Implementing the Reliable Broadcast Service in Dynamic Distributed Networks

April 22, 2014

Welles Robinson

Senior Thesis

Department of Computer Science

Georgetown University

Abstract

Abstract

1 Introduction

In the last decade, the importance of distributed computing has grown rapidly as a result of the enormous increase in the amount of data being processed and shared. However, distributed systems only works as well as their underlying algorithms. For example, systems like Hadoop, Akamai and BitTorrent, which have seriously impacted big data storage, the Internet and the music industry respectively, all rely on provably correct distributed algorithms.

Reflecting the aforementioned examples, the majority of the work in distributed algorithms has focused on what we term *static distributed networks*, which assume a relatively stable network topology and set of participants. These algorithms are designed to handle node failures but not changes to the network topology or participant set. This choice is logical when designing peer-to-peer networks overlaid on the Internet, which guarantees reliable connections between nodes. Similarly, this design decision makes sense in the context of big data centers where computers are hard wired together in a fixed topology.

However, if distributed algorithms are to play a large role in the exciting future of wireless technology, it is imperative to develop algorithms for what we term *dynamic distributed networks*, which allow for variable network topologies and participant sets. The core appeal of wireless devices is their ability to move and maintain connectivity. However, the movement of nodes will cause changes to the network topology and participant set, both of which are not handled by *static distributed networks*. Many potential applications of distributed systems exist in this rapidly growing field. For example, consider the increase in household devices with wireless capabilities, which is commonly referred to as the "Internet of things". Many of these devices move location frequently, but still need to engage in reliable, wireless coordination with the other scattered devices.

The rise of powerful smartphones portend another enormous need for algorithms in *dynamic distributed networks*. Currently, nearly all wireless communication passes through cellular towers. Distributed computing in a peer-to-peer networking between cell phones would increase the effective range of these towers and as well as the performance and efficiency of cellular networks. These are important considerations given the increasing strain on cellular networks. Even data centers, long the domain of static distributed algorithms, have begun experimenting with wireless connectivity to reduce power use and cost. The potential of distributed algorithms in the growing field of wireless technology motivates the need for significant work on algorithms for *dynamic distributed networks*.

This thesis implements the reliable broadcast service in *dynamic distributed networks*. Reliable broadcast requires nodes to disseminate messages, which are processed in the same order by all nodes in the network. The reliable broadcast problem is a key primitive for implementing replicated state machines, a common strategies for implementing fault-tolerant services in distributed systems. Reliable broadcast can also be used to solve the consensus problem, which is at the core of many distributed systems.

This thesis describes and proves correct three new implementations of the reliable broadcast service in *dynamic distributed networks*. As detailed in the related work section, some prior work investigates reliable broadcast style problems in *dynamic distributed networks*. However, these papers by Brown et al [**?**], Chockler et al [**?**] and Gao et al [**?**] make assumptions that are not practical in the real world. For example, these papers rely on predefined regions with special properties including the ability of a node to communicate with all other nodes in the same region. Moreover, these papers assume that nodes know their absolute location. To the best of our knowledge, this thesis is the first examination of a reliable broadcast style problem in a *dynamic distributed networks*.

The first implementation we provide of the reliable broadcast service assumes that all nodes in the network activate before the same global round and remain active for the duration of the execution (Simultaneous Activation). The second implementation allows nodes to activate before different rounds of execution but assumes that after a node activates, it remains active for the remainder of the execution (Staggered Activation). The third implementation assumes nodes can activate or deactivate before each and every round (Variable Activation and Deactivation). While the first two sets of assumptions do not allow for significant changes to the network topology and participant set during execution, they demonstrate the best-case efficiency and performance in a reasonably well-behaved *dynamic distributed networks*. While the third implementation correctly implements reliable broadcast for all three sets of assumptions about node behavior related to activations and deactivations, it has a significantly worse average case performance than both the first and second implementation.

