An Overview of Content Protection Techniques using Network Coding

Muriel Medard

Professor

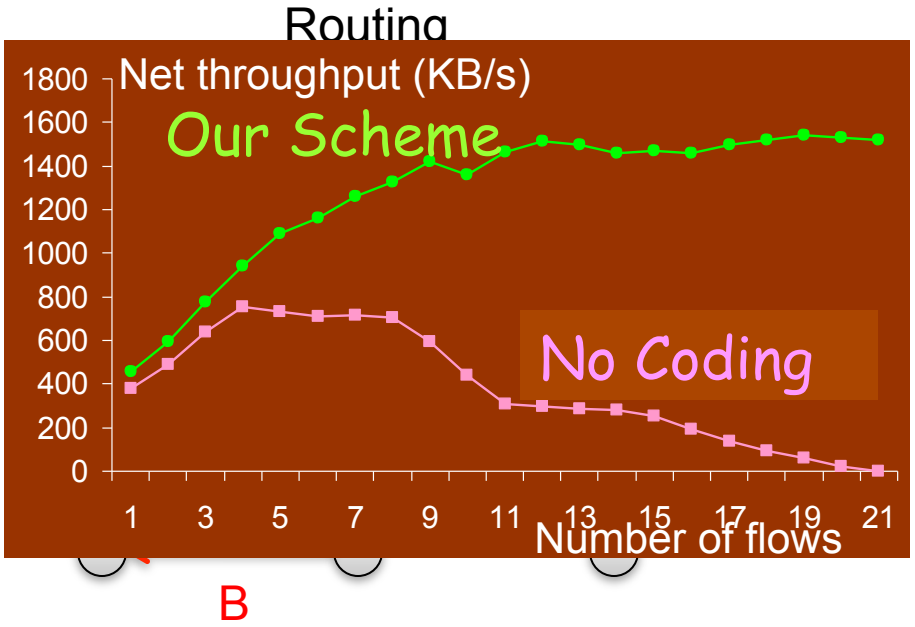
EECS

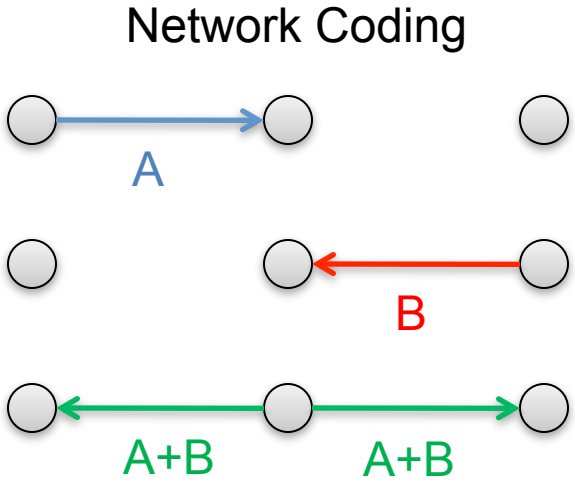
Collaborators

- MIT: Dina Katabi, Sachin Katti (now Stanford), Minji Kim, Marie-Jose Montpetit
- Technical University of Munich: Ralf Koetter
- Caltech: Michelle Effros, Tracey Ho (formerly MIT)
- Chinese University of Hong Kong: Sid Jaggi (formerly MIT)
- AFRL: Keesook Han
- HP Labs: Ton Kalker (now Huawei)
- Open University Israel: Michael Langberg
- University of Porto: Joao Barros, Luisa Lima, Paulo Oliveira, Tiago Vinhoza
- SMART: Fang Zhao (formerly MIT)
- Telefonica: Steluta Gheorghiu, Alberto Toledo

Network Coding

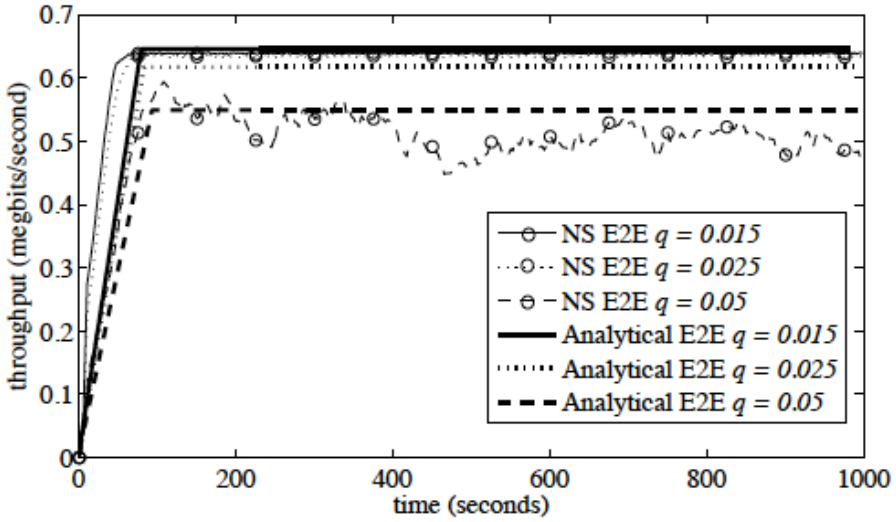
- A new means of conveying information
  - Throughput gains, robustness against failures and erasures
  - It also changes completely the security aspects of networks
- Routing/traditional network: similar to a transportation network
- Network coding: data is an algebraic entity



