From Partial to Global Asynchronous Reliable Broadcast

Diana Ghinea
Department of Computer Science, ETH Zurich, Switzerland
ghinead@student.ethz.ch

Martin Hirt
Department of Computer Science, ETH Zurich, Switzerland
hirt@inf.ethz.ch

Chen-Da Liu-Zhang
Department of Computer Science, ETH Zurich, Switzerland
lichen@inf.ethz.ch

Abstract

Broadcast is a fundamental primitive in distributed computing. It allows a sender to consistently distribute a message among \( n \) recipients. The seminal result of Pease et al. [JACM’80] shows that in a complete network of synchronous bilateral channels, broadcast is achievable if and only if the number of corruptions is bounded by \( t < n/3 \). To overcome this bound, a fascinating line of works, Fitzi and Maurer [STOC’00], Considine et al. [JC’05] and Raykov [ICALP’15], proposed strengthening the communication network by assuming partial synchronous broadcast channels, which guarantee consistency among a subset of recipients.

We extend this line of research to the asynchronous setting. We consider reliable broadcast protocols assuming a communication network which provides each subset of \( b \) parties with reliable broadcast channels. A natural question is to investigate the trade-off between the size \( b \) and the corruption threshold \( t \). We answer this question by showing feasibility and impossibility results:

- A reliable broadcast protocol \( \Pi_{\text{RBC}} \) that:
  - For \( 3 \leq b \leq 4 \), is secure up to \( t < n/2 \) corruptions.
  - For \( b > 4 \) even, is secure up to \( t < \left( \frac{b-4}{b-2} n + \frac{8}{b-2} \right) \) corruptions.
  - For \( b > 4 \) odd, is secure up to \( t < \left( \frac{b-3}{b-1} n + \frac{6}{b-1} \right) \) corruptions.

- A nonstop reliable broadcast \( \Pi_{\text{nRBC}} \), where parties are guaranteed to obtain output as in reliable broadcast but may need to run forever, secure up to \( t < \frac{b-1}{b+1} n \) corruptions.

- There is no protocol for (nonstop) reliable broadcast secure up to \( t \geq \frac{b-1}{b+1} n \) corruptions, implying that \( \Pi_{\text{RBC}} \) is an asymptotically optimal reliable broadcast protocol, and \( \Pi_{\text{nRBC}} \) is an optimal nonstop reliable broadcast protocol.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Design and analysis of algorithms; Security and privacy → Cryptography

Keywords and phrases asynchronous broadcast, partial broadcast

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Broadcast protocols constitute a fundamental building block in distributed computing. They allow a designated party, the sender, to consistently distribute a message among \( n \) recipients, even if some of them exhibit arbitrary behaviour. It is used as an important primitive in many applications, such as verifiable secret-sharing or secure-multiparty computation [9, 2, 5, 13].
The seminal result of Pease et al. [12] shows that in the standard communication model of a complete synchronous network of pairwise authenticated channels, perfectly-secure Byzantine broadcast is achievable if and only if less than a third of the parties are corrupted (i.e., $t < n/3$). To overcome this bound, a line of works [8, 6, 15] have considered using stronger communication primitives such as partial broadcast channels, which guarantee that a message is consistent among all recipients on the channel. Hence, a natural question is to investigate a generalization of the classical broadcast problem, namely the trade-off between the strength of the communication primitives and the corruptive power from the adversary.

To the best of our knowledge, all works investigating such trade-offs for broadcast achievability [8, 6, 15] operate in the so-called synchronous model, where parties have access to synchronized clocks and there is a known upper bound on the network delay.

A more realistic setting is the so-called asynchronous model, where no timing assumption is made. In the asynchronous model, the classical notion of synchronous Byzantine broadcast, where termination is guaranteed, is not achievable, since one cannot distinguish between a dishonest sender not sending a message or an honest sender being slow [4, 3, 1]. Hence, one considers the weaker notion of reliable broadcast, where parties may not obtain output if the sender is dishonest; however, if an honest party obtains output, every honest party does as well. To the best of our knowledge, constructions of reliable broadcast [3] in the asynchronous setting are only known up to $t < n/3$ corruptions.

A natural question is then to investigate such trade-offs between the communication network and the corruptive power of the adversary in the asynchronous model.¹

In the asynchronous communication network $N_b$, where parties can reliably broadcast to any subset of $b$ parties, for which $t$ is there a reliable broadcast protocol secure up to $t$ corruptions?

We answer this question by showing feasibility and impossibility results:

**Feasibility Results.** In the network communication $N_b$, we show:

- A reliable broadcast protocol $\Pi_{RBC}$ that satisfies:
  - For $b \leq 4$, secure up to $t < n/2$ corruptions.
  - For $b > 4$ even, secure up to $t < \left( \frac{b-4}{b-2} n + \frac{8}{b-2} \right)$ corruptions.
  - For $b > 4$ odd, secure up to $t < \left( \frac{b-3}{b-1} n + \frac{6}{b-1} \right)$ corruptions.

- A nonstop reliable broadcast $\Pi_{nRBC}$, where parties are guaranteed to obtain output as in reliable broadcast but may need to run forever, secure up to $t < \frac{b-1}{b+1} n$ corruptions.

**Impossibility Result.** We show that in the network $N_b$, there is no protocol for (nonstop) reliable broadcast secure up to $t \geq \frac{b-1}{b+1} n$ corruptions, implying that $\Pi_{RBC}$ is an asymptotically optimal reliable broadcast protocol when $n \to \infty$ and $b = O(n)$, and that $\Pi_{nRBC}$ is an optimal nonstop reliable broadcast protocol.

¹ Investigating such trade-off is additionally motivated as a natural way to overcome the $n/3$-bound of constructing reliable broadcast in the pairwise channels setting. Note that this bound holds even assuming public-key infrastructure (PKI), in contrast to the synchronous counterpart, where Byzantine broadcast (and reliable broadcast) can be achieved under arbitrary many corruptions with PKI [7].
1.1 Related Work

All previous results in this realm operate in the synchronous model, where parties operate in rounds. Parties then proceed synchronously, and messages sent at round $r$ are guaranteed to be delivered by round $(r + 1)$.

Fitzi and Maurer [8] showed that assuming partial Byzantine broadcast channels among every triplet of parties, global Byzantine broadcast can be realized securely if and only if $t < n/2$. Considine et al. [6] generalized this result to the $b$-cast model, i.e. a partial Byzantine broadcast channel among any $b$ parties, where it was shown that Byzantine broadcast is achievable if and only if $t < \frac{b-1}{b+1}n$. Raykov [15] generalized this result to the setting of general adversaries [10] proving that broadcast is achievable from $b$-cast channels against adversary structures $\mathcal{A}$ if and only if $\mathcal{A}$ satisfies the so-called $(b + 1)$-chain-free condition.

Some additional works focus on the setting of incomplete communication networks, where some of the partial $b$-cast channels might be missing. Ravikant et al. [14] provide necessary and sufficient conditions for 3-cast networks to satisfy so that Byzantine agreement can be achieved while tolerating threshold adversaries in the range $n/3 \leq t < n/2$. In a follow-up work, Jaffe et al. [11] provide asymptotically tight bounds on the number of necessary and sufficient 3-cast channels to construct Byzantine agreement for the same threshold adversary.

1.2 Comparison to Previous Work

We argue that the asynchronous setting has different challenges from those that arise in the synchronous setting. Compared to previous works which assume and construct Byzantine broadcast, in this work we assume and construct reliable broadcast. That is, although our constructed primitive, reliable broadcast, is weaker than the traditional Byzantine broadcast primitive, as it does not guarantee termination in the dishonest sender case, it is also the case that we assume weaker primitives, which poses new challenges. For example, in the synchronous model, the primitive $proxcast$ [6] which provides a weak form of consistency where parties output a level of confidence, and is used as a core building block to construct Byzantine broadcast, can be achieved simply by allowing the sender to partially broadcast the input value via all possible $b$-casts, and letting each recipient $R_i$ take a deterministic decision based on all the outputs: $R_i$ decides on level $\ell_i$ as the minimum number of parties with whom $R_i$ sees only zeros. However, in the asynchronous model, parties cannot wait for the outcome of all partial reliable broadcasts from the sender, because the sender may be dishonest and so some of the partial channels may not output a value. As a consequence, $R_i$ would need to make a decision without knowing the outcome of all partial channels. In general, parties have to make progress in the protocol after seeing the messages from $n - t$ parties, as all other parties could be corrupted. This is especially troublesome in the dishonest majority region, where parties need to make progress after seeing messages from $n - t \leq t$ parties, i.e., even when potentially no message from any honest party is received. The key idea to overcome this is by observing that honest parties can actually wait for messages from $n - t$ parties that are consistent, i.e., it is allowed to wait for more than $n - t$ parties if inconsistency is received. This prevents the adversary, with the help of partial channels, to send arbitrary inconsistent messages.

2 Model and Definitions

We consider a setting with $n + 1$ parties, where a designated party, called the sender $S$, distributes a value to a set of $n$ recipients $\mathcal{R} = \{R_1, \ldots, R_n\}$. We note that the insider-
sender setting where the sender is also a recipient is a special case, as it can run in parallel both processors, acting as sender and as recipient simultaneously.

2.1 Communication, Adversary and Setup

In this work, we generalize the communication model where, in addition to a complete network of pair-wise authenticated channels, parties have access to partial reliable broadcast channels as well. We refer to such a partial reliable broadcast channel \( \text{RBC}(S, \{R_1, \ldots, R_{b-1}\}) \) with a sender \( S \) and \( b - 1 \) additional recipients \( \{R_1, \ldots, R_{b-1}\} \) a \( b \)-cast channel. We denote by \( N_b \) the generalized communication model where each party \( P \), sender or recipient, in addition has access to all channels \( \text{RBC}(P, \{R_i, \ldots, R_{n_b-1}\}) \), where \( R_i, \ldots, R_{n_b-1} \) are \( b - 1 \) additional recipients.

The network is fully asynchronous. That is, we assume that the adversary has full control over the network and can schedule the messages in an arbitrary manner. However, each message must be eventually delivered.

We consider the same adversarial model and setup as in [6]. That is, we consider an adaptive adversary who can gradually corrupt parties and take full control over them. Moreover, we require our protocols to be unconditionally secure, meaning that security holds even against a computationally unbounded adversary. Note, however, that our impossibility proofs hold even with respect to a static adversary that is assumed to choose the corrupted parties at the beginning of the protocol execution and in addition is computationally bounded. Finally, we consider the setting where parties have no public-key infrastructure available.

2.2 Reliable Broadcast

Reliable broadcast is a fundamental primitive in distributed computing which allows a designated party \( S \), called the sender, to consistently distribute a message towards a set of recipients \( R = \{R_1, \ldots, R_n\} \).

Definition 1. A protocol \( \pi \) where initially the sender \( S \) holds an input \( m \) and every recipient \( R_i \) terminates upon generating output is a reliable broadcast protocol up to \( t \) corruptions, if the following properties are satisfied:

- **Validity:** If the sender is honest, the sender terminates and every honest recipient terminates with output \( m \).
- **Consistency:** If an honest recipient terminates with output \( m \), every honest recipient terminates with the same value.

We additionally define a slightly weaker version of broadcast, which requires the recipients to obtain outputs like in reliable broadcast, but may need to run forever.

Definition 2. A protocol \( \pi \) where initially the sender \( S \) holds an input \( m \) is a nonstop reliable broadcast protocol up to \( t \) corruptions, if the following properties are satisfied:

- **Validity:** If the sender is honest, every honest recipient outputs \( m \).
- **Consistency:** If an honest recipient outputs \( m \), every honest recipient outputs \( m \).
3  A Warm-Up Protocol in $N_3$

In this section, we consider the model $N_3$, where the parties have access to 3-cast channels. That is, any party can reliably broadcast messages to any subset of 2 recipients.

