Routing algorithms

Seif Haridi
The routing problem

- Routing is the decision making procedure by which one node selects one (or more) of its neighbors to forward a packet towards its ultimate destination.
  - Routing-table computation.
  - Packet forwarding.
Criteria for good routing:

– Correctness, each packet is delivered.
– Complexity (few time, storage, messages to compute tables).
– Robustness. Table computation.
  • Changes in topology. Tables are updated when a channel/node is added/removed.
– Adaptiveness, load balancing of channels and nodes (choosing those with light load).
– Fairness in delivery of packets
Some graph theory

A path of length $k$ between $v_0$ and $v_k$ is sequence $P=\langle v_0, \ldots, v_k \rangle$ such that $v_i v_{i+1} \in E$.

A path is simple if the nodes $v_0$ through $v_k$ different.

A cycle is path of which the begin node is equal to the end node.
Graphs

- A **cycle is simple** if nodes $v_1$ through $v_k$ are different.
- The **distance** between $u$ and $v$, $d(u,v)$, is the length of the shortest path between $u$ and $v$.
- The **diameter** of a graph $G$ is the largest distance between any two nodes.
- An undirected graph is **connected** if there is path between any two nodes.
- An undirected graph is **acyclic** if it contains no simple cycles of length 3 or more.
- A **tree** is an undirected, connected, acyclic graph.
Graphs

• Trees, $G=\{v_1,\ldots,v_N\}$
  – A tree is an undirected, connected, acyclic graph.

• Equivalent statements
  – Between any node there is a unique simple path.
  – $G$ is connected but becomes disconnected if any edge is removed.
  – $G$ is connected, $|E| = N-1$.
  – $G$ is acyclic, $|E| = N-1$. 
What is a best-path algorithm

(1) Minimum Hop.
(2) Shortest path, given that each channel is assigned a weight.
(3) Minimum delay, the weight depends on the load of the channel. Tables are revised to take into account the load.
Summary

• Section 4.1
  – For minimum hop and shortest path, there are routing algorithms that routes all packets for the same destination $d$ optimally via a spanning tree rooted at $d$. The source of the packets can be ignored (destination-based routing).

• Section 4.2
  – An distributed algorithm that computes the routing table for a static network. Stores the first neighbor to each destination in the node’s routing tables. The algorithm must be recomputed on topological change in the network.

• Section 4.3

• Section 4.4
  – Coding topological information in the node addresses.
Summary

• Section 4.5
  – Hierarchical routing methods.
Destination-based routing

- Optimal routing algorithm exist if the following is satisfied:
  - The cost of sending a packet $P$ via a path is independent of the actual utilization of the path (load in involved).
  - The cost of concatenation of two paths equal the sum of the costs of the two paths:
    \[
    \text{For all } i = 0, \ldots, k
    \]
    \[
    C(\langle u_0, \ldots, u_k \rangle) = C(\langle u_0, \ldots, u_i \rangle) + C(\langle u_{i+1}, \ldots, u_k \rangle)
    \]
  - The graph does not contain any cycle of negative cost.
- A path from $u$ to $v$ is optimal if there is no path from $u$ to $v$ with lower cost.
Existence of optimal paths

• Lemma 4.1
  – Let $u, v$ be in $V$. If a path from $u$ to $v$ exists in $G$, then there is a simple path that is optimal.

• Proof
  – There is a finite number of simple paths in $G$.
  – There is a finite number of simple paths from any $u$ to $v$.
  – Choose $S$ that is minimal from $u$ to $v$.
  – For all non-simple paths $P_i$, $S$ is a lower bound.
Existence of optimal paths

Assume a non-simple path from $u$ to $v$, call it $P_0$, remove the cycles resulting in $P_N$. Then $C(S) \leq C(P_N)$.

![Diagram](image)

$V=\{v_1, \ldots, v_N\}$
Minimal Spanning Trees

Theorem 4.2  
For each $d \in V$, there exists a tree $T_d = \langle V, E_d \rangle$, $E_d \subseteq E$, and such that for each node $v \in V$, the path from $v$ to $d$ is an optimal path from $v$ to $d$ in $V$.

