Distributed Computation in Dynamic Networks

Fabian Kuhn, Nancy Lynch, Rotem Oshman

Presented by: Michal Dory
Why dynamic networks?

- Can be used to model various networks: mobile networks, wireless networks.
- Can be used to model failures in the network.
The Dynamic Graph Model

- A fixed set of nodes
- Synchronous rounds
- Communicate by broadcast
- In each round the communication graph is chosen adversarially
- The communication graph is always connected
The Dynamic Graph Model (cont.)

- In every round the adversary first chooses the edges for the round, it can see the nodes’ internal states.
- At the same time each node chooses which message to send in the round, without knowing which edges were chosen by the adversary.
The Dynamic Graph Model (cont.)

- Nodes initially know nothing about the network
- Messages size is $O(\log n)$ bits
- Nodes have unique identifiers that can be represented in $O(\log n)$ bits
- All nodes start the computation in the same round
What can we compute in this model?

- **Example:** assume that one node has a token, and it want to circulate it in the network, such that at every round exactly one node has the token, and eventually all the nodes have the token in some round.

- Can we solve this problem in our model?
No!

- Assume that we have 3 nodes: u, v, w and at the beginning u has the token:
No!

- Assume that we have 3 nodes: $u,v,w$ and at the beginning $u$ has the token:
No!

- It can pass it to $v$: 

![Graph with nodes u, v, and w]

- It can pass it to $v$: 

![Graph with nodes u, v, and w]
No!

- It can pass it to $v$: 

```
  w
 /|
| v|
|
 u
```
No!

- In the next round the adversary can choose different edges:

  ![Diagram]
  
  - w
  - u
  - v
No!

- In the next round the adversary can choose different edges:
No!

- Now $v$ can pass the token only to $u$.
- The adversary can continue in the same way, and $w$ will never receive the token.
No!

- Now $v$ can pass the token only to $u$.
- The adversary can continue in the same way, and $w$ will never receive the token.
No!

- Now v can pass the token only to u.
- The adversary can continue in the same way, and w will never receive the token.
Problem Definitions

- **Counting** - An algorithm is said to solve the counting problem if whenever executed in a dynamic graph with \( n \) nodes, all nodes eventually terminate and output \( n \).

- **\( k \)-verification** - Closely related to counting. Now all the nodes begin with \( k \) as their input, and their goal is to decide whether \( n \leq k \).

Note: if we can solve the \( k \)-verification problem we can solve counting.
Problem Definitions (cont.)

- **$k$-token dissemination** - At the beginning there are $k$ tokens distributed in the network, and the goal is that all the nodes know about all the tokens by the end of the algorithm. We assume that each token can be represented by $O(\log n)$ bits.

- **All-to-all token dissemination** - A special case where $k=n$ and each node initially knows exactly one token.

Note: if we can solve all-to-all token dissemination we can compute any computable function of the initial inputs of the nodes (assuming initial inputs with size $O(\log n)$ bits).
Results:

- There is an $O(n^2)$-round protocol for the counting problem and to all-to-all token dissemination.
- If we do not restrict the message size, we can solve these problems with $O(n)$ rounds.
Basic Facts

- **Proposition:** It is possible to solve 1-token dissemination in $n - 1$ rounds, if nodes are not required to halt after they output the token.

- **The algorithm:** All nodes that know the token broadcast it in every round. When a node receives the token, it outputs it immediately, but continues broadcasting it.
Correctness proof of the algorithm:

- We will show that after round $i \leq n - 1$, at least $i + 1$ nodes know about the token. So in particular after $n - 1$ rounds all nodes know the token.

- The proof is by induction:
  
  For $i = 0$, we know that at the beginning one node knows about the token.

  **Induction step:** assume correctness for $i - 1$. If after $i - 1$ rounds, all nodes know the token we are done. Otherwise, there is at least one node that does not know the token at the beginning of round $i$. We will show that at least one node learns about the token in round $i$. 
Correctness proof of the algorithm: (cont.)

- Let $A$ be the set of nodes that know the token at the beginning of round $i$ and $B = V \setminus A$.

- $A$ and $B$ are both non-empty, so from the connectivity of the graph in round $i$, there must be an edge between a node from $A$, and a node from $B$ in round $i$. So a new node learns about the token in round $i$, as needed.
Notes:

- After $n - 1$ rounds all nodes know the token, if they do not know $n$ or an upper bound on $n$ they do not know if all nodes know the token.
- Given an upper bound $N$ on the size of the network, functions such as the minimum or maximum of inputs to the nodes can be computed in $N-1$ rounds. One application is leader election.
- Counting and all-to-all token dissemination can be solved in $O(n)$ rounds using messages of size $O(n \log n)$.
  
  The idea: all the nodes constantly broadcast all the information they have collected so far.
k-Committee Election

- In this problem, nodes must partition themselves into sets, called *committees*, such that:
  1. The size of each committee is at most $k$.
  2. If $n \leq k$ there is just one committee containing all nodes.

