Routing in Delay Tolerant Networks (DTNs)

Arobinda Gupta
Dept. of Computer Science & Engineering
IIT Kharagpur
Assumptions in Traditional Networking

- Network remains mostly connected
  - Even if topology changes due to failures, mobility
  - Even for mobile ad hoc networks
- End-to-end delay is low (typically 100s of milliseconds max.)
- Storage at intermediate nodes is transient (stored only for processing)
What If Not?

- Intermittent connectivity
  - Network remains partitioned most of the time
  - End-to-end paths may not exist for many node pairs most of the time
- Large/variable end-to-end delays (up to days!)

Can communication still take place? Will applications still work?
Delay Tolerant Networking

- Allow communication to happen in such “challenged” or “disrupted” environments
  - Should be able to tolerate
    - Network partitioning, loss of connectivity between node pairs
    - High end-to-end delays
  - Applications should be delay-tolerant also
  - Ideally, should work well also when network disruption and delay is low
Some Applications

- Interplanetary networks [www.ipnsig.org]
- Rural connectivity
  - DakNet [Pentland04], Kiosknet [Guo07], Motopost [Naidu08]
- Wildlife monitoring
  - ZebraNet [Juang02]
- Sensor Networks
  - SeNdT [McDonald07]
- Vehicular networks
  - FleetNet [Franz01], TrainNet [Zarafshan10]
- Underwater networks [Partan06]
- Military applications
One fundamental requirement in any network is the ability to send a message from one place to another.

In this talk, we focus on

**Routing in Delay Tolerant Networks**
Traditional Routing

- Assumes end-to-end path (path information maintained proactively or on-demand)
- Store-and-forward
  - Store packet in buffer
  - Process packet to find next hop
  - Forward packet if next hop found
  - Drop packet if no next hop (no route)
  - **Storage is short-term**
Delay Tolerant Routing

- Also store-and-forward
  - But doesn’t necessarily drop packets if no route is found
- Store packets at intermediate node
  - Forward to destination when contact made
  - May forward to other nodes based on some policy
- Handles cases when paths may be formed in future, or only partial paths are formed at a time
- Storage may be long-term
Main Issues in DTN Routing

- **Forwarding policy**: when to forward a stored packet?
  - Node may not come in contact with the destination for some time
    - Can the next contact time or chance of contact with destination in near future be predicted?
  - Other nodes in the current neighborhood may have a higher chance of meeting the destination
    - Packet can be given to a node on contact if it has a higher chance
    - How to determine that and rank the neighbors?
Main Issues (contd.)

- **Replication policy**: send more than one copy of a packet through different paths
  - How many total copies to create?
    - Too many copies can cause buffer overflow at intermediate nodes
  - Which nodes can create the replicas?
    - Source only, any intermediate node, specific intermediate nodes...
  - How many copies to give to a node on contact?
Main Issues (contd.)

- **Buffer management policy**
  - Longer-term storage means packets occupy buffers for more time
  - Buffer sizes are limited, may need to drop packets if buffer fills up
  - Which packets to drop?
    - New packets, FIFO, packets with hop count greater than threshold, packets with less chance of eventual delivery, packets with lower priority….
Main Issues (contd.)

- **Contact capacity/bandwidth**: how many packets can be transferred on a contact?
  - Contact time may not be large enough to transfer all the packets
  - Depends also on bandwidth and error rate of link
  - If more than one packet has to be transferred, in which order should they be sent?
- Priority assignment for waiting packets
Some Evaluation Metrics

- **Delivery Ratio**
  - Fraction of sent packets that are *eventually* delivered

- **Delay**
  - Average delay to reach all/some fraction of nodes
  - Maximum delay to reach all/some fraction of nodes

- **Packet Drops**
  - Drops due to message expiry before contact
  - Drops due to buffer overflow
  - Drops due to transmission loss during contact
Two Major Aspects of DTN Routing Protocols

- Amount of knowledge the protocol uses about the state of the system
  - None, partial, complete
  - More knowledge allows better decision making in forwarding, resource management, replication
  - More knowledge means more overhead (message, computation) to collect and maintain it

- Degree of replication (no. of copies of a packet) used
  - None, some
Examples of System Knowledge

Contact Information

- Schedule of contact between different node pairs
  - May be precise (ex., interplanetary networks)
  - May be completely unpredictable (ex., random movement of nodes)
  - May be predictable to some extent (ex. node movement following specific mobility patterns)
    - Use history of contact in recent past to build models to predict future contacts
- Contact frequency
  - Rate of contact between pairs of nodes
- Contact duration/Link uptime
  - Amount of time in the recent past for which a link was up (nodes in contact)
Examples of System Knowledge (contd.)

