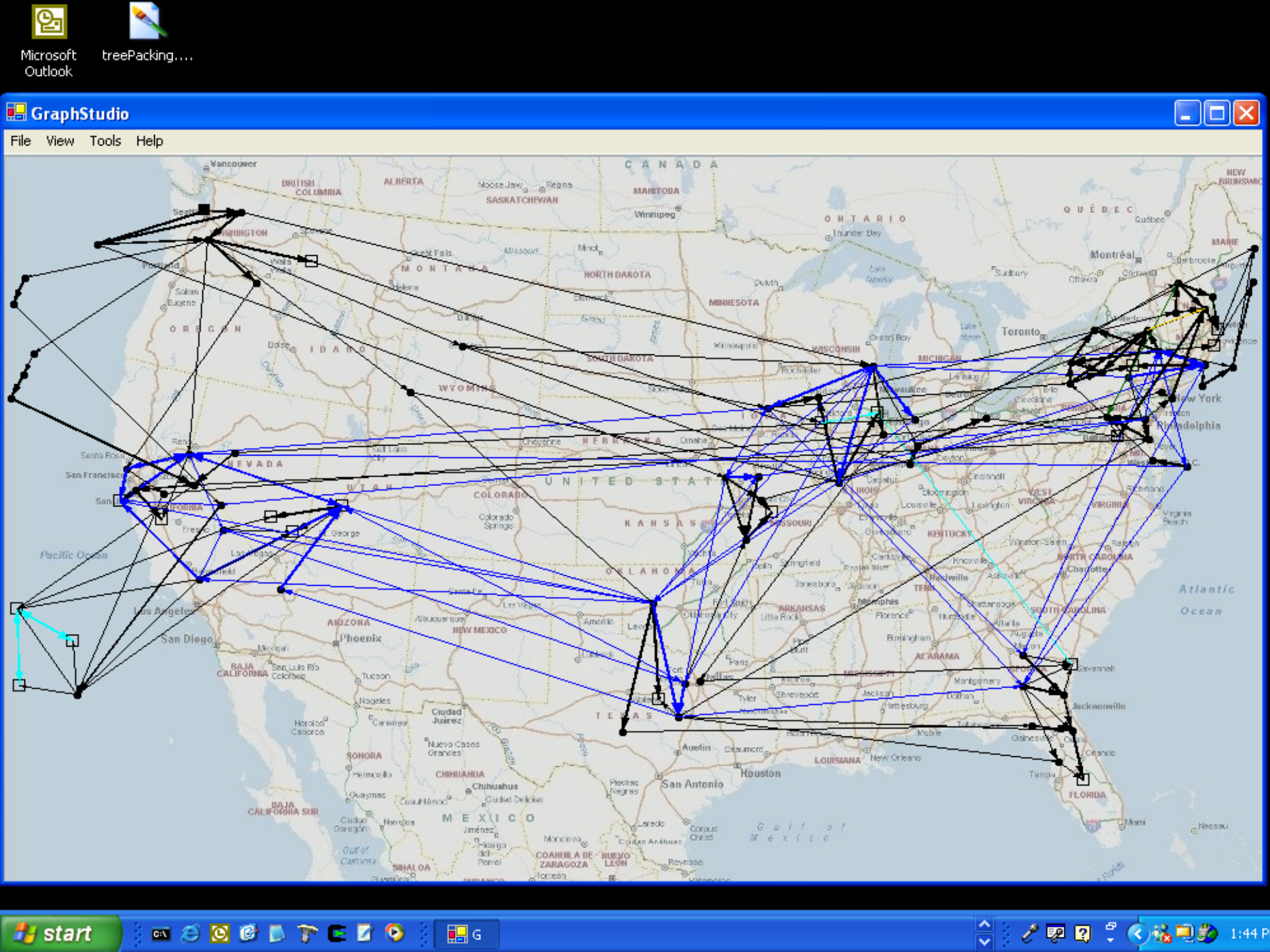


- Does not decrease overall sending rate
- Increases space between packets in each session; decreases relative delay spread

