Epidemic Algorithms for Reliable Multicast in Ad Hoc Networks

Presented By:
Vinita Apte

Outline

- Reliable Multicast Protocols in wired networks
- What about Ad Hoc networks?
- Route Driven Gossip
- Anonymous Gossip

Reliable Multicast Protocols in Wired Networks

- Strong reliability guarantees: all or nothing
  - Poor scalability
- Some practical reliability
  - ACK/NAK mechanism
  - ACK implosion
- Gossip-based protocols
  - Trade-off between reliability and scalability
  - Performance prediction

What about Ad Hoc networks?

- Gossip!
- Underlying network itself does not offer much determinism
  - Nodes are not connected by any fixed infrastructure
  - Communication between two nodes may not be possible always
  - Example: Gossip based ad hoc routing protocol
- Existing deterministic protocols (e.g.: IP multicast) provide no reliability guarantees at all.
- Not suitable when the network topology undergoes frequent changes

Route Driven Gossip

Route Driven Gossip [RDG]

- A protocol for probabilistic reliable multicast in ad hoc networks
- Goal: To achieve probabilistic reliability
  - "If some group member sends out a flow of M packets, a group member receives a fraction m of the M packets with probability $P_M(m)$"
  - m: Reliability Degree
  - P: Reliability Probability Distribution
- The reliability of the protocol given by P should be predictable given information like the packet loss ratio.
- Increasing network size and mobility should degrade the reliability only slightly

RDG…

- Uses a pure gossip scheme
- Does not assume the existence of an underlying multicast primitive
- Built on top of DSR
- Network Model:
  - ‘N’ identical nodes
    - Each with a unique ID
    - Fixed transmission range
    - Bidirectional wireless links to each other
    - Nodes fail only by crashing
  - Multicast group size G
  - CSMA/CA like MAC layer

Network Model

- ‘message’ – information unit
  - Data packets + membership information
- Each packet has a unique pid.
- Missing packets detected by observing gaps in the pid sequence
  - Pid = [group ID, source ID, packet seq number]

Design Characteristics

- View-based gossip unsuitable for ad hoc networks
- Some observations
  - Routing information is precious
  - Route requests are costly
- RDG uses ‘partial views’
- Gossiper-push and gossiper-pull

How it works…

- Basic Data Structures
  - Group ID - gid
  - Data Buffer (*Buffer*): Stores the data packets received.
    - Buffer.new: packets to be gossiped
    - Buffer.old: for gossiper-pulls
  - Active View (*AView*): IDs of known members to which atleast one routing path is known.
  - Passive View (*PView*): IDs of known members to whom no routing path is currently available.
  - Remove View (*RView*): IDs of members who want to leave.

How it works…

- Some Terminology
  - 1> Fanout (F): Number of gossip destinations randomly selected from AView for each gossip emission
  - 2> Quiescence Threshold (τ): A packet will be removed from Buffer.new after it has been gossiped τ times
- Extensions to DSR
  - 1> GROUPREQUEST: Requests multiple routing paths at the same time
  - 2> GROUPREPLY

The Protocol

- Join Session
  - Node floods the network with GROUPREQUEST message and announces its existence
  - On receiving a GROUPREQUEST from a certain member, all members update their AView with the new ID.
  - Return a GROUPREPLY with probability P_{reply}
  - The initiator also updates its AView after receiving the GROUPREPLY.

Join Session

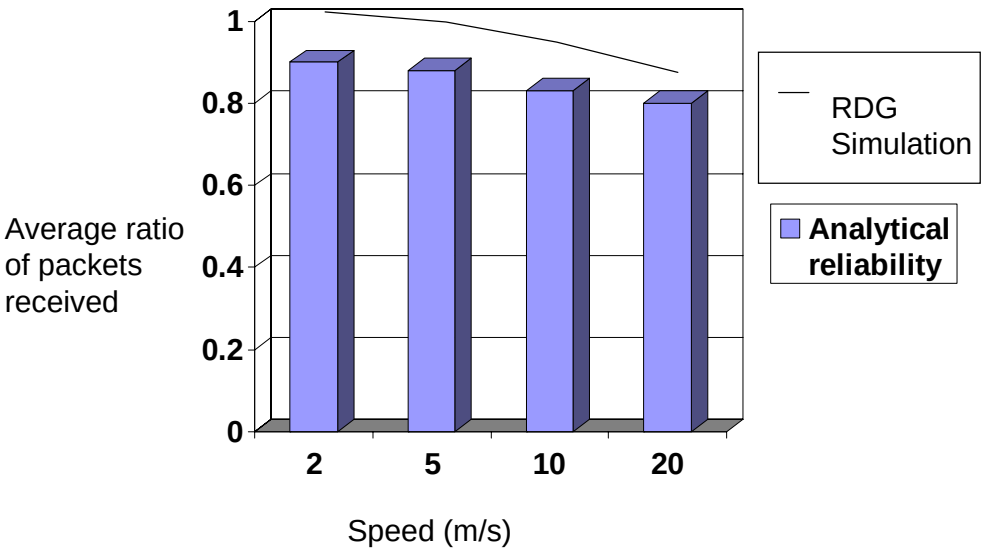
- Route of each incoming packet is recorded
- Each new element in AView has a corresponding entry in the DSR routing table
- Validity checked periodically – AView, PView updated
- If size(AView) < τ_v node has to reinitiate a Join Session