- Buffer availability at nodes
- Power remaining at nodes
- Social behavior of nodes
  - Can affect the pattern of contacts between nodes as well as buffer availability
- A node may have information of all other nodes (very costly), only a subset of nodes (ex. nodes it comes in contact with), or just itself
Contact Information

- One of the most important piece of knowledge used in DTN routing
- Governed by mobility patterns of nodes
  - Analyzing real world mobility patterns and proposing realistic mobility models is an interesting area of research in itself
- Various forms of contact information have been used in DTN routing protocols
  - Contact history, contact frequency, contact duration, predicted future contacts, contact probability
Classes of Routing Protocols

- Data ferrying based protocols
- Epidemic based protocols
- Forwarding based protocols
- Social-aware protocols
Data Ferrying Based Protocols

- Heterogeneous nodes
- Data generating nodes are mostly stationary (some protocols also have mobile data generating nodes)
- One or more mobile nodes move around on predefined trajectories among the stationary nodes, collecting/distributing data from/to them
- Mobility patterns of all nodes not very important as most nodes are static
- Simple forwarding policy, transfer to ferry when in contact
Data Ferry (contd.)

- Main issue: trajectory planning of the mobile nodes to achieve some desired goal such as lower average delivery delay and higher delivery success
  - Depends on node positions, traffic pattern
  - Also depends on nature of mobile node
    - Example: a bus can move only along roads and may have a fixed route
- Buffer management is important for low number of ferries
  - Delay in picking up data by ferries can cause buffer overflow
- Applications mentioned earlier like DakNet, KioskNet, MotoPost work on the same principle
Epidemic Based Protocols

- Variations of flooding
  - Name derives from the way infectious diseases spread on contact
- Mostly homogeneous nodes
- Do not use any mobility information
- Very high degree of replication
- Fast and robust, but extremely resource consuming in terms of buffer space and bandwidth
  - Performance can degrade very fast if resource constraints are present (less buffer, low bandwidth)
Epidemic Based Protocols (contd.)

- No forwarding decision to be made
- Buffer management can be critical for smaller buffer sizes
- Several variations proposed to reduce the resource consumption while maintaining most of the advantages of flooding
Forwarding Based Protocols

- Largest class of DTN routing protocols
- Mostly homogenous nodes
- Nodes forward to other nodes
  - Different forwarding policies
  - Most forwarding policies used exploit mobility information of nodes in some way
  - A few protocols do not use any mobility information
- Different degrees of replication used
- Buffer management is not critical, though important depending on traffic rate, mobility pattern, and degree of replication used
Social-Aware Protocols

- Social behavior of nodes affect both the contact pattern and forwarding decision
- Considers the social behavior of nodes in the design of different policies
- Mostly a subclass of forwarding-based protocols, but interesting because of the type of information used
Data Ferrying Based Protocols
DataMule [Shah03]

- Data collection in sensor networks
  - Sensors send data to base stations
  - Data routed towards base stations using other sensors
- Drains more energy from sensors close to the base station as they have to route more data
- Alternative: *Data Mules*
  - Sensors do not try to send data towards base stations
  - Mobile transport agents (*Mules*) go around and collect the data from the sensors
Three level architecture

- **Sensors, mules, access points**

- Sensors and access points are static, mules move covering the sensors and access points

- Sensors send data to mules when they are in contact, mules transfer data to access points (connected to base stations) when they are in contact

- Mules do not exchange any data between themselves

- Sensors, mules, and access points are all placed at grid points of a grid topology
- Sensors and mules exchange data if the mule is on the same grid point as the sensor.
- Mules and access points exchange data if the mule is on the same grid point as the access point.
- Mule trajectory is random walk along grid:
  - Moves to one of four neighbors with equal probability at each clock tick.
- Sensors generate 1 unit of data at each clock tick.
- Analyses the performance – fraction of data generated at the sensors delivered successfully to base stations, and buffer size.
Message Ferries  [Zhao03, Zhao04]