**Construction** 
$V = \{v_0, \ldots, v_N\}, d = v_0$
Construct a series of trees $T_0 = \langle V_0, E_0 \rangle, \ldots, T_N = \langle V_N, E_N \rangle$ with properties:
1. $T_0 = \langle \{d\}, \emptyset \rangle$.
2. $T_i$ is a tree; a sub tree of $G; V_i \subseteq V, E_i \subseteq E$.
3. $T_i$ is a subtree of $T_{i+1}$.
4. $\forall w \in V_i$, the simple path from $w$ to $d$ in $T_i$ is an optimal path from $w$ to $d$ in $G$. 

Seif Haridi
The construction

• Set \( V_0 \) to \{d\}, \( E_0 \) to \( \emptyset \).
• Construct \( T_{i+1} \) from \( T_i \): pick \( v_{i+1} \notin V_i, v_{i+1} \in V \).
• Choose an optimal path from \( v_{i+1} \) to \( d \), call it \( P \).
• \( u_l \) is the first node in the path such that \( u_l \in V_i \)
• \( V_{i+1} = V_i \cup \) the set of nodes in the prefix \( u_0, \ldots, u_{l-1} \) of \( P \).
• \( E_{i+1} = E_i \cup \) the set of edges in the prefix \( u_0, \ldots, u_{l-1} \) of \( P \).
The construction

$T_{i+1}$ is a tree; connected and the number of nodes exceeds the edges by one.

For all $w \in \{u_0, \ldots, u_{l-1}\}$, the path from $w$ to $d$ is optimal in $T_{i+1}$.

We know $C(Q) + C(P')$ is optimal. i.e. $C(Q) + C(P') \leq C(Q) + C(P'')$.

Therefore $C(P') \leq C(P'')$.

Now if assume $P''$ is better than $P'$, $C(P'') < C(P')$.

We get a contradiction.
Destination-based routing

- **Optimal sink tree for** $d$ **is a spanning tree rooted at** $d$, **where the path from any node to** $d$ **is optimal.**
- Compute the sink tree for all nodes in the network, store a table $T_u$ indexed by all destination nodes in each node $u$.
- For each node $u$, $T_u[d]$ is the parent node of $u$ in the optimal sink tree for $d$.
- **Algorithm:**
  - /* A packet with destination $d$ received or generated at node $u */
  - if $d==u$ then deliver the packet locally
  - else send the packet to $T_u[d]$ end
- The algorithm delivered each packet, because the routing tables are cycle-free.
Bifurcated Routing

- Traffic splits and takes multiple paths for each source-destination pair.
All-pairs shortest path problem

- An algorithm that computes simultaneously the routing table for all nodes in a network.
- Computes for each pair \((u,v)\) of nodes, the shortest path from \(u\) to \(v\) and stores the first channel of the path in \(u\).

Each edge \(uv\) has weight \(w_{uv}\).

Weight of a path \(\langle u_0, \ldots, u_k \rangle\) is \(\sum_{i=0}^{k-1} w_{u_iu_{i+1}}\).

The distance for \(u\) to \(v\), \(d(u,v)\), is the lowest weight of all paths from \(u\) to \(v\).
S-paths

let $S \subseteq V$.
A path $\langle u_0, \ldots, u_k \rangle$ is an $S$-path if $u_1 \in S, \ldots, u_{k-1} \in S$.
$S$-distance from $u$ to $v$, $d^S(u, v)$, is the lowest weight of any $S$-path, otherwise $\infty$.

- The algorithm starts by computing all $\emptyset$-paths, incrementally computes larger $S$-paths, and all $V$-paths are considered.