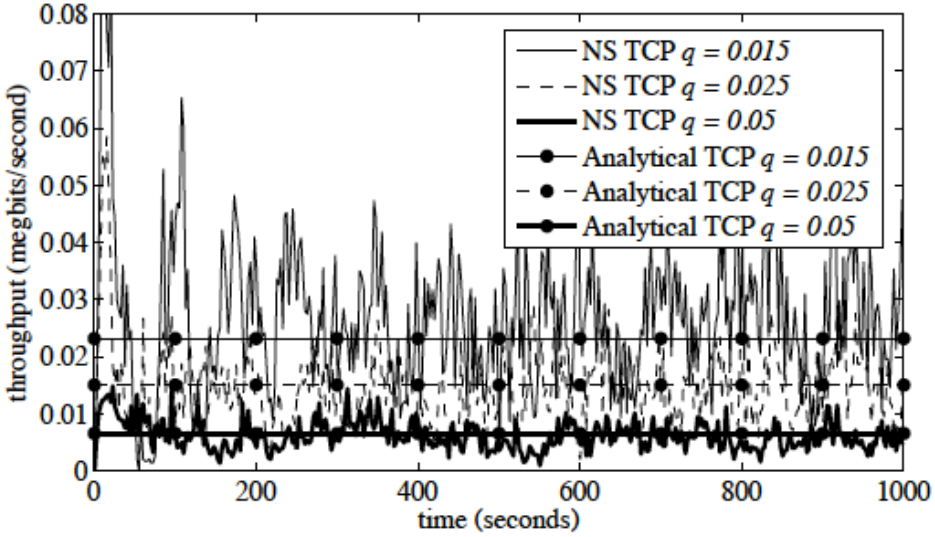


Performance Comparison

- Significantly reduces loss effects
  - Is compatible with TCP/IP
  - Based on random linear network coding (RLNC)



(a) E2E-TCP/NC



(b) TCP

[Kim et al 2010]

Content distribution with RLNC

Random Network Coding used to improve video download times:

"6 minutes → 3 seconds"[1]

Random Network Coding used for software distribution (Avalanche):

"Hours, instead of days"[2]

