Distributed Delaunay Triangulation*

Geographic Routing in $d$-dimensional Spaces with Guaranteed Delivery and Low Stretch**

*Joint work with Dong-Young Lee

**Joint work with Chen Qian

4/9/2014
Greedy Routing

- It is scalable to a large network
  - because each node stores info about its directly-connected neighbors only

- But it fails at a local minimum, where all neighbors are farther away from the destination than the node itself
Most greedy routing protocols include a recovery method

- Face routing used by GFG [Bose et al. 99] and GPSR [Karp & Kung 00]
  - for planar graphs (2D) only
  - successful planarization of a general graph requires that
    - the graph is a “unit disk” graph and
    - node location information is accurate.
  - Both assumptions are unrealistic
Delaunay triangulation (DT)?

A set of point in 2D
A triangulation of $S$

Circumcircle of this triangle is not empty
Delaunay triangulation of $S$

Circumcircle of every triangle is empty
Greedy forwarding in a DT always succeeds to find a destination node

- Theorem and proof for nodes in 2D
  [Bose & Morin 2004]
- Each node is identified by its coordinates in 2D

Multi-hop DT (Simon S. Lam)
**DT in $d$-dimensional Euclidean space**

- DT definition generalized from 2D

<table>
<thead>
<tr>
<th>2D</th>
<th>$d$-dimensional</th>
</tr>
</thead>
<tbody>
<tr>
<td>triangle</td>
<td>simplex</td>
</tr>
<tr>
<td>empty circumcircle</td>
<td>empty circum-hypersphere</td>
</tr>
</tbody>
</table>

- In any dimension, the DT of $S$ is a **graph**, denoted by $\text{DT}(S)$
  - neighbors in the graph are called **DT neighbors**
Greedy forwarding in a DT always succeeds to find a node closest to a destination location

- Theorem and proof for nodes in a \(d\)-dimensional Euclidean space, \(d \geq 2\) [Lee & Lam 2006]

- Node coordinates may be arbitrary
Distributed system model of DT

- A set $S$ of nodes in a $d$-dimensional Euclidean space
  - Each node assigns itself coordinates in the space to be used as the node's identifier
  - "$u$ knows $v$" means "$u$ knows $v$'s coordinates"

- Each node is a communicating state machine
  - a node's state is set of nodes it knows
  - protocol messages it sends and receives

No need to think about $d$-dimensional objects except when proving theorems
A distributed DT

\( C_u \) set of nodes \( u \) knows

\( DT(C_u) \) local DT computed by \( u \)

\( N_u \) neighbors of \( u \) in \( DT(C_u) \)

- The distributed DT is correct iff, for all \( u \in S \),
  \( N_u = \text{set of } u\text{'s neighbors in global DT, } DT(S) \)

- No broadcast, \( N_u \subseteq C_u \) and \( |C_u| \ll |S| \)

Multi-hop DT (Simon S. Lam)
Node $u$ finds nodes and computes its local DT

How does $u$ search?

When does $u$ stop?

$C_u=\{u, a, b, c, d\}$

$D_T(C_u)$

$N_u=\{a, b, c\}$
Application to Layer 2 routing

- Layer 2 network represented by an arbitrary graph of nodes and physical links (connectivity graph)

- Minimal assumptions:
  - graph is connected
  - each physical link is bidirectional

How to make use of distributed DT?
- connectivity graph is usually not a DT graph
Extension - Multi-hop DT

- Connectivity graph - nodes and physical links

- DT graph

- In a multi-hop DT, neighbors can be
  - directly connected
  - multiple hops apart and communicate via a virtual link

A physical link that is not a DT edge
Each node has a forwarding table

- Each entry in the forwarding table is a 4-tuple
  \[<\text{source}, \text{pred}, \text{succ}, \text{dest}>\]

- For the DT edge \(a-d\), to provide the path \(a-b-c-d\), each node stores a tuple, e.g.,
  - Node \(b\) stores \(<a, a, c, d>\)

The tuple is used by \(b\) for forwarding in both directions.
In a multi-hop DT, each node $u$

- maintains tuples in its forwarding table $F_u$ as soft state

$C_u = \text{set of destination nodes in tuples of } F_u$

$N_u = \text{set of neighbors in } DT(C_u)$

node $u$’s local DT
A multi-hop DT is correct iff

1. for all $u \in S$, $N_u = \text{set of } u\text{'s neighbors in } DT(S)$ (the distributed DT is correct)

2. for every DT edge $(u, v)$, there exists a unique $k$-hop path between $u$ and $v$ in the forwarding tables of nodes in $S$
MDT’s 2-step greedy forwarding

node $u$ receives a packet with destination $d$

**greedy step 1**

∃ a physical neighbor $v$ closest to $d$?  