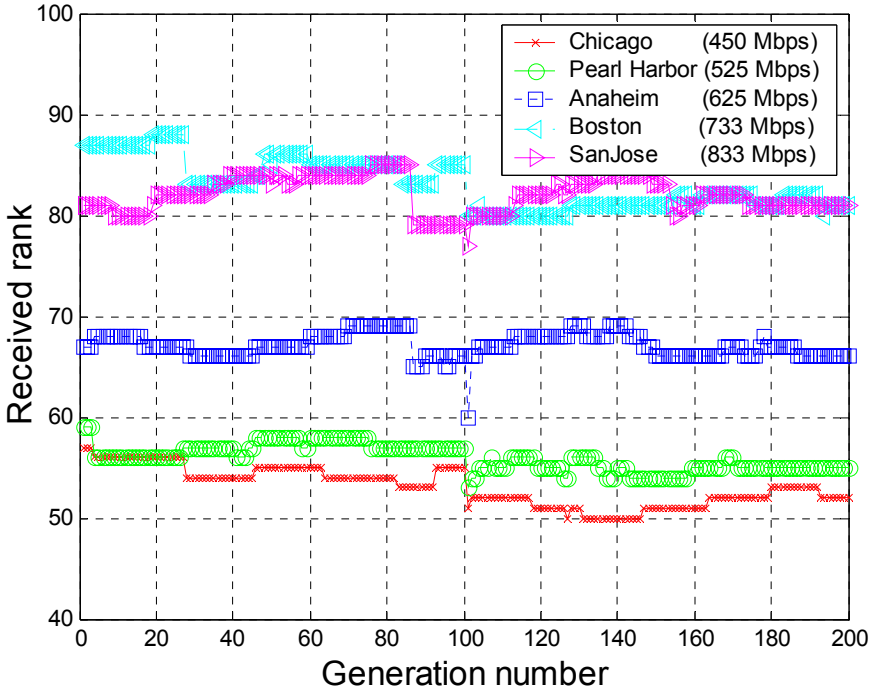


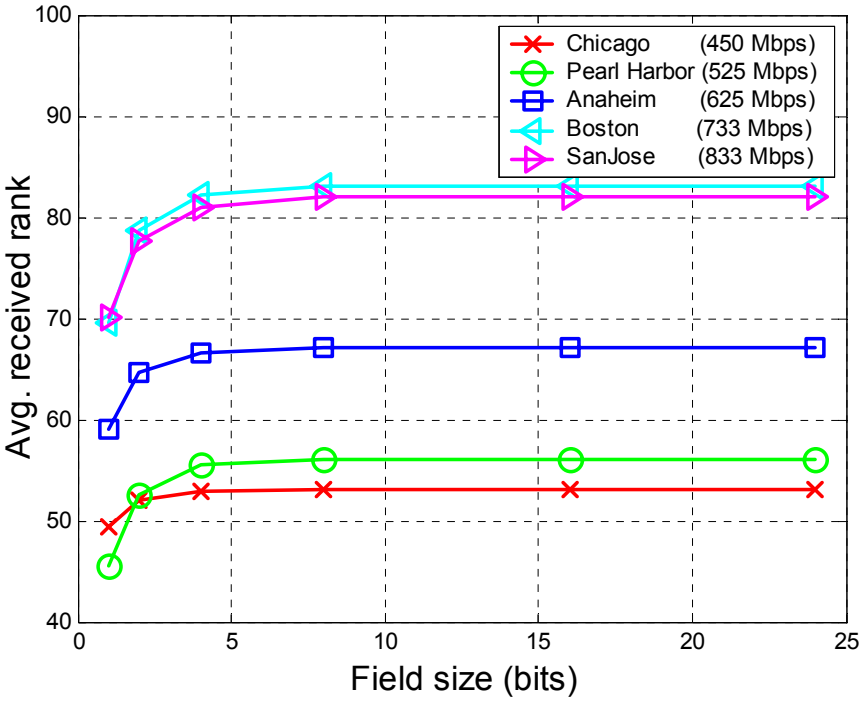
Simulations

- Implemented event-driven simulator in C++
- Six ISP graphs from Rocketfuel project (UW)
  - SprintLink: 89 nodes, 972 bidirectional edges
  - Edge capacities: scaled to 1 Gbps / "cost"
  - Edge latencies: speed of light x distance
- Sender: Seattle; Receivers: 20 arbitrary (5 shown)
  - Broadcast capacity: 450 Mbps; Max 833 Mbps
  - Union of maxflows: 89 nodes, 207 edges
- Send 20000 packets in each experiment, measure:
  - received rank, throughput, throughput loss, decoding delay vs. sendingRate(450), fieldSize(2^{16}), genSize(100), intLen(100)