[1] Presented verbally by Baochun Li at InfoCom 2010

[2] Chris Craft Blog September 2007

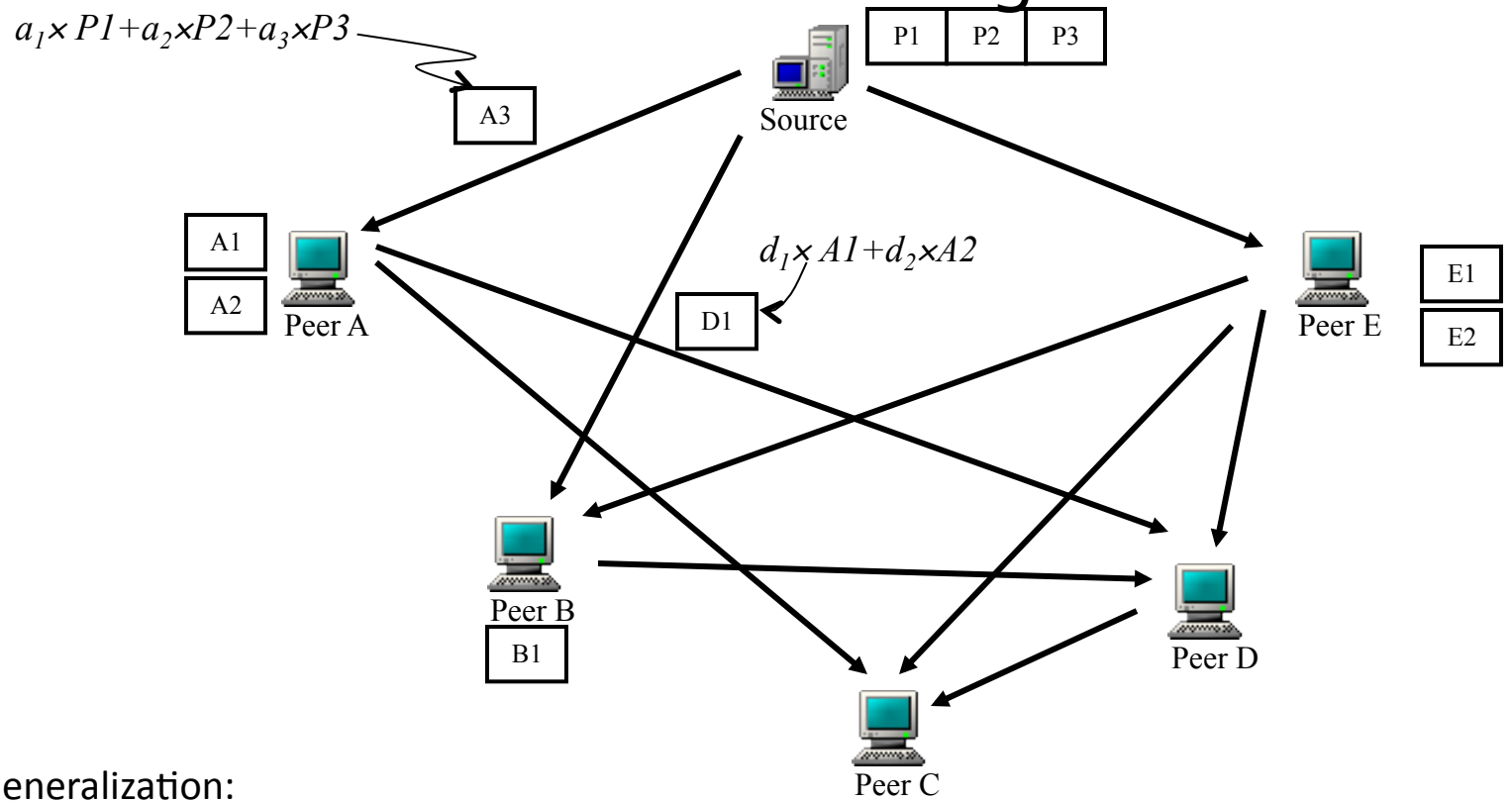
Content distribution with P2P

- Distribution of large files to many users.
- Traditional solutions are based on a client-server model.
- Alternative technique - P2P swamping.
- Example - BitTorrent
  - Divide file into many pieces.
  - Client requests different pieces from server(s) or other users.
  - Client becomes server to pieces downloaded.
  - When a client has obtained all pieces, re-construct the whole file.
  - Problem: hard to do optimal scheduling of pieces to nodes.

Content distribution using network coding

- Use network coding to increase the efficiency of network coding in a P2P cooperative architecture.
- Instead of storing pieces on servers, store random linear combination of the pieces on servers.
- Clients also generate random linear combination of the pieces they have received to send out.
- When a client has accumulated enough degrees of freedom, decode to obtain the whole file.

Content distribution using network coding



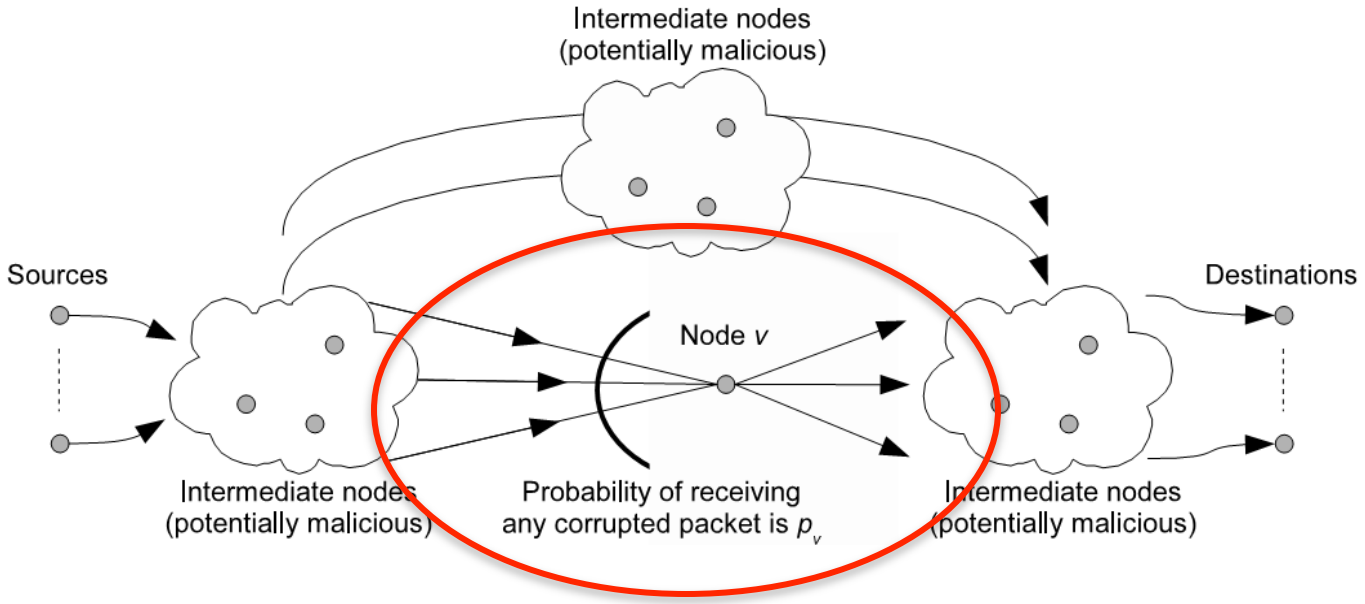
- Generalization:
  - Some pieces are uncoded and some are coded
  - Example:
    - Suppose 10 % bandwidth deficit in reaching back to server for interruption-free service
    - If we cache locally 10 % of content uncoded, it is only providing 1% acceleration
    - If we cache 10% coded, we get full interruption-free service