We present a reliable broadcast protocol $\Pi^{n,3}_{RBC}$ in the communication network $N_3$ secure up to $t < n/2$ corruptions inspired by Brachas reliable broadcast protocol [3]. In Section E, we show that the construction is optimal with respect to the corruption threshold. $\Pi^{n,3}_{RBC}$ first lets the sender $S$ mega-send its input message $m$ (distribute $m$ to any two recipients, via all available 3-cast channels). Any recipient $R_i$ that mega-receives the same message ($MSG$, $m$) from $S$ (receives consistently ($MSG$, $m$) via all (in total $n$ - 1) 3-cast channels from $S$), mega-sends a message ($READY$, $m$) notifying all recipients that it is ready to output $m$. Any recipient $R_i$ that receives consistent notification messages ($READY$, $m$) from $t + 1$ different recipients mega-sends ($READY$, $m$). Finally, any recipient $R_i$ that mega-sent ($READY$, $m$) and mega-received consistent notification messages ($READY$, $m$) from $n - t - 1$ other different recipients than himself, outputs $m$ and terminates.

Intuitively, the usage of 3-cast channels guarantees that honest parties send consistent $READY$ messages, because two recipients cannot mega-receive different messages from the sender (they have a common 3-cast channel with the sender). Moreover, note that if an honest recipient $R_i$ mega-receives ($READY$, $m$) from $R_j$, then any honest recipient $R_k$ receives ($READY$, $m$) from $R_j$ via the 3-cast containing $R_j$ as the sender and ($R_i$, $R_k$) as the recipients. This ensures that if an honest recipient outputs a message $m$, meaning that it sent ($READY$, $m$) and mega-received ($READY$, $m$) from $n - t - 1$ different recipients, then any honest recipient eventually receives ($READY$, $m$) from $n - t + 1$ different recipients. It then follows that all honest parties mega-send ($READY$, $m$), so all honest parties mega-receive ($READY$, $m$) from at least $n - t - 1$ parties and terminates. The protocol is described below, and a formal analysis of it can be found in Appendix A.

**Protocol $\Pi^{n,3}_{RBC}$**

**Code for the sender $S$**

1: On input $m$, send ($MSG$, $m$) to every pair of recipients via 3-cast and terminate with output $m$.

**Code for recipient $R_i$**

1: Upon receiving ($MSG$, $m$) via all 3-cast $RBC(S, \{R_i, R_j\})$, $R_j \in R$, send ($READY$, $m$) to every pair of recipients via 3-cast.

2: Upon receiving ($READY$, $m$) from $t + 1$ different recipients, i.e. from $RBC(R_i, \{R_i, \cdot \})$, $R_j \in T$, $|T| \geq t + 1$, if no READY message was sent, send ($READY$, $m$) to every pair of recipients via 3-cast.

3: Upon receiving ($READY$, $m$) via all 3-cast $RBC(R_k, \{R_i, R_j\})$, $R_j \in R$, $R_k \in T$, $|T| \geq n - t - 1$, if ($READY$, $m$) was sent, output $m$ and terminate.

4  Notation for Protocols in $N_b$

We consider the model $N_b$ where parties, sender or recipients, have access to $b$-cast channels. That is, any party can reliably broadcast a message to any subset of $b - 1$ recipients.

We introduce some definitions that will be convenient to describe our protocols. Generalizing the terminology of parties mega-receiving messages in Section 2.2, we add a definition for parties that receive a message via all $b$-cast channels including a certain subset of recipients.
Definition 3. Let $P \in \{S\} \cup \mathcal{R}$ be a party and $U \subseteq \mathcal{R} \setminus \{P\}$. We denote $\mathcal{B}_P(U) := \{\text{RBC}(P, V) : U \subseteq V \subseteq \mathcal{R} \land |V| = b - 1\}$ the set of $b$-cast channels that include $P$ as the sender and any subset of $b - 1$ recipients that includes $U$ as receivers.

In particular, note that if $U' \subseteq U$, then $\mathcal{B}_P(U) \subseteq \mathcal{B}_P(U')$.

Definition 4. We say that a party $P \in \{S\} \cup \mathcal{R}$ $U$-sends a message $m$ if it sends $m$ through every channel in $\mathcal{B}_P(U)$.

Definition 5. We say that a recipient $R \in \mathcal{R}$ $U$-receives a message $m$ from a party $P$ if it receives $m$ through every channel $\mathcal{B}_P(U)$. Moreover, we say that $R$ $l$-receives $m$ if such $U \subseteq \mathcal{R}$ with $|U| = l$ exists.

Remark 6. Note that if $|U| \geq b$, $\mathcal{B}_P(U) = \emptyset$, and any recipient $b$-receives any message.

With these definitions, we prove some properties about $l$-receiving values among parties.

Lemma 7. Let $l \leq b$. If a recipient $R_i$ $l$-receives $m$ from $P$, then all recipients eventually $(l + 1)$-receive $m$.

Proof. If $l \geq b - 1$, then by definition every recipient $b$-receives $m$. If $l < b - 1$, this means that if $R_i$ receives $m$ from all channels containing a fixed set of $l$ recipients, then any recipient $R_j$ received $m$ via all channels containing the same set and $R_j$. More formally, let $U$ be the set such that $R_i$ $U$-receives $m$ from $P$. Then, any $R_j$ eventually $(U \cup \{R_j\})$-receives $m$, since $\mathcal{B}_P(U \cup \{R_j\}) \subseteq \mathcal{B}_P(U)$.

Lemma 8. If $R_i \in \mathcal{R}$ $U$-receives $m$ from $P$ and $R_j \in \mathcal{R}$ $V$-receives $m'$ from $P$, with $m \neq m'$, then $|U \cup V| \geq b$.

Proof. If $|U \cup V| < b$, then $\exists c \in \mathcal{B}_P(U \cup V)$. Hence, $c$ outputs $m$ to $R_i$ and $m'$ to $R_j$, contradicting the consistency of the $b$-cast channel $c$.

4.1 Predicate LEVELS

Our protocols follow a specific pattern. They first allow the sender $S$ to send its input $m$ to each subset of $b - 1$ recipients. From now on, only the recipients interact with each other. Whenever a condition $C_1$ is met for $R_i$, it sends a message via all available $b$-cast channels, and as soon as a (stricter) condition $C_2$ is met, it outputs its final message.

At the core of the conditions is the predicate LEVELS, which is reminiscent of the notion of proxcast [6]. Intuitively, the predicate LEVELS can be understood as an indicator of the consistency level achieved so far. Level 1 is the strongest, and indicates that from the view of $R_i$, the sender “looks honest”. Level 2 indicates that there might be an honest receiver $R_j$ for whom the sender looks honest, i.e., that $R_j$ is at level 1. Level 3 indicates that there might be an honest $R_k$ at level 2, and so on.

Roughly speaking, to achieve level 1, $R_i$ checks that it $1$-received a message $m$ from the sender $S$, and that $n - t$ parties confirm to be at level 1 as well. Intuitively, a recipient $R_i$ is at level $l > 1$ if from its point of view there could be an honest recipient that is at level $l - 1$. Note that if an honest recipient is at level $l - 1$, then all the $n - t$ honest recipients eventually receive enough messages from the sender and enough confirmations to place themselves on level $l - 1$ or $l$. Hence, for level $l$, $R_i$ needs to $l$-receive $m$ from $S$ and there must be a sequence of subsets of parties $L_1, \ldots, L_l$, where $L_k$ can be interpreted as a set containing parties confirming to be at level $k$, that satisfies $|L_1| \geq n - t$ and
We present a protocol for any finite input space achieving reliable broadcast in \( N \). The protocol generalizes the simple protocol for broadcast presented in Section 2.2. In the region of dishonest majority, \( n - t < t \) confirmations are not sufficient to make a decision since all confirmations could come from dishonest parties. Instead, we store the recipients that send confirmations and make use of the LEVELS predicate to evaluate the consistency level.

\( \forall 1 \leq k \leq l - 1: |L_k| + |L_{k+1}| \geq n - t \). For technical reasons, it will be useful to consider different sizes for each of the two conditions above (see Figure 1).

More concretely, the predicate satisfies level \( l \) for parameters \( n \), \( t_v \) and \( t_c \) if:

\[
\text{LEVELS}_{n,t_v,t_c}(L_1, \ldots, L_l) = (\forall 1 \leq k < k' \leq l : L_k \cap L_{k'} = \emptyset) \land \\
(\forall 1 \leq k \leq l : |L_k| \geq n - t_v) \land (\forall 2 \leq k \leq l : |L_k| \geq 1) \land \\
(\forall 1 \leq k \leq l - 1 : |L_k| + |L_{k+1}| \geq n - t_c)
\]

For \( l = 0 \), the predicate is true by default.

We include a few properties of the predicate, which will be useful in the proofs.

\( \blacktriangleright \text{Lemma 9.} \) If \( \text{LEVELS}_{n,t_v,t_c}(L_1, \ldots, L_l) \) holds, then \( \text{LEVELS}_{n,t_v,t_c}(L_1, \ldots, L_k) \) holds for any \( 0 \leq k \leq l \).

\( \text{Proof.} \) Follows from the definition of the LEVELS predicate.

We denote \( \lambda(l) \) the minimum number of recipients that can be placed into sets \( L_1, \ldots, L_l \) satisfying \( \text{LEVELS}_{n,t_v,t_c}(L_1, \ldots, L_l) \). That is, the minimum \( |\bigcup_{k=1}^l L_k| \), for sets \( L_1, \ldots, L_l \) satisfying \( \text{LEVELS}_{n,t_v,t_c}(L_1, \ldots, L_l) \). Naturally, \( \lambda(0) = 0 \). The next lemma computes \( \lambda(l) \), \( l > 0 \). The proof can be found in Appendix B.

\( \blacktriangleright \text{Lemma 10.} \) Given \( 0 \leq t_c \leq t_v < n \) and \( l > 0 \), \( \lambda(l) \) can be computed as follows.

\[
\lambda(l) = \begin{cases} 
\frac{t_v + 1}{2}(n - t) & \text{if } l \text{ is odd}, \\
\frac{t_v}{2}(n - t) + 1 & \text{if } l \text{ is even.}
\end{cases}
\]

\[
\lambda(l) = \begin{cases} 
(n - t_v) + \frac{l-1}{2}(n - t_c) & \text{if } l \text{ is odd,} \\
\frac{1}{2}(n - t_c) & \text{if } l \text{ is even.}
\end{cases}
\]

5. Asymptotically Optimal Reliable Broadcast Protocol

We present a protocol for any finite input space achieving reliable broadcast in \( N_0 \) that is:

- for \( 3 \leq b \leq 4 \), secure up to \( t < n/2 \) corruptions;
- for \( b > 4 \) even, secure up to \( t < \left(\frac{b-4}{b-2}n + \frac{8}{b-2}\right) \) corruptions;
- for \( b > 4 \) odd, secure up to \( t < \left(\frac{b-3}{b-1}n + \frac{6}{b-1}\right) \) corruptions.

5.1 Protocol Description

The protocol generalizes the simple protocol for 3-cast presented in Section 2.2. In the region of dishonest majority, \( n - t < t \) confirmations are not sufficient to make a decision since all confirmations could come from dishonest parties. Instead, we store the recipients that send confirmations and make use of the LEVELS predicate to evaluate the consistency level.
Initially, the sender forwards its input \( m \) to every subset of \( b-1 \) recipients via all available \( b \)-cast channels. If \( S \) is honest, once a recipient \( R_i \) \( 1 \)-receives \( m \) from \( S \), it sends (READY, \( m \)) via all \( b \)-cast channels to notify the other recipients. It outputs \( m \) when in addition it \( 1 \)-receives (READY, \( m \)) from \( n-t-1 \) other recipients, completing the protocol at level 1.