Gossip/Leave Session

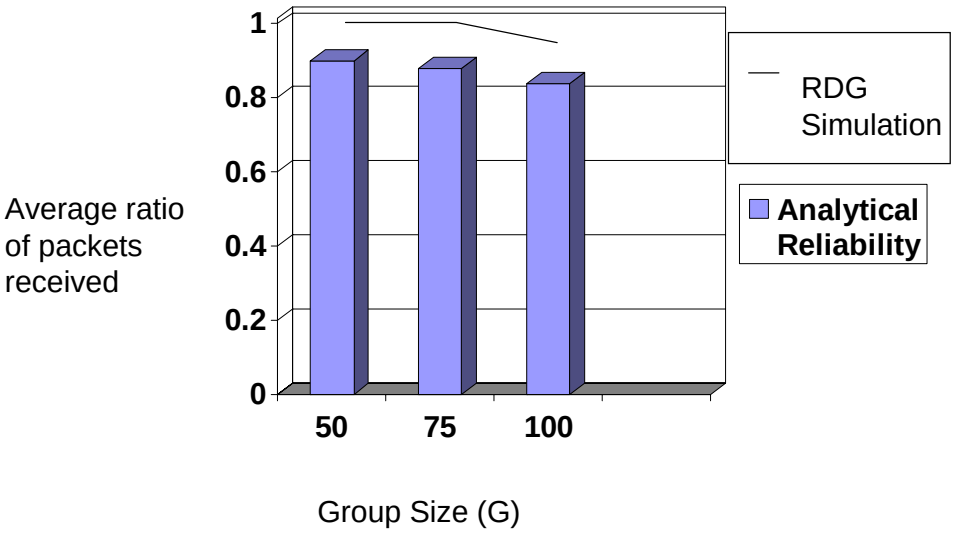
- Each member periodically (every T ms) generates a gossip message and gossips it to F other nodes randomly chosen from AView
- Message includes packets from Buffer.new and the id of the most recent missing packet
- Receiver of a gossip message:
  - Remove obsolete members
  - Add new member
  - Update data buffer
  - Respond to the gossiper-pull

- Nodes along routing paths to gossip destinations belonging to the same group also update their buffers when they see a new packet
- A variant – Topology Aware RDG (TA-RDG)
  - Relies on underlying routing protocol for some topological information
  - Different weights assigned to members in AView
  - Longer the path, lower the weight so that a node chooses a 'near' member with high probability
  - Example: weight = 1/path length

Results: Reliability with G = 50



Results: Reliability with speed = 2 m/s



Anonymous Gossip [AG]

What is Anonymous Gossip?

- A protocol to improve reliability of multicast in ad hoc networks
- A new method of gossip
- Does not require a group member to have knowledge of any other group members
- Can be implemented on top of any tree-based or mesh-based routing protocols
- Protocol used here: MAODV

A slight diversion - MAODV

- Multicast routing protocol
- Dynamically creates and maintains a multicast tree for each group
- Maintains two tables
  - Route Table [RT]

    Next hop for routes to other nodes in the network
  - Multicast Route Table [MRT]

    Contains entries for multicast groups of which the node is a router

    Any node can join the multicast tree by a series of RREQ/RREP messages

    To leave group: prune flag

Back to Anonymous Gossip

- 2 phases
  - Unreliable multicast protocol used to multicast message m to the group
  - Gossip used to recover lost messages
- A single round of gossip can recover many lost messages
- One Gossip round
  - A chooses node B to gossip to
  - A tells B what messages it has received and not received
  - B checks if it has the messages missed by A
  - A and B exchange messages

Anonymous Gossip

- New *gossip_message*
  - *Group_Address*
  - *Source_Address*
  - *Lost_Buffer*
  - *Number_Lost*
  - *Expected Sequence Number*
- Each node randomly selects a neighbor and sends it a gossip message
- This node in turn selects one of its neighbors and propagates the message

Anonymous Gossip

- If receiving node ε Multicast Group, randomly decides to accept message or propagate it
- Accepting node unicasts a gossip reply to initiator
- Locality of Gossip
  - AG gossips locally with a very high probability and with distant nodes occasionally
- Retrieving Lost Messages
  - *Lost_Table* at each node for every multicast group
  - *History_Table* – most recently received messages
  - *Lost_Buffer* stores most recent entries in Lost_Table

Retrieving Lost Messages…

- When a node prepares a gossip message, it adds the lost_buffer to it
- When a node receives a gossip message,

  compare(lost_buffer in message, own history_table)

  If a message is found, unicast it to the gossip initiator.

To sum up…

- Gossip possible without the knowledge of other group members
- AG can improve the reliability of multicast routing protocols without the use of ACKs

References

- Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks
  Jun Luo, Patrick Eugster, Jean-Pierre Hubaux
  School of Computer and Communication Sciences
  Lausanne
- Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks
  Ranveer Chandra, Venugopalan Ramasubramanian,
  Kenneth Birman,
  Cornell University
- Gossip-based ad hoc routing
  - Z Haas, J Halpern, L Li