Implementing the Reliable Broadcast Service in Dynamic Distributed Networks

April 24, 2014

Welles Robinson

Senior Thesis

Department of Computer Science

Georgetown University

Abstract

The importance of distributed computing has grown in recent years with the creation and accumulation of massive amounts of data, the prevalence of mobile devices and the development of peer-to-peer networks. The vast majority of the existing distributed theory literature has focused on static distributed networks, which assume a relatively constant network topology and participant set. While reasonable for relatively stable settings such as data centers, the assumptions of this approach will not hold for wireless networks, which feature dynamic sets of participants and variable topologies because the nodes are extremely mobile and the links unstable. Unlocking the potential of distributed computing in wireless networks requires the implementation of primitives used by static distributed algorithms in this more dynamic distributed network setting. To this end, this thesis describes and proves correct three novel algorithms that implement a reliable broadcast service in dynamic distributed networks with varying assumptions for node behavior. The first two proposed algorithms assume nodes will not deactivate during execution and consequently demonstrate strong performance and efficiency. The third algorithm allows nodes to activate and deactivate during execution although this comes at a slight cost to performance. Finally, to demonstrate the practicality of these algorithms in realistic networks, this thesis also presents simulation results on both synthetic and real world data sets of varying size and topology.

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 Related Work

Extensive theoretical work already exists for static distributed networks [?].

Kuhn et al use the model of dynamic distributed networks that most closely matches our own model [?]. In contrast to our model, which permits a dynamic participant set, their model assumes a stable set of participants. Their paper studies token passing and message dissemination, the latter of which we leverage in this thesis.

Other papers that examine dynamic distributed networks create a reliable virtual layer on top of the variable topology of the underlying dynamic distributed network [?, ?, ?]. Two shortcomings of this approach are the requirements that nodes have knowledge of their absolute location and remain within a physical area that is determined prior to execution.

3 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.

4 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

5 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.

5.1 Simultaneous Activation Reliable Broadcast Algorithm

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 finished message to their parent confirming that they have received message m. When a node has received finished messages from all of its children, it will send a finished message to its parent confirming that it (and all its children) have received m. When the leader has received finished 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.

5.2 Staggered Activation Reliable Broadcast Algorithm

The simultaneous activation reliable broadcast algorithm works for reliable broadcast under the assumption of 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.

5.3 Variable Activation and Deactivation Reliable Broadcast Algorithm

The variable activation and deactivation reliable broadcast algorithm is very different from the algorithms for reliable broadcast under the assumption of only activations. This algorithm 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.

6 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.

6.1 Simultaneous Activation Reliable Broadcast Algorithm

In this section, we prove the correctness of the algorithm assuming simultaneous activation of nodes. First, we prove the correctness of the leader election subroutine and then leverage this knowledge to prove the correctness of the overall simultaneous activation reliable broadcast algorithm.

6.1.1 Leader Election Subroutine

We prove the correctness of leader election by separately proving the liveness and safety property of the subroutine.

Theorem 6.1. *The simultaneous activation reliable broadcast algorithm will eventually elect a single node to be leader as indicated by the leader node setting its leader variable to true.*

Proof. To prove that a single node elects itself leader, it is sufficient to prove both the liveness and safety property of the given leader election subroutine. Liveness guarantees that at least one node will eventually elect itself leader (Lemma 6.2) while safety guarantees that no more than one node will have its leader variable set to true at any point (Lemma 6.4). □

We now define notation that will simplify the statment and proof of the lemmata that follow. Unless redefined, this notation will also be used in the lemmata for the leader election theorem for the staggered activation reliable broadcast algorithm.

