
Presented by:
Kyu-Wook Cho
11/02/05

Outline
- Background
- Routing
- Forwarding
- Evaluation
- Related Work & Conclusion

Assumptions
- Direct access only to neighbor nodes
- Resource-constrained nodes
- Many producers and few consumers
  - Producers – sensor node
  - Consumers – base stations
- Nodes are stationary

Key Features
- DV/DRP is augmented distance-vector routing protocol
- Forwarding uses receiver predicates
- Avoid message duplicates and loops
- Receiver allows to limit the message delivered rate
Outline

- Background
- Routing
- Forwarding
- Evaluation
- Related Work & Conclusion

Routing Table

- Kind of information in routing table
  - Path information
    - primary/alternative path information
  - Path-independent information:
    - receiver predicates $p$
    - sequence number $s_r$
    - bit-vector position $b_r$
    - the minimum inter-arrival interval $\Delta_r$
    - time of last matching message $t_r$

Routing Algorithm

- New entry added if new predicate
- More recent predicate updates entry
- Obsolete predicate dropped
- Keep same sequence # predicate with different distance
- Zero sequence # uses to reset the sequence counter
- Null predicate uses to remove a route entry

Message and Predicate Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>receiver (16 bits)</td>
<td></td>
</tr>
<tr>
<td>next hop (16 bits)</td>
<td></td>
</tr>
<tr>
<td>distance (8 bits)</td>
<td></td>
</tr>
<tr>
<td>sequence num. (8 bits)</td>
<td></td>
</tr>
<tr>
<td>bit-vector pos. (5 bits)</td>
<td></td>
</tr>
<tr>
<td>max rate (27 bits)</td>
<td></td>
</tr>
<tr>
<td>predicate (variable size)</td>
<td></td>
</tr>
<tr>
<td>receiver set (32 bits)</td>
<td></td>
</tr>
<tr>
<td>route-failure (1 bit)</td>
<td></td>
</tr>
<tr>
<td>opt. message id (31 bits)</td>
<td></td>
</tr>
<tr>
<td>message content (variable size)</td>
<td></td>
</tr>
</tbody>
</table>
Routing Paths

Routing Example

<table>
<thead>
<tr>
<th>id</th>
<th>Pred.</th>
<th>primary</th>
<th>2nd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recovery Algorithm

- if failed, node set “route-failure flag”
- Random message ID to message
- Store the message ID into local cache
- Send message using alternate route
- If all fails, flood the message
- A message with failure flag set received, node check the message ID from its local cache
  - If ID found, the packet got trapped in a loop and flood packet
  - Route-failure flag inform failure to receiver
- After several failure packets, reissue a predicate advertisement
Data Rate Limitation

- No data rate limitation by default
  - Problem in large networks: receiver and its surrounding relay nodes overwhelmed by message flow
- DV/DRP allows receiver to specify a limit on the message delivery rate
  - Receiver \( r \) set minimum inter-arrival interval \( \Delta_r \) (\( \Delta_r = 0 \) means no limit)
  - Advertise \( \Delta_r \) with predicate \( p_r \)

Outline

- Background
- Routing
- Forwarding
- Evaluation
- Related Work & Conclusion

Forwarding

- Node has complete information about receivers in the network
- Receiver set – result of evaluation in message header field
- Relay messages – check the receiver is in the receiver set attached to the message
- Fixed-size bit vector – dynamically assign bit position for each receiver

Fixed-size Bit Vector

- Advantage
  - Compact representation of receivers
- Disadvantage
  - Node negotiate their bit-vector position
Duplicates when two copies of a message \( m \) cross two distinct branches of content-based spanning tree of \( r \).

**Forwarding Algorithm**

```
proc cdp.init.forward(message m) {  
  map<node, set<int>> D := 0  
  foreach i in Available {  
    if time() - t_r > Delta and match(i,m) {  
      D[i] := D[i] \( m \)  
      t_r := time()  
    }  
  }  
  foreach i in D {  
    //for each next-hop node  
    send(m, P[i,a])  
  }  
}
```

```
proc cdp.forward(message m, set<int> D) {  
  map<node, set<int>> D := 0  
  foreach i in Available {  
    if time() - t_r > Delta and r \( D \) {  
      D[i] := D[i] \( m \)  
      t_r := time()  
    }  
  }  
  foreach i in D {  
    //for each next-hop node  
    send(m, P[i,a])  
  }  
}
```

**Outline**

- Background
- Routing
- Forwarding
- Evaluation
- Related Work & Conclusion
Evaluation

DV/DRP Functionality

![Graph showing false negatives and false positives](image)

Data and Control Traffic

![Graph showing number of packets per second](image)

Scalability

![Graphs showing scalability with no failure and 10% node failures](image)

Transient Failures

![Graphs showing transient failures with different failure scenarios](image)
Outline
- Background
- Routing
- Forwarding
- Evaluation
- Related Work & Conclusion

Evaluation
Permanent Failures

Related Work
- Optimizing link-layer communication
  - Data aggregation, node clustering, minimizing radio listening time, energy efficient MAC protocols
- Directed diffusion
  - DD shares many of DV/DRP
  - DV/DRP isolates application from routing decisions

Conclusion
- Maximizes proper message delivery
- Minimizes power consumption with shortest path
- Avoid duplicates and loops
- Tolerate both transient and permanent network failure
Appendix

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>P1</th>
<th>primary</th>
<th>2nd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>p1</td>
<td>1-&gt;2</td>
<td>2-&gt;1</td>
<td>3-&gt;2</td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td>2-&gt;1</td>
<td>3-&gt;2</td>
<td>1-&gt;2</td>
</tr>
</tbody>
</table>