Reliable Network Coding/Privacy in Microblogging

Srikanth V. Krishnamurthy

UC Riverside
Progress from last meeting

○ Work on Network Coding (with T. La Porta)
  ○ Papers in Infocom 2010, ICNP 2010

○ Work on Jamming
  ○ Paper in ACM CoNext 2009
  ○ Paper in IEEE Transactions on Wireless Communications.

○ Work on reliability with MIMO diversity (with S. Kasera)
  ○ Paper in IEEE ICDCS 2010

○ Trust in wireless networks (part funded by CTA; with F. Wu)

○ Preserving Privacy in Microblogging

○ Coping with attacks
  ○ Replay attacks (with M. Faloutsos)
  ○ Route attraction attacks (with P. Krishnamurthy, P. Mohapatra)
  ○ Exploitation of Carrier Sensing (publication in Infocom 2009)
A Framework for Joint Network Coding and Transmission Rate Control in Wireless Networks

Tae-Suk Kim*, Serdar Vural*, Ioannis Broustis*, Dimitris Syrivelis‡, Srikanth V. Krishnamurthy*, and Thomas F. La Porta§

*University of California, Riverside, ‡University of Thessaly, § Penn State University
Network coding is a technique that can potentially increase transport capacity of wireless networks. Conventional network coding schemes do not consider the effect of using diverse transmission rates. Higher transmission rates can improve the link-level throughput, but can degrade the encoding capacity by reducing packet overhearing probabilities.
Goal

To maximize network throughput by achieving the best trade-off between two contradictory goals:

- To use higher transmission rates for improving link level throughputs
- To ensure effective overhearing at receivers to preserve high encoding gain
Network coding with COPE*

- Encode packets at routers into a single packet to make a single transmission
  - 3 transmissions instead of 4
- Encoding function: XOR
- Based on these functionalities:
  - Packet overhearing (packet pools)
  - “Probe packets” for link quality estimation
  - Periodic “Reception Reports” for native packets received at receivers
  - A fixed transmission rate at all nodes

Our Approach

For transmission of native packets:
- Choose rate to maximize throughput to the router
- Consider overhearing probabilities

For transmission of encoded packets:
- Choose rate to maximize total throughput at receivers
- Properly choose the primary receiver (ACKer) of the encoded packet
Notations

- Transmission time at rate $r$ of packet of length $L$: $T_{L}^{r}$
- Probability of overhearing the transmission of rate $r$ from $x$ to $y$ by $z$: $P_{\{x,y\},z}^{r}$
- Number of transmissions from $x$ to $y$ at rate $r$: $N_{x,y}^{r}$
- Rate of transmission at node $x$: $R_{x}$
- Packet length of node $x$: $L_{x}$
Select a rate to maximize throughput to Jack:

\[
\max_{R_{Alice} \in R} \frac{L_{Alice}}{N^{R_{Alice}}_{Alice,Jack} \cdot T^{R_{Alice}}_{L_{Alice}}}
\]

Constrained by overhearing probabilities at common neighbors of Alice and Jack:

\[
s.t. \quad P^{R_{Alice}}_{\{Alice,Jack\},Dave} \geq \beta
\]

\[
P^{R_{Bob}}_{\{Alice,Jack\},Bob} \geq \beta
\]

R: Set of transmission rates
(e.g. R: {6, 9, 12, 18, 24, 36, 48, 54} Mbps at 802.11a)
Example (ACKer Selection)

- **Perfect overhearing**
- **Choice 1: ACKer is Chloe:**
  - $1 / 0.1 = 10$ expected retransmissions before receiving an ACK for $A \oplus B$
  - Total packets: 2
  - Expected throughput:
    - $2/10 = 0.2$
- **Choice 2: ACKer is Dave:**
  - $1/0.8 = 1.25$ expected retransmissions before receiving an ACK for $A \oplus B$
  - Total packets: 2
  - Expected throughput:
    - $2/1.25 = 1.6$
ACKer Selection Module

- Jack selects one of the next hops of the encoded packet as the primary receiver (ACKer) node.
- Maximize the throughput by considering all next hops as the ACKer over all transmission ranges:

\[
\max_{\{\text{Chloe, Dave}\}} \frac{L'_t}{D'_t}
\]