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

Definition 2. *Let SynchBFS instance b_i refer to an instance of terminating synchronous breadth-first search (SynchBFS) initiated by the process with ID i.*

Lemma 6.2. *One node will eventually set its leader variable to true.*

Proof. A SynchBFS instance b_i will eventually terminate if every node in the network participates in b_i. Every node in the network will eventually participate in $b_{u_{min}}$ because when a node hears about $b_{u_{min}}$, it will stop participating in its current SynchBFS instance and start participarting in $b_{u_{min}}$ because it is the SynchBFS instance with the minimum ID in the network. As a result, $b_{u_{min}}$ will eventually terminate and the process with ID u_{min} will set leader = true. □

Lemma 6.3. *A SynchBFS instance b_i will only terminate if i equals u_{min}.*

Proof. Termination of a SynchBFS instance b_j requires all other processes in the network to send a "done" message to b_j. A process sends done to b_j by broadcasting a "done" message that is received by its parent in b_j. A process will only broadcast a "done" message when it knows that all of its neighbors are participating in 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 participate in b_j. Therefore, b_j will never terminate. □

Lemma 6.4. *For every round r, at most one node has its leader variable equal to true at the beginning of r.*

Proof. A node with ID i will only set leader = true if its BFS instance, b_i, terminates. A BFS tree b_i will terminate only if i equals u_{min} (Lemma 6.3). Only the process with ID u_{min} will set its leader variable equal to true. □

6.1.2 Overall Correctness

Drawing on the proof of correctness of the leader election subroutine, we prove the simultaneous activation reliable broadcast algorithm satisfies the four properties of reliable broadcast and therefore correctly implements the reliable broadcast service.

Theorem 6.5. *The simultaneous activation reliable broadcast algorithm correctly implements the reliable broadcast service.*

Proof. This simultaneous activation reliable broadcast algorithm satisfies the liveness property (Lemma 6.9), the first safety property (Lemma 6.10), the second safety property (Lemma 6.11) and the third safety property (Lemma 6.12) of the reliable broadcast problem under the assumption of simultaneous activation. □

Lemma 6.6. *Under the assumption of simultaneous activation, any message m that is disseminated by the leader will reach every node in the network.*

Proof. For a message m to be disseminated by the leader, it means the leader must broadcast m, which will be received and processed by all of the leaders children. These children will broadcast m in the next round and so m will be received and processed by all of the their children. This pattern continues until eventually every node in the network receives m because every node in the network is a descendant of the leader.

□

Lemma 6.7. *Under the assumption of simultaneous activation, for any message m disseminated by the leader, the leader will eventually receive a finished message that all the nodes in the network have seen message m.*

Proof. According to the subroutine, a leaf node will send a finished message to its parent upon receiving m. Every child of a non-leaf node eventually sends a finished 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 finished message from its children.

□

Lemma 6.8. *Under the assumption of simultaneous activation, for any node u that is a confirmed member of the tree, there exists a stable path between u and the leader.*

Proof. If Node u is a confirmed member of the tree, node u can broadcast a message m to its parent, which will broadcast m to its parent. In this way, m is guaranteed to reach the leader of the tree because the leader is an ancestor of node u. Once a node is a confirmed message of the tree, its parent will not change so this path is guaranteed to be stable. According to the definition of the subroutine, each node keeps track of which child node forwarded the message from u so the leader can respond to a message from u along the same path, which is guaranteed to be stable.

□

Lemma 6.9. *The simultaneous activation reliable broadcast algorithm satisfies the liveness property of the reliable broadcast service under the assumption of simultaneous activation.*

Proof. Assume node u running algorithm $\mathcal{A}$ is passed a message m by E_u through its *send* command. If u is not a confirmed member of the spanning tree, it will store message m until it is becomes a confirmed member of the spanning tree. Node u will eventually become a confirmed member of the tree because the leader election subroutine will eventually result in one node electing itself leader (Lemma 6.1) and any message disseminated by the leader, including the "confirm" message, will be seen by every node in the network (Lemma 6.6).

