Election in Mesh, Cube and Complete Networks

T-79.4001 Seminar on Theoretical Computer Science

Heikki Kallasjoki

28.2.2007
Outline

Meshes and Tori
Mesh
Oriented Torus
Unoriented Torus

Hypercubes
Oriented Hypercube
Unoriented Hypercube

Complete Networks
Complete Networks with Arbitrary Labelings
Complete Networks with Chordal Labeling
Notation

- $n$ is the number of nodes, $m$ the number of edges
- $\mathcal{N}(x)$ denotes the neighbors of node $x$
- $\mathcal{M}[\text{Alg}], \mathcal{T}[\text{Alg}], \mathcal{B}[\text{Alg}]$: message, time and bit costs
- Standard restrictions for election:
  - $\mathcal{IR} = \{\text{Initial Distinct Values}\} \cup \mathcal{R}$
  - $\mathcal{R} = \{\text{Bidirectional Links, Connectivity, Total Reliability}\}$
Outline

Meshes and Tori
   Mesh
   Oriented Torus
   Unoriented Torus

Hypercubes
   Oriented Hypercube
   Unoriented Hypercube

Complete Networks
   Complete Networks with Arbitrary Labelings
   Complete Networks with Chordal Labeling
An $a \times b$ mesh contains $n = ab$ nodes of three types:

- 4 corner nodes with two neighbors
- $2(a + b - 4)$ border nodes with three neighbors
- $n - 2(a + b - 2)$ interior nodes with four neighbors

Can be either unoriented or oriented
Election in an Unoriented Mesh

- Actual election can happen in the outer ring, with corner nodes as the only candidates

- Election process:
  1. Wake-up, started by $k_*$ initiators: initiators send wake-up to all neighbors, noninitiators forward, at most $3n + k_*$ messages
  2. Election in the outer ring with the Stages protocol, two stages so at most $6(a + b) - 16$ messages
  3. Termination notification sent by the leader, at most $2n$ messages

- Total cost at most $6(a + b) + 5n + k_* - 16$ messages

- Possible to save $2(a + b - 4)$ messages, so

$$M[ElectMesh] \leq 4(a + b) + 5n + k_* - 8$$
Message Cost of the Actual Election in MeshElect

- Each election stage requires $2n'$ messages, where $n' = 2(a + b - 2)$ is the length of the outer ring.
- In the first stage there are also unnecessary $2(a + b - 4)$ messages to interior nodes, because the border nodes do not know which links are part of the border.
- In Stages the number of candidates is at least halved every time, so for four corners only two stages are needed.
- Maximum amount of messages for the election process is therefore

\[ 4(a + b - 2) + 2(a + b - 4) = 6(a + b) - 16 \]
Election in an Oriented Mesh

- Trivial to select an unique node, for example the single “north-east” corner of the mesh
- Only wake-up needed, can be done in fewer than $2n$ messages
- Whether the mesh is oriented or not, a leader can be elected with $O(n)$ messages
- No election protocol can use fewer than $n$ messages, so

$$\mathcal{M}(\text{Elect/IR} ; \text{Mesh}) = \Theta(n)$$
Topology of a Torus

- Mesh with a “wrap-around”
- An $a \times b$ torus contains $n = ab$ nodes, each node has four neighbors
- In an oriented torus the links are **consistently** labeled as “east”, “west”, “north”, “south”

**Figure:** A $4 \times 5$ torus
Election in an Oriented Torus

- Election in an oriented torus uses electoral stages combined with marking of territory.
- In stage $i$ each candidate marks the border of a rectangular region of size $d_i$ in the torus; $d_i = \alpha^i$ for some $\alpha > 1$.
- The marking is done by sending a message which travels first $d_i$ steps north, then east, south, west.
- The candidate survives to the next stage, if either
  - The marking message does not encounter anyone in stage $i$.
  - The marking message encounters a border of a candidate with a larger id, and the candidate also receives a note that its border has been seen by a larger id.
Correctness and Cost of *MarkBoundary*

- At least one candidate (with the smallest id) survives
- After $p > \lceil \log(2 - \alpha^2)^{-1} \rceil$ additional stages after wraparouned, there is only one candidate left
- With $\alpha \approx 1.1795$, $\mathcal{M}[\text{MarkBoundary}] = \Theta(n)$
Unoriented Torus