1. $d^S(u, u) = 0$.
2. for $u \neq v$, there is a $\emptyset$-path from $u$ to $v$ $\iff uv \in E$.
3. If $uv \in E$ then $d^\emptyset(u, v) = d_{uv}$.
4. If $S' = S \cup \{w\}$, then a simple path from $u$ to $v$ is either
   - an $S$-path from $u$ to $v$, or an $S'$-path from $u$ to $w$ concatenated
     by an $S'$-path from $w$ to $v$.
5. If $S' = S \cup \{w\}$, then $d^{S'}(u, v) = \min(d^S(u, v), d^{S'}(u, w) + d^{S'}(w, v))$
6. A path from $u$ to $v$ exists iff a $V$-path from $u$ to $v$ exists.
7. $d(u, v) = d^V(u, v)$. 

Seif Haridi
S-paths

$S_0 = \emptyset$

$S_1 = \{1\}$

$S_2 = \{1, 2\}$

$S_3 = \{1, 2, 3\}$
S-paths

$S_4 = \{1,2,3,4\}$

$S_5 = \{1,2,3,4,5\}$
Floyd-Warshall sequential algorithm

% Initialize S to ∅ and D to ∅ - distance
S := ∅

forall u, v do
    if u = v then D[u, v] := Ø
    elseif uv ∈ E then D[u, v] := w_{uv}
    else D[u, v] := ∞ end
end

% Expand S by pivoting

while S ≠ V do
    % Loop invariant : ∀u, v : D[u, v] = d^{S}(u, v)
    pick w from V \ S
    forall u ∈ V do
        % Execute a global w - pivot
        forall v ∈ V do
            D[u, v] := min(D[u, v], D[u, w] + D[w, v]) end
        end
    S := S ∪ {w}
end
The algorithm

Computes in $\Theta(N^3)$ steps.

% Initialize $S$ to $\emptyset$ and $D$ to $\emptyset$ - distance
$S := \emptyset$

for all $u, v$ do
  if $u = v$ then $D[u, v] := \emptyset$
  elseif $uv \in E$ then $D[u, v] := w_{uv}$
  else $D[u, v] := \infty$ end
end

% Expand $S$ by pivoting

while $S \neq V$ do
  % Loop invariant: $\forall u, v : D[u, v] = d^S(u, v)$
  pick $w$ from $V \setminus S$
  for all $u \in V$ do
    for all $v \in V$ do
      $D[u, v] := \min(D[u, v], D[u, w] + D[w, v])$ end
    end
  end
  $S := S \cup \{w\}$
end

Seif Haridi
Toueg’s shortest path

- A distributed version of Floyd and Warshall algorithm.
- Assumptions:
  - Each cycle in the network has a positive weight.
  - Each node initially knows the identities of all nodes (the set V).
  - Each node u knows its neighbors stored in Neigh_u, and the weight of outgoing channels.
  - Described in two refinement steps.

variables:

\( S_u \) : set of nodes.
\( D_u \) : array of weights.
\( Nb_u \) : array of nodes.
Version 1

\[ S_u := \emptyset \]

\textbf{for all} \( v \in V \) \textbf{do}

\begin{align*}
\text{if} \quad u &= v \quad \text{then} \quad D_u[v] := \emptyset; \quad Nb_u[v] := udef \\
\text{elseif} \quad v &\in \text{Neigh}_u \quad \text{then} \quad D_u[v] := \omega_{uv}; \quad Nb_u[v] := v \\
\text{else} \quad D_u[v] := \infty; \quad Nb_u[v] := udef \quad \text{end}
\end{align*}

\textbf{end}

\textbf{while} \( S_u \neq V \) \textbf{do} \quad \% \text{Loop invariant} : \forall u, v : D_u[v] = d^{S_u}(u, v) 

\begin{align*}
\text{pick} \ w \text{ from} \ V \setminus S_u \quad \% \text{All pick uniformly the same element} \\
\text{if} \quad u &= w \quad \text{then} \quad " \text{broadcast the} \ D_w " \\
\text{else} \quad " \text{receive the tables} \ D_w " \quad \text{end}
\end{align*}