- *Message Ferries (MF)* – a set of special mobile nodes
- Ferries move around among data generating nodes collecting and distributing data
- Different problems addressed
  - Single ferry vs. multiple ferries
  - Static vs. mobile data generating nodes
  - *Node-initiated* vs, *ferry-initiated* message ferry
- We look at the version with single ferry, mobile data generating nodes, ferry-initiated message ferry
Ferry Initiated Message Ferry

- Single ferry moves along default route
- Sends its location periodically using long range radio
- Nodes also send service request (with location info) to ferries using long range radio
- Ferry, on hearing service request from node, modifies its route to go to the node and transfers data using short range radio
- Nodes also send location update messages periodically
  - Helps the ferry track a mobile node and recompute its route accordingly
Default route

- Total area broken up into grids
- Ferry moves on the grid row by row
- Cell size $\leq \sqrt{2} R$ where $R =$ range of long range radio
  - Allows every node in the cell to communicate with the ferry

Route re-computation

- Among all nodes whose service requests have been received, go to the nearest one
- Another more complex heuristic also proposed
Ferry Replacement [Yang05]

- An improvement over node-initiated MF to rotate ferry duty among nodes
  - Necessary as the ferry consumes more energy and is a single point of failure
- Two schemes for replacement
  - Successor designation
    - Current ferry designates one node as its successor
  - Distributed election
    - Nodes elect a new ferry in a distributed manner on failure
Ferry Replacement: Distributed Election

- Assumes a predetermined ferry route
- Nodes approach ferry route for data delivery (Node Initiated Message Ferrying)

Failure Detection
- Any node that approaches the ferry route and does not hear hello messages from the ferry detects failure and enters the election

Each node can be ferry candidate or elector
- A node computes a backoff delay such that larger node capability implies lower delay
- Node waits for backoff delay interval before becoming candidate
- While waiting, if it hears from any other candidate, stops waiting and becomes elector
- Keeps number of candidates low
- Candidate nodes move along predetermined ferry route advertising themselves to nodes with their capabilities
- Electors compare advertised capability with highest capability advertisement seen so far
  - If current advertisement higher, send WIN message to candidate
  - Else send LOSE message to candidate with highest capability seen so far
  - Then become stationary and do not move until leader elected
- Candidate node becomes *disrupted*
  - On getting a LOSE message, or
  - On meeting another candidate with higher capability, or
  - On meeting another disrupted candidate with higher capability stored

- Disrupted candidate goes back to its original location, and behaves as an elector now (but stores the highest capability candidate seen)

- *Leader* – candidate that finishes two rounds and not get any LOSE message
  - Because this elector must have seen advertisements from all candidates
TrainNet [Zarafshan10]

- Trains and stations equipped with racks of hard disks
- Trains carry data from source stations to destination stations on their route
- Service providers load non real time data to be transferred on to disks in stations
- Before a train arrives at a station
  - Station fills outgoing traffic on disks
  - Train fills data destined for the arriving station on disks
- Data transfer is achieved by swapping the contents of the disk by one of the following
  - Manual exchange of disks
  - Manual interconnection of the disks
  - Wireless transfer through some high-speed wireless link
Epidemic Based Protocols
Epidemic Routing [Vahdat00]

- No knowledge used, maximum replication
- On contact, two nodes first exchange summary vectors – list of messages they each have
- Then copies of messages they don’t have are exchanged
- Eventually all nodes may have replicas of all messages
- High delivery ratio, low delay (if no resource constraints are there)
- Creates too many copies of a packet, resource-hungry
  - Can cause packet drops due to buffer overflow or bandwidth constraints
- Approaches to reduce copies
  - Remove unnecessary copies
    - TTL based
      - Remove copies that have been in circulation for long
    - Anti-packet based
      - Remove copies that have already been delivered explicitly
  - Flood limited copies
    - Get some (but not full) benefit of epidemic, but at lower cost
- Several protocols proposed based on the above
Epidemic with Immunity Messages [Mundur08]

- Aims at removing messages that are already delivered
- Each node keeps a list of delivered messages
- Destination, on receiving a message, updates its list of delivered messages
- List is propagated to other nodes on contact
- Messages in delivered list are not exchanged between nodes anymore
(p,q)-Epidemic [Matsuda08]

