Delay Tolerant Networks (DTN)
Motivation

• Rural area (buses, mail trucks, infostations)
• Mobile routers w/disconnection (e.g., ZebraNet)
• Sensor networks (e.g., Data mules)
• Deep space
• Underwater
• ...
Internet vs. DTN

Unstated Internet assumptions

- Exist some end-to-end paths
- End-to-end RTT is low
  - At most a few seconds, and typically less than 500 ms
- Use retransmission for reliability
- Packet switching is the right abstraction
Internet vs. DTN

Unstated Internet assumptions
- Exist some end-to-end paths
- End-to-end RTT is low
  - At most a few seconds, and typically less than 500 ms
- Use retransmission for reliability
- Packet switching is the right abstraction

DTN characteristics
- May not exist e2e paths
  - Contact connectivity is intermittent and hard to predict
- Large delay (can be hours or even days!)
- High link error and low capacity
  - Resource budget can limit transmissions
- Different network architectures (e.g., TCP/IP won’t work)
DTN Issues

- Naming, addressing, location management
- Routing on dynamic graphs
- Scheduling
- Security
- Applications
- …
Routing on Dynamic Graphs

- DTN routing takes place on a time-varying topology
  - Links come and go, sometimes unpredictably
  - Use any/all links that can possibly help

- Inputs
  - Time varying topologies (S, D, c(t), d(t))
  - Traffic demands
  - Vertex buffer limits
  - Mobility patterns

- Goal
  - Determine route and schedule to optimize some metric (e.g., delay, throughput, resource consumption)
What's the routing scheme that minimizes delay?
Solution 1: Flooding
Flooding

- Each node forwards any non duplicated msg to any other node it encounters

- Pros: low delay

- Cons: high transmission overhead
What’s the routing scheme that minimizes resource consumption?
Solution 2: Direct contact
Direct Contact

• The source holds the data until it comes in contact with the destination

• Pros: minimal resources

• Cons: long delay
Solution 3: Simple Replication
Simple Replication

- Source sends $r$ identical copies over the first $r$ contacts
- Relay nodes directly send to the destination
Illustration
Solution 4: History-based replication
History-based Replication

• Each node keeps track of the probability a given node delivers its message
• It replicates to $r$ highest ranked relays based on delivery probability
Solution 5: Erasure-coding based Replication
Erasure Codes

• Rather than seeking particularly “good” contacts, we “split” messages and distribute to more contacts to increase chance of delivery
  – Same number of bytes flowing in the network, now in the form of coded blocks
  – Partial data arrival can be used to reconstruct the original message
    • Given a replication factor of $r$, (in theory) any $1/r$ code blocks received can be used to reconstruct original data

• Potentially leverage more contacts opportunity $\Rightarrow$ reduce worse-case latency
  – Reduces “risk” due to outlier bad contacts
Erasure Codes

- **Message n blocks**
- **Encoding (to m blocks, where m > n)**
- **Split message blocks among r*k relays**
- **Decoding (Require n+alpha blocks)**
- **Message n blocks**
Simple Replication vs. Erasure Coding

• Simple replication
  – Use first $r$ relays, where each relay gets 1 copy
  – 1 out of $r$ relays to succeed

• Erasure coding
  – Use first $r\times k$ relays, where each relay gets $1/k$ copy
  – $k$ out of $r\times k$ relays to succeed
  – If $k$ is large, the delay distribution converges to a constant

Which one is better?
Simple Replication vs. Erasure Coding

• **Simple replication**
  – Use first $r$ relays, where each relay gets 1 copy
  – 1 out of $r$ relays to succeed $\Rightarrow$ low average-case delay