The major contributions of this thesis are as follows. First, it provides a correct and useful definition for the problem of reliable broadcast in *dynamic distributed networks* . Second, it describes and proves correct new implementations for reliable broadcast in *dynamic distributed networks*given the different sets of assumptions regarding node activation and deactivation during execution. Third, these algorithms use general strategies that will be useful for the development of other algorithms for *dynamic distributed networks*. Finally, this thesis provides simulation results for the algorithms in both synthetic and real world networks to explore average case performance beyond the formally proven worst-case bounds.

2 Model

We consider a broadcast variant of the standard synchronous message passing model of distributed computation [**?**, **?**]. The synchronous message passing model is defined with respect to a directed graph $G = (V, E)$. We define $n = |V|$ as the number of nodes in the graph. An algorithm is a set of instructions to be followed by the nodes. When we say the network executes an algorithm $\mathcal{A}$, this means each node in the network is running a copy of $\mathcal{A}$. (For simplicity, we refer to a copy of the algorithm running on node u as simply node u.) In the execution, the nodes proceed in lock-step repeatedly performing the following two steps:

1. Following the algorithm, decide which messages, if any, to send to their neighbors in G.

2. Receiving and processing all incoming messages from their neighbors.

The combination of these two steps is called a *round*.

The synchronous broadcast model that we study in this paper is different from the synchronous message passing model in three significant ways. First, the synchronous broadcast model is defined with respect to a connected graph $G = (V, E)$ with bi-directional edges. Second, nodes do not pass individual messages directly to their neighbors. Instead, nodes broadcast their messages at the end of every round and every message is received by every neighbor. Finally, in the synchronous message passing model, nodes know their neighbors in advance while nodes do not have any prior knowledge of their neighbors in the synchronous broadcast model.

Additionally, we assume that nodes have comparable unique identifiers and that nodes are in one of two high-level states, active or deactive. When a node is active, it performs the two steps, sending and receiving messages, that constitute a round. When a node is deactive, it performs neither of the two steps that constitute a round. We say a node is activated when its state changes from deactive to active. We assume nodes only change states in between rounds. When a node is activated for the first time, it always begins in an initial state with knowledge of a global round counter. We say a node is deactivated when its state changes from active to deactive. When a node is deactivated, it maintains all local variables such that if it reactivates, it has knowledge of its previous state and knowledge of the current global round. The ability to retain previous state knowledge after deactivations leaves the decision of how to treat the reactivation of a node up to the algorithm.

This thesis uses three sets of assumptions about the behavior of nodes with respect to activations and deactivations. The first set assumes simultaneous activation of nodes, meaning that all nodes in the network must activate before the same round and remain active for the duration of the execution (Simultaneous Activation). The second set allows for staggered activation of nodes, meaning that a node in the network may activate before any round of execution but must remain active for the duration of the execution once it has activated (Staggered Activation). The third set allows for nodes to activate and deactivate without limit before the beginning of any round of execution (Variable Activation and Deactivation). All three sets of assumptions assume that the active subset of the graph G must be connected as a single component before the beginning of every round.

3 Problem Definition

The reliable broadcast problem provides messages to arbitrary nodes in the synchronous broadcast model to send to all active nodes in the network. This problem assumes there is an environment at each node u that communicates with u through an interface with three commands, *send*, *receive* and *acknowledge*. We refer to the environment at node u as E_u.

Using the *send* command, E_u can pass a message m to u, which u is expected to send to all other nodes in the network. Once all the other nodes have received m, u is expected to pass a "done" signal to E_u using the *acknowledge* command. We assume E_u will not pass another message to u until it has received a "done" signal from u. When a node u learns about a message m, it uses the *receive* command to notify E_u about m.

An algorithm A is said to solve the reliable broadcast problem if it implements the *send*, *receive* and *acknowledge* commands and satisfies the following properties (in the following, assume without loss of generality that all messages are unique):

1. *Liveness Property* : If a node u running algorithm $\mathcal{A}$ is passed a message by E_u through its *send* command, u will eventually send a "done" signal to E_u using the *acknowledge* command unless u deactivates at some point after receiving the *send* command.

2. *Safety Property #1* : Assume node u is passed a message m by E_u at

round r and u sends a "done" signal in some later round r'. Let $A(r, r')$ be the set of nodes that are active in every round in the interval from r to r'. It must be the case that every node in $A(r, r')$ passes message m to its environment through its *receive* command at some point between rounds r and r'. In addition, no node, including those not part of $A(r, r')$, will pass m to its environment after round r'.