If \( S \) is corrupted, it is possible that \( R_i \) is the only honest recipient that \( 1 \)-receives \( m \) from \( S \) and \( 1 \)-receives (READY, \( m \)) from the other \( n-t-1 \) recipients, which are corrupted. However, any other honest recipient \( R_j \) eventually \( 2 \)-receives \( m \) from \( S \), \( 2 \)-receives (READY, \( m \)) from the \( n-t-1 \) corrupted recipients and \( 1 \)-receives (READY, \( m \)) from \( R_i \). Once \( R_j \) receives these messages, it sends (READY, \( m \)) and outputs \( m \), completing the protocol at level 2.

Following this line of reasoning, for \( l > 1 \), an honest recipient \( R_i \) that \( l \)-receives \( m \) from \( S \) sends (READY, \( m \)) when it believes that an honest recipient \( R_j \) completed the protocol on level \( l - 1 \). Then, \( R_i \) outputs \( m \) when it is sure that any honest recipient that eventually \((l + 1)\)-receives \( m \) from \( S \) will have enough evidence that \( R_i \) terminated at level \( l \). This guarantees that when an honest recipient \( R_i \) completes the protocol with output \( m \), every honest recipient eventually sends (READY, \( m \)). Additionally, we set the threshold so that it ensures that honest recipients cannot send READY for different messages, and moreover, if all honest recipients send (READY, \( m \)), all honest recipients complete the protocol with output \( m \).

Each recipient \( R_i \) keeps sets \( R^i(m,k) \), \( 1 \leq k \leq b \), where it stores the recipients from whom it \( k \)-received (READY, \( m \)). We define two predicates which will be helpful when describing the protocol. Predicate \( \text{DONE}^i(m,l) \) indicates that the \( l \) levels have been completed for the message \( m \), and hence \( R_i \) can complete the protocol. Predicate \( \text{NOTIFY}^i(m,l) \) indicates that there is a seemingly honest recipient \( R_j \) who satisfied the predicate \( \text{DONE}^i(m,l-1) \), meaning that \( R_i \) should send a notification for level \( l \) and message \( m \). In the following, we formally describe the two predicates \( \text{NOTIFY}, \text{DONE} \) (see Figure 2).

\[
\text{NOTIFY}^i(m,1) = \text{true}
\]

\[
\text{NOTIFY}^i(m,l) = \exists L_1^i \subseteq R^i(m,l), \ldots, L_k^i \subseteq R^i(m,l-k+1), \ldots, L_{l-1}^i \subseteq R^i(m,2) \text{ s.t.}
\]

\[
(\forall 1 \leq k \leq l-1 : L_k^i \cap R^i(m,l-k) \neq \emptyset) \land
\]

\[\text{LEVELS}_{n,t}(L_1^i, \ldots, L_{l-1}^i) \text{ holds.}\]

\[
\text{DONE}^i(m,l) = \exists L_1^i \subseteq R^i(m,l), \ldots, L_k^i \subseteq R^i(m,l-k+1), \ldots, L_l^i \subseteq R^i(m,1) \text{ s.t.}
\]

\[
(\forall 1 \leq k \leq l-1 : L_k^i \cap R^i(m,l-k) \neq \emptyset) \land
\]

\[\text{LEVELS}_{n,t}(L_1^i, \ldots, L_l^i) \text{ holds.}\]

![Figure 2](image-url) The condition \( \text{NOTIFY}^i(m,l) \), in contrast to the condition \( \text{DONE}^i(m,l) \).

We hereby formally describe the protocol.
5.2 Resilience Proof

We first state a number of intermediate lemmas that help in proving validity and consistency.

▶ Lemma 11. If \( R \in R^i(m, k) \) for an honest recipient \( R_i \) and for \( k < b \), then for any honest recipient \( R_j \), eventually \( R \in R^i(m, k + 1) \) and also \( R \notin R^i(m', k') \), for any \( m' \neq m \) and \( k' < b - k \).

Proof. Follows from Lemmas 7 and 8. 

▶ Lemma 12. Assume that \( t < \lambda(b - 2) \) and let \( U \) denote the smallest set containing an honest recipient such that the sender \( U \)-sends \( m \). If \( |U| \geq b - 1 \), then no honest recipient can send \( \text{READY}, m \) or complete the protocol with output \( m \).

Proof. Assume that \( R_i \) is the first honest recipient that sends \( \text{READY}, m \).

Then, \( \text{NOTIFY}^i(m, b - 1) \) holds based on messages sent by corrupted recipients only. The definition of the \( \text{NOTIFY} \) predicate hence implies that \( t \geq \lambda(b - 2) \), contradicting the hypothesis. Since \( \text{NOTIFY}^i(m, b - 1) \) is false, \( \text{DONE}^i(m, b - 1) \) is false as well, and therefore \( R_i \) cannot output \( m \).

▶ Lemma 13. If \( t < \lambda(b - 2) \) and an honest recipient \( R_i \) completes the protocol with output \( m \), then any other honest recipient eventually sends \( \text{READY}, m \).

Proof. Assume that \( R_i \) is the first honest recipient that completes the protocol with output \( m \). Then, it \( l \)-received \( m \) from the sender and \( \text{DONE}^i(m, l) \) holds, which implies that \( \text{NOTIFY}^i(m, l) \) also holds. According to Lemma 12, \( l < b - 1 \). Hence, from Lemma 11 and from the definitions of \( \text{DONE} \) and \( \text{NOTIFY} \), it follows that \( \text{NOTIFY}^i(m, l + 1) \) eventually holds for every other honest recipient \( R_j \). In addition, according to Lemma 7, every honest recipient eventually \((l + 1)\)-receives \( m \) from the sender, and hence sends \( \text{READY}, m \).

▶ Lemma 14. If \( t < \lambda(b - 2) \) and every honest recipient sends \( \text{READY}, m \), then every honest recipient can output \( m \).

Proof. Let \( R_i \) denote the first honest recipient that sends \( \text{READY}, m \). Then, \( R_i \) \( l \)-receives \( m \) and \( \text{NOTIFY}^i(m, l) \) holds for \( l < b - 1 \), according to Lemma 12. From the definition of \( \text{NOTIFY}^i(m, l) \), it follows that \( \exists L_1^i \subseteq R^i(m, l), \ldots, L_k^i \subseteq R^i(m, l - k + 1), \ldots, L_{l-1}^i \subseteq R^i(m, 2) \) such that \( L_k^i \cap R^i(m, l - k) \neq \emptyset \) for every \( 1 \leq k \leq l - 1 \) and \( \text{LEVELS}_{n,t}(L_1^i, \ldots, L_{l-1}^i) \) holds. Since \( R_i \) is the first honest recipient to send \( \text{READY}, m \), every recipient in \( \bigcup_{k=1}^{l-1} L_k^i \) is corrupted.
The following lemma shows that for certain thresholds, the \textit{READY} messages that honest parties send are unique. The proof is enclosed in Appendix C.

\begin{lemma}
Assume that \( t < \lambda(l-1) + \lambda(l'-1) - \lambda(l+l'-b+1) + 2 \) for any \( l > 0, l' > 0 \) such that \( l + l' \geq b \). Then, if \textit{NOTIFY}(m, l) holds for an honest recipient \( R_i \), \textit{NOTIFY}(m', l') is false for any honest recipient \( R_j \), where \( m \neq m' \).
\end{lemma}
Lemma 20. Assume that \( t < \lambda(l-1) + \lambda(l'-1) - \lambda(l + l' - b + 1) + 2 \) for any \( l > 0, l' > 0 \) such that \( l + l' \geq b \). Then, if an honest recipient \( R_i \) outputs \( m \), it sent \((\text{READY}, m)\).

Proof. \( R_i \) outputs \( m \), there is an \( l \) such that \( \text{DONE}^t(m,l) \) holds and \( R_i \) has \( l\)-received \( m \), which implies that \( \text{NOTIFY}^t(m,l) \) holds. According to Lemmas 19 and 8, \( \text{NOTIFY}^t(m',l') \) is false for any \( m' \neq m \) and \( l' \) such that \( R_i \) can \( l\)-receive \( m' \). Hence, \( R_i \) cannot send \((\text{READY}, m')\) and therefore it sent \((\text{READY}, m)\) before completing the protocol.

Lemma 21. Assume that \( t < \lambda(l-1) + \lambda(l'-1) - \lambda(l + l' - b + 1) + 2 \) for any \( l > 0, l' > 0 \) such that \( l + l' \geq b \). Then, if an honest recipient \( R_i \) sends \((\text{READY}, m)\), no honest recipient completes the protocol with output \( m' \neq m \).

Proof. If an honest recipient \( R_j \) outputs \( m' \), then according to Lemma 20, \( R_j \) sent \((\text{READY}, m')\), which contradicts Lemma 19.

Lemma 22. If \( t < \lambda(b-2) \), validity is satisfied.

Proof. Let \( m \) denote the input of the honest sender and let \( m' \neq m \). According to Lemma 12 and since the recipients only \( b\)-receive \( m' \), no honest recipient sends \((\text{READY}, m')\) or outputs \( m' \). Every recipient can 1-receive \( m \) from \( S \), hence every honest recipient eventually sends \((\text{READY}, m)\). From Lemma 14, we obtain that every honest recipient eventually outputs \( m \).

Lemma 23. If \( t < \lambda(b-2) \) and \( t < \lambda(l-1) + \lambda(l'-1) - \lambda(l + l' - b + 1) + 2 \) for every \( l, l' > 0 \) such that \( l + l' \geq b \), consistency holds.

Proof. Assume an honest recipient \( R_i \) completes the protocol with output \( m \). According to Lemmas 19 and 21, no honest recipient sends \((\text{READY}, m')\) or outputs \( m' \), where \( m' \neq m \). Consequently, it follows from Lemmas 20 and 13, that every honest recipient outputs \((\text{READY}, m)\). Then, we obtain from Lemma 14 that every honest recipient outputs \( m \).

Theorem 24. If \( b > 4, t < \lambda(b-2) \) and \( t < \lambda(l-1) + \lambda(l'-1) - \lambda(l + l' - b + 1) + 2 \) for any \( l, l' > 0 \) such that \( l + l' \geq b \), \( \Pi^{n,b}_{\text{RBC}} \) achieves reliable broadcast in \( N_b \).

Proof. We obtain from Lemma 22 that \( \Pi^{n,b}_{\text{RBC}} \) achieves validity, and it follows from Lemma 23 that \( \Pi^{n,b}_{\text{RBC}} \) achieves consistency.

Since from Lemmas 52 and 53 \( t < \frac{b-4}{b-2} n + \frac{8}{b-2} \) (resp. \( t < \frac{b-4}{b-2} n + \frac{6}{b-1} \)) implies the hypothesis of Theorem 24, we obtain the following:

Corollary 25. If \( b > 4 \) is even, \( \Pi^{n,b}_{\text{RBC}} \) achieves resilient reliable broadcast in \( N_b \) secure against \( t < \frac{b-4}{b-2} n + \frac{8}{b-2} \) corruptions.

Corollary 26. If \( b > 4 \) is odd, \( \Pi^{n,b}_{\text{RBC}} \) achieves resilient reliable broadcast in \( N_b \) secure against \( t < \frac{b-4}{b-1} n + \frac{5}{b-1} \) corruptions.

Optimal Nonstop Broadcast

In this section, we present a protocol that achieves nonstop reliable broadcast secure against \( t < \frac{b-4}{b-1} n \) corruptions. In Appendix E, we show that the construction is optimal with respect to the corruption threshold.
The construction follows an information gathering approach, where parties recursively invoke a two-threshold nonstop reliable broadcast, along the lines of [6]. We denote a \((t_v,t_c)\)-nonstop reliable broadcast a protocol achieving validity (resp. consistency) up to \(t_v\) (resp. \(t_c\)) corrupted recipients.