- yes: transmit to $v$
- no: greedy step 2

**greedy step 2**

∃ a DT neighbor $w$ closest to $d$?  

- yes: forward to $w$  
  (using a tuple in forwarding table)
- no: node $u$ is closest to $d$
MDT's 2-step greedy - example

- Source c, dest. k
- At node c, physical neighbor closest to k is b
  - c transmits msg to b
2-step greedy example (cont.)

- Node b is a local minimum with multi-hop DT neighbor j closest to k.
- Node b forwards msg to j by transmitting it to e.
- Node e forwards msg to j by transmitting it to h.
  - h does not perform greedy step 1.
- h transmits msg to j.
- j finds itself closest to k.
In a correct multi-hop DT

- MDT’s 2-step greedy forwarding provides guaranteed delivery to a node that is closest to the destination location

Theorem and proof [Lam and Qian 2011]

We next present a join protocol for nodes to construct a correct multi-hop DT
**MDT join protocol: initial step**

- **Given:** a correct multi-hop DT of S
- **Node a boots up**

- **To join S, a needs to find the closest node in S**
  - It must be a neighbor of a in the DT of $S \cup \{a\}$
2-step greedy in existing DT finds node closest to a

- a sends JOIN_req to b with a's location as destination
  - It is greedily forwarded to node c which is closest to a
  - Each node along the path of JOIN_req stores a forwarding tuple for the path
Closest node c found

- c sends JOIN_rep to a along the reverse path
- a begins an iterative search
- a sends NB_req to c

Multi-hop DT (Simon S. Lam)
Finding more DT neighbors

- $c$ adds $a$ to its set $C_c$
- $c$ recomputes $\text{DT}(C_c)$
- Set of $a$'s new neighbors in $\text{DT}(C_c)$ is $N_a^c = \{ j, d \}$
- $c$ sends $\text{NB}_{-\text{rep}}(N_a^c)$ to $a$
Iterative search by node $u$

repeat
for all $x \in N_u^{\text{new}}$ do
remove $x$ from $N_u^{\text{new}}$
send NB_req to $x$
receive NB_rep$(N_u^x)$
$C_u = C_u \cup \{N_u^x\}$
compute DT$(C_u)$; update $N_u$
update $N_u^{\text{new}}$

node $x$
receive NB_req from $u$
$C_x = C_x \cup \{u\}$
compute DT$(C_x)$; update $N_x$
$N_u^x = u$'s neighbors in DT$(C_x)$
send NB_rep$(N_u^x)$ to $u$

until $N_u^{\text{new}}$ is empty (successfully joined)

$N_u^{\text{new}}$ new neighbors that have not been sent a NB_req

Multi-hop DT (Simon S. Lam)
Path to a multi-hop DT neighbor

- Node $a$ has learned $j$ from node $c$
  - $a$ sends NB_req
  - $a$-$c$ path has been established
  - $c$-$j$: the existing multi-hop DT is correct; a forwarding path exists between $c$ and $j$

- The virtual link $a$-$j$ is set up
Physical-link shortcut

- j received NB_req and sends NB_rep to a
- At any intermediate node along the reverse path j-h-e-c-b-
  - if a node (h in this example) finds that dest. a is a physical neighbor, the msg is transmitted directly to a
- h updates its tuple for a and j

Tuples for a and j in nodes b, c, and e will time out
When join protocol terminates the multi-hop DT of $S \cup \{u\}$ is correct

- For a single join
  - Theorem and proof [Lam and Qian 2011]

- Theorem also holds for concurrent joins that are independent

- A correct multi-hop DT can be constructed by nodes joining serially
Concurrent events