3. *Safety Property #2* : Assume some node u passes a message m to its environment through its *receive* command in some round r and then passes a different message m' in a later round r'. It follows that no node in the network passed message m' to its environment before message m.

4. *Safety Property #3* : Assume some node u passes a message m to its environment through its *receive* command, the following two conditions must hold:

   (a) u has not previously passed m to its environment

   (b) some node previously received m from its environment through its *send* command

4 Algorithm

In the model section, we introduced three sets of assumptions for node behavior with respect to activation and deactivation. Each set of assumptions requires its own algorithm. While the algorithms for the first two sets of assumptions share many similarities, they are very different from the algorithm for the third set of assumptions.

4.1 Reliable Broadcast with Simultaneous Activation

The first step of the algorithm, the leader election subroutine, is for each node to run the terminating synchronous breadth-first search algorithm (SynchBFS) as described and analyzed by Lynch [**?**]. The nodes will use SynchBFS to elect a leader as follows. When a node u activates, it starts running an instance of SynchBFS with itself as its source. The messages for SynchBFS include the ID of the source of that instance. Nodes stop participating in an instance of SynchBFS if they hear about an instance of SynchBFS with a smaller ID. Therefore only the SynchBFS instance started by the node with the minimum ID in the network will terminate. Once this instance of SynchBFS terminates, it's source will elect itself leader and disseminate a confirm message throughout the network in the manner described by Lynch [**?**]. When a node receives the confirm message, it becomes a confirmed member of the spanning tree rooted at the source of the SynchBFS instance that terminated.

The message dissemination subroutine begins when a node u receives a message m from E_u using the *send* command. While a node can receive a *send* command from its environment at any point during execution, it won't begin the message dissemination subroutine until it has become a confirmed member of the tree. First, node u sends the message m to the leader of the network by broadcasting m with the instructions that only its parent should forward m. Each node that forwards m keeps track of from which child node it received m. Message m continues to be passed up the tree from child to parent until it reaches the leader of the tree. Message m is guaranteed to reach the leader because the leader is an ancestor of every node in the tree. When the leader receives a message, it adds that message to its send queue.

If a message is not being disseminated through the tree, the leader will dequeue the next message m, if any exists, off its send queue. Next, the leader will notify its environment of message m using the *receive* command and then broadcast m. All of the children of the leader will receive m and perform the same two actions as the leader, notifying their respective environments and broadcasting m, and in this way, m will eventually reach every node, which will pass m to its environment only once because a node will only process m when it is sent by its parent and a node can only have one parent in the tree. A node that receives m and doesn't have any children will send a message to their parent confirming that they have received message m. When a node has received confirmation messages from all of its children, it will send a message to its parent confirming that it (and all its children) have received m. When the leader has received confirmation messages from all of its children, the leader will dequeue the next message m', if any exists, off its send queue and repeat the message dissemination subroutine.

At the same time, the leader will send a message to u telling it that all nodes in the network have received m. This message to u follows the same path as the original message from u to the leader as each node broadcasts it with the instruction that it only should be forwarded by the specified node, the child that sent it to the parent. When u receives this message, it notifies its environment using its *acknowledge* command.

4.2 Reliable Broadcast with Staggered Activation

The algorithm for reliable broadcast with simultaneous activation works for reliable broadcast with staggered activation with the following modifications.

In the leader election subroutine, the source of a SynchBFS instance started by node u is now identified by the combination of the global round when u activated and the ID of u. A node will stop running instances when they see smaller ids; A node will stop running a SynchBFS instance if it hears about a SynchBFS instance with a lowest global round of activation or a SynchBFS instance with the same global round of activation and a smaller ID. By the definition of the model, nodes have unique IDs so multiples BFS instances won't have the same ID.

When a node is a confirmed member of the spanning tree, it broadcasts a message at the beginning of every round declaring that it is part of the spanning tree and telling its neighbors to join. Once the leader has been elected and sent a confirm message through the tree, it is still possible for nodes to activate.