- Two parts
  - Forwarding
  - Recovery
- **Forwarding strategy**
  - A relay node receives a copy from the source node with probability $q$
  - A relay node receives a copy from another relay node with probability $p$
  - Destination node receives a copy from the source or a relay node with probability 1
  - Note that $p = q = 1$ gives epidemic routing
Recovery Strategy: **VACCINE**

- Destination, on receiving a packet, generates an *anti-packet* with packet id. The relay node that delivers to destination also generates an anti-packet (so two copies initially)
- Anti-packets are flooded to all nodes as per conventional epidemic routing
- A node, on receiving an anti-packet, deletes the corresponding packet if it holds it. It also remembers the anti-packet to avoid accepting the corresponding packet later
- Overall operation
  - Node has list of data packets and anti-packets
  - List exchanged between nodes on contact
  - Data packets corresponding to anti-packets are deleted at each node
  - Outstanding data packets are exchanged

- Additional overhead of flooding anti-packets is expected to be small as anti-packets are very small in size
- **Scheme for Epidemic Routing with Active Curing**
- Anti-packets used as earlier are termed as *passive curing*, as the anti-packets (ACK) are propagated in the same manner as data packets.
- Should propagate faster than data packets, but with low resource consumption.
- **Active Curing**: ACKs are propagated preferentially over data packets.
Active Curing

- Data packets, after being forwarded once, are marked as dormant for some time
- A dormant data packet will not be considered for transmission on contact with another node
- ACKs are always active (never dormant)
- So a message exchange will always close with the two nodes having the same ACKs, but maybe without the same data. So data propagation is delayed such that an ACK has a higher chance of killing redundant data before it spreads too much
**Prioritized Epidemic**

- Partial knowledge, some replication
- Uses epidemic routing, with limited resources
- Knowledge used here is not to determine route, but to decide which packets to drop (if buffer fills up) and which packets to transmit first (useful if contact capacity is low)
- **Average Availability (AA)** of a link – fraction of time in near future that the link will be available
  - Calculated based on the amount of time the link is up (two ends are in contact) in a past time window
  - If down for > a threshold, link is assumed to be dead
- AA values flooded using epidemic routing
- All nodes create a graph $G$ with (slightly modified) AA values as edge weights, computes shortest paths in $G$
- Assigns drop and transmit priorities to each packet in buffer based on the path lengths
- **Drop priority** – determines order in which packets will be dropped if buffer gets filled up
  - Function of no. of hops traversed so far and the distance to the destination in the graph (higher → lower priority for both)

- **Transmit priority** – determines order in which packets will be transmitted when a contact is made
  - Function of distance to destination and packet creation time and expiry time (earlier → higher priority for both)

- Otherwise similar to epidemic routing
Spray and Wait [Spyropoulos05]

- No knowledge (in basic scheme), some replication
- Source makes $L$ copies of a packet and sprays it to $L$ relays (relays are normal nodes chosen to act as relay only for this packet)
- If destination is not one of these relays, the relays will deliver to destination only on direct contact with it
- Relays don’t make copies
- A node can be a source for some packets and relays for others at the same time
- Choice of $L$ can be static or depend on expected delay to be achieved (uses some knowledge of the system in this scheme)
- Efficiency depends on spray method
  - Different possibilities for choosing relays

- **Binary spray-and-wait**
  - Source A has \( L \) copies initially
  - Any node with \( n > 1 \) copies of a packet, on encountering any other node B with no copies, forwards \( n/2 \) copies to B and keeps the rest
  - If a node has exactly 1 copy, it switches to act as a relay and does direct transmission to destination only (no more forwarding)
Variation: Multi-period Spray-and-Wait

[Bulut10b]

- Multi-phase spray
- In first phase, spray less than $L$ copies
- Wait to see if it is delivered to the destination
- If not, go to next phase and spray some more copies
- Repeat for the no. of periods chosen
Forwarding Based Protocols
First-Contact Routing [Jones05]

- No knowledge used, no replication
- A node forwards a packet to the first node it comes in contact with
- Large delay
- Low delivery ratio
- Extremely simple to implement
Minimum Estimated Expected Delay

Partial knowledge used, no replication

Use observed contact history to estimate expected contact schedule

For each node, record connection and disconnection time of each contact with another node over a sliding history window to get average waiting time till next contact with that node