- Jack unicasts encoded packet to the ACKer:
  - Retransmits until ACK is received
  - Other next hops receive the packet by overhearing
  - \( P_{\text{success}} \): Probability of successful delivery to ACKer

Example: Dave is ACKer, then:

\[
L'_t = P_{\text{Success}} \cdot P_{\{\text{Alice, Jack}\}, \text{Dave}} \cdot L_{\text{Bob}} + P_{\{\text{Jack, Dave}\}, \text{Chloe}} \cdot P_{\{\text{Bob, Jack}\}, \text{Chloe}} \cdot L_{\text{Alice}}
\]

\[
D'_t = N_{\text{Jack, ACKer}}^{R_{\text{Jack}}} \cdot T_{\max(L_{\text{Alice}}, L_{\text{Bob}})}^{R_{\text{Jack}}}
\]
Experiments

- Click Router v.1.4.2 (as in COPE)
- Madwifi-2005 wireless driver
- 802.11b (4 bit rates: 1, 2, 5.5, 11 Mbps)
- Our scheme on top of COPE
  - COPE operates by default at 1 Mbps
- Probing mechanism of Roofnet routing protocol (SRCR)
- Two topologies:
  - X-Topology
  - Cross Topology
Both indoor and outdoor links
- Soekris net5501 nodes
- Debian Linux distribution
- Kernel v2.6.16.19 over NFS
- 500 MHz CPU, 512 Mbytes of memory
- WN-CM9 wireless mini-PCI card
- AR5213 Atheros as main chip
- 5dBi omnidirectional antenna
- Transmission power set to 10 dBm
- RTS/CTS disabled

Sample topology for experiments
Gain in Throughput wrt COPE

- Significant improvement in throughput over COPE

Ratio of encoded packets at router
Scenario 1

- Good-channel quality links (PDR of links are 70% or above)
- Up to 250% improvement
- Our scheme efficiently exploits good channel conditions by utilizing higher transmission rates
- Our scheme does not hurt encoding gain while using higher transmission rates

Scenario 2

- Good channel quality links (PDR of links are 70% or above)
- Bi-directional traffic flows
  - Can encode up to 4 packets
- Up to 272% improvement
- We can obtain 20% higher throughput than X topology since higher encoding opportunities occur with 4 traffic flows.
### Scenario 3
- **Poor quality links:**
  - < Jack to Chloe >
  - < Chloe to Dave >
- Up to 189% improvement
- To increase probability of reception by Chloe, Bob uses lower transmission rates compared to Scenarios 1&2: Less gain is obtained

### Scenario 4
- **Poor quality links:**
  - < Chloe to Jack >
  - < Chloe to Dave >
- Up to 150% improvement
- Both Alice and Bob use lower transmission rates to increase overhearing probabilities. Hence, throughput gain is lower than Scenario 3
Network Simulator 2 (ns2)
802.11a (8 bit rates: 6, 9, 12, 18, 24, 36, 48, 54 MBps)
Performance results of the following schemes are compared:
- COPE (basic rate)
- COPE + rate adaptation
- Our scheme with only ACKer Selection
- Our scheme with both ACKer Selection and Rate Selection
Small-Scale Topologies

- **PDR<Jack to Dave> = 1**
- **PDR<Jack to Chloe>: varied**
  - With this topology, up to 390% improvement is obtained over COPE
  - ACKer selection is important when link qualities to receivers are diverse
  - Rate Selection is important when link qualities are similar
  - Gain in throughput: 75% over COPE, 30% over COPE + Rate Adaptation
Dense "Wheel" Topologies

Encoding Ratio = \frac{\text{Total number of encoded packets sent}}{\text{Total number of packets sent}}

COPE + Rate
Adaptation is coding unaware

Slightly less than COPE
Similar to COPE

\sim \text{Half of COPE}
Larger-scale Multihop Settings

- 1000 x 1000 m²
- Random node locations
- Randomly selected source-destination pairs
- Paths established by DSR
- Fully-saturated UDP flows