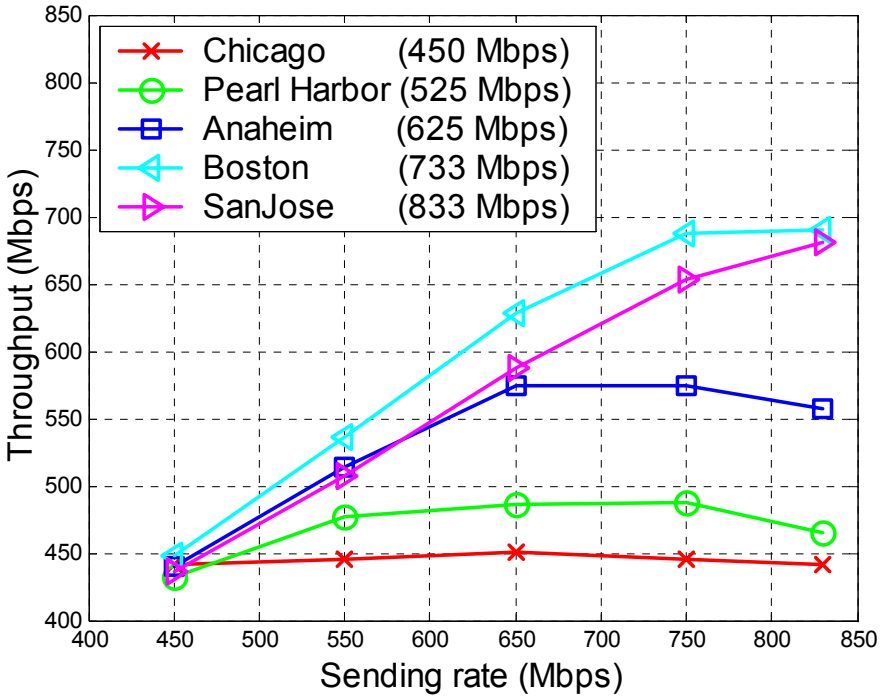

Microsoft Outlook
treePacking....
GraphStudio
File View Tools Help
CANADA
UNITED STATES
MEXICO
Pacific Ocean
Atlantic Ocean
Gulf of Mexico
start
1:44 P