- Two practical problems
  1. At network initialization, all nodes join concurrently to construct a correct multi-hop DT
  2. Dynamic topology changes occurring at a high rate (churn)
     - Nodes
     - Links

- MDT solution - Each node runs the iterative search protocol repeatedly and asynchronously (controlled by a timer)
Initialization - Accuracy vs. time

Concurrent joins of 300 nodes in 3D, ave. msg delay = 15 ms

Each node has run iterative search 2 or 3 times

Accuracy = 1 ⇔ correct MDT

- max. token delay = 1.5s
- max. token delay = 1s
- max. token delay = 0.5s

10 sec TO
Convergence to a correct multi-hop DT

300 nodes in 3D join concurrently, 50 experiments

![Graph showing cumulative fraction of experiments against max. no. of iterative searches by a node. The graph indicates that for N = 300, the max. no. is 6.]

Multi-hop DT (Simon S. Lam)
Convergence to a correct multi-hop DT

700 nodes in 3D join concurrently, 50 experiments

Cumulative fraction of experiments vs. max. no. of iterative searches by a node for N = 300 and N = 700. The max. no. = 8.
Achieving 100% routing success rate is faster

300 nodes in 3D join concurrently, 50 experiments
Achieving 100% routing success rate is faster

700 nodes in 3D join concurrently, 50 experiments

max. no. = 4
- **500 simulation experiments**
  - 300 - 1500 nodes in 3D and 2D, ran on some difficult graphs
  - *Convergence to a correct multi-hop DT in every experiment*

- **Conjecture.** The iterative search protocol when run repeatedly by a set of nodes is **self-stabilizing.**
  - No proof, but no counter example has been found in simulations
  - What assumptions are needed?
Churn - Accuracy vs. time

300 nodes in 3D, churn rate = 20 nodes/second from time 0 to 5 sec, ave. msg delay = 15 ms

Each node has run iterative search 2 or 3 times

churn stopped

correct multi-hop DT

multi-hop DT

10 sec TO
Msg cost/node/sec vs. churn rate

300 nodes in 3D, ave. msg delay = 15 ms

- Control message cost depends more on TO interval than on churn rate
- TO interval should be adaptive

10 sec TO per iterative search

Multi-hop DT (Simon S. Lam)
Comparison of 5 protocols in 2D

Routing stretch vs. e

300 nodes with inaccurate coordinates, static topologies, density = 9.7

only for packets delivered by GPSR

Multi-hop DT (Simon S. Lam)
Initialization msg cost vs. $N$

log scale

node density = 12

MDT costs do not increase with $N$
Virtual vs. physical coordinates

Log scale

Routing stretch vs. Number of nodes

- 4D arbitrary coordinates
- 3D e=1
- 4D virtual coordinates (VPoD)

Inaccurate physical coordinates

VPoD

Multi-hop DT (Simon S. Lam)
Multi-hop DT - overview

- Nodes in a d-dimensional Euclidean space
  - Each node assigns itself coordinates in the space
  - Any connectivity graph, bidirectional links

- MDT protocols
  - 2-step greedy forwarding
  - Join protocol - each node runs iterative search once
  - Leave and failure protocols for repairing node states after a single leave or failure
  - Maintenance protocol - each node runs optimized iterative search periodically to repair node states
  - Network initialization by concurrent joins - each node runs iterative search once followed by optimized iterative search repeatedly
MDT protocols performance

- An efficient and effective search method for nodes to construct and maintain a correct multi-hop DT - *fast convergence*
- 2-step greedy forwarding provides guaranteed delivery to a node closest to a given location - *basis for a DHT*
- Scalable and highly resilient to dynamic topology changes
- Every node runs the same protocols - no special nodes
Routing applications in layer 2

- Wireless routing for nodes with inaccurate coordinates in 2D or 3D
  - Lowest routing stretch compared to other geographic routing protocols
- Wired or wireless routing using virtual coordinates
  - VPoD and GDV provide end-to-end routing cost close to that of shortest path routing [Qian & Lam 2011]
- Finding a node closest to a location in a virtual space
  - Delaunay DHT - highly resilient to churn [Qian and Lam 2012]
References


Multi-hop DT (Simon S. Lam) 45
The end