First issue – Byzantine attacks

- Combining symbols in the network may complicate keeping track of information to ensure its integrity or provenance
  - One corrupted packet, if mixed with others, can pollute the entire flow
  - Packets are no longer individually recognizable

- **Detection:**
  - Generation-based Byzantine detection (Batch)
  - Packet-based Byzantine detection scheme (Signature)
  - Watchdog (wireless)
- **Correction:**
  - End-to-end network error correction
  - Eliminate corrupted packets with erasure correction

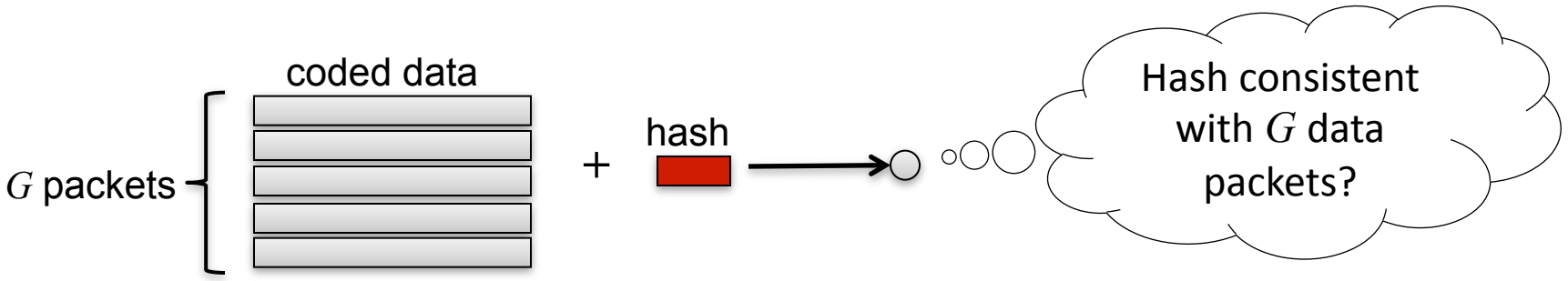
Byzantine detection with Network Coding



- General network
- Node v: non-malicious, receives m packets (n bits each) per unit time.

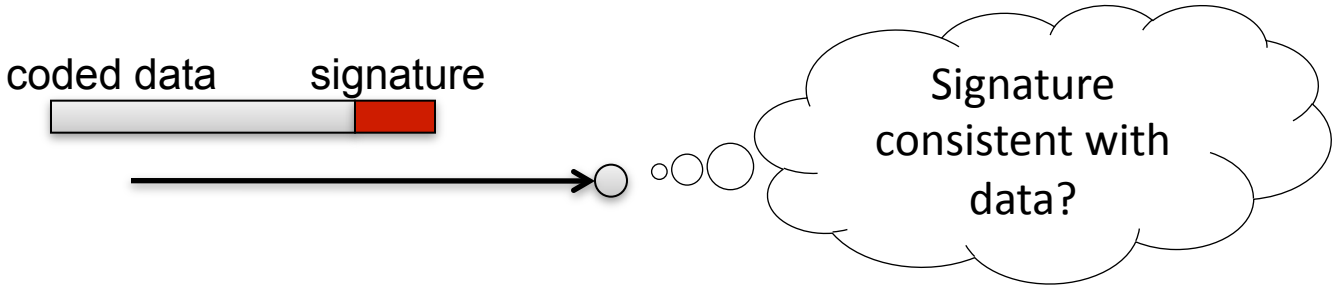
Generation-based detection scheme

- [Ho et al. '04] Information-theoretic approach to detect Byzantine adversaries
- **IDEA:**
  - Data and hash symbols must be consistent with its coding vector
  - Node v checks for error on a generation.
    - If error, then discards the *entire generation* of G packets; otherwise, it forwards the data.
  - Can extend to a *local* Byzantine detection scheme.
  - Overhead: ~2% (for detection probability ≥ 98.9%.)