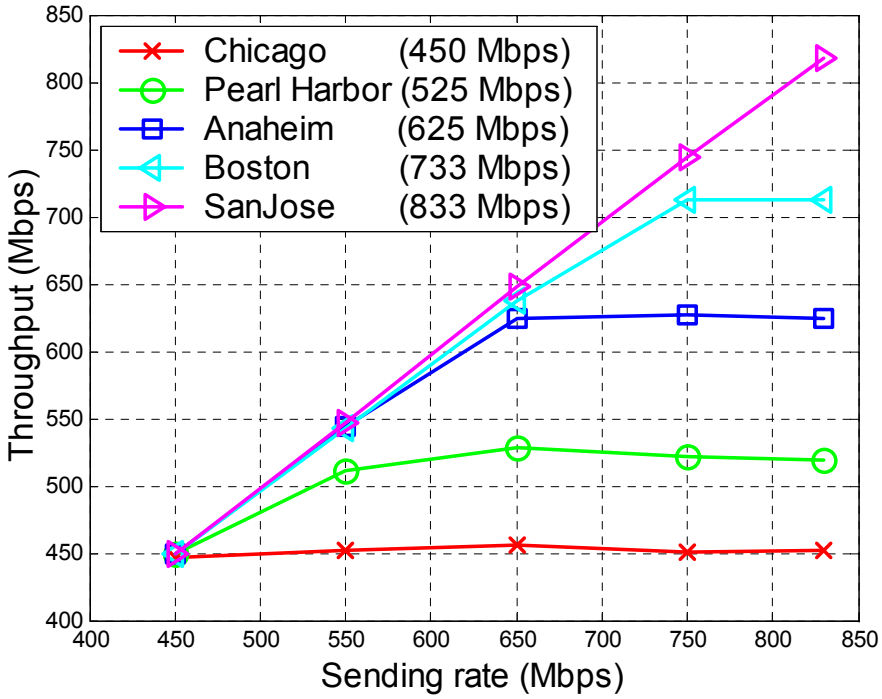
Received Rank



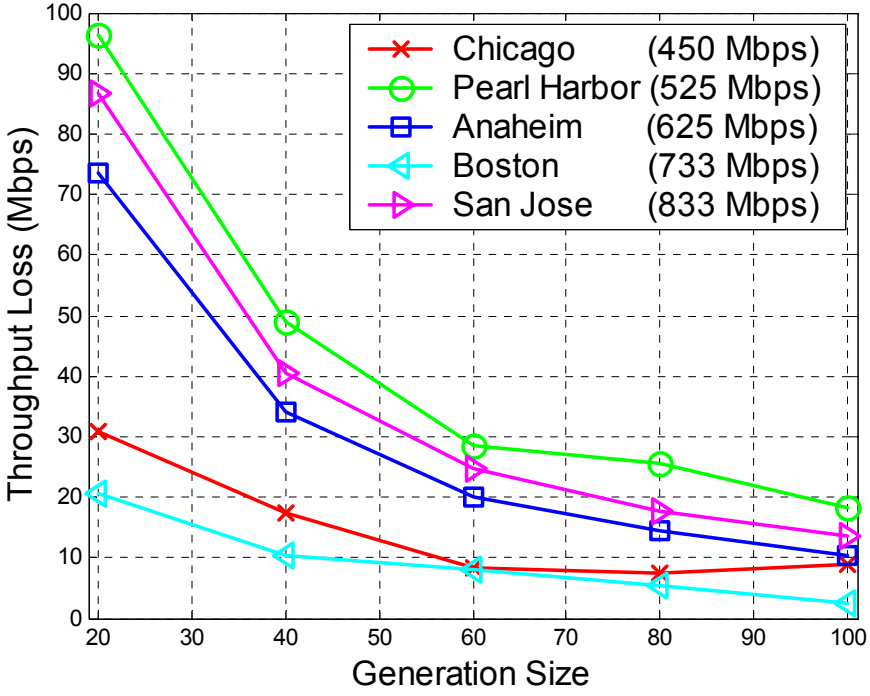


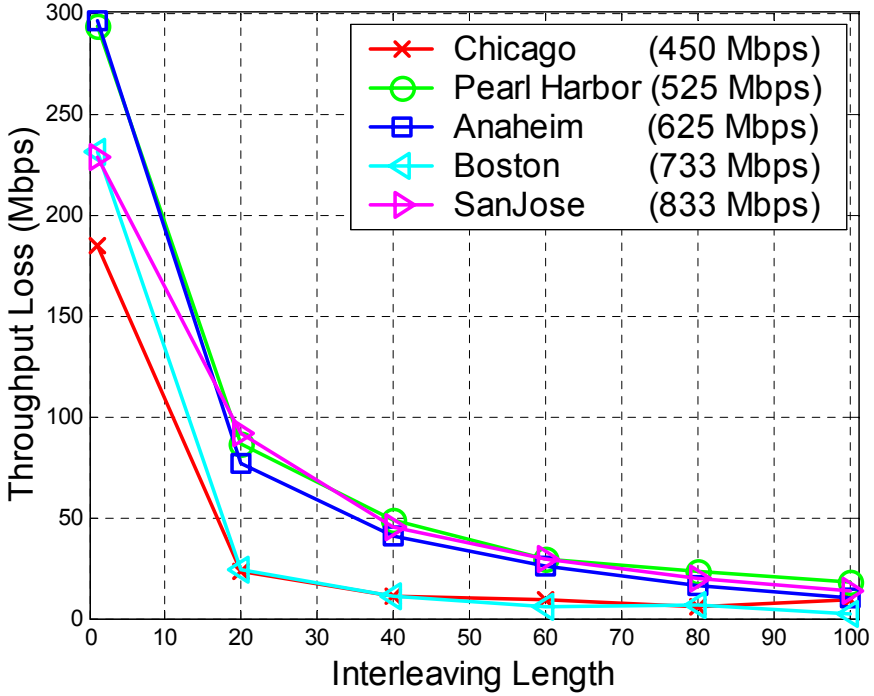
Throughput


Chicago (450 Mbps)
Pearl Harbor (525 Mbps)
Anaheim (625 Mbps)
Boston (733 Mbps)
SanJose (833 Mbps)
Throughput (Mbps)
Sending rate (Mbps)
400
450
500
550
600
650
700
750
800
850