If significant change in contact schedule, flood this to all nodes (similar to link-state routing)
Each node creates an undirected weighted graph $G = (V, E)$
- $V$ = set of nodes
- $E$ = set of node pairs $(u, v)$ with some contact schedule value
  - weight $w(u, v) = $ contact schedule value from $u$ to $v$
- Run Dijkstra’s algorithm to find shortest paths in $G$
- Forwarding done on a per-contact basis
  - If $u$ comes in contact with $v$, then $w(u, v)=0$ during contact
  - Recalculate shortest path based on this new value before forwarding any packet
PRoPHET [Lindgren04]

- Probabilistic ROuting Protocol using History of Encounters and Transitivity
- Partial knowledge, some replication
- Every node $u$ maintains a probabilistic metric, delivery predictability, $P(u,v) \in (0, 1]$, for every other node $v$
  - Signifies the probability that $u$ will deliver to $v$
- $P$ values updated on contact between nodes
  - Tries to exploit underlying mobility pattern, if any, to get stable $P$ values
- Direct update: when \( u \) comes in contact with \( v \)
  \[
P(u, v) = P(u, v)_{\text{old}} + (1- P(u, v)_{\text{old}}) \times P_{\text{init}}
  \]
  \( P_{\text{init}} \in (0, 1] \)

- Transitive update: when \( u \) comes in contact with \( v \)
  \[
P(u, w) = P(u, w)_{\text{old}} + (1- P(u, w)_{\text{old}}) \times P(u, v) \times P(v, w) \times \beta
  \]
  \( \beta \) is a transitivity constant, determines how much effect transitivity will have

- Aging: if \( u \) and \( v \) do not come in contact for some time
  \[
P(u, v) = P(u, v)_{\text{old}} \times \gamma^k
  \]
  \( k \) is the time since last aging, \( \gamma \in (0, 1] \) is an aging constant
- Forwarding strategy at node $u$ for packet with destination $d$
  - On contact with node $v$, forward to $v$ if $P(v,d) > P(u,d)$
  - Hold a copy at $u$ (subject to buffer space) in case other better nodes are encountered
SCaTR [Boice07]

- *Space Content adaptive Time Routing*
- Partial knowledge, some replication
- Integrates DTN and MANET routing
- Extension of AODV
- If only one partition, behaves same as AODV
- If destination not found in current partition (AODV fails to find a route), forward to one or more proxies
- *Proxy* – nodes in the current partition that have more chance of having a route to the destination in the near future
- **Contact Table** at each node, one entry for each destination (destinations added on-demand)
- For each destination $d$
  - Maintains a *contact value*
  - $d$ advertises its own contact value as maximum
  - Nodes periodically broadcast their contact table to neighbors
  - For each period, a node takes the maximum contact value for $d$ received from all its neighbors
  - The maximum is averaged over a sliding window of number of periods to get the final contact value for $d$
Discovering proxies for destination $d$

- Node sends *Proxy Request (PREQ)* messages with its contact value for $d$ to all nodes in its partition
- A node, on receiving a *PREQ* the first time
  - Replies with the route to source if it has an active route to $d$ (route discovery)
  - Replies with *Proxy Reply (PREP)* messages with their contact values only if their contact value for $d$ is significantly better
  - Rebroadcasts *PREQ* otherwise

- *PREQ/PREP* similar to *RREQ/RREP* in AODV
Source node selects one or more proxies based on the received PREPs, creates a route to them (similar to AODV routes, only that these are routes to proxies of the destination in local partition).

Forwarding packets

- Source selects at most two proxies and sends to them
- Proxies can send to other proxies in local partition if better proxies are discovered later (by getting PREPs with better contact values)
  - Proxy discovery initiated if better contact value received for destinations of buffered packets
- Packet is delivered when proxies get an active route to the destination
Context-Aware Adaptive Routing

- Partial knowledge used, no replication
- Integrates DTN and MANET routing
- Two modes of transfer – *synchronous* and *asynchronous*
- Transfers synchronously (no intermediate storage) if destination is in same network partition using some routing protocol (DSDV used)
  - So a normal routing protocol runs underneath
- Transfers asynchronously (store-carry-forward) if destination not in same network partition
  - Transferred to node in the same partition (using routing protocol) with highest delivery probability to destination
  - That node gives to destination when it goes to that partition
- Delivery probabilities computed by applying Kalman filter prediction from two context information
  - Change degree of connectivity – no. of neighbor changes (neighbors added or deleted) by a host over last T time, normalized by total no. of hosts met
  - Host colocation information for last T time
    - Colocation of host i with host j at time t = 1 if i and j are in contact at time t, 0 otherwise
  - Weights of individual context information adapted
- Delivery probability values sent through routing updates of the underlying routing protocol
- Based on these, each node builds entries of the form 
  <destination, best_hop, delivery probability>
Forwarding policy