For simplicity, in this section we focus on protocols with binary input domain. One can always extend it to any finite domain by invoking the protocol in parallel for each bit of the message to be sent.

### 6.1 Protocol Description

Initially, the sender forwards his input via \(b\)-cast to every subset of \(b-1\) recipients. Each recipient \(R_i\), now as sender, recursively invokes the two-threshold broadcast protocol with parameters \(t'_v = t_v\) and \(t'_c = t_c - 1\) towards the \(n-1\) left recipients, to distribute messages. The idea is that if \(R_i\) is honest, then validity holds in the recursive calls, and this will be enough to achieve validity. On the other hand, if the sender is dishonest, it means that among the \(n-1\) recipients, there is one less corrupted party, and so resilience up to \(t'_c\) corruptions is enough.

The message that each \(R_i\) distributes indicates a level \(l\) of confidence on a message \(m\), meaning that it \(l\)-receives \(m\) from the sender, and did not send a message for \((b-l)\)-receiving \(m' \neq m\). In particular, \(R_i\) sends \((\text{READY}, k, m_k, l_k)\) to announce that it \(l_k\)-received \(m_k\) from the sender and that this is the \(k\)-th message that it sends. We think about the protocol as the recipients having available a designated channel for each level, so that any receiver can verify the order of the messages. Following the reasoning presented in Section 4.1, a recipient outputs once it receives enough confirmations that other recipients achieved the same consistency level. Given that the recursive broadcast works, it is then guaranteed that when an honest recipient outputs \(m\) with level \(l\), every other recipient confirms with level \(l+1\), and eventually every recipient outputs at level \(l+1\). Intuitively, the reason why the protocol does not allow parties to terminate, is because recipients do not know whether they will need to confirm messages for other recursive calls of the protocol (there might be some honest recipient still waiting for a confirmation message to terminate).

#### Formal condition to output

Formally, a recipient \(R_i\) outputs \(m\) on level \(l\) when it \(l\)-receives \(m\) from the sender and the following predicate is satisfied:

\[
\text{DONE}^l(m, l) = \exists L_{1} \subseteq R^1(m, 1), \ldots, L_{l} \subseteq R^l(m, l) \text{ s.t. LEVELS}_{n, t, t_c}(L_1, \ldots, L_l) \text{ holds.}
\]

- **Definition 27.** A message \((\text{READY}, k, m_k, l_k)\) is consistent with respect to the set of messages \(\{(\text{READY}, k', m_{k'}, l_{k'}) | 1 \leq k' < k\}\) if for every such \(k'\), \(l_k + l_{k'} > b\) when \(m_{k'} \neq m_k\) and \(l_k \neq l_{k'}\) when \(m_{k'} = m_k\).

- **Definition 28.** A recipient \(R_i\) accepts a message \((\text{READY}, k, m_k, l_k)\) from a recipient \(R_j\) if it has received the messages \(\{(\text{READY}, k', m_{k'}, l_{k'}) | 1 \leq k' \leq k\}\) from \(R_j\) and \((\text{READY}, k, m_k, l_k)\) is consistent with respect to them.

- **Definition 29.** \(R^l(m, l) \subseteq \mathcal{R}\) denotes the set of recipients that \(R_i\) has accepted a message \((\text{READY}, \cdot, m, l)\) from. We use \(R^l(m, \leq l)\) to denote \(\bigcup_{j=1}^{l} R^j(m, j)\).

We now formally present our protocol \(\Pi_{\text{non}}^{n,b}(t_v, t_c, S, \mathcal{R})\) for \(n\) recipients in the communication network \(\mathcal{N}_b\), where \(S\) is the sender, and \(\mathcal{R}\) is the set of recipients.
Lemma 30.

Lemma 31.

On input \(P\) when receiving \(I\) for any honest recipient \(R\) \(\Pi\) are at most \(Q\).

Proof. for every honest recipient \(R\) \(\Pi\) consistent with respect to your previously sent messages, send \((\text{READY}, k + 1, m, l)\) to the other recipients by invoking \(\Pi_{\text{ABRC}}^{n-1,b}(\min(t_v, |R \setminus \{R_i\}| - 1), t_e - 1, R_i, R \setminus \{R_i\})\), adds \(R_i\) to \(R'(m, l)\), and increment \(k\).

When observing for the first time that \(\text{DONE}'(m, l)\) holds and you have \(l\)-received \(m\) from the sender, output \(m\).

6.2 Resilience Proof

In this section, we define a predicate \(Q_n^b(t_v, t_e)\) which reflects the conditions that \(t_v\) and \(t_e\) must satisfy such that \(\Pi_{\text{ABRC}}^{n,b}\) achieves \((t_v, t_e)\)-nonstop reliable broadcast.

\[ Q_n^b(t_v, t_e) = \text{true} \] since \(\Pi_{\text{ABRC}}^{n,b}\) implements an ideal b-cast. Then, for \(n > b\), \(Q_n^b(t_v, t_e) = \bigwedge_{k=0}^{n-b-1} P_n^{b-k}(n - k - 1, t_e)\), where \(P_n^b(t_v, t_e)\) denotes a local predicate enclosing the conditions that \(t_v\) and \(t_e\) must satisfy assuming that \(\Pi_{\text{ABRC}}^{n-1,b}\) achieves \((t_v, t_e)\)-nonstop reliable broadcast. We prove that \(P_n^b(t_v, t_e)\) can be defined as follows.

\[ P_n^b(t_v, t_e) = [t_v < \lambda(b - 1) \lor n < \lambda(b)] \land 
\forall 1 \leq l \leq b : (t_v < \lambda(l - 1) + \lambda(b - l - 1) \lor n < \lambda(l) + \lambda(b - l)) \]

6.2.1 Validity

In each result enclosed in this subsection, we assume that the sender is honest, that there are at most \(t_v\) corrupted recipients, and that \(Q_{n-1}^b(\min(t_v, |R| - 2), t_e - 1)\) holds, i.e., that \(\Pi_{\text{ABRC}}^{n-1,b}(\min(t_v, |R \setminus \{R_i\}| - 1), t_e - 1, R_i, R \setminus \{R_i\})\) achieves validity (resp. consistency) up to \(\min(t_v, |R \setminus \{R_i\}| - 1)\) (resp. \(t_e - 1\)) corruptions. We show that satisfying \(P_n^b(t_v, t_e)\) suffices for \(\Pi_{\text{ABRC}}^{n,b}\) to achieve validity.

Lemma 30. If an honest recipient \(R_i\) sends \((\text{READY}, i, m, l)\), then eventually \(R_i \in R^j(m, l)\) for any honest recipient \(R_j\).

Proof. \(R_i\) invokes \(\Pi_{\text{ABRC}}^{n-1,b}(\min(t_v, |R \setminus \{R_i\}| - 1), t_e - 1, R_i, R \setminus \{R_i\})\) with at most \(t_v\) corrupted recipients. It follows that \(\Pi_{\text{ABRC}}^{n-1,b}\) achieves validity. Hence, every honest recipient \(R_j\) eventually receives the messages sent by \(R_i\) and, since \(R_i\) is honest, \(R_j\) accepts each such message and adds \(R_i\) to \(R^j(m, l)\).

Lemma 31. If \(m\) is the input of the honest sender, then every recipient eventually outputs \(m\).
Lemma 36. Let \( U \subseteq \mathcal{R} \) be the smallest set containing an honest recipient such that \( S \) sends \( m \). Then, for any honest \( R_t \in \mathcal{R} \), there is no honest party in \( \mathcal{R}^n \leq m \leq b - l - 1 \) where \( l = |U| \).

Proof. Assume that \( R_j \) is an honest recipient in \( \mathcal{R}^j(m, l - 1) \cup \mathcal{R}^j(m', l - 1) \). Since Lemma 34 guarantees that \( R_l \) receives the messages that \( R_j \) sends, \( R_j \) has \( (l - 1) \)-received \( m \), which contradicts the hypothesis, or has \( (b - 1) \)-received \( m' \), contradicting Lemma 8.

Lemma 37. Let \( U \subseteq \mathcal{R} \) be the smallest set containing an honest recipient such that \( S \) sends \( m \) and let \( l = |U| \). Then, it eventually holds that \( |\mathcal{R}^j(m, l + 1) \cup \mathcal{R}^j(m, l - 1)| \geq n - t_e \) for any honest recipient \( R_i \).
Let $R_j$ denote an arbitrary honest recipient. From Lemma 36, $R_j \notin R^i(m, \leq l - 1) \cup R^i(m', \leq b - l - 1)$. $R_j$ eventually $(l + 1)$-receives $m$ according to Lemma 7, and it sends (READY, $m, l + 1$) since it is consistent with respect to any messages it previously sent. Consequently, $R_j \in R^i(m, \leq l + 1) \setminus R^i(m, \leq l - 1)$ since according to Lemma 34, $R_i$ eventually receives each message that $R_j$ sends.

\begin{lemma}
If $m \neq m'$ and $l + l' \leq b$, $R^i(m, \leq l) \cap R^i(m', \leq l') = \emptyset$ for any honest $R_i$.
\end{lemma}

\begin{proof}
Suppose that there is an honest $R_j \in R^i(m, \leq l) \cap R^i(m', \leq l')$. It follows from Lemma 34 that $R_j$ sent (READY, $m, l + 1$) such that $l_1 \leq l$ and (READY, $m', l_2$) such that $l_2 \leq l' \leq b - l$. However, as $l_1 + l_2 \leq b$, $R_i$ only accepts one of these messages.
\end{proof}

\section*{Properties of the DONE Predicate}

\begin{lemma}
If $\text{DONE}'(m, l)$ holds for an honest recipient $R_i$, then $\text{DONE}'(m, l)$ eventually holds for any other honest $R_j$.
\end{lemma}

\begin{proof}
Follows from Lemma 35, as $R_j$ eventually receives the same messages as $R_i$.
\end{proof}

\begin{lemma}
Let $U \subseteq R$ be the smallest set containing an honest recipient such that $S$ $U$-sends $m$ and let $l = |U|$. Assume that $\text{DONE}'(m, l)$ holds for an honest $R_i$.

If $P^b_n(t_v, t_c)$ holds, then $\text{DONE}'(m', b - l)$ cannot be satisfied for $m' \neq m$.
\end{lemma}

\begin{proof}
Assume that $\text{DONE}'(m', b - l)$ holds. Note that, according to Lemma 38, $R^i(m, \leq l) \cap R^i(m', \leq b - l) = \emptyset$. Then, $n \geq |R^i(m, \leq l) \cup R^i(m', \leq b - l)| \geq \lambda(l) + \lambda(b - l)$, as shown in Figure 4. Additionally, using Lemma 36, we obtain that every recipient in $R^i(m, \leq l - 1) \cup R^i(m', \leq b - l - 1)$ is corrupted. It follows that $t \geq |R^i(m, \leq l - 1) \cup R^i(m', \leq b - l - 1)| \geq \lambda(l - 1) + \lambda(b - l - 1)$, contradicting $P^b_n(t_v, t_c)$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{levels.png}
\caption{Levels if both $\text{DONE}'(m, l)$ and $\text{DONE}'(m', b - l)$ hold.}
\end{figure}

\section*{Achieving Consistency}

\begin{lemma}
If an honest recipient $R_i$ outputs $m$, every honest recipient eventually outputs $m$.
\end{lemma}

\begin{proof}
Since $R_i$ has output $m$, it $l$-received $m$ such that $\text{DONE}'(m, l)$ holds. If every honest recipient can $l$-receive $m$, $\text{DONE}'(m, l)$ is enough to guarantee their output, by Lemma 39.