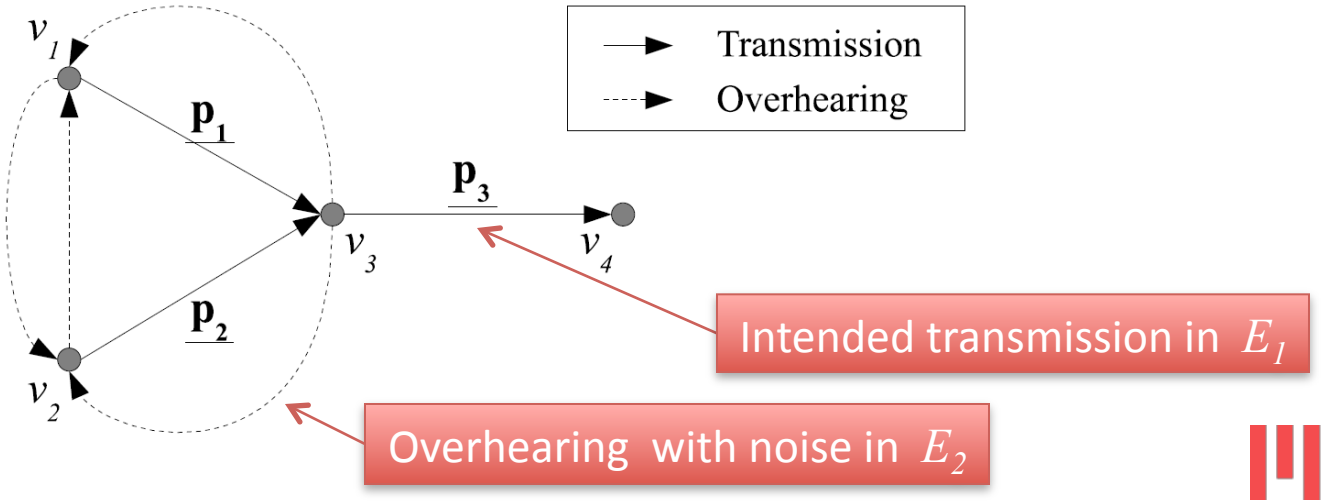
Packet-based detection scheme

- [Zhao et al. '07] Signature scheme for linear network coding.
- **IDEA:**
  - **Without decoding** can validate **each packet**: belongs to a specified subspace?
  - Add signature to help check membership to the subspace
  - Node v checks the validity of every packet using public key $\mathbf{K}$
  - Overhead:
    - Initialization: ~6% (setting up signature and public key $\mathbf{K}$)
    - Subsequent files: <1% (incremental updates)
  - When attack high, then, "bandwidth saved" > "cost of detection"



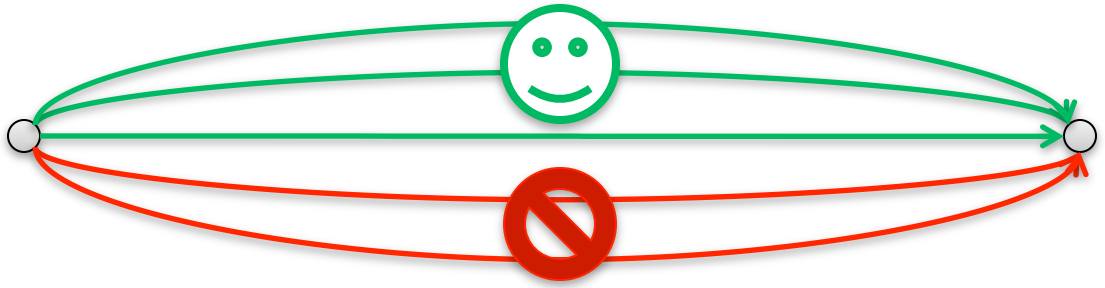
Watchdog: Self-checking network

- Combine benefits of network coding with Watchdog
  - Wireless setting: Use overhearing
  - Probabilistically police downstream neighbors
- Challenges:
  - Network Coding: packets mixed unlike routing
  - Errors from channel and attacker: probabilistic detection



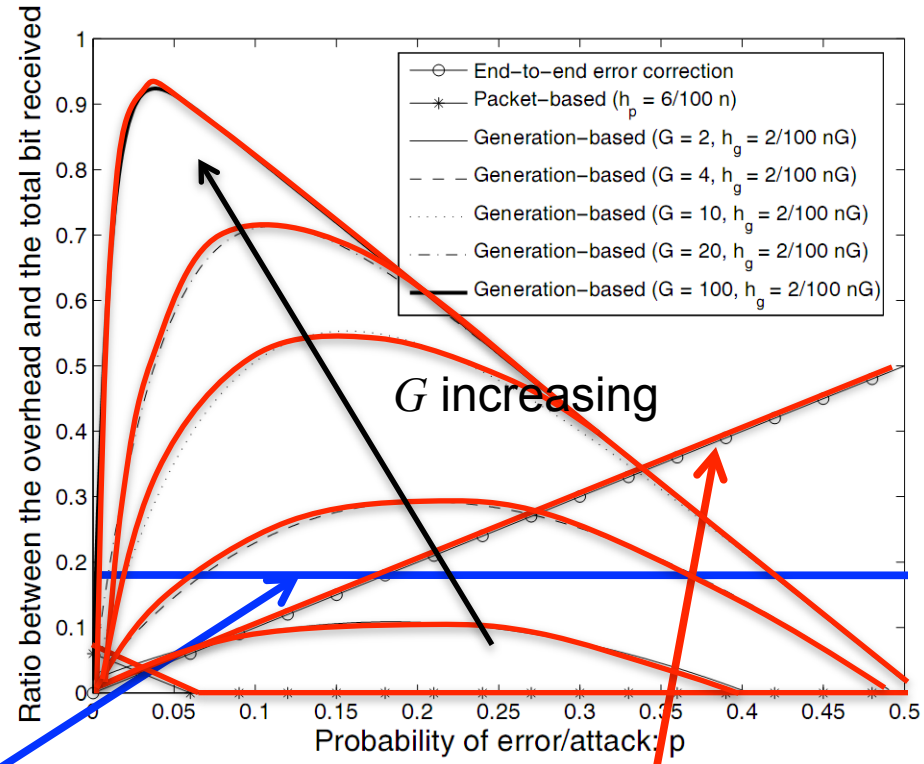
End-to-end network error correction

- [Jaggi et al. '07] offers distributed, polynomial-time, rate-optimal network codes that are information-theoretically secure against Byzantine attacks.
- **IDEA:**
  - Byzantine adversaries = secondary sources. Adds redundancy to distinguish the packets.
  - Node v does not check for attacks, and naively performs network coding.
  - Transmits at the remaining network capacity.
  - Error correction at destinations (more expensive than erasure correction).
  - Expected ratio of corrupted bits transmitted and total bits received is: p.