- Check if destination is in routing table
- If yes, destination is in same partition, forward using next hop in routing table
- If not, destination in different partition
  - Send to node with best delivery probability (which is in the same partition) using next hop in routing table
SCAR [Mascolo06]

- **Sensor Context Aware Routing**
- Based on CAR, but adapted for sensor network
- No underlying routing protocol as in CAR
- Delivery probability values computed similarly, battery power also part of context
- Delivery probability values exchanged on contact as before
- Each node maintains an ordered list of neighbors (including itself) in decreasing order of delivery probability to each destination
Source replicates packet to first R nodes in the list (one of which may be itself)

Replica sent to the highest delivery probability node is called the *master copy*, others are *backup copies*

Backups can be dropped if buffers are full, master is dropped only after delivery to sink

Packets are forwarded to nodes with better delivery probability on contact (actually, with delivery probability larger than a threshold)

When battery is low, master copies copied to neighboring nodes with free slots irrespective of delivery probability
MaxProp [Burgess06]

- Partial knowledge used, no replication
- Basic Idea
  - Rank stored packets based on a cost (estimate of delivery likelihood) of destination
  - For packets with same cost, packets that traversed less number of hops get priority
  - Higher priority packets are transferred first on a contact
  - Lower priority packets are to be deleted first when buffer is full
- Acks used to notify packet delivery
  - Can be used to remove packets
Cost computation

- Let no. of nodes = \( n \)
- Each node \( i \) defines a probability \( p_i(j) \) for meeting \( j \) next
  - Initialized to \( 1/(n-1) \)
  - Incremented by 1 on meeting \( j \), and then all \( p_i \) values are normalized again (i.e., sum to 1)
- These values are exchanged on contact between nodes
- So each node \( i \) knows all the values in the network
- Cost for each possible path to the destination is now computed
  - Summation of the prob. that each link in the path does NOT occur
- Cost for destination = cost of least cost path
Utility Based Distributed Routing Algorithm with Multiple Copies

- Partial knowledge, some replication
- Source replicates a certain number of copies
- Each node holds a utility vector – meeting possibility value for every node it has met
- Relay nodes transfer to other nodes with higher utility (chance to meet destination)
- Destination sends message with ids of received message, which are used to remove copies
- Higher utility packets replace lower utility packets if buffer is full
- **Utility** of node $j$ at node $i$
  - Ratio of total meeting time between $i$ and $j$ in a past interval $T$ divided by $T$
  - Exponential averaging done for smoothening

- **Replication policy**
  - If a node holds $x$ copies, it gives $x/2$ copies to a node while forwarding

- **Buffer management policy**
  - *Core-copy*: packets with destination utility greater than a threshold
  - *Backup copy*: otherwise
  - Backups copies overwritten first if buffer is full
Encounter Based Routing [Nelson09]

- Partial knowledge used, some replication
- Two basic ideas
  - Future rate of contact can be predicted from past encounters
  - Nodes that have a higher encounter rate has a higher chance of meeting the destination
- The source starts with some no. of replicas of the packet
- At every contact, some of these replicas are transferred to the other node
How many replicas should be transferred?

- Use the encounter rate of the two nodes

Each node maintains an *Encounter Value (EV)*

- Exponentially weighted moving average of past encounter rate

If node A meets node B, transfer

\[ m \times \left( \frac{EV_B}{(EV_A + EV_B)} \right) \] replicas to B, where \( m \) is the current no. of replicas of the message in A
VeRo [Kang08]

- Vector Routing
  - Assumes location information is available
  - Each node maintains its location coordinates of past $n$ intervals
  - Computes an average vector of node movements for the $n$ intervals
    - Compute vector for each interval
    - Take exponential average over the intervals
  - Nodes exchange direction and velocity on contact
Replication policy on contact

- Replicate no packet if the two nodes move in same direction (the carrier node itself will cover)
- Replicate less packets if the two nodes move in opposite direction (probably the carrier node has already covered)
- Replicate all packets if the two nodes move in orthogonal direction
- For any other directions, number of packets replicated is a function of the difference in direction and difference in velocity
  - Higher velocity means will reach faster
- Note that policy decides how many packets to replicate, but still one replica per packet at most
Orion Routing \[Medijah11\]