Less Interference
- Higher transmission rates possible
- Delay is reduced mostly by transmission rate increase

Higher interference
- Higher and more diverse packet loss rates
- ACKer selection is predominant factor in delay reduction
Conclusions

- Performance gain of our framework in throughput with network coding as much as 390% compared to COPE
- A coding-unaware rate adaptation scheme degrades coding gain and achievable throughput
- Our scheme conserves the coding gain of COPE even with higher transmission rates
- Routers can boost throughput performance by intelligently choosing the recipient of the encoded packets
Preserving Privacy in Microblogging

Indrajeet Singh, Michael Butciewicz, Harsha V. Madhyastha and Srikanth V. Krishnamurthy

University of California, Riverside,
Introduction

- In a tactical setting it is important to distribute messages towards allowing people to see messages in a *selective* fashion.

- As an example, a commander may seek to communicate with a group of selective users.

- A blogging system can be useful for real-time communications – but needs to be adapted to the scenario under consideration.
Architectural possibilities

- Conceptually there can be a Single Server having all the users’ data (needs to be secured)
  - Can be implemented as a distributed system eg NFS, Twitter

- Every user has unique credentials using which the system allows access.

- Users have “sensitive” information i.e., information that the user would like to selectively pass on.
The System Under Consideration

- We want to take a System that is
  - Leverage existing possibilities
  - Popular i.e. has a large user base.
  - Open to developers

- For these reasons we consider Twitter, which consists of
  - Small messages
  - Convenient for mobile usage
The Problem

- Twitter has the following associated problems:
  - Location Privacy (geo-tagging)
  - Message Secrecy

- Both these stem from the binary nature of privacy (as implemented in Twitter)

- Therefore, we need to provide the end users a finer grain of privacy control, allowing the user (Jack) to choose who can see specific “posts” (say only Chloe and Tony).
Design goals

- We aim to guarantee the following properties:
  - The System designed should not increase the load on either the client’s side or the servers side drastically which would make it unusable.
  - The System should not interfere or “spoil” the way data is managed by the original system.
  - The general concept of social blogging should remain.
  - Backward compatibility with Twitter
  - Lastly we should not have access to the users passwords, in case our system is compromised.
The System Architecture

User’s Mobile Device
  User Input
  MiddleWare

Micro Blogging Server

Custom Server
Basic Structure

- We aim to have 2 components of as a part of our solution.
  - One runs on the client and
  - The other behaves as an intermediary proxy
    - Only for certain communications between the client and the server/other clients.

- The system (both at the client and the server) should not have access to the end users credentials
  - The client side should still be able to access the users data,
  - We use the OAuth standard
    - Authenticates the user to Twitter which provides the client side with a token.

- The Client’s middleware communicates with the custom server only in certain conditions. There are privacy and efficiency tradeoffs while considering what all the client talks to the custom server
Current methodology

- User creates lists which contain sub-set of private friends to whom the tweet should go to.
- Server takes this information and sends direct (private) messages to each of the listed recipients.
- Clients using our application will see the post as a private “tweet”.
- The recipients can then reply – which is intercepted by the custom server and sent out as private tweets to all the users on the list.
Securing Communications

- Need to secure against server compromise
- A secure hash function to validate data
- Use of a separate channel for key distribution among users
  - Data can then be encrypted to keep it secret from server
  - However, more processing on mobile devices
    - Power implications
Current State and Future Work

- We have a simple multi-threaded Server which communicates to Twitter and to our clients.
  - We currently store no private information on this server, which specifically means message contents or other details.

- Clients right now have basic functionality like talking to the server and Twitter as well.
  - We will also be porting our clients in the future to popular handheld devices.
  - We plan to automate the fetching of new messages to our client, while not causing excessive redundant communication.

- We are currently implementing the additional methods that introduce the added fine-grain privacy control.
To do...

- In such a system, there is always a trade off in performance and privacy. We have to find a soft spot.
  - Perform a worst case analysis
- Quantify the privacy achieved.
  - Formulate metrics
- How to avoid the custom server from becoming a bottleneck in the connection between the client and the server.
Thank you