Greedy Distance Vector Routing

Reference
Chen Qian and Simon S. Lam, “Greedy Distance Vector Routing,” Proceedings IEEE ICDCS, June 2011
Design space of routing protocols

Node routing state

- large
- small

Path quality
- good
- bad

Routing states:
- shortest path
- hierarchical routing
- greedy routing

4/11/2014  Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
Distance Vector (DV) routing

- Node $u$ chooses a neighbor $v$ as the next hop to destination $t$ such that, for every neighbor $x$ of $u$,

\[ c(u, v) + D(v, t) \leq c(u, x) + D(x, t) \]

where
- $c(u, x)$ denotes the cost of link $u - x$
- $D(x, t)$ denotes the least cost from $x$ to $t$
Distance Vector routing

- Periodically, node u computes its distance vector, \( \{D(u,t): t \in N\} \) which it sends to its neighbors, where \( N \) is set of network nodes.

- From each neighbor \( x \), node u receives a distance vector, \( \{D(x,t): t \in N\} \), which is used to update node u’s own distance vector.

- For a fixed topology and no link cost changes, distance vectors will converge to least costs.
Distance Vector routing

**Pros**
- any additive routing metric can be used
- provides least cost paths *if* distance vectors are *least costs*

**Con**
- Each node’s routing state is large, $O(|N|)$

**Practical issues**
- Convergence to least costs only if network is static
- Link cost increase -> “count to infinity” problem
Greedy routing

Node $u$ chooses a neighbor $v$ as the next hop to destination $t$ such that, for every neighbor $x$ of $u$,

$$d(v,t) < d(u,t) \text{ and } d(v,t) \leq d(x,t)$$

where $d(y,t)$ denotes the distance from $y$ to $t$ in a Euclidean space.

If no such $v$
- $u$ is closest to $t$
- $u$ may be a local minimum
Greedy routing

Cons

- Routing cost metric limited to *distance* or *hop count*
  - Other metrics such as ETX, ETT, or latency may provide higher throughput in a wireless network
- No information about the remaining path
- Localization method required

Pro

- Each node’s routing state is independent of network size
### Greedy Distance Vector (GDV) Protocol Design Objectives

<table>
<thead>
<tr>
<th></th>
<th>Node Routing State</th>
<th>Routing Metric</th>
<th>Path Quality</th>
<th>Localization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DV</strong></td>
<td>(O(</td>
<td>N</td>
<td>))</td>
<td>any additive metric</td>
</tr>
<tr>
<td><strong>Greedy</strong></td>
<td>independent of (N)</td>
<td>distance or hop count only</td>
<td>good only if metric is hop or distance</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>GDV</strong></td>
<td>independent of (N)</td>
<td>any additive metric</td>
<td>near-optimal</td>
<td>No</td>
</tr>
</tbody>
</table>

4/11/2014  
Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
**GDV idea**

- Embed nodes into a virtual space, such that the Euclidean distance between any pair of nodes in the virtual space is a good estimate of the routing cost between them
  - Can be done for any additive cost metric
  - Each node knows link costs to its physical neighbors

4/11/2014  
Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
GDV forwarding

Node $u$ selects a neighbor, as the next hop, that minimizes

$$c(u, x) + \tilde{D}(x, t)$$

where $\tilde{D}(x, t)$ is the estimated routing cost from locally computing the distance between the virtual positions of $x$ and $t$.
Prior Virtual Positioning Systems

- Proposed to embed latencies in a virtual space for Internet hosts
  - GNP [Ng & Zhang 02] - requires landmarks
  - Vivaldi [Dabek et al. 04] - fully distributed

- Not applicable in wireless (or layer-2) networks because
  - they require any-to-any routing support for measuring latencies to some *distant nodes*
  - latency only (rather than any additive metric)
Using Vivaldi for a layer-2 network

121-node network in 2D physical space

Running 2-hop Vivaldi for 20 adjustment periods
Requirements for good virtual positions

- **Local relationships**: nodes with low routing costs between them should be nearby in the virtual space

- **Global relationships**: nodes with high routing costs between them should be far apart in the virtual space
  - Vivaldi needs measurements to some distant nodes to be effective
Virtual Position by Delaunay (VPoD)

- Nodes use MDT protocols to construct and maintain a multi-hop DT [Lam & Qian 2011]
  - Each node assigns itself an arbitrary location in a virtual space (2D, 3D, or higher dimension)
  - MDT provides routing support