- `MarkBoundary` can also be used in an unoriented torus
- A candidate needs to mark off a square of any orientation
- Two operations needed:
  - Forwarding a message “in a straight line”
  - Making the “appropriate turn” consecutively
Outline

Meshes and Tori
  Mesh
  Oriented Torus
  Unoriented Torus

Hypercubes
  Oriented Hypercube
  Unoriented Hypercube

Complete Networks
  Complete Networks with Arbitrary Labelings
  Complete Networks with Chordal Labeling
Topology of an Oriented Hypercube

- A $k$-dimensional hypercube $H_k$ has $n = 2^k$ nodes
- Removing all links with labels greater than $i$ from $H_k$ results in $2^{k-i}$ disjoint hypercubes $H_i$, denoted $H_{k:i}$
Topology of an Oriented Hypercube

A $k$-dimensional hypercube $H_k$ has $n = 2^k$ nodes.

Removing all links with labels greater than $i$ from $H_k$ results in $2^{k-i}$ disjoint hypercubes $H_i$, denoted $H_{k:i}$.

Figure: $2^{4-3} = 2$ disjoint hypercubes $H_3$
Topology of an Oriented Hypercube

A $k$-dimensional hypercube $H_k$ has $n = 2^k$ nodes

- Removing all links with labels greater than $i$ from $H_k$ results in $2^{k-i}$ disjoint hypercubes $H_i$, denoted $H_{k:i}$

Figure: $2^4 - 2 = 4$ disjoint hypercubes $H_2$
Topology of an Oriented Hypercube

A $k$-dimensional hypercube $H_k$ has $n = 2^k$ nodes.

Removing all links with labels greater than $i$ from $H_k$ results in $2^{k-i}$ disjoint hypercubes $H_i$, denoted $H_{k:i}$.

Figure: $2^{4-1} = 8$ disjoint hypercubes $H_1$
Election in an Oriented Hypercube

- The *HyperElect* protocol uses **electoral stages**
- At each stage, every **candidate (duelist)** is paired with another duelist and will have a match (id comparison) with it; only one survives to the next stage
- At the end of stage $i - 1$, only one duelist will be left in each of the separate hypercubes $H_{k:i-1}$
- For stage $i$, the opponent of each duelist can be found from the $(i - 1)$-dimensional hypercube behind link $i$
- The defeated nodes remember the shortest path to the winner, so that further duels can be done efficiently (without flooding)
- Messages “from the future” need to be delayed locally
Practical Considerations for *HyperElect*

- The defeated nodes need the shortest path to the winner
- This is accomplished by recording paths in the messages
- In a hypercube, paths containing any pair of identical labels are equivalent to the paths with those labels removed; ⟨231345212⟩ leads to the same place as ⟨245⟩
- In a $k$-dimensional hypercube maximum path length is $k$
- Because the path elements are unique integers between 1 and $k$, with no repetitions, the path can be stored as a single $k$-bit integer
- $k = \log n$, so the path does not use more bits than a counter
Correctness and Costs of *HyperElect*

- *HyperElect* terminates when a duelist wins the $k$th stage
- Correctness depends on the following fact: (proof by omission)
  
  Let $\text{id}(x)$ be the smallest id in one of the hypercubes of dimension $i$ in $H_{k:i}$. Then $x$ is a duelist at the beginning of stage $i + 1$.

- At most $1 + \frac{i \cdot (i-1)}{2}$ messages required for a match message

- In stage $i$ there are $n_i = 2^{k-i+1}$ duelists (one for each hypercube $H \in H_{k:i-1}$)

- Summing the messages over all stages, and adding the termination broadcast, we get:

  $$M[\text{HyperElect}] \leq 7n - (\log n)^2 - 3 \log n - 7$$
Election in an Unoriented Hypercube

- *HyperElect* obviously will not work for hypercubes with arbitrary labelings
- It is still possible to do better than in rings:

\[ M(\text{Elect}/\text{IR}; \text{Hypercube}) \leq O(n \log \log n) \]

(Problem 3.10.8)
- It is not known whether it can be done in \( O(n) \) messages
(Problem 3.10.9)
Outline