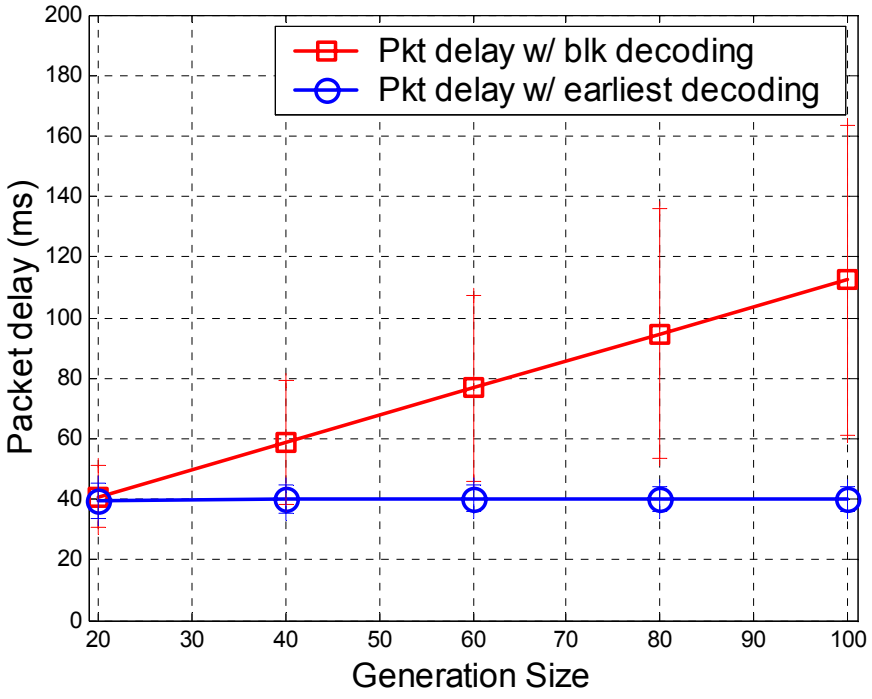

Chicago (450 Mbps)
Pearl Harbor (525 Mbps)
Anaheim (625 Mbps)
Boston (733 Mbps)
SanJose (833 Mbps)
Throughput (Mbps)
Sending rate (Mbps)
400
450
500
550
600
650
700
750
800
850