Otherwise, $l$ must be the size of the smallest set $U \subseteq R$ containing an honest recipient such that $S$ $U$-sends $m$, and there is at least one honest recipient that does not belong to any $V \subseteq R$ such that $S$ $V$-sends $m$ and $|V| = l$. This recipient however eventually $(l + 1)$-receives $m$, by Lemma 7.

According to Lemma 37, \( |R^j(m, \leq l + 1) \setminus R^j(m, \leq l - 1)| \geq n - t \) eventually holds for any honest \( R_j \), and, since \( |R^j(m, l + 1)| \geq 1 \), \( \text{DONE}^j(m, l + 1) \) holds. It follows that every honest recipient eventually outputs \( m \).

\[ \text{Lemma 42.} \] If \( P^b_n(t_v, t_c) \) and an honest recipient \( R_i \) outputs \( m \), then no honest recipient outputs \( m' \neq m \).

\[ \text{Proof.} \] Assume that an honest recipient \( R_j \) outputs \( m' \neq m \).

Let \( U \subseteq R \) be the smallest set containing an honest recipient such that \( S \) \text{-}\( U \)-sends \( m \) and let \( l = |U| \). Since \( R_i \) completed the protocol, and according to Lemma 9, \( \text{DONE}^i(m, l) \) holds. It follows from Lemma 39 that \( \text{DONE}^j(m, l) \) eventually holds as well. According to Lemma 40, \( \text{DONE}^j(m, b - l) \) must be false. Then, \( R_j \) has completed the protocol by \((b - l - 1)\)-receiving \( m' \), contradicting Lemma 8.

\[ \text{Lemma 43.} \] If \( P^b_n(t_v, t_c) \) holds, then \( \Pi_{\text{nRBC}}^n(t_v, t_c, S, R) \) achieves consistency.

\[ \text{Proof.} \] Follows from Lemmas 41 and 42.

### 6.2.3 Corruption Threshold

We assemble the results proved in the sections 6.2.1 and 6.2.2.

\[ \text{Theorem 44.} \] If \( Q^b_n(t_v, t_c) \) holds, then \( \Pi_{\text{nRBC}}^n \) achieves \((t_v, t_c)\)-nonstop reliable broadcast.

\[ \text{Proof.} \] Follows from Lemmas 33 and 43.

In Appendix D, we show the technical lemmas that prove that if \( 2t_v + (b - 1)t_c < (b - 1)n \), then predicate \( Q^b_n(t_v, t_c) \) holds, leaving us with the next theorem.

\[ \text{Theorem 45.} \] If \( 2t_v + (b - 1)t_c < (b - 1)n \), then \( \Pi_{\text{nRBC}}^n(t_v, t_c) \) achieves \((t_v, t_c)\)-nonstop reliable broadcast in \( N_b \). Additionally, \( \Pi_{\text{nRBC}}^n(t) \) is a \( t \)-resilient nonstop reliable broadcast in \( N_b \), for any \( t < \left( \frac{b - 1}{b + 1} \right)n \).
References


Appendix

A Resilience proof for $\Pi_{RBC}^{n,3}$

We show that protocol $\Pi_{RBC}^{n,3}$ is a reliable broadcast protocol secure up to $t < \frac{n}{2}$ corruptions. Throughout the results of this section, we assume that $t < \frac{n}{2}$.

Lemma 46. If a recipient $R_i$ consistently receives $(\text{MSG}, m)$ from the sender, then no honest recipient sends $(\text{READY}, m')$ for $m' \neq m$.

Proof. Let $R_j$ denote the first honest recipient that sends $(\text{READY}, m')$. Note that $R_j$ did not send this message in Step 2, since it requires that $t + 1$ recipients sent such a message, while $R_j$ is the first honest receiver that sends $(\text{READY}, m')$. However, $R_j$ cannot send $(\text{READY}, m')$ in Step 1 either, since it receives at least one message $(\text{MSG}, m)$.

Lemma 47. If an honest recipient sends $(\text{READY}, m)$, then no honest recipient can send $(\text{READY}, m')$, for $m' \neq m$.

Proof. Let $R_i$ denote the first honest recipient that sends $(\text{READY}, m)$. This implies that $R_i$ consistently received the message $(\text{MSG}, v)$ from the sender. According to Lemma 46, no honest recipient can send $(\text{READY}, m')$, for $m' \neq m$.

Lemma 48. $\Pi_{RBC}^{n,3}$ achieves validity.

Proof. Assume that an honest recipient $R_i$ outputs $m$. It follows that $R_i$ sent $(\text{READY}, m)$, and hence, according to Lemma 47, no honest recipient sends $(\text{READY}, m')$. Hence, no honest recipient can output $m'$. Since $R_i$ outputs $m$, it received $(\text{READY}, m)$ via all 3-cast from other $n - t - 1$ recipients and it also sent $(\text{READY}, m)$ via all 3-cast. It follows that all the honest recipients eventually execute Step 2 (as they receive from the $n - t - 1$ and $R_i$) and send $(\text{READY}, m)$ via all 3-cast. Therefore, every honest recipient sends $(\text{READY}, m)$ via all 3-cast. Eventually, each honest recipient receives these messages and outputs $m$ and terminates.

Lemma 49. $\Pi_{RBC}^{n,3}$ achieves consistency.

Proof. Assume that an honest recipient $R_i$ outputs $m$. It follows that $R_i$ sent $(\text{READY}, m)$, and hence, according to Lemma 47, no honest recipient sends $(\text{READY}, m')$. Hence, no honest recipient can output $m'$. Since $R_i$ outputs $m$, it received $(\text{READY}, m)$ via all 3-cast from other $n - t - 1$ recipients and it also sent $(\text{READY}, m)$ via all 3-cast. It follows that all the honest recipients eventually execute Step 2 (as they receive from the $n - t - 1$ and $R_i$) and send $(\text{READY}, m)$ via all 3-cast. Therefore, every honest recipient sends $(\text{READY}, m)$ via all 3-cast. Eventually, each honest recipient receives these messages and outputs $m$ and terminates.

Theorem 50. $\Pi_{RBC}^{n,3}$ is an asynchronous reliable broadcast protocol in $N^3$ secure up to $t < \frac{n}{2}$ corruptions.

Proof. Follows from Lemmas 48 and 49.

B Technical Lemmas for LEVELS

We give the proof of Lemma 10, that computes $\lambda(l)$.
Lemma 10. Given $0 \leq t_0 \leq t_v < n$ and $l > 0$, $\lambda(l)$ can be computed as follows. 

If $t_v = t_c = l$: 

$$\lambda(l) = \begin{cases} \frac{l+1}{2}(n-l) & \text{if } l \text{ is odd,} \\ \frac{1}{2}(n-l) + 1 & \text{if } l \text{ is even.} \end{cases}$$

Otherwise, if $t_v > t_c$: 

$$\lambda(l) = \begin{cases} (n - t_v) + \frac{l+1}{2}(n-t_c) & \text{if } l \text{ is odd,} \\ \frac{1}{2}(n-t_c) & \text{if } l \text{ is even.} \end{cases}$$

Proof. For any $0 \leq t_c \leq t_v < n$, assuming the existence of the pair-wise disjoint sets $L_1, L_2, \ldots, L_l$ such that $\text{LEVELS}_{n, t_0, t_v}(L_1, L_2, \ldots, L_l)$ holds and $|\bigcup_{k=1}^l L_k| < \lambda(l)$ leads to a contradiction. Hence, we need to show that it is possible to construct $L_1, L_2, \ldots, L_l$ such that $|\bigcup_{k=1}^l L_k| = \lambda(l)$ and the predicate $\text{LEVELS}_{n, t_0, t_v}(L_1, L_2, \ldots, L_l)$ is satisfied.

If $t_v = t_c = l$, let $L_1, L_2, \ldots, L_l$ be pair-wise disjoint sets such that $|L_1| = n - t$, $|L_{2k}| = 1$ for $2 \leq 2k \leq l$, and $|L_{2k+1}| = n - t - 1$ for $3 \leq 2k+1 \leq l$. Since $n > t$, the sets are non-empty.

Otherwise, if $t_v > t_c$, let $L_1, L_2, \ldots, L_l$ be pair-wise disjoint sets such that $|L_{2k-1}| = n - t_v$ for $1 \leq 2k - 1 \leq l$ and $|L_{2k}| = t_v - t_c$ for $2 \leq 2k \leq l$. Since $n > t_v > t_c$, the sets are non-empty.

In both cases, it is easy to see that $\text{LEVELS}_{n, t_0, t_v}(L_1, L_2, \ldots, L_l)$ is satisfied, and, since the sets are pair-wise disjoint, $|\bigcup_{k=1}^l L_k| = \sum_{k=1}^l |L_k| = \lambda(l)$. □

Consider a sequence of sets $L_1, \ldots, L_l$ satisfying $\text{LEVELS}_{n, t_0, t_v}(L_1, L_2, \ldots, L_l)$. The following lemma allows us to bound the total number of parties in $\bigcup_{k=1}^l L_k$, even when a subsequence of sets $L_1, \ldots, L_{l'}$, $l' < l$ has non-minimal assignment of parties, i.e., $|\bigcup_{k=1}^{l'} L_k| > \lambda(l')$. The proof of the following lemma is enclosed in Appendix B.

Lemma 51. Let $L_1, L_2, \ldots, L_l$ such that $\text{LEVELS}_{n, t_0, t_v}(L_1, L_2, \ldots, L_l)$ is satisfied and let $x_1 = |L_1| - n - t$, $x_{2k} = |L_{2k}| - 1$ for $2 \leq 2k \leq l$, and $x_{2k+1} = |L_{2k+1}| - n - t - 1$ for $3 \leq 2k+1 \leq l$.

Then, for any $l' \leq l$, if $l$ and $l'$ have the same parity, then $|\bigcup_{k=1}^{l'} L_k| \geq \lambda(l) + \sum_{k=1}^{l'} x_k$, and otherwise $|\bigcup_{k=1}^{l'} L_k| \geq \lambda(l') + \sum_{k=1}^{l'} x_k - x_{l'}$.

Proof. Using the definition of the LEVELS predicate, we obtain that $x_1 \geq 0$ and $x_k + x_{k+1} \geq 0$ for any $k < l$. Additionally, the following inequalities hold.

\begin{align*}
(1) \quad |L_1| & \geq n - t + x_1 \\
(2) \quad |L_1| + |L_2| & \geq n - t + 1 + x_1 + x_2 \\
(3) \quad |L_2| + |L_3| & \geq n - t + x_2 + x_3 \\
\cdots \\
(l') \quad |L_{l'-1}| + |L_{l'}| & \geq n - t + x_{l'-1} + x_{l'} \\
(l' + 1) \quad |L_{l'}| + |L_{l'+1}| & \geq n - t + x_{l'} + x_{l'+1} \\
\cdots \\
(l) \quad |L_{l-1}| + |L_l| & \geq n - t + x_{l-1} + x_l
\end{align*}

We can obtain a lower bound for $|\bigcup_{k=1}^{l'} L_k|$ by adding the inequalities indexed with even numbers if $l$ is even, and by adding the inequalities indexed with odd numbers if $l$ is odd. In both cases, we obtain that $|\bigcup_{k=1}^l L_k| \geq \lambda(l) + (x_1 + x_2 + \ldots + x_{l'}) + (x_{l'+1} + \ldots + x_l)$.

If $l-l'$ is even, we obtain that $x_{l'+1} + \ldots + x_l \geq 0$ and therefore $|\bigcup_{k=1}^{l'} L_k| \geq \lambda(l) + \sum_{k=1}^{l'} x_k$.