Overhead of Byzantine detection

- Cost of error correction scheme
  - Linear with attack = $O(p)$.
  - When attack low ($p<0.03$): cost of detection >> cost introduced by the attacker.
- Cost of packet-based scheme
  - Small p : costly hashes
  - Large p : the hashes become "cheaper".
  - Infrastructure needed (PKI).
- Cost of generation-based scheme:
  - When attack low $(p<<0.2)$, cost effective, hash across G packets.
  - When attack high $(p>0.2)$, many invalid generations.
  - Balance between generation size (throughput) and probability of detection.



Ratio between the expected overhead and the total bits received by a node v with $h_p \approx 0.06n$ and $h_g \approx 0.02nG$

Secure routing protocols for uncoded systems (especially for wireless ad hoc networks) has on average 24% overhead [Marti et al. '00].

■ Coded systems need authentication as well; but also benefit from the throughput gain.

At the very best, the uncoded system will achieve this (assuming no losses in the channel).

■ In a non-coded system, overhead is equal to probability of attack.

■ Coding gives throughput gains as well as robustness against erasures.

Second issue - eavesdropping

- A conventional approach will encrypt packets and encode on those encoded packets
- We have introduced a new approach which consists of **protecting the code itself**
  - It is still possible to perform combinations on top of encrypted codes, by keeping the collective effect of the successive encodings upon the encrypted codes
  - We can model the encryption of the codes as saying that the encoding matrices, based upon variants of random linear network coding, are given only to the source and sinks
- We can use coding to make secure use of untrusted servers with very strong information-theoretic guarantees

Different flavors

- Lock coefficients (vanilla)
  - Simple multi-layer system
  - Suffers from known plaintext-attack

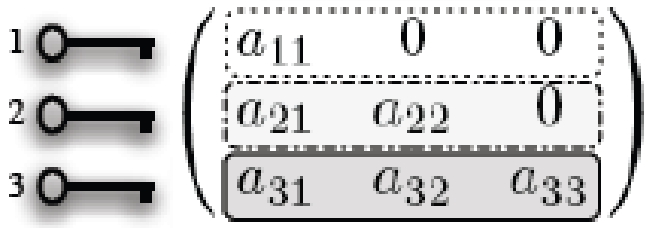
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

- Strawberry flavor:
  - Prevent plaintext attacks by encrypting one symbol per use of the encoding matrix

$$\begin{cases} a_{11}\ \mathbf{E}(\mathbf{b_1}) + 0 + 0 = \gamma_1 \\ a_{21}\ \mathbf{E}(\mathbf{b_1}) + a_{22} b_2 + 0 = \gamma_2 \\ a_{31}\ \mathbf{E}(\mathbf{b_1}) + a_{32} b_2 + a_{33} b_3 = \gamma_3 \end{cases}$$