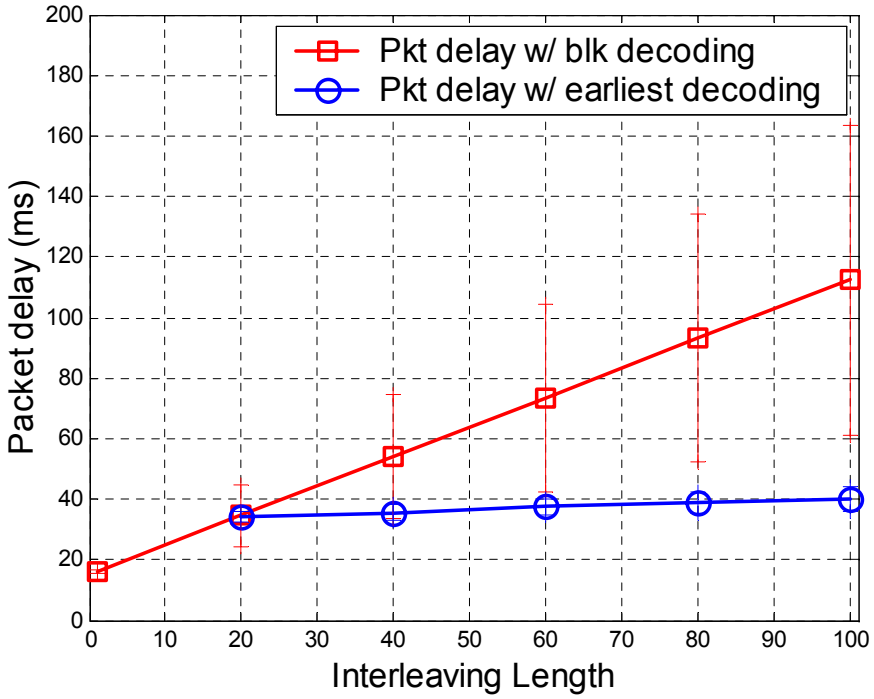
Throughput Loss



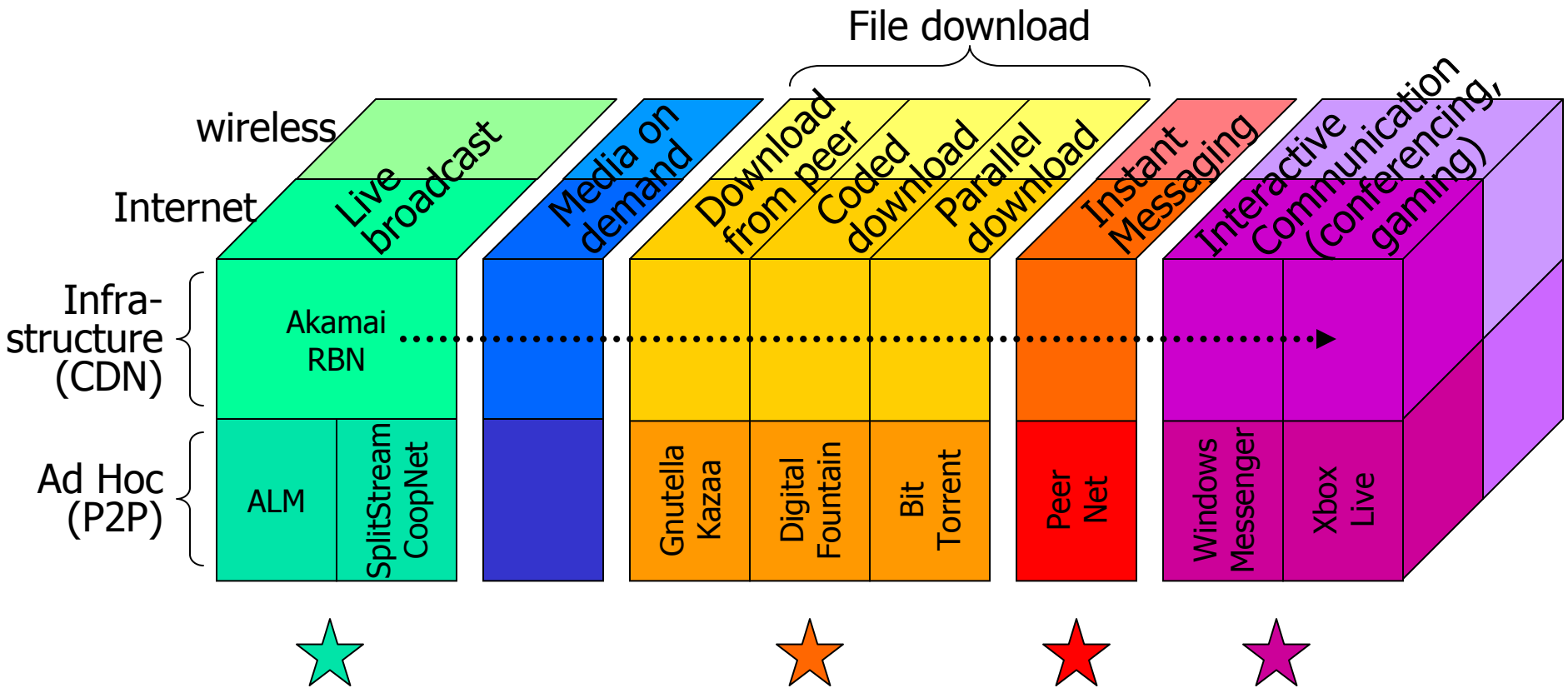


Decoding Delay



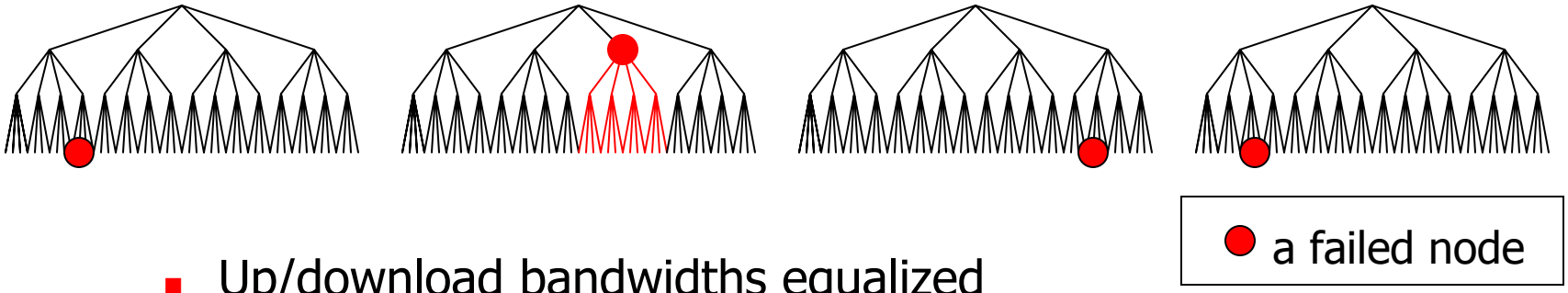


Network Coding for Internet and Wireless Applications

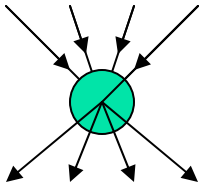


Live Broadcast (1/2)