- Each node $u$ uses an adjustment algorithm to move its position in the virtual space by comparing its distance to a neighbor $v$ with its routing cost to $v$
  - for each neighbor *(physical or DT)*
    - sufficient for convergence, no need for distant nodes
  - using any additive cost metric
VPoD for the same layer-2 network

121-node network in 2D physical space

Running VPoD for 20 adjustment periods

4/11/2014

Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
Outline

- Motivation
- MDT Protocols
- Virtual Position Construction
- Greedy Distance Vector (GDV) Routing
- Performance Evaluation
- Conclusion
MDT protocols [Lam & Qian 11]

MDT protocols construct and maintain a multi-hop DT such that each node knows all its DT neighbors and has a multi-hop path to each of them.
MDT routing

For a correct multi-hop DT of a finite set of nodes in a d-dimensional Euclidean space (d ≥ 2), given a destination location ℓ, MDT forwarding succeeds to find a node in S closest to ℓ in a finite number of hops

- nodes may be specified by arbitrary coordinates
Outline

- Motivation
- MDT Protocols
- Virtual Position Construction
- Greedy Distance Vector (GDV) Routing
- Performance Evaluation
- Conclusion
Main ideas of VPOD

- Each node **chooses a position** in the virtual space.
- Nodes run MDT protocols to construct a **multi-hop DT**.
- Each node then **iteratively adjusts its position** by checking the positions of its physical and multi-hop DT neighbors.

![Diagram](image)

**new multi-hop DT**

4/11/2014  Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
Two types of adjustment by a node

- **With physical neighbors**
  - If its distance to a physical neighbor $v$ is larger than its link cost to $v$, it adjusts its position so that its distance to $v$ is smaller.

- **With multi-hop DT neighbors**
  - If its distance to a multi-hop DT neighbor $v$ is smaller (larger) than its routing cost to $v$, it adjusts its position so that its distance to $v$ is larger (smaller).
  - Adjustments with multi-DT neighbors are sufficient to preserve global relationships.
Multi-hop DT extensions

Each forwarding entry in a multi-hop DT used by VPoD is extended to a 6-tuple:

\[ \langle \text{source}, \text{pred}, \text{succ}, \text{dest}, \text{cost}, \text{error} \rangle \]

- **cost** is the link or routing cost of a link/path to \text{dest} node
- **error** is the estimated position error of \text{dest}
- \text{cost} and \text{error} fields are included in NB_SET_REQ and NB_SET_REPLY messages exchanged between a node and its DT neighbors
Multi-hop DT extensions (cont.)

- In VPoD, nodes have global IDs but their virtual positions change
  - MDT protocol running under VPoD do not need mapping from global IDs to virtual positions
  - During multi-hop DT construction, whenever a node $u$ learns a new node $x$ from node $v$, the message from $v$ to $u$ includes both the global ID and virtual position of $x$
Adjustment algorithm in detail (at node u)

Adjustment():
1. \( e_{\text{sum}} \leftarrow 0 \); // summed error of this adjustment, initialized to 0;
2. for all \( v \) in \( P_u \cup N_u \) do
3.   if \( (v \in P_u \text{ and } \tilde{D}(u, v) > D(u, v)) \text{ or } v \in N_u - P_u \) then
4.     \( t \leftarrow \text{tuple in } F_u \text{ such that } t.\text{dest} = v; \)
5.     \( e_v \leftarrow t.\text{error}; \)
6.     \( f \leftarrow e_u(e_u + e_v); \) // confidence of this update
7.     \( x_u \leftarrow x_u + c_e \times f \times [D(u, v) - \tilde{D}(u, v)] \times \hat{u}(x_u - x_v); \)
       // \( \hat{u}(x_u - x_v) \) is a unit vector in the direction of \( x_u - x_v \)
8.     \( e_{\text{sum}} \leftarrow e_{\text{sum}} + |D(u, v) - \tilde{D}(u, v)| / \tilde{D}(u, v); \)
       // add the error of this sample
9.   end if
10. end for
11. \( e_{\text{new}} \leftarrow e_{\text{sum}} / |P_u \cup N_u|; \) // average error
12. \( e_u \leftarrow e_u \times (1 - c_e) + e_{\text{new}} \times c_e; \)
13. Send the updated \( x_u \) and \( e_u \) to all nodes in \( P_u \cup N_u; \)

\( \tilde{D}(u, v) \) distance in virtual space
\( D(u, v) \) routing cost
Adjustment algorithm in detail