Meshes and Tori
- Mesh
- Oriented Torus
- Unoriented Torus

Hypercubes
- Oriented Hypercube
- Unoriented Hypercube

Complete Networks
- Complete Networks with Arbitrary Labelings
- Complete Networks with Chordal Labeling
Election in a Complete Network

- *CompleteElect* is based on both electoral stages and territory acquisition
- Nodes are either candidates, captured or passive
- Each candidate tries to capture all other nodes, one at a time
- To capture nodes the candidate sends them a message containing its own id and the number of nodes captured (the stage)
- Node $x$ succeeds in capturing node $y$ when:
  - $y$ is a candidate and either in a lower stage, or in the same stage but with a larger id
  - $y$ is passive
  - $y$ is captured, and $x$ could capture its current owner
- If the attack fails, $x$ becomes passive
The \textit{CompleteElect} Protocol

\begin{align*}
S &= \{\text{ASLEEP}, \text{CANDIDATE}, \text{PASSIVE}, \\
& \quad \text{CAPTURED}, \text{FOLLOWER}, \text{LEADER}\}; \\
S_{\text{INIT}} &= \{\text{ASLEEP}\}; \\
S_{\text{TERM}} &= \{\text{FOLLOWER}, \text{LEADER}\}. \\
\text{Restrictions:} & \quad \text{IR} \cup \text{CompleteGraph}.
\end{align*}

\textbf{ASLEEP}

\textit{Spontaneously}

\begin{align*}
\text{begin} \\
\text{stage} := 1; \text{value} := \text{id}(x); \\
\text{Others} := N(x); \\
\text{next} & \leftarrow \text{Others}; \\
\text{send}(& \text{"Capture", stage, value}) \text{ to next}; \\
\text{become} \text{ CANDIDATE}; \\
\end{align*}

\textit{Receiving("Capture", stage*, value*)}

\begin{align*}
\text{begin} \\
\text{if} & \quad (\text{stage} < \text{stage*}) \text{ or } ((\text{stage} = \text{stage*}) \text{ and } \text{value} > \text{value*}) \text{ then} \\
& \quad \text{send("Reject", stage) to sender;} \\
\text{else} & \quad \text{send("Accept", stage*, value*) to sender;} \\
& \quad \text{owner} := \text{sender}; \\
& \quad \text{ownerstage} := \text{stage*} + 1; \\
& \quad \text{become} \text{ CAPTURED}; \\
\end{align*}

\textbf{CANDIDATE}

\textit{Receiving("Capture", stage*, value*)}

\begin{align*}
\text{begin} \\
\text{if} & \quad (\text{stage} < \text{stage*}) \text{ or } ((\text{stage} = \text{stage*}) \text{ and } \text{value} > \text{value*}) \text{ then} \\
& \quad \text{send("Reject", stage) to sender;} \\
\text{else} & \quad \text{send("Accept", stage*, value*) to sender;} \\
& \quad \text{owner} := \text{sender}; \\
& \quad \text{ownerstage} := \text{stage*} + 1; \\
& \quad \text{become} \text{ CAPTURED}; \\
\end{align*}

\textit{Receiving("Accept", stage, value*)}

\begin{align*}
\text{begin} \\
\text{stage} := \text{stage} + 1; \\
\text{if} & \quad \text{stage} \geq 1 + n/2 \text{ then} \\
& \quad \text{send("Terminate") to } N(x); \\
& \quad \text{become} \text{ LEADER}; \\
\text{else} & \quad \text{next} \leftarrow \text{Others}; \\
& \quad \text{send("Capture", stage, value*) to next;} \\
\end{align*}

\textit{Receiving("Capture", stage, value*)}

\begin{align*}
\text{begin} \\
\text{send}(& \text{"Accept", stage*, value*) to sender; \\
\text{stage} := 1; \\
\text{owner} := \text{sender}; \\
\text{ownerstage} := \text{stage*} + 1; \\
\text{become} \text{ CAPTURED}; \\
\end{align*}
The CompleteElect Protocol, cont.