- Each committee has a unique committee ID, and the goal is for all nodes eventually terminate and output a committee ID such that the two conditions are satisfied.
Counting through k-committee

- Note that given $k$, we can use a $k$-committee protocol, in order to determine if $n \leq k$ (k-verification).
- $n \leq k$ iff there is only one committee at the end of the protocol.
- We next present a $k$-round protocol that checks if there is more than one committee in the network.
Each node has a local variable $x$, which is initially set to 1.

While $x_u = 1$ node $u$ broadcasts its committee ID.

If it hears from some neighbor a different committee ID from its own, or the special value $\bot$, it sets $x_u = 0$ and broadcasts $\bot$ in all subsequent rounds.

After $k$ rounds all nodes output the value of their $x$ variable.
Correctness proof of the protocol:

- **Lemma:** If the initial state of the execution represents a solution to $k$-committee election, at the end of the $k$-verification protocol each node outputs 1 iff $n \leq k$.

- **Proof:** If $n \leq k$ there is only one committee, so no node ever hears a different committee ID from its own, at the end of the protocol all nodes still have $x = 1$, and all output 1.
Correctness proof of the protocol: (cont.)

- Assume now that $k < n$.
- Let’s look at a particular committee $C$, such that $|C| = m \leq k$.
- We will show that after round $i$ of the protocol for $i \leq m$, at least $i$ nodes in $C$ have $x = 0$. So after $m$ rounds (and also after $k$ rounds) all nodes in $C$ have $x = 0$.
- So after $k$ rounds all nodes have $x = 0$, and output 0 as needed.
- Denote by $A$ all the nodes in $C$ that have $x = 1$ at the beginning of round $i$. If $A = \emptyset$ we are done. Let $B = V \setminus A$. Note that $k < n$, so $B \neq \emptyset$. 
Correctness proof of the protocol: (cont.)

- $A$ and $B$ are both non-empty, so from the connectivity of the graph in round $i$, there must be an edge between a node from $A$, and a node from $B$ in round $i$.

- So there is a node $u \in A$, such that $u \in C$ and has $x_u = 1$ at the beginning of round $i$, that is connected to a node $v$ that is in a different committee or has $x_v = 0$. So $u$ hears a different committee ID from its own, or the special value $\bot$, and it sets $x_u = 0$ as needed.
Counting through k-committee (cont.)

- So we can use a k-committee protocol, in order to determine if $n \leq k$.
- We can use that in order to solve counting:
  
  We will solve k-committee for the values $k=1,2,4,8,...$ until we get a value such that $n \leq k$, in that case we will know $n$ (The k-committee protocol we will see has the property that if $n \leq k$, every node knows the IDs of all other nodes in the graph at the end of the protocol).
A k-Committee Election Protocol

- We will show $O(k^2)$-round protocol for k-committee election.
- **The idea:** assume we have a leader in the network, and it invites k nodes to join its committee. Nodes that did not receive invitations create committees of their own.
- We do not really have a pre-elected leader. Initially all nodes consider themselves leaders, but throughout the protocol, any node that hears an ID smaller than its own adopts this ID as its leader. $Leader_u$ contains the smallest ID that $u$ have heard so far.
- A node that has not yet joined a committee is called *active*, and a node that has joined a committee is *inactive*. Once nodes join a committee they do not change their choice.
A k-Committee Election protocol (cont.)

- The protocol proceeds in $k$ cycles, each consisting of two phases:
  - **Polling phase:** for $k-1$ rounds, all nodes propagate the ID of the smallest active node of which they are aware.
  - **Invitation phase:** each node that considers itself a leader, selects the smallest ID it heard in the previous phase and sends a message inviting that node to its committee. All nodes propagate the smallest invitation of which they are aware of for $k-1$ rounds. A node that receives an invitation to join its leader’s committee does so and becomes inactive.
  - At the end of the $k$ cycles, any node that has not been invited to join a committee, join its own committee.
Correctness proof of the protocol:

- It is immediate to see that the protocol takes $O(k^2)$ rounds.

- The size of each committee is at most $k$: after $k$ rounds at most $k$ nodes joined each committee (note that the first invitation is always from a leader to itself), so the size of each committee is at most $k$.

- If $n \leq k$ there is just one committee containing all nodes: note that if $n \leq k$, we can propagate one piece of information in the network in $k-1$ rounds (as seen before).
Correctness proof of the protocol: (cont.)

- So after the first polling phase all the nodes hear the ID of the smallest node in the network, so there is only one leader for the rest of the protocol. In particular, no other node ever sends an invitation.

- \( k-1 \) rounds of each polling phase are sufficient for the leader to successfully identify the smallest node that has not yet joined its committee.

- The invitation phase is long enough for that node to receive the leader’s invitation.

- Since \( n \leq k \), eventually all the nodes join the leader’s committee.
This protocol gives $O(n^2)$ protocol for counting.

When $n \leq k$, the protocol can also be used to solve all-to-all token dissemination. To do so we simply have nodes attach their token to their ID in every message they send. At the end all the nodes see the tokens (and IDs) of all the nodes in the network.

The algorithm relayed on the fact that after $n-1$ rounds we can propagate one piece of information in the network, in more stable graphs we can do better.
Conclusion

- We saw a model for dynamic networks which makes very few assumptions about the network.
- We talked about the problems we can solve in this model.
- We saw an $O(n^2)$-round protocol for counting and all-to-all token dissemination.