If node u is a confirmed member of the spanning tree, it sends m to the leader of the tree. If node u is a confirmed member of the tree, there exists a stable path between u and the leader so m is guaranteed to reach the leader (Lemma 6.8). When m reaches the leader, it is placed in its send queue. The message in the front of the leader's send queue will be dequeued if the network is not already in the process of disseminating another message. A message that is disseminated through the network is guaranteed to terminate (Lemma 6.7). Given that all of the messages in front of m in the queue will eventually be disseminated and terminate, m is guaranteed to reach the front of the queue, be disseminated and terminate. When message m terminates, the leader will notify u using an "acknowledge" message along the stable path between the leader and u (Lemma 6.8). When u receives this message from the leader, it sends a "done" signal to E_u using the *acknowledge* command thus satisfying the liveness property of reliable broadcast.

□

Lemma 6.10. *The simultaneous activation reliable broadcast algorithm satisfies the first safety property of the reliable broadcast service under the assumption of simultaneous activation.*

Proof. Under the assumption of simultaneous activation, $A(r, r')$ is the set of all nodes in the network so the first safety property of reliable broadcast 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 along a stable path (Lemma 6.8). The leader will disseminate m to the network at some round after r and all the nodes in the network will receive m and pass m to their environments through their *receive* command (Lemma 6.6). The leader eventually receives a confirmation message and then sends a acknowledge message to u, which then sends a "done" signal at round r' to E_u using the *acknowledge* command (Lemma 6.7).

□

Lemma 6.11. *The simultaneous activation reliable broadcast algorithm satisfies the second safety property of the reliable broadcast service under the assumption of 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 disseminated by the leader. If the leader disseminates m, every node in the network will receive m and notify its environment and eventually the leader will receive a confirm message (Lemma 6.6) (Lemma 6.7). 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 6.12. *The simultaneous activation reliable broadcast algorithm satisfies the third safety property of the reliable broadcast service with simultaneous activation.*

Proof. The third safety property of reliable broadcast service with simultaneous activation requires that if a node u passes a message m to its environment, it has not passed m to its environment before and some node in the network previously received m from its environment. It follows from the definition of the algorithm that any node u that is not the leader of the tree will pass a message m to its environment only if and when it receives m from its parent in the tree. A node will also only pass a message to its environment if it is a confirmed member of the tree. If a node is a confirmed member of the tree, it has only one parent, which does not change during the execution. As a result, u will only process and pass m to its environment once. If node u is the leader of the tree, it will only pass m to its environment when it dequeues m from its queue, which will only happen once for each message. Therefore, a node u will only pass a message m to its environment at most one time.

The only way in which u will receive a message m to pass to its environment is if m is disseminated to the network by the leader. The leader will only disseminate a message m if m is sent to it by its own environment or by a node that received m from its environment. Therefore, a node u will only pass a message m to its environment if m has been received by a node in the network from its environment.

□

6.2 Staggered Activation Reliable Broadcast Algorithm

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 leverage this knowledge to prove the correctness of the overall staggered activation reliable broadcast algorithm.

6.2.1 Leader Election Subroutine

We prove the correctness of leader election by separately proving the liveness and safety property of the subroutine.

Theorem 6.13. *The staggered activation reliable broadcast algorithm will eventually elect a single node to be leader as indicated by the leader node setting its leader variable to true.*

Proof. To prove that a single node elects itself leader, it is sufficient to prove both the liveness and safety property of the given leader election subroutine.

Liveness guarantees that at least one node will eventually elect itself leader (Lemma 6.14) while safety guarantees that no more than one node will have its leader variable set to true at any point (Lemma 6.16).

□

We now define notation that will simplify the statment and proof of the lemmata that follow. We redefine u_{min} but do not redefine b_i and will use the definition of b_i from the simultaneous activation reliable broadcast algorithm section.