CANDIDATE

Receiving("Reject", stage*)
begin
    become PASSIVE;
end

Receiving("Terminate")
begin
    become FOLLOWER;
end

Receiving("Warning", stage*, value*)
begin
    if (stage* < stage) or ((stage* = stage) and (value* > value)) then
        send("No", stage) to sender;
    else
        send("Yes", stage*) to sender;
    become PASSIVE;
end

PASSIVE

Receiving("Capture", stage*, value*)
begin
    if (stage* < stage) or ((stage* = stage) and (value* > value)) then
        send("Reject", stage) to sender;
    else
        send("Accept", stage*, value*) to sender;
        owner:= sender;
        ownerstage:= stage* +1;
        become CAPTURED;
end

end

Receiving("Warning", stage*, value*)
begin
    if (stage* < stage) or ((stage* = stage) and (value* > value)) then
        send("No", stage) to sender;
    else
        send("Yes", stage*) to sender;
    become PASSIVE;
end

end

Receiving("Terminate")
begin
    become FOLLOWER;
end
The *CompleteElect* Protocol, cont.

**Receiving**("Capture", stage*, value*)

begin
  if stage* < ownerstage then
    send("Reject", ownerstage) to sender;
  else
    attack := sender;
    send("Warning", value*, stage*) to owner;
    close N(x) −{owner};
  end
end

**Receiving**("No", stage*)

begin
  open N(x);
  send("Reject", stage*) to attack;
end

**Receiving**("Yes", stage*)

begin
  ownerstage := stage*+1;
  owner := attack;
  open N(x);
  send("Accept", stage*, value*) to attack;
end

**Receiving**("Warning", stage*, value*)

begin
  if (stage* < ownerstage) then
    send("No", ownerstage) to sender;
  else
    send("Yes", stage*) to sender;
  end
end

**Receiving**("Terminate")

begin
  become FOLLOWER;
end
Efficiency of *CompleteElect*

- With suitable tweaks to ensure that territories of candidates in stage $i$ remain disjoint, the overall costs will be:
  \[ M(\text{CompleteElect}) \leq 2.76n \log n - 1.76n + 1 \]
  \[ T(\text{CompleteElect}) = O(n) \]
- There is a simple strategy for $O(1)$ time and $O(n^2)$ messages
- Combining the two results in a protocol using $O(n \log n)$ messages and $O(n/ \log n)$ time (Exercise 3.10.68)
- Even more generally, $O(nk)$ messages and $O(n/k)$ time for any $\log n \leq k \leq n$ is achievable (Exercise 3.10.69)
A “Surprising Limitation”

- The $O(n \log n)$ $\text{CompleteElect}$ is no better than election protocols in rings, and even has a worse constant factor.
- In fact,
  \[ \mathcal{M}(\text{Elect/IR} ; K) = \Omega(n \log n) \]
- Justification: any election protocol also solves $\text{wake-up}$, which has a lower bound of $0.5n \log n$ messages.
Chordal Labeling

The complete graph $K_n$ can be viewed as a ring, with additional links (chords) added between nonneighbors.

Port labeling is chordal, if the label for link $(x, y)$ at $x$ is simply the clockwise distance from $x$ to $y$ in the ring.

Figure: Complete graph $K_5$ with chordal labeling
Election in a Complete Graph with Chordal Labeling

- Links labeled 1 and $n - 1$ form a ring, so any ring election protocol can be used directly.
- Basic idea: add a distance counter to the Election messages in Stages.
- When the distances are known, defeated nodes can be directly bypassed.
- End result: each election stage is executed in a smaller ring.
- Message costs:
  \[
  M[K_{\text{elect}} - \text{Stages}] < 7n
  \]
- Using \textit{Alternate} instead of \textit{Stages} uses even less messages.
Summary

Meshes

*ElectMesh* uses $O(n)$ messages in an unoriented mesh. Oriented meshes are even easier, so $\mathcal{M}(\text{Elect/IR} ; Mesh) = \Theta(n)$.

Tori

*MarkBoundary* works in oriented as well as unoriented tori, and has a message complexity of $\Theta(n)$, so $\mathcal{M}(\text{Elect/IR} ; Torus) = \Theta(n)$.

Hypercubes

*HyperElect* uses $O(n)$ messages in an oriented hypercube. $\mathcal{M}(\text{Elect/IR} ; Hypercube) \leq O(n \log \log n)$, not known if an $O(n)$ protocol exists for unoriented cubes.

Complete networks

$O(n \log n)$ messages for *CompleteElect*, $O(n)$ possible with chordal labeling.