- State-of-the-art: Application Layer Multicast (ALM) trees with disjoint edges (e.g., CoopNet)
  - FEC/MDC striped across trees

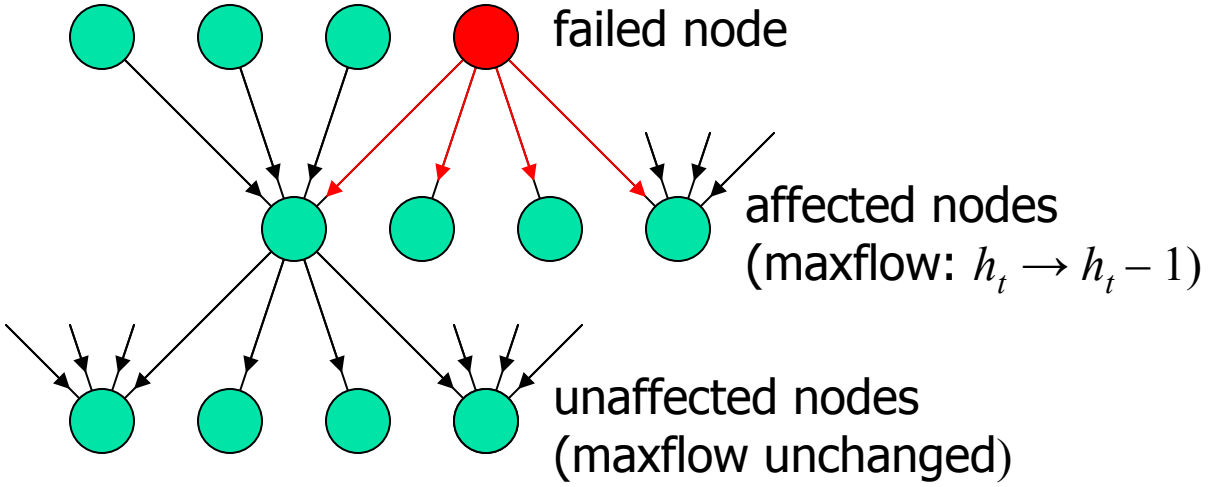


  - Up/download bandwidths equalized



Live Broadcast (2/2)

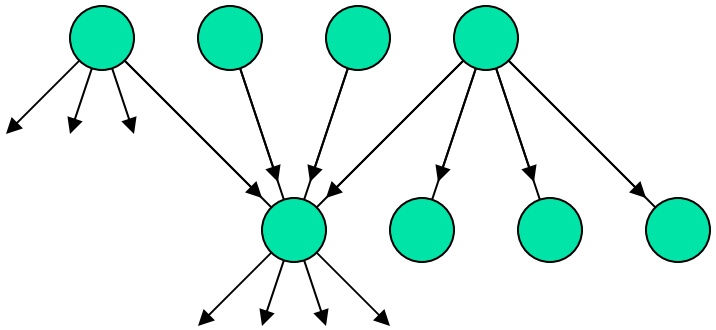
- Network Coding [Jain, Lovász, Chou (2004)]:
  - Does not propagate losses/failures beyond child