\textbf{for all} \( v \in V \) \textbf{do}

\begin{align*}
\text{if} \quad D_u[w] + D_w[v] < D_u[v] \quad \text{then} \\
\quad D_u[v] := D_u[w] + D_w[v]; \\
\quad Nb_u[v] := Nb_u[w]
\end{align*}

\textbf{end}

\textbf{end}

\( S_u := S_u \cup \{w\} \)

\textbf{end}
• After each pivot round:

\[ \forall u, D_u[w] = d^S(u, w). \]
if \( d^S(u, w) < \infty \) and \( u \neq w \), then \( Nb_u[w] \) is the first channel of a shortest \( S \)-path to \( w \).

The directed graph \( T_w = (V_w, E_w) \) where 
\[
\begin{align*}
&u \in V_w \iff D_u[w] \neq \infty, \\
&\text{and } u x \in E_w \text{ if } x \text{ is the first channel from } u \text{ to } w, \\
&\text{is a tree rooted at } w.
\end{align*}
\]

For each destination \( w \), the nodes that computed the way to \( w \) form a spanning tree rooted at \( w \).
The improved algorithm

- At the start of the $w$-pivot round a node $u$ with $D[w]=\infty$ does not improve its table.
- Only the nodes in $T_w$ need to receive $w$’s table, to extend their table.
- The table is sent via the channels of the tree $T_w$.
- Each node knows its father in $T_w$ but not its sons, therefore sons must inform the father (needed to do the broadcast).
The skeleton at node $u$

- Initialize $D_u$ and $Nb_u$ table by self and immediate neighbors
- Start the $w$-pivot rounds, for each $w$ round:
  - Establish the son-father chain in $T_w$.
    - Send $(ys,w)$ message to the father neighbor, $(nys,w)$ to the non-father neighbors.
    - Receive $(ys,w)$ and $(nys,w)$ messages from neighbors.
- Participate in the $w$-pivot round
  - Receive $(dtab,w,D)$ from father in $T_w$ ($u\neq w$).
  - Send $(dtab,w,D)$ to sons in $T_w$.
  - Extends $D_u$ and $Nb_u$ tables
  - Extend $S_u$ with $w$. 

Seif Haridi
Messages

- \((\text{ys}, w)\): your-son message in the spanning tree of \(w\).
- \((\text{nys}, w)\): not-your-son message in the spanning tree of \(w\).
- \((\text{dtab}, w, D)\): the D-table of \(w\).
- Requires FIFO channels for not mixing rounds, or storing messages for round \(w'\) if \(w'\) is after \(w\).
Tree construction phase

\[
\text{forall } x \in \text{Neigh}_u \text{ do}
\]
\[
\quad \text{if } \text{Nb}_u[w] = x \text{ then send } \langle \text{ys}, w \rangle \text{ to } x
\]
\[
\quad \text{else send } \langle \text{nys}, w \rangle \text{ to } x \text{ end}
\]
\[
\text{end}
\]
\[
\text{num}_{\text{rec}}_u := 0
\]
\[
\text{while } \text{num}_{\text{rec}}_u < |\text{Neigh}_u| \text{ do}
\]
\[
\quad \text{receive } \langle \text{ys}, w \rangle \text{ or } \langle \text{nys}, w \rangle \text{ message}
\]
\[
\quad \text{num}_{\text{rec}}_u := \text{num}_{\text{rec}}_u + 1
\]
\[
\text{end}
\]
Broadcast and computation phase

if $D_u[w] \neq \infty$ then
    if $u \neq w$ then receive $\langle dtab, w, D \rangle$ from this $Nb_u[w]$ end
    forall $x \in \text{Neigh}_u$ do
        if $\langle ys, w \rangle$ was received from $x$ then send $\langle dtab, w, D \rangle$ to $x$ end
    end
    { *local table computation* }
end

$S_u := S_u \cup \{w\}$
Complexity

We have $N$ rounds
At each round:
At most on each edge 2 $ys/nys$ messages + 1 $dtab$ message.
Since a $dtab$ message traverses a spanning tree we have at most
$N$ $dtab$ messages in per/round and $N^2$ $dtab$ in total.
Number of message = $O(2N \cdot |E| + \frac{N^2}{dtab})$