When a node u activates, it follows the leader election subroutine. If at any point during running the leader election subroutine u receives a confirm message, it will stop running its instance of SynchBFS, join the spanning tree rooted at the leader and begin the message dissemination protocol.

4.3 Reliable Broadcast with Activations and Deactivations

The algorithm for reliable broadcast with activations and deactivations is very different from the algorithms for reliable broadcast with only activations. The deactivation case does not involve running SynchBFS to elect a leader or rely on a stable tree. Instead, it relies on flooding messages through the network.

In more detail, when a node u receives a message m from its environment at round r using the *send* command, u disseminates m through the network, with the instructions that every node that receives m should execute the *receive* command on m at a specified round r'. The round of execution r' is equal to round $r + n$ where n is an upper bound on the number of nodes in the network. The node u will execute the *acknowledge* command on m at round $r' + 1$.

If a node has multiple *receive* commands to execute in the same round, it executes them in ascending order of the ID of the message, which is the ID of the node that received the message from its environment. Every node in the network maintains an internal list of messages. At the beginning of every round, a node will broadcast all of the messages. When a node hears a broadcast about a message for the first time, it adds that message to its internal list. At the end of every round, a node checks its list of messages and executes a *receive* command on the message(s), which correspond to the given round and remove all messages that correspond to previous rounds from its internal list.

5 Analysis

In this section, we prove the correctness of the three algorithms presented in this paper. The analysis sections for the first two algorithms begin with a proof of the leader election subroutine in their respectives settings as correct leader election is integral to the reliable broadcast protocols.

5.1 Reliable Broadcast with Simultaneous Activation

In this section, we prove the correctness of the algorithm assuming simultaneous activation of all nodes. First, we prove the correctness of the leader election subroutine and then we prove the correctness of the message dissemination subroutine.

5.1.1 Leader Election Subroutine

Theorem 5.1. *A node will eventually elect itself leader and no more than one node will have leader equal to true at the beginning of any round.*

Proof. One node will eventually elect itself leader (Lemma 5.2). No more than one node will have leader set to true at any point (Lemma 5.4). □

Definition 1. *Let u_{min} be the ID of the process with the minimum UID in the network.*

Definition 2. *Let BFS instance b_i refer to an instance of the terminating breadth-first search protocol initiated by process with ID i.*

Lemma 5.2. *One node will eventually set its variable leader to true.*

Proof. A BFS instance b_i will eventually terminate if every node in the network runs b_i. Every node in the network will eventually run $b_{u_{min}}$ so $b_{u_{min}}$ will eventually terminate and the process with ID u_{min} will set leader = true. □

Lemma 5.3. *A BFS instance b_i will only terminate if i equals u_{min}.*

Proof. Termination of a BFS instance b_j requires all other processes in the network to send a done message to b_j. Given BFS instance b_j where $j > u_{min}$, there is at least one process, the process with ID u_{min}, that will never reply done to b_j. Therefore, b_j will never terminate. □

Lemma 5.4. *For every round r, at most one node has leader = true at the beginning of r.*

Proof. A node with ID i will only set leader = true if the BFS instance b_i terminates. A BFS tree b_i will terminate only if i equals u_{min} (Lemma 5.3). Only the process with ID u_{min} will set leader = true. □

5.1.2 Message Dissemination Subroutine

Theorem 5.5. *The algorithm solves the reliable broadcast problem with simultaneous activation.*

Proof. This algorithm satisfies the liveness property (Lemma 5.7), the first safety property (Lemma 5.8), the second safety property (Lemma 5.9) and the third safety property (Lemma 5.10) of the reliable broadcast problem with simultaneous activation. □

Lemma 5.6. *A message m that is broadcast by the leader through the spanning tree formed by the leader election protocol will eventually be seen by every node and the leader will eventually receive a confirmation message that all nodes have seen m.*

Proof. m will be broadcast by the leader and received by all of the leader's children. The children will similarly broadcast m. Eventually, every node in the tree will receive m because every node in the tree is a descendant of the leader.