Definition 3. *Let u_{min} be the ID of the process which has the minimum global round of activation in the network. If multiple processes have the minimum global round of activation, then u_{min} will be the process with the minimum ID out of the processes with the minimum global round of activation.*

Lemma 6.14. *At least one node will eventually set its leader variable equal to true.*

Proof. A SynchBFS instance b_i will eventually terminate if every node in the network participates in b_i. Every node in the network will eventually participate in $b_{u_{min}}$ because when a node hears about $b_{u_{min}}$, it will stop participating in its current SynchBFS instance and start participarting in $b_{u_{min}}$ because it is the SynchBFS instance whose source has the minimum combination of global round of activation and ID in the network. As a result, $b_{u_{min}}$ will eventually terminate and the process with ID u_{min} will set leader = true. □

Lemma 6.15. *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. A process sends done to b_j by broadcasting a "done" message that is received by its parent in b_j. A process will only broadcast a "done" message when it knows that all of its neighbors are participating in 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 6.16. *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 6.15). Only the process with ID u_{min} will set leader = true. □

6.2.2 Overall Correctness

Drawing on the proof of correctness of the leader election subroutine, we prove the staggered activation reliable broadcast algorithm satisfies the four properties of reliable broadcast and therefore correctly implements the reliable broadcast service.

Theorem 6.17. *The staggered activation reliable broadcast algorithm correctly implements the reliable broadcast service.*

Proof. This staggered activation reliable broadcast algorithm satisfies the liveness property (Lemma 6.20), the first safety property (Lemma 6.22), the second safety property (Lemma 6.23) and the third safety property (Lemma 6.24) of the reliable broadcast problem under the assumption of staggered activation. □

Lemma 6.18. *The path from the leader of a tree to any confirmed node in the tree will not change with the activation of any node.*

Proof. When a node is a confirmed member of the tree, it is part of the spanning tree rooted at the leader. There exists a stable path from every node u that is a confirmed member of the tree to the leader because the leader is an ancestor of every node in the network. This path goes from a node to its parent. If its parent is not the leader, then it goes to its parent. Eventually, this path reaches the leader as the tree is rooted at the leader. This path is stable because nodes do not change their parent after becoming a confirmed member of the tree and nodes do not deactivate under the assumptions of staggered activation. □

Lemma 6.19. *A given node u that receives a message m at round r will eventually send a finished message to its parent.*

Proof. Assume node u receives a message m from its parent at round r. By definition of the algorithm, a node that receives a message m at round r will execute the message at round $r+2$. Also according to the algorithm, a node that has no children when it executes the message will immediately send a finished message to its parent. If node u is a leaf node at round $r+2$, u will immediately send a finished message to its parent.

If u is not a leaf node at round $r+2$, u will broadcast m at $r+2$, which will be processed by all of its children. All active nodes that consider u to be their parent before $r+2$, will process m. At round $r+4$, the children of u will execute the message in the same way as node u. If they are leaf nodes, they will send a finished message u. Otherwise, they will broadcast m to their children. In this way, the message m will travel through the tree until it reaches a leaf node. These leaf nodes will respond finished to their parents. When a non-leaf node receives finished messages from all of its children, it sends a finished message to its parent. In this way, finished messages will propagate back up the tree until u receives finished messages from all of its children. When this happens, u will send a finished message to its parent

□

Lemma 6.20. *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. If u is not a confirmed member of the spanning tree rooted at the leader, it will become a confirmed member soon after the leader election subroutine terminates (Lemma 6.13). When u is a confirmed member of the spanning tree rooted at the leader, it sends m to the leader of the tree. There is a stable path between the leader and any confirmed node so m is guaranteed to reach the leader (Lemma 6.18). When m reaches the leader, it is placed in its send queue. The message in the front of the leader's send queue will be dequeued if the network is not already in the process of disseminating another message. A message that is disseminated through the network is guaranteed to terminate as all of the leader's children will eventually send the leader a finished message (Lemma 6.19). Given that all of the messages in front of m in the queue will eventually be disseminated and terminate, m is guaranteed to reach the front of the queue, be disseminated and terminate. When message m terminates, the leader will notify u using an "acknowledge" message along the stable path between the leader and u (Lemma 6.18). When u receives this message from the leader, it sends a "done" signal to E_u using the *acknowledge* command thus satisfying the liveness property of reliable broadcast.