Assume each entry (node - id or weight) takes $W$ bits.
Number of bits transferred = $O(2N \cdot W \cdot |E| + N^3) = O(N^3W)$

\[ \begin{array}{c}
\text{w} \\
\text{x} \\
\text{u}
\end{array} \quad \uparrow \quad \begin{array}{c}
\text{ys/nys} \\
\downarrow \\
\downarrow
\end{array} \quad \text{Dtab (table of N entries)} \]
From sequential to distributed algorithms

- Variables of a sequential algorithm are distributed over a number of nodes. Computation on the variables are done locally.
- Whenever a remote variable is needed communication is performed.
- Minimize amount of communication by exploiting properties of the sequential algorithm.
- Two bad properties of Toueg’s algorithm.
  - Agreement of pivot nodes require knowledge of the nodes in the system. In general we need to execute first a wave algorithm to get acquire this knowledge.
  - Requires information that is not available in the node, nor in the neighbors.
    - \( d(u,w) + d(w,v) < d(u,v) \)
Alternative solutions

Computing distances from u to v can be instead based on the following equation:

\[
d(u, v) = \begin{cases} 
0 & \text{if } u = v \\
\min_{w \in \text{Neigh}_u} (\omega_{uw} + d(w, v)) & \text{otherwise}
\end{cases}
\]

- Communication is local (only information from neighbors).
- Computing different destinations is independent.
- Requires more total computation ??.
- Locality makes it easy to design an algorithm that adapts to topological changes.
Chandy-Misra Algorithm

Variables:
\( D_u[v_0] \): weight, initially \( \infty \).
\( Nb_u[v_0] \): node, initially \( \text{undef} \).

For node \( v_0 \): (The root of the spanning tree)
\( D_{v_0}[v_0] := 0 \)
\( \text{forall } w \in \text{Neigh}_{v_0} \text{ do} \)
    \( \text{send } \langle \text{mydist}, v_0, 0 \rangle \)
\( \text{end} \)

Processing a \( \langle \text{mydist}, v_0, d \rangle \) from neighbor \( w \) to \( u \)
receive \( \langle \text{mydist}, v_0, d \rangle \) from \( w \)
\( \text{if } d + \omega_{uw} < D_u[v_0] \text{ then} \)
    \( D_u[v_0] := d + \omega_{uw} \)
    \( Nb_u[v_0] := w \)
    \( \text{forall } x \in \text{Neigh}_u \text{ do } \text{send } \langle \text{mydist}, v_0, D_u[v_0] \rangle \text{ to } x \text{ end} \)
\( \text{end} \)
Reasoning

\( \forall j \leq N - 1 \), a configuration is reached where
\( \forall i < j, D_{v_i}[v_0] \leq d(v_i, v_0) \)

Proof by induction, assume it holds for \( i < j \), and prove it for \( i < j + 1 \).

Example:
assume it hold for \( v_0, \ldots, v_4 \), consider \( v_5 \)

The complete algorithm contains also termination detection mechanism.

\( O(N \cdot |E|) \) of messages to compute \( v_0 \).
\( O(N^2 \cdot |E|) \) of messages to compute all nodes.
\( O(N^2 \cdot W \cdot |E|) \) bits total.
Netchange algorithm

• Assumptions
  – The nodes know the size of the network (N).
  – The channels are fifo.
  – Nodes are notified of failure and repair of their adjacent channels.
  – The cost of the path equals the number of channels in the path.
  – Failure of a node is observed as a failure of its connecting channels.

• If the topology of the network remains constant after a finite number of topological changes, the algorithm terminates after a finite number of steps.