By the definition of the algorithm, a leaf node will send a confirm message to its parent upon receiving m. Every child of a non-leaf node eventually sends a confirm message because the subtree of every node ends with all leaf nodes.

As a result, every non-leaf node, including the leader, will eventually receive a confirm message from its children.

□

Lemma 5.7. *The algorithm satisfies the liveness property of the reliable broadcast problem with simultaneous activation.*

Proof. Assume node u running algorithm $\mathcal{A}$ is passed a message m by E_u through its *send* command. When u is a confirmed member of the spanning tree, it sends m to the leader of the tree. There will be a stable path between u and the leader so m is guaranteed to reach the leader. When m reaches the leader, it is placed in its send queue. Eventually, m reaches the front of the send queue and is dequeued by the leader. The leader broadcasts m to the network and eventually receives a confirm message that all the nodes in the network have seen m (Lemma 5.6). When the leader has received confirmation messages from all of its children, it will notify u using the same stable path. When u receives this message from the leader, it sends a "done" signal to E_u using the *acknowledge* command.

□

Lemma 5.8. *The algorithm satisfies the first safety property of the reliable broadcast problem with simultaneous activation.*

Proof. In the simultaneous activation case, $A(r, r')$ is the set of all nodes in the network so the first safety property requires that all nodes in the network receive m between round r and r'. Assume node u is passed a message m by E_u at round r and u sends a "done" signal in some later round r'. u will first send m to the leader of the tree. The leader will broadcast m at some round after r. The leader eventually receives a confirmation that all nodes have seen m and passed m to their environments through their *receive* command (Lemma 5.6). Then, the leader sends a confirm message to u, which then sends a "done" signal at round r' to E_u using the *acknowledge* command.

□

Lemma 5.9. *The algorithm satisfies the second safety property of the reliable broadcast problem with simultaneous activation.*

Proof. Assume some node u passes a message m to its environment through its *receive* command in some round r and then passes a different message m' in a later round r'.

If u passes m to its environment, m must have been broadcast by the leader. If the leader broadcasts m, every node in the network will receive m and notify its environment and eventually the leader will receive a confirm message. If u receives m' after m, then the leader must have broadcast m' after m because the leader will not begin to broadcast another message, like m', until it has received a confirm message that all of the nodes in the network, including u, have seen m. As a result, no node in the network will pass m' to its environment before m.

□

Lemma 5.10. *The algorithm satisfies the third safety property of the reliable broadcast problem with simultaneous activation.*

Proof. Per the algorithm, nodes only pass a message m to their environment when they receive m from their parent in the tree. A given node has only one parent and so only receives m once. □

5.2 Reliable Broadcast with Staggered Activation

In this section, we prove the correctness of the algorithm assuming staggered activation of nodes. First, we prove the correctness of the leader election subroutine and then we prove the correctness of the message dissemination subroutine.

5.2.1 Leader Election Subroutine

Theorem 5.11. *A node will eventually elect itself leader and no more than one node will have leader equal to true at the beginning of any round.*

Proof. One node will eventually elect itself leader (Lemma 5.12). No more than one node will have leader set to true at any point (Lemma 5.14). □

Definition 3. *Let u_{min} be the ID of the process with the minimum global round of activation and the minimum UID of the nodes that at that round in the network.*

Definition 4. *Let BFS instance b_i refer to an instance of the terminating breadth-first search protocol initiated by process with ID i.*

Lemma 5.12. *One node will eventually set its variable leader to true.*

Proof. A BFS instance b_i will eventually terminate if every node in the network runs b_i. Every node in the network will eventually run $b_{u_{min}}$ so $b_{u_{min}}$ will eventually terminate and the process with ID u_{min} will set leader = true. □

Lemma 5.13. *A BFS instance b_i will only terminate if i equals u_{min}.*

Proof. Termination of a BFS instance b_j requires all other processes in the network to send a done message to b_j. Given BFS instance b_j where $j > u_{min}$, there is at least one process, the process with ID u_{min}, that will never reply done to b_j. The process with ID u_{min} will have been activate before or at the same round as every process in the network, including the process with ID j, so j will require the process with ID u_{min} to reply done, which it never will. Therefore, b_j will never terminate. □