□

Lemma 6.21. *Assume node u activates before round r and is q hops from the closest node that is a confirmed member of the spanning tree rooted at the leader. Node u will receive any message m that is sent to a given node v from an environment during or after round r.*

Proof. This proof implicitly assumes that a leader has been elected by round r. If a leader has not been elected, there can be no confirmed members of the spanning tree rooted at the leader. This assumption does not invalidate the proof. No node will start the reliable broadcast for a message received from the environment until they are a confirmed member of the spanning tree rooted at the leader, which requires the election of a leader. For the purpose of the proof, we can allow r to be the round at which v begins the reliable broadcast service for m because the round at which v begins the reliable broadcast service for m will be greater than or equal to the round at which v receives m from its environment. In short, this assumption reduces the amount of time that u has to become a confirmed member of the spanning tree, making this proof stronger not weaker.

We prove the lemma statement by induction on q, the number of hops from u to the closest confirmed member of thespanning tree rooted at the leader.
Inductive Hypothesis: The theorem holds for all values of q up to k
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 6.10).
Inductive Step: Assuming the theorem holds for all values of q up to k, let $q = k + 1$

If node u is $k+1$ hops from the nearest confirmed member of the spanning tree rooted at the leader, there must exist a node v, which is a neighbor of u and is k hopes from the nearest confirmed member. By the inductive hypothesis, v receives m at a later round r'. By the algorithm, only confirmed nodes will process reliable broadcast messages. Consequently, node v is a confirmed member of the tree by round r'. All confirmed members of the tree broadcast a message telling its neighbors to join the spanning tree rooted at the leader and become confirmed. Node v therefore broadcast a confirm to its neighbors, including u, to join the tree during round r'. All neighbors of v, including u, are confirmed members of the tree by round r'. By the definition of the algorithm, node v will broadcast message m two rounds after it received m. When v broadcasts message m in round $r+2$, this message will be received by u.

□

Lemma 6.22. *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 6.10) given that the activation of any nodes won't affect the path between a node and the leader (Lemma 6.18) and any node that receives a message will eventually send a confirmation message to its parent (Lemma 6.19).

If all the nodes in $A(r, r')$ are not part of the spanning tree of by round r, the nodes that are not part of the tree before round r will become part of the tree in time to receive the message (Lemma 6.21) and any node that receives a message will eventually send a confirmation message to its parent (Lemma 6.18).

□

Lemma 6.23. *The staggered activation reliable broadcast algorithm satisfies the second safety property of the reliable broadcast service under the assumption of staggered 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 disseminated by the leader. If the leader disseminates m, every node in $A(r, r')$ will receive m and notify its environment and eventually the leader will receive a confirm message (Lemma 6.22) (Lemma 6.19). 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 6.24. *The staggered activation reliable broadcast algorithm satisfies the third safety property of the reliable broadcast problem with staggered activation.*

Proof. The third safety property of reliable broadcast service with staggered activation requires that if a node u passes a message m to its environment, it has not passed m to its environment before and some node in the network previously received m from its environment. It follows from the definition of the algorithm that any node u that is not the leader of the tree will pass a message m to its environment only if and when it receives m from its parent in the tree. A node will also only pass a message to its environment if it is a confirmed member of the tree. If a node is a confirmed member of the tree, it has only one parent, which does not change during the execution. As a result, u will only process and pass m to its environment once. If node u is the leader of the tree, it will only pass m to its environment when it dequeues m from its queue, which will only happen once for each message. Therefore, a node u will only pass a message m to its environment at most one time.

The only way in which u will receive a message m to pass to its environment is if m is disseminated to the network by the leader. The leader will only disseminate a message m if m is sent to it by its own environment or by a node that received m from its environment. Therefore, a node u will only pass a message m to its environment if m has been received by a node in the network from its environment.