Adjustment():
1. \( e_{\text{sum}} \leftarrow 0; \) // summed error of this adjustment, initialized to 0;
2. for all \( v \) in \( P_u \cup N_u \) do
3.     if \( (v \in P_u \text{ and } \tilde{D}(u, v) > D(u, v)) \) or \( v \in N_u - P_u \) then
4.         \( t \leftarrow \) tuple in \( F_u \) such that \( t.\text{dest} = v; \)
5.         \( e_v \leftarrow t.\text{error}; \)
6.         \( f \leftarrow e_u/(e_u + e_v); \) // confidence of this update
7.         \( x_u \leftarrow x_u + c_c \times f \times [D(u, v) - \tilde{D}(u, v)] \times \hat{u}(x_u - x_v); \)
       // \( \hat{u}(x_u - x_v) \) is a unit vector in the direction of \( x_u - x_v \)
8.         \( e_{\text{sum}} \leftarrow e_{\text{sum}} + |D(u, v) - \tilde{D}(u, v)| / \tilde{D}(u, v); \)
       // add the error of this sample
9.     end if
10. end for
11. \( e_{\text{new}} \leftarrow e_{\text{sum}}/|P_u \cup N_u|; \) // average error
12. \( e_u \leftarrow e_u \times (1 - c_c) + e_{\text{new}} \times c_c; \)
13. Send the updated \( x_u \) and \( e_u \) to all nodes in \( P_u \cup N_u; \)

update for neighbor \( v \) that causes a position change
\( c_c \) tuning parameter
\( f \) confidence of this adjustment
Adjustment algorithm in detail

Adjustment():
1. \( e_{\text{sum}} \leftarrow 0; \) // summed error of this adjustment, initialized to 0;
2. \textbf{for} all \( v \) in \( P_u \cup N_u \) \textbf{do}
3. \hspace{1em} if \( (v \in P_u \text{ and } \hat{D}(u,v) > D(u,v)) \text{ or } v \in N_u - P_u \) then
4. \hspace{2em} \( t \leftarrow \text{tuple in } F_u \text{ such that } t.\text{dest} = v; \)
5. \hspace{2em} \( e_v \leftarrow t.\text{error}; \)
6. \hspace{2em} \( f \leftarrow e_u/(e_u + e_v); \) // confidence of this update
7. \hspace{2em} \( x_u \leftarrow x_u + c_e \times f \times [D(u,v) - \hat{D}(u,v)] \times \hat{u}(x_u - x_v); \)
   \hspace{2em} // \( \hat{u}(x_u - x_v) \) is a unit vector in the direction of \( x_u - x_v \)
8. \hspace{2em} \( e_{\text{sum}} \leftarrow e_{\text{sum}} + |D(u,v) - \hat{D}(u,v)| / \hat{D}(u,v); \)
   \hspace{2em} // add the error of this sample
9. \hspace{1em} \textbf{end if}
10. \textbf{end for}
11. \( e_{\text{new}} \leftarrow e_{\text{sum}}/|P_u \cup N_u|; \) // average error
12. \( e_u \leftarrow e_u \times (1 - c_e) + e_{\text{new}} \times c_e; \)
13. Send the updated \( x_u \) and \( e_u \) to all nodes in \( P_u \cup N_u; \)

error of this sample
average error over all neighbors
\( u \) updates its local error by a moving average

4/11/2014    Greedy Distance Vector Routing (Chen Qian and Simon S. Lam) 26
Adaptive adjustment timeout

The number of times node u runs \( Adjustment() \) is

\[
\left\lfloor \frac{T_A}{\Delta_u} \right\rfloor
\]

where \( T_A \) is the duration of adjustment period and \( \Delta_u \) is the adjustment timeout of node u

- At the beginning, using a small timeout can help nodes rapidly find approximate positions
- When positions are relatively stable, they should be refined slowly for convergence
Adaptive adjustment timeout (cont.)