Lemma 5.14. *For every round r, at most one node has leader = true at the beginning of r.*

Proof. A node with ID i will only set leader = true if the BFS instance b_i terminates. A BFS tree b_i will terminate only if i equals u_{min} (Lemma 5.13). Only the process with ID u_{min} will set leader = true. □

5.2.2 Message Dissemination Subroutine

Theorem 5.15. *The algorithm solves the reliable broadcast problem with staggered activation.*

Proof. This algorithm satisfies the liveness property (Lemma 5.18), the first safety property (Lemma 5.20), the second safety property (Lemma 5.21) and the third safety property (Lemma 5.22) of the reliable broadcast problem with staggered activation. □

Lemma 5.16. *The path from the leader of a tree to any node in the tree will not change with the activation of any node.*

Proof. Every node is connected to the leader through their parent and nodes do not change their parent after they are a confirmed member of the spanning tree. □

Lemma 5.17. *A given node u that receives a message m at round r will eventually send a confirmation message to its parent.*

Proof. If u is a leaf node at round $r + 2$, u will immediately send a confirmation message so any nodes joining the tree as children of u won't affect the propagation of the confirm message up the tree.

If u is a non-leaf node at round $r + 2$, u will broadcast m. All active nodes that consider u to be their parent before $r + 2$, will process m. If a node v considers u to be its parent at round $r + 2$, u will consider v to be its child no later than round $r + 3$. According to the algorithm, u will only wait to hear confirm messages from nodes that it considers to be its children at the end of round $r + 3$ so u will only wait to hear from nodes that processed m and are guaranteed to eventually send a confirm message to u.

As a result, u is guaranteed to send a confirm message to its parent.

□

Lemma 5.18. *The algorithm satisfies the liveness property of the reliable broadcast problem with staggered activation.*

Proof. Assume node u running algorithm $\mathcal{A}$ is passed a message m by E_u through its *send* command. When u is a confirmed member of the spanning tree, it sends m to the leader of the tree. There will be a stable path between u and the leader so m is guaranteed to reach the leader (Lemma 5.16). When m reaches the leader, it is placed in its send queue. Eventually, m reaches the front of the send queue and is dequeued by the leader. The leader broadcasts m to the network and eventually receives a confirm message that all the nodes in the network have seen m (Lemma 5.17). When the leader has received confirmation messages from all of its children, it will notify u using the same stable path. When u receives this message from the leader, it sends a "done" signal to E_u using the *acknowledge* command.

□

Lemma 5.19. *By induction, any node u that activates before round r and is q hops from the tree will receive a message m that is sent to any given node from an environment during round r.*

Proof. By Induction.

Base Case $q = 0$: A node u that is zero hops away from the tree is part of the tree and will receive the message m (Lemma 5.8).
Inductive Hypothesis: Suppose the theorem holds for all values of q up to k
Inductive Step: Let $q = k + 1$.

v, the node that is k hops from the tree at round r, receives m at a later round r'. For v to have received m in round r', it must have been a confirmed member of the tree in round r'. As a confirmed member of the tree, v must have broadcast a message telling its neighbors to join the tree. u, which is a neighbor of v, receives this message in round r' and is a confirmed member of the tree by round $r' + 1$. As the parent of u, v will broadcast m to u in round $r' + 3$ after the two round wait period.

□

Lemma 5.20. *The algorithm satisfies the first safety property of the reliable broadcast problem with staggered activation.*

Proof. Assume node u is passed a message m by E_u at round r and u sends a "done" signal in some later round r'. In the staggered activation case, $A(r, r')$ is identical to $A(r)$ which is the set of all nodes that have activated by round r. If all the nodes in $A(r, r')$ are part of the spanning tree of by round r, this is identical to the first safety property in the case of simultaneous activation (Lemma 5.8) given that the activation of any nodes won't affect the path between a node and the leader (Lemma 5.16) and any node that receives a message will eventually send a confirmation message to its parent (Lemma 5.17).

If all the nodes in $A(r, r')$ are not part of the spanning tree of by round r, the nodes that aren't part of the tree before round r will become part of the tree in time to receive the message (Lemma 5.19) and any node that receives a message will eventually send a confirmation message to its parent (Lemma 5.16).