- Partial knowledge used, no replication
- Network of buses and trams (regular nodes), cars (random nodes), and hotspots (fixed nodes)
- Assumes location information is available at each node
- Each node uses two time series data of contact durations and disconnection durations (recorded from past contacts)
- Standard prediction model applied on them to predict next contact time and the duration of next contact
- A score is assigned by each node to each possible neighbor based on the predicted time and duration of next contact
  - Score function tuned to prioritize delivery speed (time of next contact) or delivery certainty (duration of next contact) or both
Forwarding strategy

- Look for closest connected neighbor to destination (greedy geographic routing) and send to it
- If none, look for most advancing connected neighbor to destination and send to it
  - Connected neighbor that has reduced its distance to the destination the most in the recent past
- If none, schedule to send to best future connected neighbor
  - One with the highest score
  - This node is not in contact now, so schedule for later. If contact happens as predicted, packet ill be forwarded. If not, the forwarding strategy will be applied again on it
**Link Contact Duration Based Routing**

- Partial knowledge used, no replication
- Three phases
  - Contact history computation
  - Contact history distribution
  - Message transmission
- Phase 1: Contact History Computations
  - Monitors every contact make/break and update contact history accordingly
  - $LDT$: Average time for which a link is broken in past $T$ time
  - $LCT$: Average time for which the link is connected in $T$ time
  - Network history table entry added:
    $<$link ID, LDT, LCT, update time$>$

- Phase 2: Contact History Distribution
  - $<$link id, LDT, update time$>$ forwarded to currently connected node ($LCT$ used only locally)
    - Both its own links and for other links received indirectly
  - Receiver uses update time field to ignore obsolete entries
  - Eventually, network history table has information of all links in the network
Phase 3: Message Transmission

- \( E(A) \) = links containing node A
- \( E_e(A) \) = effective links of A = \( E(A) - e_c \) where \( e_c \) = currently connected node
- Sender computes \( EDT \) (Expected Disconnection Time) and \( ECT \) (Expected Connection Time)
  - \( EDT = \) half of LDT initially (as LDT only measures disconnection, but not actual delay, as message can be generated well into the disconnection)
  - \( ECT = LCT \)
- Given a routing path, path cost = sum of EDT link costs
- Shortest path = path with least cost
- Connected shortest path \( SP_c \) = path with currently connected node as next hop
- Potential shortest path \( SP_p \) = shortest path with one link from \( E_e(A) \) as next hop
Forwarding strategy

- Take each message \( m \) and try to forward once a contact is made
- If no path to destination (possible initially), send to currently encountered node
- If only SP\(_c\), send through SP\(_c\)
- If only SP\(_p\), defer till next contact and apply same rule
- If both SP\(_p\) and SP\(_c\), select one with the smaller cost
  - If it is SP\(_c\), send immediately
  - If it is SP\(_p\), \( ECT(E_p) = ECT(e_p) - t(m) \), where \( t(m) \) is the expected transmission delay for the message \( m \)
- Sort of reserving, so that next messages considered gets only remaining connection time in its forwarding decision

- Also has a scheme to change link costs adaptively based on contact duration and no. of buffered messages
Social-Aware Protocols
**SimBet** [Daly07]

- *Betweenness centrality* of a node $n$
  - Extent to which node $n$ lies in paths linking other nodes
    - A measure of the importance of the node
  - Let $N = \text{total no. of shortest paths between any pair of nodes } j \text{ and } k, \ j, k \neq n$
  - Let $M = \text{no. of those paths that include } n$
  - Betweenness centrality of $n$, $\text{Bet}_n = M/N$

- *Similarity* of node $n$ with node $d$, $\text{Sim}_n(d)$
  - No. of common neighbors between $n$ and $d$
  - Higher degree of similarity gives higher chance of the nodes coming in contact
    - Nodes with more common friends are more likely to be friends themselves
- **Similarity Utility** of node $n$ for destination $d$ compared to node $m$
  \[ \text{SimUtil}_n(d) = \frac{\text{Sim}_n(d)}{\text{Sim}_n(d) + \text{Sim}_m(d)} \]