□

6.3 Variable Activation and Deactivation Reliable Broadcast

In this section, we prove the correctness of the algorithm assuming variable activation and deactivation of nodes. This section differs considerably from the previous sections in that it does not have a proof of correctness for leader election as this algorithm does not have a leader election subroutine. This section proves correct the variable activation and deactivation reliable broadcast algorithm by separately proving the four properties of reliable broadcast.

Theorem 6.25. *The variable activation and deactivation reliable broadcast algorithm correctly implements a reliable broadcast service under the assumption of variable activation and deactivation of nodes.*

Proof. The variable activation and deactivation reliable broadcast algorithm satisfies the liveness property(Lemma 6.26), the first safety property(Lemma 6.28), the second safety property(Lemma 6.29) and the third safety property(Lemma 6.30) of the reliable broadcast service assuming variable activation and deactivation of nodes. □

Lemma 6.26. *The variable activation and deactivation reliable broadcast algorithm satisfies the liveness property of the reliable broadcast problem assuming variable activation and deactivation of nodes.*

Proof. By the definition of 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 our definition of liveness. □

Lemma 6.27. *For round r and a later round $r' = r + n + 1$, every node in $A(r, r')$ will receive by round $r' - 1$ a message m broadcasted by node u during round r*

Proof. By the model definition, the active subset of the network graph is connected as a single component before the beginning of every round. In the worst case, the active subset of the network graph will satisfy the 1-interval connectivity property [**?**]. K-interval connectivity stipulates that there must exist a stable connected spanning sub-graph for every K consecutive rounds. If the active subset of the network graph is connected before the beginning of every round and nodes do not deactivate during a round of execution, 1-interval connectivity must hold under the assumption of variable activation and deactivation of nodes.

As proven by Kuhn et al, information can be disseminated between nodes in a network with 1-interval connectivity in n' rounds where n' is the size of the network [**?**]. In the variable activation and deativation reliable broadcast algorithm, the pieces of information being disseminated are the messages received by nodes from their environment. This proof only holds for nodes that activated before round r and do not deactivate before round r' because Kuhn et al did not examine networks where nodes activate and deactivate. $A(r, r')$ is the subset of nodes that activate before round r and do not deactivate before round r' so every node in $A(r, r')$ will receive any piece of information, including message m, that is disseminated during round r by round $r' - 1$.

□

Lemma 6.28. *The variable activation and deactivation reliable broadcast algorithm satisfies the first safety property of the reliable broadcast problem assuming variable activation and deactivation of nodes.*

Proof. For round r and a later round $r' = r + n + 1$, every node in $A(r, r')$ will receive message m broadcasted by node u during round r (Lemma 6.27). By definition of the algorithm, message m will have an execution round of $r' - 1$ if it is received by u from its environment at round r. If a message m' has an execution round of r'', any node that receives m' and is active for round r'' will pass m to its environment during round r''. Therefore, every node that receives m and is active for round $r' - 1$, which is a superset of $A(r, r')$ will pass message m to its environment during round $r' - 1$, which is between round r and round r'.

□

Lemma 6.29. *The variable activation and deactivation reliable broadcast algorithm satisfies the second safety property of the reliable broadcast problem assuming variable activation and deactivation of nodes.*

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

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

Proof. The third safety property of reliable broadcast service assuming variable activation and deactivation requires that if a node u passes a message m to its environment, it has not passed m to its environment before and some node in the network previously received m from its environment. By definition of the algorithm, any node u maintains a list of every unique message that it has received with a round of execution greater than or equal to round r where r is the current global round of execution. Since node u only passes messages to its environment from its list, which will only have a single copy of each message, u will only pass each message to its environment one time.

By the definition of the algorithm, the only way for a node will only add a message m to its list without receiving a message from a neighbor, is if it received m from its environment.

□