• **Erasure coding**
  – Use first $r \times k$ relays, where each relay gets $1/k$ copy
  – $k$ out of $r \times k$ relays to succeed
  – If $k$ is large, the delay distribution converges to a constant $\Rightarrow$ almost assured constant delay
## Summary: Forwarding Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Who</th>
<th>When</th>
<th>To whom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flood</td>
<td>All nodes</td>
<td>New contact</td>
<td>All new</td>
</tr>
<tr>
<td>Direct</td>
<td>Source only</td>
<td>Destination</td>
<td>Destination</td>
</tr>
<tr>
<td>Simple Replication(r)</td>
<td>Source only</td>
<td>New contact</td>
<td>$r$ first contacts</td>
</tr>
<tr>
<td>History (r)</td>
<td>All nodes</td>
<td>New contact</td>
<td>$r$ highest ranked</td>
</tr>
<tr>
<td>Erasure Coding (ec-r)</td>
<td>Source only</td>
<td>New contact</td>
<td>$kr$ ($k \geq 1$) first contacts ($k$ is related to coding algorithm)</td>
</tr>
</tbody>
</table>
Evaluation Methodology

• Use a real-world mobility trace collected from the initial ZebraNet test deployment in Kenya, Africa, July, 2004

• Only one node returned 32-hour uninterrupted movement data
  – Weather and waterproofing issues

• Semi-synthetic group model
  – Statistics of turning angles and walking distance
Trace Results

- Contact duration
- Inter-contact time
Performance Evaluation: Routing Overhead

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Overhead (34 nodes)</th>
<th>Overhead (66 nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec-rep2-p8</td>
<td>3.96</td>
<td>—</td>
</tr>
<tr>
<td>ec-rep2-p16</td>
<td>3.96</td>
<td>3.98</td>
</tr>
<tr>
<td>ec-rep2-p32</td>
<td>—</td>
<td>3.98</td>
</tr>
<tr>
<td>srep-rep2</td>
<td>3.98</td>
<td>3.99</td>
</tr>
<tr>
<td>direct</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>history</td>
<td>30.28</td>
<td>59.61</td>
</tr>
<tr>
<td>flood</td>
<td>68.0</td>
<td>132.0</td>
</tr>
</tbody>
</table>

History and flooding replicates messages even after a copy of original message has been delivered.
Performance Evaluation: Latency
(64 nodes)
Performance Evaluation: Success Rate

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>0.25 day</th>
<th>1 day</th>
<th>2 days</th>
<th>4 days</th>
<th>8 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>ec-rep2-p8</td>
<td>22.6%</td>
<td>95.9%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>ec-rep2-p16</td>
<td>9.2%</td>
<td>94.6%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>srep-rep2</td>
<td>51.8%</td>
<td>92.5%</td>
<td>99.6%</td>
<td>99.9%</td>
<td>99.9%</td>
</tr>
<tr>
<td>direct</td>
<td>32.0%</td>
<td>74.6%</td>
<td>94.2%</td>
<td>99.5%</td>
<td>99.9%</td>
</tr>
<tr>
<td>history</td>
<td>58.4%</td>
<td>87.9%</td>
<td>92.7%</td>
<td>94.6%</td>
<td>95.3%</td>
</tr>
<tr>
<td>flood</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

ec has low success rate when deadlines are small.

ec has high success rate for longer deadlines (due to lower 99th percentile latency distr).
Theoretical Results on Delay Distribution

Erasure Coding (32 nodes):
- Get rid of the ‘bad’ cases
- Has few very low delay cases

99th percentile
Simple Replication ~ 3 Erasure Coding
Summary

- Flooding
- HR
- SR
- EC
- Direct

Overhead vs. Average-case Delay
Summary

Overhead

Worst-case Delay

Flooding

EC

SR

HR

Direct
How to improve?
Enhancements

- **Optimize common case and guarantee worst-case**
  - Whom to replicate to?
    - Currently based on first r contacts
    - Could use delivery probability for selection
  - How much to replicate?
    - Currently every node selected is replicated an equal amount of data
    - Could use delivery probability for deciding the amount to replicate
  - Different coding schemes
    - Adapt the coding parameters based on delivery probability and performance requirement
    - Apply network coding

- **Provide differentiated services**
  - Adapt who to replicate, how much to replicate, when to replicate based on the urgency of message