If $l-l'$ is odd, we can write $x_{l'+1} + \ldots + x_l = x_{l'} + x_{l'+1} + \ldots + x_l - x_{l'} \geq -x_{l'}$ and therefore $|\bigcup_{k=1}^{l'} L_k| \geq \lambda(l) + \sum_{k=1}^{l'} x_k - x_{l'}$. □
C Technical Lemmas for $\Pi_{\text{RBC}}^{a,b}$

\[ \text{Lemma 19.} \] Assume that $t < \lambda(l-1) + \lambda(l'-1) - \lambda(l + l' - b + 1) + 2$ for any $l > 0, l' > 0$ such that $l + l' \geq b$. Then, if NOTFY$(m, l)$ holds for an honest recipient $R_i$, NOTFY$(m', l')$ is false for any honest recipient $R_j$, where $l + l' \geq b$ and $m \neq m'$.

**Proof.** We argue by contradiction. Assume that $R_i$ ($R_j$) is the first honest recipient for whom NOTFY$(m, l)$ (resp. NOTFY$(m', l')$) holds.

Let $L_1, \ldots, L_{l-1}$ be sets satisfying the conditions in NOTFY$(m, l)$ such that $|L_{k}^{+1}|$ is minimal. That is, such that (1) LEVELS$_{t,k}(L_1, \ldots, L_{l-1})$ holds, and for each $k \in [1, l-1]$:

(2) $L_k \subseteq \text{R}^t(m, l - k + 1)$ and (3) $L_k \cap \text{R}^t(m, l - k) \neq \emptyset$. Similarly, let $L'_1, \ldots, L'_{l'-1}$ be sets satisfying the conditions in NOTFY$(m', l')$ such that $|L'_{k}^{+1}|$ is minimal.

Since $R_i$ and $R_j$ are the first honest recipients to send (READY, m), respectively (READY, m'), each recipient in $\bigcup_{k=1}^{l-1} L_k$ or $\bigcup_{k=1}^{l'-1} L'_k$ is corrupted. Hence, $t \geq \left| \left( \bigcup_{k=1}^{l-1} L_k \right) \cup \left( \bigcup_{k=1}^{l'-1} L'_k \right) \right| = \left| \bigcup_{k=1}^{l-1} L_k \right| + \left| \bigcup_{k=1}^{l'-1} L'_k \right| - \left| \bigcup_{k=1}^{l'-1} L'_k \cap \bigcup_{k=1}^{l-1} L_k \right|.$

From Lemma 11, it follows that $L_k \cap L'_k = \emptyset$ if $(l - k + 1) + (l' - k' + 1) < b$ (i.e. $k + k' > l + l' - b + 2$). Hence, recipients in $L_k$ can intersect with $\bigcup_{k=1}^{l'-1} L'_k$ only if $k \leq l + l' - b + 1$. Hence, $\left( \bigcup_{k=1}^{l-1} L_k \right) \cap \left( \bigcup_{k=1}^{l'-1} L'_k \right) = \left( \bigcup_{k=1}^{l'-1} L'_k \right) \cap \left( \bigcup_{k=1}^{l'-1} L'_k \right)$. 

Figure 5 displays the levels of $R_i$ and $R_j$.

![Figure 5 Levels if $R_i$ sends (READY, m), while $R_j$ sends (READY, m').](image)

Condition (3) from the NOTIFY predicate implies that there is a recipient $R_1 \in L_{l'+b+1}^{t} \cap \text{R}^t(m, b - l' - 1)$. $R_i$ (b - l' - 1)-received m from $R_1$, and so by Lemma 8, $R_j$ cannot $t'$-receive $m'$ from $R_1$, implying that $R_j \notin \bigcup_{k=1}^{l'-1} L_k$. Similarly, there is a recipient $R_2 \in L_{l'-b+1}^{t'} \cap \text{R}^t(m', b - l - 1)$, such that $R_2 \notin \bigcup_{k=1}^{l'-1} L'_k$.

Hence, these two recipients are different and not in $\bigcup_{k=1}^{l'-1} L_k \cup \bigcup_{k=1}^{l'-1} L'_k$. Using Lemma 9, we obtain that both $|L_{k}^{t'-b+1}|$ and $|L'_k| \geq \lambda(l - b + 1)$.

We use Lemma 51 to obtain a lower bound on the total number of corrupted recipients involved. Hence, consider the sets $L_1, \ldots, L_l$ such that $|L_i| = n - t, |L_{2k+1}| = 1$ and $|L_{2k+1}| = n - t - 1$ and we compare the size of each level $L_k$ to the size of $L'_k$. It is easy to see that the sets satisfy the LEVELS predicate and, additionally, $|L_k \cup \cdots \cup L'_k| = \lambda(k)$ for any $k \leq l$.

Then, we use $x_k$ to denote $|L_k^+| - |L'_k|$. Hence, $x_1 = |L_1^-| - n - t, x_{2k+1} = |L_{2k+1}^-| - 1$, and $x_{2k+1} = |L_{2k+1}^-| - (n - t - 1)$. Since $L_1, \ldots, L_l$ satisfy the LEVELS predicate, $x_1 \geq 0$ and $x_k + x_{k+1} \geq 0$. Using a similar approach, we define $x_1' = |L_1^+| - n - t, x_{2k+1}' = |L_{2k+1}^-| - 1$, and $x_{2k+1}' = |L_{2k+1}^-| - (n - t - 1)$.

We show that each of the following cases contradicts the hypothesis, using Lemma 51. We mention that since the sets $L_1, \ldots, L_l$ and $L'_1, \ldots, L'_{l'}$ are minimal, it follows that if
if \( l + l' - b + 1 \) is odd, then \( x_{t+l' -b+1} \leq 0 \) and if \( l + l' - b + 1 \) is even, then \( x_{t+l' -b+1} \leq n - t - 2 \) and \( x_{l + l' - b + 1} \leq n - t - 2 \). Moreover, if \( l + l' - b + 1 \) is even, \( x \geq x_{t+l' -b+1} \). Otherwise, if \( x = \sum_{k=1}^{t+l'-b-1} x_k < 0 \), we obtain that \( x_1 = -(x_2 + x_3) - \ldots - (x_{t+l' -b-1} + x_{t+l' -1}) \leq 0 \), which contradicts the LEVELS predicate. A similar argument implies that \( x' \geq x_{t+l' -b+1} \).

\[ \text{b even} \]

- If \( l \) and \( l' \) are both even, then \( l + l' - b + 1 \) is odd. Since \( l - 1, l' - 1, \) and \( l + l' - b + 1 \) are odd, \( \sum_{k=1}^{t+l'-b+1} L_k \geq \lambda(l - 1) + x \) and \( \sum_{k=1}^{t+l'-b+1} L_k \geq \lambda(l' - 1) + x' \). Therefore, \( t \geq \lambda(l - 1) + x + \lambda(l' - 1) + x' - \lambda(l + l' - b - 1) - \min(x, x') + 2 \geq \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b -1) + 2. \)

\[ \text{b odd} \]

- If \( l \) and \( l' \) are both odd, then \( l + l' - b + 1 \) is odd, which implies that \( x_{t+l' -b+1} \leq 0 \) and \( x'_{t+l' -b+1} \leq 0 \). Then, \( \sum_{k=1}^{t+l'-b+1} L_k \geq \lambda(l - 1) + x - x_{t+l' -b+1} \geq \lambda(l - 1) + x', \) and \( \sum_{k=1}^{t+l'-b+1} L_k \geq \lambda(l' - 1) + x' \). It follows that \( t \geq \lambda(l - 1) + x + \lambda(l' - 1) + x' - \lambda(l + l' - b - 1) \geq \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b - 1) + 2. \)

- If \( l \) is even and \( l' \) is odd, then \( l + l' - b + 1 \) is even. Then, \( \sum_{k=1}^{t+l'-b+1} L_k \geq \lambda(l - 1) \) since the predicate LEVELS is satisfied, while \( \sum_{k=1}^{t+l'-b+1} L_k \geq \lambda(l' - 1) + x' \) since \( t < 1 \) and \( l + l' - b + 1 \) are both even. We obtain that \( t \geq \lambda(l - 1) + \lambda(l' - 1) + x' - \lambda(l + l' - b - 1) - \min(x, x') + 2 \geq \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b - 1) + 2. \)

**Lemma 52.** If \( b > 4 \) is even and \( t < \frac{b-4}{2} + \frac{b}{2} \), then \( t < \lambda(b - 2) \) and \( t < \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b + 1) + 2 \) for any \( l, l' > 0 \) such that \( l + l' \geq b \).

**Proof.** The result follows from Lemma 10

Firstly \( t < \lambda(b - 2) = \frac{b-4}{2} + \frac{b}{2} \) is equivalent to \( t < \frac{b-4}{2} + \frac{b}{2} \), which is implied by \( t < \frac{b-4}{2} + \frac{b}{2} \).

For the second inequality, we need to consider the following cases.
Lemma 54. If $l$ and $l'$ are even, then $l + l' - b + 1$ is odd and we obtain that $t < \frac{l}{2}(n - t) + \frac{l'}{2}(n - t) - \frac{t}{2}(n - 2) - \frac{l}{2}(n - 2) - \frac{t}{2}(n - 2) = \frac{b - 2}{2}(n - t) + 2$, which is equivalent to $t < \frac{b - 2}{2}n + \frac{4}{B}$.

If $l$ is odd and $l'$ is even, then $l + l' - b + 1$ is even and we obtain that $t < \frac{l}{2}(n - t) + \frac{l'}{2}(n - t) - \frac{t}{2}(n - 2) - \frac{l}{2}(n - 2) + 1 = \frac{b - 2}{2}(n - t) + 2$, which is equivalent to $t < \frac{b - 2}{2}n + \frac{4}{B}$.

If $l$ and $l'$ are odd, then $l + l' - b + 1$ is odd and we obtain that $t < \frac{l}{2}(n - t) + \frac{l'}{2}(n - t) + 1 - \frac{l}{2}(n - 2) + 1 = \frac{b - 2}{2}(n - t) + 4$, which is equivalent to $t < \frac{b - 2}{2}n + \frac{4}{B}$. Hence, if $t < \frac{b - 1}{2}n + \frac{6}{B}$, then $t < \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b + 1) + 2$ for any $l, l' > 0$ such that $l + l' \geq b$.

Lemma 53. If $b > 4$ is odd and $t < \frac{b - 1}{2}n + \frac{6}{B}$, then $t < \lambda(b - 2)$ and $t < \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b + 1) + 2$ for any $l, l' > 0$ such that $l + l' \geq b$.

Proof. The result follows from Lemma 10.

Firstly, $t < \lambda(b - 2) = \frac{b - 1}{2}(n - t)$ is equivalent to $t < \frac{b - 1}{2}n$, which is implied by $t < \frac{b - 3}{2}n + \frac{6}{B}$.

For the second inequality, we need to consider the following cases.

If $l$ and $l'$ are even, then $l + l' - b + 1$ is even and we obtain that $t < \frac{l}{2}(n - t) + \frac{l'}{2}(n - t) - \frac{t}{2}(n - 2) - \frac{l}{2}(n - 2) - \frac{t}{2}(n - 2) = \frac{b - 3}{2}(n - t) + 2$, which is equivalent to $t < \frac{b - 2}{2}n + \frac{4}{B}$. Hence, if $t < \frac{b - 1}{2}n + \frac{6}{B}$, then $t < \lambda(l - 1) + \lambda(l' - 1) - \lambda(l + l' - b + 1) + 2$ for any $l, l' > 0$ such that $l + l' \geq b$.

D Technical Lemmas for $Q_n^b(t_v, t_c)$

In this section, we provide an expression implying $Q_n^b(t_v, t_c)$. We will have to consider two separate cases: when $b$ is odd and when $b$ is even.

Odd $b$

If $b$ is odd, we show that if $t_v$ and $t_c$ satisfy $2t_v + (b - 1)t_c < (b - 1)n$, then $\Pi_n^{ab}$ achieves $(t_v, t_c)$-asynchronous broadcast.