□

Lemma 5.21. *The algorithm satisfies the second safety property of the reliable broadcast problem with staggered activation.*

Proof. Safety Property #2: Assume some node u passes a message m to its environment through its *receive* command in some round r and then passes a different message m' in a later round r'. It follows that no node in the network passed message m' to its environment before message m.

Assume some node u passes a message m to its environment through its *receive* command in some round r and then passes a different message m' in a later round r'.

The leader of the tree only broadcasts one message at a time as after sending a message, it waits to receive a confirmation message before sending the next message in its send queue. If u passes m to its environment before passing m' then the leader must have broadcast m before it broadcast m'. As a result, any node that received m and m' must have received m before m'.

□

Lemma 5.22. *The algorithm satisfies the third safety property of the reliable broadcast problem with staggered activation.*

Proof. Per the algorithm, nodes only pass a message m to their environment when they receive m from their parent in the tree. A given node has only one parent and so only receives m once.

□

5.3 Reliable Broadcast with Activations and Deactivations

In this section, we prove the correctness of the algorithm allowing nodes to activate and deactivate.

Theorem 5.23. *The algorithm solves the reliable broadcast problem with activations and deactivations.*

Proof. This algorithm satisfies the liveness property(Lemma 5.24), the first safety property(Lemma 5.26), the second safety property(Lemma 5.27) and the third safety property(Lemma 5.28) of the reliable broadcast problem with activations and deactivations. □

Lemma 5.24. *The algorithm satisfies the liveness property of the reliable broadcast problem with activations and deactivations.*

Proof. By the algorithm, u will send a "done" signal to E_u using the *acknowledge* command after n+1 rounds. The only way u will not send the "done" signal is if u deactivates, which is acceptable under liveness. □

Lemma 5.25. *Every node in $A(r, r')$ sees m within n rounds*

Proof. By the model definition, the network is guaranteed to have the stability property of at worst 1-interval connectivity. In a network with 1-interval connectivity where nodes broadcast every round, a message is guaranteed to be seen for the first time by at least one node every round if any node in the network has not yet seen the message (Distributed Computing in Dynamic Networks). After $n-1$ rounds of broadcasting message m in a network with 1-interval connectivity where n is the total number of nodes in the network, every node that was active for all $n-1$ rounds will have seen m.

□

Lemma 5.26. *The algorithm satisfies the first safety property of the reliable broadcast problem with staggered activation.*

Proof. Every node in $A(r, r')$ sees m within n rounds (Lemma 5.25). By definition of the algorithm, every node that receives m passes m to its environment during round $r' - 1$. Therefore, every node in $A(r, r')$ passes m to its environment during round $r' - 1$, which is between round r and round r'.

□

Lemma 5.27. *The algorithm satisfies the third safety property of the reliable broadcast problem with deactivation.*

Proof. By definition of the algorithm, every node that receives a message will pass that message to its environment at a pre-determined round. Assume two messages, m and m', are received by multiple nodes. m will be executed by all nodes at round r and m' will be executed by all nodes at round r'.

If r is greater than r', then all nodes that receive m and m' and are active for both r and r' will pass m' to their environment before they pass m. If r' is greater than r, then all nodes will that receive m and m' and are active for both r and r' will pass m to their environment before they pass m'.

If r is equal to r', then nodes will use the UID attached to the message to determine the order of execution. If m_{ID} is greater than m'_{ID}, then all nodes that receive m and m' and are active for both r and r' executes m before m'. If $m'_{sourceID}$ is greater than $m_{sourceID}$, then all nodes that receive m and m' and are active for both r and r' executes m' before m. $m'_{sourceID}$ cannot be equal to $m_{sourceID}$ because a single node cannot receive a message from its environment while it has an unacknowledged message out there.

□

Lemma 5.28. *The algorithm satisfies the third safety property of the reliable broadcast problem with staggered activation.*

Proof. Per the algorithm, nodes keep a list of messages to execute and only add a message m if m is not already in the list. A node will only execute message m at its specified round if m is in its list at that round.

□