  - ALM/CoopNet average throughput: $(1-\varepsilon)^{\text{depth}}$ * sending rate
    Network Coding average throughput: $(1-\varepsilon)$ * sending rate

File Download

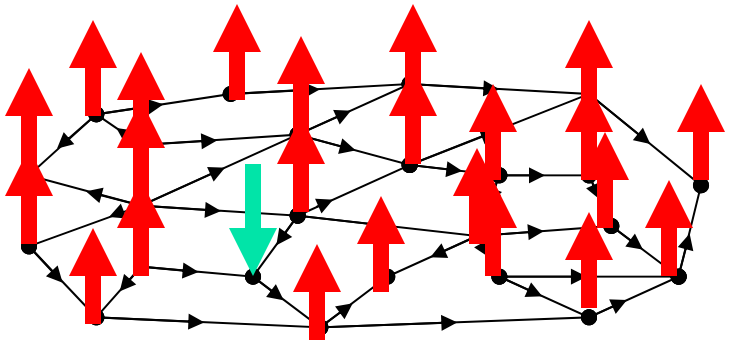
- State-of-the-Art: Parallel download (e.g., BitTorrent)
  - Selects parents at random
  - Reconciles working sets
  - Flash crowds stressful



- Network Coding:
  - Does not need to reconcile working sets
  - Handles flash crowds similarly to live broadcast
    - Throughput ⟷ download time
  - Seamlessly transitions from broadcast to download mode

Instant Messaging

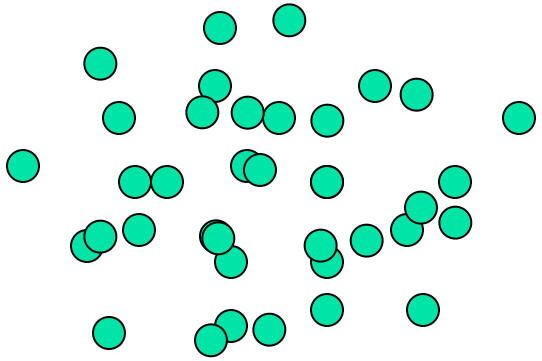
- State-of-the-Art: Flooding (e.g., PeerNet)
  - Peer Name Resolution Protocol (distributed hash table)
  - Maintains group as graph with 3-7 neighbors per node
  - Messaging service: push down at source, pops up at receivers
  - How? Flooding
    - Adaptive, reliable
    - 3-7x over-use



- Network Coding:
  - Improves network usage 3-7x (since all packets informative)
  - Scales naturally from short message to long flows

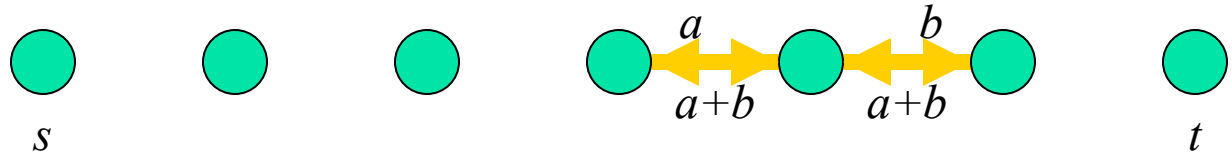
Interactive Communication in mobile ad hoc wireless networks

- State-of-the-Art: Route discovery and maintenance
  - Timeliness, reliability

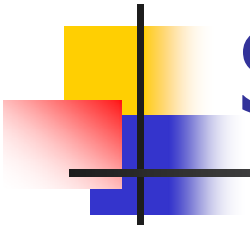


- Network Coding:
  - Is as distributed, robust, and adaptive as flooding
    - Each node becomes collector and beacon of information
  - Minimizes delay without having to find minimum delay route

Physical Piggybacking



- Information sent from t to s can be piggybacked on information sent from s to t
- Network coding helps even with point-to-point interactive communication
  - throughput
  - energy per bit
  - delay



Summary

- Network Coding is Practical
  - Packetization
  - Buffering
- Network Coding can improve performance
  - in IP or wireless networks
  - in infrastructure-based or P2P networks
  - for live broadcast, file download, messaging, interactive communication
  - by improving throughput, robustness, delay, manageability, energy consumption
  - even if all nodes are receivers, even for point-to-point communication