Firstly, we compute $P_n^b(t_v, t_c)$.

$$P_n^b(t_v, t_c) = \begin{cases} (b + 1)t < (b - 1)n + 2 & \text{if } t_v = t_c = t, \\ 2t_v + (b - 1)t_c < (b - 1)n & \text{if } t_v > t_c. \end{cases}$$

Lemma 54. If $b$ is odd and $2t_v + (b - 1)t_c < (b - 1)n$, then $Q_n^b(t_v, t_c)$ holds.

Proof. We have to show that $P_n^b(\min(t_v, n - k - 1), t_c - k)$ is satisfied for every $0 \leq k < n - b$.

For $t_v > t_c$ or $k > 0, 2 \cdot \min(t_v, n - k - 1) + (b - 1)(t_c - k) \leq 2t_v + (b - 1)t_c - k(b - 1) < (b - 1)n - k(b - 1) = (b - 1)(n - k)$. Hence, $P_n^b(\min(t_v, n - k - 1), t_c - k)$ is satisfied. If $t_v = t_c = t$, $P_n^b(t, t)$ is satisfied since $(b + 1)t < (b - 1)n < (b - 1)n + 2$.

Theorem 55. If $b$ is odd and $2t_v + (b - 1)t_c < (b - 1)n$, then $\Pi_n^{ab}(t_v, t_c)$ achieves $(t_v, t_c)$-nonstop reliable broadcast.
From Lemma 54, it follows that $Q_n^b(t_v, t_c)$ holds. Using Theorem 44, we obtain that $\Pi_{n,b}^{a,b}(t_v, t_c)$ achieves $(t_v, t_c)$-nonstop reliable broadcast.

Even $b$

For even $b$, we show that if $t_v$ and $t_c$ satisfy $[2t_v + (b - 1)t_c < (b - 1)n] \lor [bt_c < (b - 2)n]$, then $\Pi_{n,b}^{a,b}$ achieves $(t_v, t_c)$-asynchronous broadcast.

If $b$ is even, we can write $P_n^b(t_v, t_c)$ as follows.

$$P_n^b(t_v, t_c) = \begin{cases} (b + 2)t < bn & \text{if } t_v = t_c = t, \\ [bt_c < (b - 2)n] \lor [4t_v + (b - 2)t_c < bn] & \text{if } t_v > t_c. \end{cases}$$

Lemma 56. If $bt_c < (b - 2)n$, then $b(t_c - k) < (b - 2)(n - k)$ for every $0 \leq k < n - b$.

Proof. If $bt_c < (b - 2)n - bk < (b - 2)n - (b - 2)k = (b - 2)(n - k)$, we obtain that $bt_c < (b - 2)n$.

Lemma 57. If $2t_v + (b - 1)t_c < (b - 1)n$, then $b(t_c - k) < (b - 2)(n - k)$ or $4 \cdot \min(t_v, n - k) + 2(b - 2)k < n$.

Proof. We can write $2t_v + (b - 1)t_c < (b - 1)n$ as $4t_v + (b - 2)t_c < bn - 2n + 2 \cdot \frac{b}{b - 1}t_v$. If $\frac{b}{b - 1}t_v \leq n$, then $4t_v + (b - 2)t_c < (b - 2)n$. It follows that $4 \cdot \min(t_v, n - k) + 2(b - 2)k < bn$. If $\frac{b}{b - 1}t_v > n$, then $2 \cdot \frac{b}{b - 1}n + (b - 1)t_c < 2t_v + (b - 1)t_c < (b - 1)n$, which implies that $bt_c < (b - 2)n$. From Lemma 56, it follows that $b(t_c - k) < (b - 2)(n - k)$.

Lemma 58. If $b$ is even and $2t_v + (b - 1)t_c < (b - 1)n$ or $bt_c < (b - 2)n$, then $Q_n^b(t_v, t_c)$ holds.

Proof. We have to show that $P_{n-k}^b(\min(t_v, n-k-1), t_c-k)$ is satisfied for every $0 \leq k < n-b$. If $t_v > t_c$ or $k > 1$, the result follows from Lemmas 56 and 57. For $t_v = t_c = t$, we obtain that $(b + 1)t < (b - 1)n$ or $bt < (b - 2)n$, both implying $P_n^b(t, t)$.

Theorem 59. If $b$ is even and $2t_v + (b - 1)t_c < (b - 1)n$ or $bt_c < (b - 2)n$, then $\Pi_{n,b}^{a,b}(t_v, t_c)$ achieves $(t_v, t_c)$-nonstop reliable broadcast.

Proof. From Lemma 58, it follows that $Q_n^b(t_v, t_c)$ holds. Using Theorem 44, we obtain that $\Pi_{n,b}^{a,b}(t_v, t_c)$ achieves $(t_v, t_c)$-nonstop reliable broadcast.

Conclusion

To close this chapter, we add the following result.

Theorem 60. If $2t_v + (b - 1)t_c < (b - 1)n$, then $\Pi_{n,b}^{a,b}(t_v, t_c)$ achieves $(t_v, t_c)$-nonstop reliable broadcast in $N_b$.

Additionally, $\Pi_{n,b}^{a,b}(t, t)$ achieves nonstop reliable broadcast secure against $t < \frac{b - 1}{b + 1}n$ corruptions in $N_b$.

Proof. Follows from Theorems 55 and 59.

E Impossibility

We consider the network $N_b$, with sender $S$ and $n$ recipients. We closely follow the proof of Considine et al. [6] to show that nonstop reliable broadcast is unachievable if the number of corrupted recipients $t \geq \frac{b - 1}{b + 1}n$. Since nonstop reliable broadcast is impossible, it follows that achieving reliable broadcast is also impossible. We start with the concrete case $n = b + 1$ and $t = b - 1$. Then, we generalize the result.
E.1 Impossibility for $n = b + 1$ and $t = b - 1$

Let $\Pi$ denote an arbitrary nonstop reliable broadcast protocol for $n = b + 1$ recipients $\mathcal{R} = \{R_0, R_1, \ldots, R_{n-1}\}$. We show that there is an admissible adversary that can make $\Pi$ fail by corrupting $t = b - 1$ recipients and, in some cases, the sender. We denote $\pi_S$ the sender’s program and $\pi_i$ the recipient $R_i$’s program according to protocol $\Pi$.

We execute these programs in two different systems: $\Sigma$ and $\Sigma'$. $\Sigma$ denotes the original system with $n$ recipients and up to $t$ corruptions, where $\Pi$ is secure. The system $\Sigma'$ extends $\Sigma$ by making an identical copy of each party and adjusting the connections between them. The idea is that for certain sets of honest parties, their joint views are indistinguishable with respect to systems $\Sigma$ and $\Sigma'$. $\Sigma'$ simulates an admissible adversary in $\Sigma$ with respect to specific sets of parties, implying that the validity and consistency conditions hold even in system $\Sigma'$.

$\Sigma'$ denotes the identical copy of the sender in $\Sigma'$, running the same program $\pi_S$, however with different input. We use $m$ and $m'$ to denote the input of $S$, respectively the input of $\Sigma'$. $R_{i+n}$ denotes the identical copy of the recipient $R_i$, running the same program $\pi_i$. $\mathcal{R}' = \{R_0, R_1, \ldots, R_{2n-1}\}$ denotes the set of $2n = 2b + 2$ recipients in $\Sigma'$.

E.1.1 Defining the Communication Ports

Firstly, we define the type of a recipient $R_i \in \mathcal{R}'$ as $i \mod n$.

We assume that each party has ports for the communication channels it shares with other parties.

**Bilateral Ports** The bilateral port of type $i$ of a party $P \in \{S\} \cup \mathcal{R}$ denotes the port $P$ uses to communicate via pair-wise asynchronous channels with $R_k$ in $\Sigma$. Similarly, the bilateral port of type $S$ of $R_i$ denotes the port $R_i$ uses to communicate via pair-wise asynchronous channels with $S$ in $\Sigma$.

**b-cast Ports** $R_i$'s b-cast port of type $k$ denotes the port $R_i$ uses to communicate via b-cast channels with $\mathcal{R} \setminus \{R_i, R_k\}$ in $\Sigma$. $R_i$'s b-cast port of type $(S, j, k)$ ($j < k$) denotes the port $R_i$ uses to communicate via b-cast channels with $S \cup \mathcal{R} \setminus \{R_i, R_j, R_k\}$ in $\Sigma$.

The b-cast port of type $(j, k)$ ($j < k$) of $S$ denotes the port $S$ uses for its communication via b-cast channels with $\mathcal{R} \setminus \{R_j, R_k\}$ in $\Sigma$.

E.1.2 Connecting the Recipients and their Copies in $\Sigma'$

Figure 6 shows how the recipients are connected in $\Sigma$ and in $\Sigma'$, for $b = 3$.

$\Sigma'$ places the recipients in a circle and connects their channels in a cyclically identical manner. It is enough to describe the reconnection of the channels of $R_0$; for $1 \leq k \leq b - 1$, each bilateral write port of type $k$ of $R_0$ is connected to the bilateral read port of type 0 of $R_k$, while $R_0$’s bilateral write port of type $b$ is connected to the bilateral read port of type 0 of $R_{2n-1}$. $R_0$’s b-cast port of type $1 \leq k \leq b$ is connected to the b-cast port of type $k$ of $R_1, R_2, \ldots, R_{k-1}$ and $R_{k+1+n}, R_{k+2+n}, \ldots, R_{2n-1}$.

E.1.3 Connecting the Sender and its Copy in $\Sigma'$

$\Sigma'$ connects $S$ and $S'$ to the recipients by following the connections of $R_0$ and of $R_n$. Figure 7 shows how the copies of the sender are connected to the recipients in $\Sigma'$ for $b = 3$.

The pair-wise channels of $S$ and $S'$ follow the pair-wise channels of $R_0$, respectively of $R_n$. Hence, each bilateral write (read) port of type $0 \leq k \leq b - 1$ of $S$ is connected to the
bilateral read (write) port of type $S$ of $R_k$, and $S$’s bilateral write (read) port of type $b$ is connected to the bilateral read (write) port of type $S$ of $R_{2n-1}$. Then, each bilateral write (read) port of type $0 \leq k \leq b - 1$ of $S'$ is connected to the bilateral read (write) port of type $S$ of $R_{n+k}$, and $S$’s bilateral write (read) port of type $b$ is connected to the bilateral read (write) port of type $S$ of $R_{n-1}$.

Similarly, the $b$-cast ports of $S$ follow the connections of $R_0$. Each $b$-cast port of type $(0, k)$ of $S$ is hence connected to the same $b - 1$ parties as $R_0$’s $b$-cast port of type $k$. Then, for $0 < j < k \leq b$, the $b$-cast port of type $(j, k)$ of $S$ is connected to the $b$-cast ports of type $(S, j, k)$ of $R_0$ and of the set of parties $R_0$’s $b$-cast port of type $k$ is connected to, excluding the recipients of type $j$.

Then, the $b$-cast ports of $S'$ follow the connections of $R_n$. Each $b$-cast port of type $(0, k)$ of $S'$ is hence connected to the same $b - 1$ parties as $R_n$’s $b$-cast port of type $k$. Then, for $0 < j < k \leq b$, the $b$-cast port of type $(j, k)$ of $S'$ is connected to the $b$-cast ports of type (null)
(S, j, k) of Rn and of the set of parties Rn’s b-cast port of type k is connected to, excluding the recipients of type j.

E.1.4 Properties of the Connections in Σ′

Exclusive Assignment of Bilateral Ports The cyclical symmetry of the structure guarantees that the bilateral ports among the recipients are exclusively connected. More precisely, R′i’s bilateral read (write) port of type j mod n is exclusively connected to Rj′s bilateral write (read) port of type i mod n.

Similarly, the bilateral write (read) port of type S of each recipient Ri is either exclusively connected to the bilateral read (write) port of type i of S or of S′.