Chocolate flavor

- Different keys for different layers

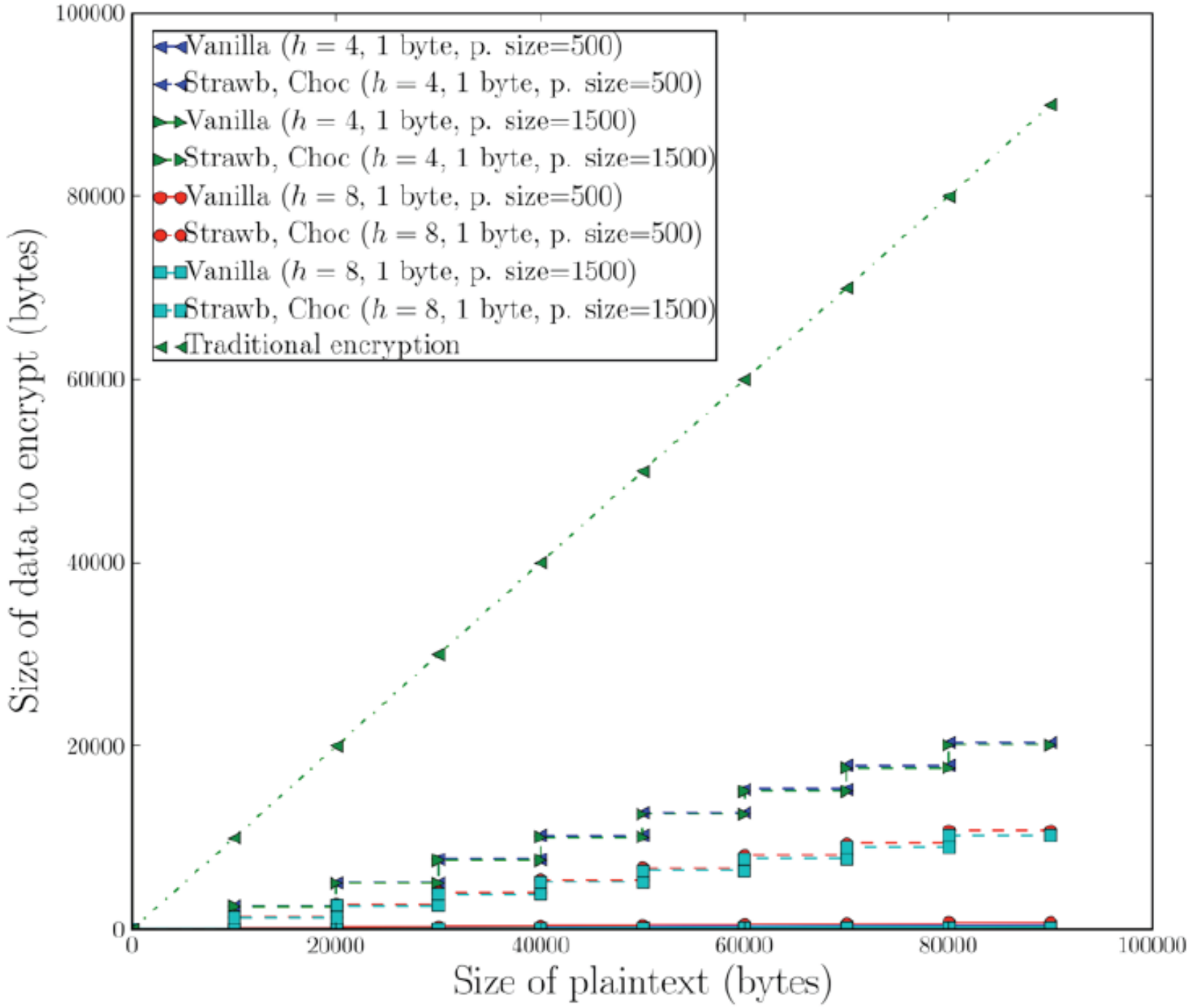


  - Example: protecting all layers

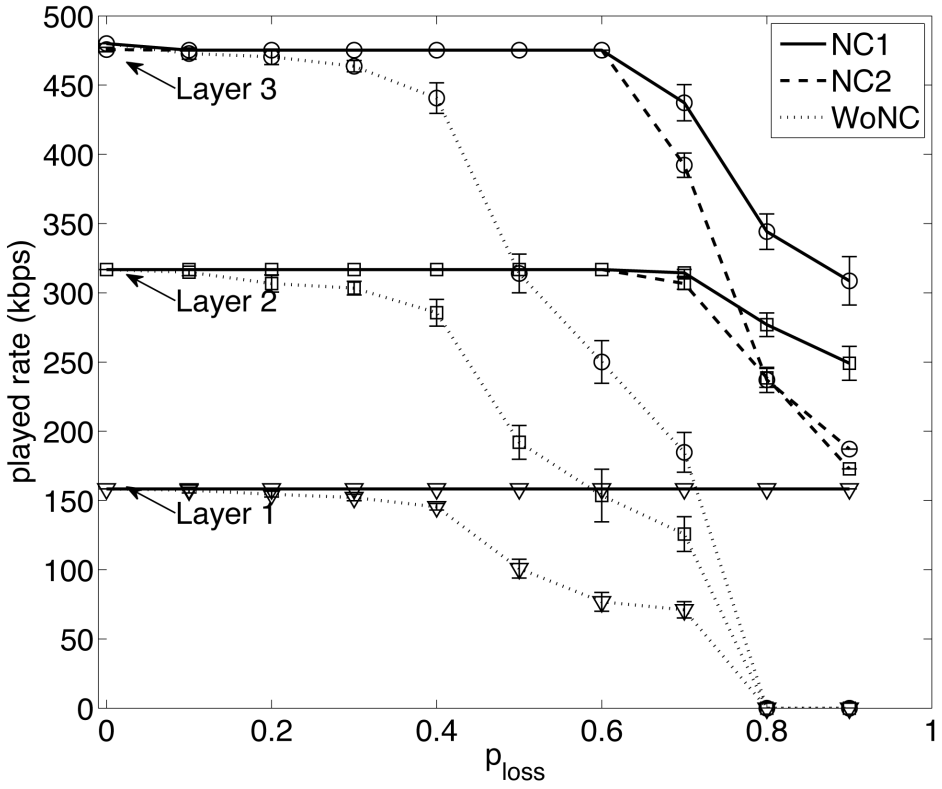
$$\begin{cases} a_{11}\ \mathbf{E}(\mathbf{b_1}, \mathbf{K_1}) + 0 + 0 = \gamma_1 \\ a_{21}\ \mathbf{E}(\mathbf{b_1}, \mathbf{K_1}) + a_{22} b_2 + 0 = \gamma_2 \\ a_{31}\ \mathbf{E}(\mathbf{b_1}, \mathbf{K_1}) + a_{32} b_2 + a_{33} b_3 = \gamma_3 \end{cases}$$