• when the algorithm terminates the following holds for node $u$:
  – $Nb_u[v] = \text{local}$, if $u = v$,
  – $Nb_u[v] = w$, where $w$ is the first neighbor on a shortest path to $v$
  – $Nb_u[v] = u\text{def}$, if there is no path from $u$ to $v$
Description

$Neigh_u$ the neighbors of $u$

$D_u[v]$ estimate of $d(u,v)$

$D_u$ : array of $1 \cdots N$

$Nb_u[v]$ preferred neighbor in path to $v$

$ndis_u[w,v]$ estimate of $d(w,v)$

$d(u,v) = 1 + \min_{w \in Neigh_u} d(w,v)$

Has to be maintained

- Network of N nodes
- Initial estimate of $d(u,u)=0$, $d(u,v)=N$ where $u \neq v$.
- Maintains an estimate of each neighbor’s distance to $v$, initially N.
- Initially $(\text{mydist},u,0)$ is sent to all neighbors.
If the estimate \( n\text{dist}[w,v] \) is different from \( d \):
- \( d(u,v) \) is recomputed
- if \( d(u,v) \) has changed, \((\text{mydist},v,d)\) is sent to all neighbors.
channel failure

- Messages may be lost, therefore distance to all nodes have to be recomputed after removing $w$ as neighbor.
channel repair

• $u$ uses $N$ as an estimate of $d(w, v)$
• $u$ sends its estimate $d(u, v)$ for all $v$. 
Variables

$Neigh_u$ the neighbors of $u$

$D_u[v]$ estimate of $d(u, v)$
$D_u$ : array of $1 \cdots N$

$Nb_u[v]$ preferred neighbor in path to $v$

$ndis_u[w, v]$ estimate of $d(w, v)$

Initializations

forall $w \in Neigh_u, v \in V$ do $ndis_u[w, v] := N$ end
forall $v \in V$ do
  $D_u[v] := N; Nb_u[v] := undef$
end
dforall $w \in Neigh_u$ do send $\langle \text{mydis}, u, 0 \rangle$ to $w$ end
Recompute($v$)

if $u = v$ then $D_u[v] := 0; \text{Nb}_u[v] := \text{local}$
else
    $d := 1 + \min\{\text{ndis}_u[w,v] : w \in \text{Neigh}_u\};$
    if $d < N$ then
        $D_u[v] := d; \text{Nb}_u[v] := w \text{ s.t.} 1 + \text{ndis}_u[w,v] = d$
    else
        $D_u[v] := N; \text{Nb}_u[v] := \text{undef}$
    end
end
if $D_u[v]$ has changed then
    forall $x \in \text{Neigh}_u$ do send $\langle\text{mydist,} v, D_u[v]\rangle$ to $x$ end
end
Netchange part two

Node $u$ receiving the message $\langle \text{mydist}, v, d \rangle$ from neighbor $w$:
receive $\langle \text{mydist}, v, d \rangle$ from $w$;
$ndis_{u}[w,v] = d$; recompute($v$)

Failure of a channel $uw$:
receive $\langle \text{fail}, w \rangle$;
$\text{Neigh}_{u} := \text{Neigh}_{u} \setminus \{w\}$;
forall $v \in V$ do recompute($v$) end

Upon repair of channel $uw$:
receive $\langle \text{repair}, w \rangle$;
$\text{Neigh}_{u} := \text{Neigh}_{u} \cup \{w\}$;
forall $v \in V$ do
  $ndis_{u}[w,v] = N$; send $\langle \text{mydist}, v, D_{u}[v] \rangle$ to $w$
end
Tree labeling scheme

- Exploit the destination address in the packet to reduce table size.
- Tree labeling scheme routes packets in certain address interval through one channel.
- Assume the network has a tree structure (or route via a logical tree structure, e.g. a spanning tree for a fixed root).
Tree Labeling

- Nodes are labeled in a pre-order way, (root, left subtree, right subtree).
- This classifies packets into class according to intervals modulo the N (the number of nodes).
- Not good if the network is general:
  - some channels are not used
  - leads to congestion
  - single point of failure partitions the network.
- Interval routing extends the scheme so that (almost) every channel is used.