We propose

\[ \Delta_u = \min(\Delta_{u0} / \bar{e}, T_a) \]

where \( \bar{e} \) is the average error of \( u \)'s physical and DT neighbors

Initially \( \bar{e} = 1 \) as position errors are all initialized to 1
Outline

- Motivation
- MDT Protocols
- Virtual Position Construction
- Greedy Distance Vector (GDV) Routing
- Performance Evaluation
- Conclusion
GDV using MDT

\[ \text{GDV}(u, t): \]
1. For each physical neighbor \( y \),
   \[ R_y \leftarrow c(u, y) + \tilde{D}(y, t); \]
2. For each multi-hop DT neighbor \( y \),
   \[ R_y \leftarrow D(u, y) + \tilde{D}(y, t); \]
3. Let \( v \) be the neighbor that minimizes \( R_y \);
4. if \( R_v < \tilde{D}(u, t) \) then
   send the packet to \( v \) directly or by the multi-hop path;
5. else
   \[ \text{MDT\_greedy}(u, t); \]
6. end if

- \( R_y \) is the routing cost from \( x \) to \( t \) via \( y \)
- \( u \) selects \( v \) from all physical and DT neighbors such that \( v \) minimizes \( R_y \)

This condition avoids routing loops

If the condition in 4 is not satisfied, use greedy forwarding in the multi-hop DT (tag packet with “recovery mode”)

Lines 4 and 5 ensure guaranteed delivery
Outline

- Motivation
- MDT Protocols
- Virtual Position Construction
- Greedy Distance Vector (GDV) Routing
- Performance Evaluation
- Conclusion
Performance metrics

- Two routing metrics: hop count and ETX

- We evaluate the following metrics for GDV
  - With hop-count as the routing metric, the routing stretch is:
    \[
    \frac{\text{number of physical links in the selected route}}{\text{number of physical links in the shortest route}}
    \]
  - With ETX as the routing metric, we evaluate the average number of transmissions to deliver a packet.
  - Storage cost: # of nodes stored per node
  - Communication cost for VPoD
Simulating the link layer

- Use a well-known link-layer simulator [Sead et al. 04] to generate the connectivity graphs in 2D and packet reception rates (PRRs)

- ETX value is the inverse of the PRR value

- We also randomly place large obstacles in some experiments
Greedy routing aware of link cost

For comparison \(\text{\textcopyright NADV [Lee et al. 05]}\)

- A node \(u\) choose as the next hop a physical neighbor that maximizes

\[
\frac{d(u,t) - d(x,t)}{c(u,x)} \quad \text{for} \quad x \in P_u
\]

where \(c(u,x)\) is link cost,
\(P_u\) is \(u\)'s set of physical neighbors

\(\text{\textcopyright No delivery guarantee}\)
Adaptive adjustment timeout

- 200-node network, VPoD uses virtual positions in 3D

![Graph](image)

- Large timeout: convergence is slower
- Small timeout: stretch fluctuates
- Adaptive timeout: stable stretch and fast convergence

(a) Metric is hop count
Adaptive adjustment timeout

- 200-node network, VPoD uses virtual positions in 3D.

**Graph:**
- Large timeout: convergence is slower.
- Small timeout: stretch fluctuates.
- Adaptive timeout: stable stretch and fast convergence.

(b) Metric is ETX
Choice of Dimensionality

- 200-node network

3D has better stretch and faster convergence than 2D. But 4D has only a little advantage over 3D in stretch and convergence time.

(a) metric is hop count

4/11/2014  Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
Obstacles - routing stretch

200-node network

GDV on VPoD in 3D performs the best, when there are more obstacles

(a) metric is hop count
Obstacles - no. of transmissions

200-node network

With ETX as routing metric, performance of GDV on VPoD in 3D and 2D are close. They are also close to optimal routing.
Storage cost

- 200-node network

Vivaldi stores 2-hop neighbors in these experiments

Storage costs of 3D and 2D are much less than that in 4D
Control message cost

200-node network

VVoD in 2D is the most efficient in communication cost, because DT in 2D has fewer neighbors per join-and-adjustment period.

4/11/2014 Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
Scalability - routing stretch

Increasing the number of nodes does not have much impact on the routing stretch of GDV

Better than MDT

(a) metric is hop count
Scalability - no. of transmissions

When the number of nodes is large, the average number of transmissions of GDV is about half of that of NADV.
Scalability - storage cost

Storage cost remains low for both MDT and GDV as no. of nodes increases
Resilience to Churn

After the 10th adjustment period, 150 nodes (out of 200) failed and 150 new nodes joined.

Routing stretch re-converges after only 2-3 adjustment periods.

(a) metric is hop count
**GDV in the design space of routing protocols**

- Any additive routing metric
- No localization

Node routing state:
- *large*
- *small*

Path quality:
- *good*
- *bad*

- Shortest path
- GDV
- Hierarchical routing
- Greedy routing

---

4/11/2014  Greedy Distance Vector Routing (Chen Qian and Simon S. Lam)
The end