  - Example: protecting layers 2 and above

$$\begin{cases} a_{11} b_1 + 0 + 0 = \gamma_1 \\ a_{21} b_1 + a_{22}\mathbf{E}(\mathbf{b_2}, \mathbf{K_2}) + 0 = \gamma_2 \\ a_{31} b_1 + a_{32}\mathbf{E}(\mathbf{b_2}, \mathbf{K_2}) + a_{33} b_3 = \gamma_3 \end{cases}$$


Vanilla ($h = 4$, 1 byte, p. size=500)
Strawb, Choc ($h = 4$, 1 byte, p. size=500)
Vanilla ($h = 4$, 1 byte, p. size=1500)
Strawb, Choc ($h = 4$, 1 byte, p. size=1500)
Vanilla ($h = 8$, 1 byte, p. size=500)
Strawb, Choc ($h = 8$, 1 byte, p. size=500)
Vanilla ($h = 8$, 1 byte, p. size=1500)
Strawb, Choc ($h = 8$, 1 byte, p. size=1500)
Traditional encryption
Size of data to encrypt (bytes)
Size of plaintext (bytes)
0
20000
40000
60000
80000
100000

Multilayer Codecs and NC 1 layer at a time

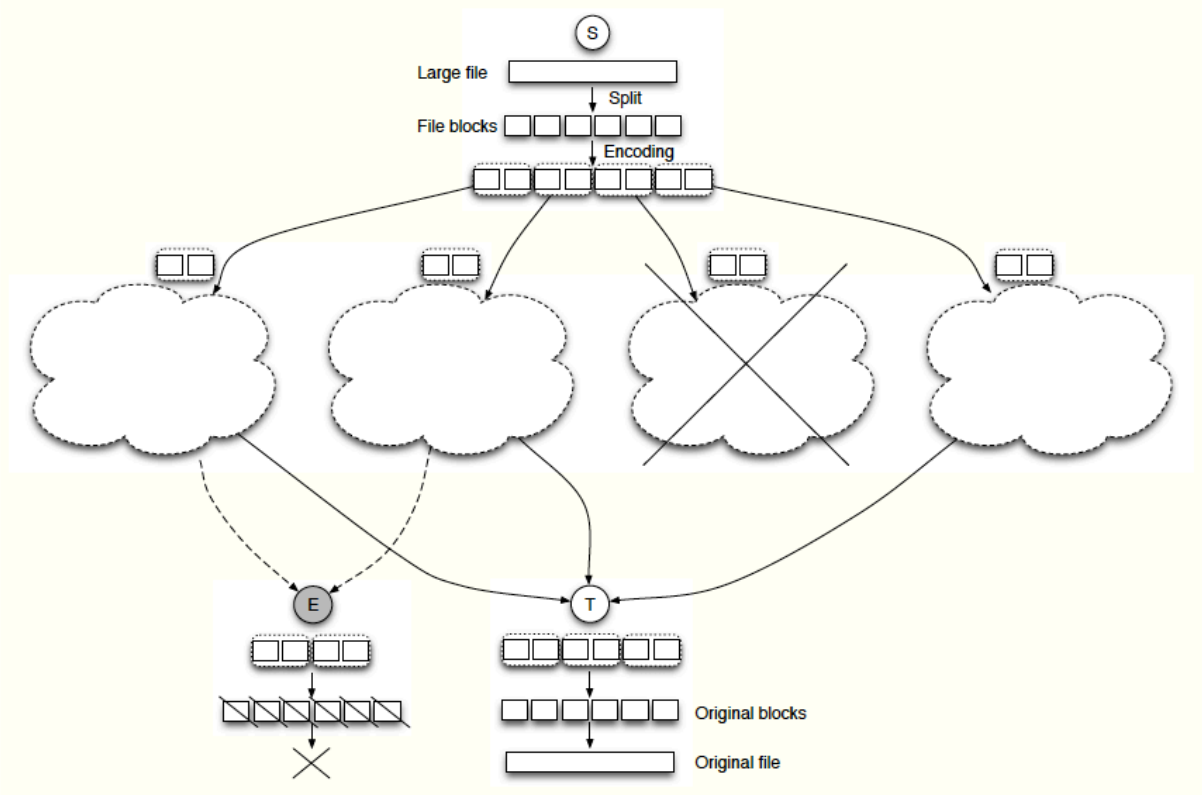


Trusted Storage over Untrusted Networks

A large file is coded and spread over four networks

Our coding scheme allows legitimate users with access to any three networks for perfect reconstruction of the original data

An eavesdropper with access to two networks of its choice is unable to decode any original symbol, even if it is able to guess some of the missing blocks



Conclusions

- Network coding opens new venues of low overhead protection of content against pollution and eavesdropping
  - Detection of attacks
  - Correction of attacks
  - Lightweight encryption
  - Secure distributed storage
- Coding in the context of content protection allows for new distributed approaches
- The ability of attackers to modify and inspect content is significantly impacted by the use of coding