Exclusive Assignment of b-cast Ports R′i’s b-cast port of type j is exclusively connected to the b-cast ports of type j of b − 1 recipients of distinct types k / {i mod n, j}. The assignment guarantees that R′i’s b-cast port of type k is connected to Rj′s b-cast port of type j if and only if Rj′s b-cast port of type k is connected to R′i’s b-cast port of type k.

Similarly, R′i’s b-cast port of type (S, j, k) is connected to the b-cast port of type (j, k) of S (S′) if and only if the b-cast port of type (j, k) of S (S′) is connected to the R′i’s b-cast port of type (S, j, k).

Mutual assignment of bilateral ports For every pair Ri, R′((i+1) mod 2n), Ri’s bilateral read (write) port of type (i + 1) mod n is connected to R′((i+1) mod 2n)’s bilateral write (read) port of type i.

Mutual Assignment of b-cast Ports For every pair R′i, R′((i+1) mod 2n), and for every j / {i, (i + 1) mod n}, their b-cast ports of type j are mutually connected. Similarly, for every j, k / {i, (i + 1) mod n} such that j < k, their b-cast ports of type (S, j, k) are mutually connected.

E.1.5 Identical Joint View of Adjacent Recipients

We show that the joint view in Σ′ of any two recipients R′i, R′((i+1) mod 2n) can be simulated by an admissible adversary in the original system Σ.

If in Σ′ Ri and R′((i+1) mod 2n) are not connected to the same copy of the sender, the adversary A corrupts the sender to simulate the virtual S and S′ towards Ri mod n and R′((i+1) mod 2n). Then, A corrupts the b − 1 recipients in R \ {R′i mod n, R′((i+1) mod 2n)} to simulate the virtual recipients in R′ \ {R′i, R′((i+1) mod 2n)}, with the inputs that they would receive from S and S′ in Σ′.

Section E.1.4 guarantees that R′i and R′((i+1) mod 2n) are consistently interconnected in Σ′. Any message received (sent) by a recipient R′i via its bilateral port of type (i + 1) mod n is sent (eventually received) by R′((i+1) mod 2n). The same holds for the b-cast ports: any message sent through a b-cast channel not excluding recipients of type i mod n and (i + 1) mod n is either eventually received by both R′i and R′((i+1) mod 2n), or by none of them. Hence, A can ensure that the communication of R′i mod n and R′((i+1) mod 2n) in Σ is identical to the interactions that R′i and R′((i+1) mod 2n) would have in Σ′, by forwarding the messages from and to the virtual recipients accordingly to the connections in Σ′.

Therefore, this adversary strategy guarantees that the joint view of R′i and R′((i+1) mod 2n) in Σ′ is identical to the joint view of R′i mod n and R′((i+1) mod n) in Σ.
E.1.6 Identical Joint View of the Sender and Special Recipients

Note that in $\Sigma'$, the bilateral ports of type $S$ and the $b$-cast ports of type $(S, i, j)$ of $R_{2n-1}$, $R_0$ and $R_1$ are only connected to $S$. Similarly, the bilateral ports of type $S$, and the $b$-cast ports of type $(S, i, j)$ of $R_{n-1}$, $R_n$ and $R_{n+1}$ are only connected to $S'$.

Hence, to simulate $\Sigma'$ towards $R_0$, $R_1$, $S$, or $R_0$, $R_{n-1}$, $S$ in $\Sigma$, the adversary corrupts the remaining $b-1$ recipients to simulate the system $\Sigma'$ as described in Section E.1.5. Figure 8 displays the parties that $A$ has to corrupt to achieve the identical joint view of $R_0$ and $R_1$ in $\Sigma$ and $\Sigma'$, for $b = 3$.

![Figure 8](image)

**Figure 8** Parties corrupted to achieve the identical joint view of $S$, $R_0$ and $R_1$ in $\Sigma$ and $\Sigma'$, for $b = 3$. The arrows represent the pair-wise channels, while the triangles represent the 3-cast channels, and the channels of the senders are omitted. On the left side, the red area encloses the parties that $A$ corrupts to simulate the system $\Sigma'$ towards $S$, $R_0$ and $R_1$. On the right side, the red area includes the parties that $A$ simulates using the corrupted recipients in $\Sigma$.

In $\Sigma'$, any message $S$ receives through its bilateral port of type 0 (1) is guaranteed to be sent by $R_0$ ($R_1$). Additionally, any message $S$ received from a recipient of type $k \in \{0, 1\}$ through its $b$-cast port of type $(i, j)$ such that $i \neq k$ or $j \neq k$, is guaranteed to be sent by $R_k$. Therefore, this adversary strategy guarantees that the joint view of $S$, $R_0$ and $R_1$ in $\Sigma'$ is identical to the joint view of $S$, $R_0$ and $R_1$ in $\Sigma$.

The same holds for $S'$: any message $S'$ receives through its bilateral port of type 0 ($n-1$) is guaranteed to be sent by $R_n$ ($R_{n-1}$). Additionally, any message $S'$ received from a recipient of type $k \in \{0, n-1\}$ through its $b$-cast port of type $(i, j)$ such that $i \neq k$ or $j \neq k$, is guaranteed to be sent by $R_n$ if $k = n-1$, respectively $R_n$ if $k = 0$. Therefore, this adversary strategy guarantees that the joint view of $S'$, $R_{n-1}$ and $R_n$ in $\Sigma'$ is identical to the joint view of $S$, $R_{n-1}$ and $R_0$ in $\Sigma$.

E.1.7 Contradiction

Let $y_i \in \{0, 1\}$ denote the value that $R_i \in R'$ eventually outputs in $\Sigma'$, if any. If $R_i$ never outputs a value, we assume that $y_i = \bot$. 

CVIT 2016
Lemma 61. Assuming that $\Pi$ is secure against $b-1$ corrupted recipients, $y_0 = y_1 = m$, and $y_{n-1} = y_n = m'$ in $\Sigma'$.

Proof. We only need to prove the result for $R_0$ and $R_1$, since the same argument applies for $R_{1-n}$ and $R_{n+1}$, where we obtain that $y_{n-1} = y_n = m'$.

Section E.1.6 shows that the joint view of $S, R_0$ and $R_1$ in $\Sigma$ is identical to their joint view in $\Sigma'$. Since $\Pi$ achieves validity, $R_0$ and $R_1$ output $m$ in $\Sigma$, including when the adversary corrupts the other $b-1$ recipients to simulate the system $\Sigma'$. Hence, as $R_0$ and $R_1$ cannot distinguish between $\Sigma$ and $\Sigma'$, $y_0 = y_1 = m$.

Lemma 62. Achieving nonstop reliable broadcast for $n = b + 1$ recipients $\mathcal{R} = \{R_0, R_1, \ldots, R_{n-1}\}$ in $\mathcal{N}_b$ is impossible if, for any pair of recipients $\{R_i, R_{i+1} \mod n\}$, the adversary can corrupt the sender and the recipients in $\mathcal{R} \setminus \{R_i, R_{i+1} \mod n\}$.

Proof. Since the joint view of any adjacent recipients in $\Sigma'$ can be simulated by an admissible adversary in $\Sigma$, the consistency property of broadcast requires that $y_i = y_{i+1}$ for any $0 \leq i < n-1$ both in $\Sigma$ and in $\Sigma'$, assuming that $R_i$ and $R_{i+1}$ are honest in $\Sigma$.

According to Lemma 61, $y_0 = y_1 = m$, while $y_{n-1} = m' \neq m$. It follows that $\exists i \in [n-2]$ such that $y_i \neq y_{i+1}$ in $\Sigma'$. The adversary can corrupt the $b-1$ parties in $\mathcal{R} \setminus \{R_i, R_{i+1}\}$ to simulate the system $\Sigma'$, contradicting that consistency is achieved.

E.2 Extending the Result

We now generalize the result obtained in the previous section.

Lemma 63. Assume $Q$ is a finite set and $n = |Q|$. Then, we can partition $Q$ into $b+1$ sets $Q_0, \ldots, Q_b$ such that $|Q_i \cup Q_{i+1} \mod (b+1)| \geq n-t$ for every $0 \leq i < b$.

Proof. $Q$ can be partitioned arbitrarily in $b+1$ sets $Q_i$ of at least $\left\lfloor \frac{n}{b+1} \right\rfloor$ elements. If $n \mod (b+1) < \frac{b+1}{2}$, $|Q_i \cup Q_{i+1}| \geq 2 \left\lfloor \frac{n}{b+1} \right\rfloor = \left\lfloor \frac{2n}{b+1} \right\rfloor \geq n-t.$

Otherwise, when $n \mod (b+1) \geq \frac{b+1}{2}$, we can enforce that if $|Q_i| = \left\lceil \frac{n}{b+1} \right\rceil$, then $|Q_{i+1} \mod (b+1)| \geq \left\lceil \frac{n}{b+1} \right\rceil$. It follows that $|Q_i \cup Q_{i+1}| \geq \left\lfloor \frac{n}{b+1} \right\rfloor + \left\lceil \frac{n}{b+1} \right\rceil \geq \left\lfloor \frac{2n}{b+1} \right\rfloor \geq n-t.$

Theorem 64. Nonstop reliable broadcast for $n > b$ recipients in $\mathcal{N}_b$ cannot be achieved if $t \geq \frac{b+1}{2}n$.

Proof. Assume that a protocol $\Pi$ achieves nonstop reliable broadcast for $n$ recipients $Q = \{Q_0, \ldots, Q_{n-1}\}$. We show that $\Pi$ serves as a basis for constructing a $(b-1)$-resilient nonstop reliable broadcast protocol $\Pi'$ for $b+1$ recipients $\mathcal{R} = \{R_0, \ldots, R_{b-1}\}$.

According to Lemma 63, we can partition $Q$ into $b+1$ sets $Q_0, \ldots, Q_b$ such that $|Q_i \cup Q_{i+1} \mod (b+1)| \geq n-t$ for every $0 \leq i < b$. In $\Pi'$, each recipient $R_i$ simulates the recipients in $Q_i$, in an instance of $\Pi$. Similarly, the sender of $\Pi'$ simulates the sender of $\Pi$.

Each party in $\Pi'$ simulates the message system for $\Pi$ as follows: when a simulated recipient $Q \in Q_i$ (or the simulated sender) sends a message $m$ to $Q_0 \in Q_0, Q_1 \in Q_1, \ldots, Q_{b-2} \in Q_{b-2}$, then $R_i$ (respectively the sender) will forward $m$ through a $b$-cast channel having $R_0, \ldots, R_{b-2}$ as recipients, together with the identities of $Q$ (respectively the sender) and of the receivers $Q_0, \ldots, Q_{b-2}$. Once a recipient $R_j$ receives such a message, it forwards $m$ as expected in $\Pi$ to any simulated recipient $Q' \in Q_j \cap \{Q_0, \ldots, Q_{b-2}\}$.
According to Lemma 62, \( \exists R, R_{(i+1) \mod (b+1)} \in \mathcal{R} \) such that \( \Pi' \) cannot achieve nonstop reliable broadcast if the adversary corrupts the parties in \( \mathcal{R} \setminus \{R, R_{(i+1) \mod (b+1)}\} \). Since \( |Q_i \cup Q_{(i+1) \mod (b+1)}| \geq n - t \), we contradict that \( \Pi \) achieves nonstop reliable broadcast secure against \( t \) corrupted recipients.

\[\text{Corollary 65.} \quad \text{Reliable broadcast for } n > b \text{ recipients in } \mathcal{N}_b \text{ cannot be achieved for } t \geq \frac{b-1}{b+2}n. \]

\textbf{Proof.} Follows from Theorem 64, since any protocol that achieves reliable broadcast also achieves nonstop reliable broadcast. \[\blacksquare\]