- **Betweenness Utility** of $n$ compared to $m$
  \[ \text{BetUtil}_n = \frac{\text{Bet}_n}{\text{Bet}_n + \text{Bet}_m} \]

- **SimBetUtil** of $n$ for destination $d$
  \[ \text{SimBetUtil}_n(d) = \alpha \times \text{SimUtil}_n(d) + \beta \times \text{BetUtil}_n \]
  where $\alpha + \beta = 1$
• Forwarding policy
  - Nodes $n$ and $m$, on contact, exchange summary vectors of list of destinations for the packets stored, and betweenness and similarity value
  - Node $n$ computes the SimBetUtility of $n$ and $m$ for each destination
  - If $n$ has higher SimBetUtility value for the destination of a packet held by $m$, $n$ requests $m$ for the packet
  - $m$ forwards the packet to $n$ and removes from its own buffer
  - Node $m$ does the same as above
Problem

- Computation of betweenness centrality and similarity computation with all nodes require knowledge of entire network

Solution

- Nodes exchange list of other nodes they have encountered on contact
- Local computation of the measures based on the graph known so far
Friendship Routing [Bulut10a]

- Uses three main features of close friendship – high frequency, longevity, regularity

- Social Pressure Metric ($SPM$)
  - Average message forwarding delay to $j$ if $i$ had a new message to deliver at each time unit

$$SPM(i, j) = \frac{\int_0^T f(t) dt}{T}$$

where $f(t) = \text{remaining time to next encounter of these nodes at time } t$
- If there are \( n \) intermeeting times in \( T \) between \( i \) and \( j \),

\[
SPM(i, j) = \frac{\sum_{x=1}^{n} t_x^2}{2T}
\]

where \( t_x \) is the \( x \)-th intermeeting time
- A node \( i \) can compute SPM values based on its past contacts
- Each node \( i \) computes a link quality for each link \((i, j)\)

\[
w(i, j) = \frac{1}{SPM(i, j)}
\]

- Higher \( w(i, j) \) means closer \textit{friendship} between \( i \) and \( j \)
  - Higher forwarding opportunity between them
- **Friendship community** formation
  - Direct friends of a node – nodes with link quality with it larger than a threshold
  - Indirect friends – Informally, nodes with a common direct friend
    - A Relative SPM (RSPM) metric (average delay to transmit from $i$ to $d$ through $j$) used to compute upto 2-hop indirect friends
  - Friendship community of a node – its direct and indirect friends
Friendship community formed from each period $T$ of the contact history

Forwarding strategy

- Node $i$ having message for $d$ forwards it to $j$ on contact if and only if $j$’s friendship community in the current period includes $d$ and $j$ is a stronger friend of $d$ than $i$
- However, if time is very near to the end of current period, use the next period’s community instead of current
- More relevant if community changes
SSAR [Li11]

- **Social Selfishness Aware Routing**
- All nodes do not forward packets received from all other nodes
- A node only forwards packets for those with which it has social ties
  - Higher priority given to packets from nodes with stronger social ties
- **Willingness** value
  - Maintained by each node for every other node in the network
  - 0 : unwilling to forward, 1 : most willing to forward
  - Manually configured

- **Delivery probability** value
  - Maintained by each node for every other node
  - Based on contact probability and drop probability
- **Priority** assigned to each incoming packet
  - Based on *willingness* value of source and the previous hop
  - If node has social ties with
    - Neither source nor previous hop, priority = 0
    - Only source, priority = willingness of source
    - Only previous hop, priority = willingness of previous hop
    - Both source and previous hop, priority = max of willingness of source and previous hop
  - Note that willingness of node with destination not considered
- If priority 0, not buffered
- Lower priority packets get dropped first
- Higher priority packets can preempt lower priority packets
- Forwarding rule from M to N
  - Find packets for which N has higher delivery probability
  - Compute
    \[ \text{selfish gain} = \text{priority} \times \text{delivery probability} \]
  - Forward packets with higher \textit{selfish gain} first
Finally

- Many other DTN routing protocols in literature
- Problems of multicasting and broadcasting, are also addressed
- Work on standardizing communication in DTNs have also been done and reference implementations conforming to them are available (www.dtnrg.org)
- Interesting real world mobility traces are also available for experimentation (www.crawdad.org), building realistic mobility models have also been an important research area
- Proliferation of smart mobile phones can enable interesting future DTN based applications
References


Thank You