From Partial to Global Asynchronous Reliable Broadcast

Diana Ghinea
Department of Computer Science, ETH Zurich, Switzerland
ghinead@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 Π_{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 Π_{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 Π_{RBC} is an asymptotically optimal reliable broadcast protocol, and Π_{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.DISC.2020.23

Related Version A full version of the paper is available at `https://eprint.iacr.org/2020/963`.

1 Introduction

Broadcast protocols constitute a fundamental building block in distributed computing. They allow a 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] has 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 network $\mathcal{N}_b$ 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 $\mathcal{N}_b$, we show:

- A reliable broadcast protocol $\Pi_{\texttt{RBC}}$ 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 $\Pi_{\texttt{nRBC}}$, 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. Following the impossibility of [6], we show in the full version of the paper that, in the network $\mathcal{N}_b$, there is no protocol for (nonstop) reliable broadcast secure up to $t \geq \frac{b-1}{b+1}n$ corruptions, implying that $\Pi_{\texttt{RBC}}$ is an asymptotically optimal reliable broadcast protocol when $n \to \infty$ and $b = O(n)$, and that $\Pi_{\texttt{nRBC}}$ 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

Previous results operate in the synchronous model, where parties proceed in *rounds*, 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, showing 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 the ones 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 is weaker than the traditional Byzantine broadcast primitive (it does not guarantee termination in the dishonest sender case), we also assume a weaker primitive, 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 ℓ_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 needs 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 setting, 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 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 $\mathsf{RBC}(S, \{R_1, \ldots, R_{b-1}\})$ with a sender S and $b-1$ additional recipients $\{R_1, \ldots, R_{b-1}\}$ as b-cast channel. We denote by $\mathcal{N}_b$ the generalized communication model where each party P, sender or recipient, in addition has access to all channels $\mathsf{RBC}(P, \{R_{i_i}, \ldots, R_{i_{b-1}}\})$, where $R_{i_i}, \ldots, R_{i_{b-1}}$ 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]. 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 $\mathcal{R} = \{R_1, \ldots, R_n\}$.

▶ **Definition 1.** *A protocol π 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 output m.*

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 π 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 $\mathcal{N}_3$

In this section, we consider the model $\mathcal{N}_3$, 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 $\Pi^{n,3}_{\mathsf{RBC}}$ in the communication network $\mathcal{N}_3$ secure up to $t < n/2$ corruptions inspired by Bracha's reliable broadcast protocol [3]. In the full version of the paper, we show that the construction is optimal with respect to the corruption threshold.

$\Pi^{n,3}_{\text{RBC}}$ 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 $(\texttt{MSG}, m)$ from S (receives consistently $(\texttt{MSG}, m)$ via all (in total $n-1$) 3-cast channels from S), mega-sends a message $(\texttt{READY}, m)$ notifying all recipients that it is ready to output m. Any recipient R_i that receives consistent notification messages $(\texttt{READY}, m)$ from $t+1$ different recipients mega-sends $(\texttt{READY}, m)$. Finally, any recipient R_i that mega-sent $(\texttt{READY}, m)$ and mega-received consistent notification messages $(\texttt{READY}, m)$ 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 $(\texttt{READY}, m)$ from R_j, then any honest recipient R_k receives $(\texttt{READY}, m)$ 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 $(\texttt{READY}, m)$ and mega-received $(\texttt{READY}, m)$ from $n-t-1$ different recipients, then any honest recipient eventually receives $(\texttt{READY}, m)$ from $n-t \geq t+1$ different recipients. It then follows that all honest parties mega-send $(\texttt{READY}, m)$, so all honest parties mega-receive $(\texttt{READY}, m)$ from at least $n-t-1$ parties and terminate. The protocol is described below, and a formal analysis of it can be found in the full version of the paper.

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

Code for the sender S

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

Code for recipient R_i

1: Upon receiving $(\texttt{MSG}, m)$ via all 3-cast $\mathsf{RBC}(S, \{R_i, R_j\})$, $R_j \in \mathcal{R}$, send $(\texttt{READY}, m)$ to every pair of recipients via 3-cast.
2: Upon receiving $(\texttt{READY}, m)$ from $t+1$ different recipients, i.e. from $\mathsf{RBC}(R_j, \{R_i, \cdot\})$, $R_j \in T$, $|T| \geq t+1$, if no READY message was sent, send $(\texttt{READY}, m)$ to every pair of recipients via 3-cast.
3: Upon receiving $(\texttt{READY}, m)$ via all 3-cast $\mathsf{RBC}(R_k, \{R_i, R_j\})$, $R_j \in \mathcal{R}$, $R_k \in T$, $|T| \geq n-t-1$, if $(\texttt{READY}, m)$ was sent, output m and terminate.

4 Notation for Protocols in $\mathcal{N}_b$

We consider the model $\mathcal{N}_b$ 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 $\mathcal{B}_P(U) := \{\mathsf{RBC}(P, V) : U \subseteq V \subseteq \mathcal{R} \wedge |V| = b-1\}$ 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 $|U| \geq b$, $\mathcal{B}_P(U) = \varnothing$, and any recipient b-receives any message.

We add some properties about l-receiving values among parties. Their proofs are included in the full version of the paper.

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

▶ **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 $|U \cup V| \geq b$.*

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 $\left|L_1\right| \geq n-t$ and $\forall\ 1 \leq k \leq l-1 : \left|L_k\right| + \left|L_{k+1}\right| \geq n-t$. For technical reasons, it will be useful to consider different sizes for each of the two conditions above (see Figure 1).

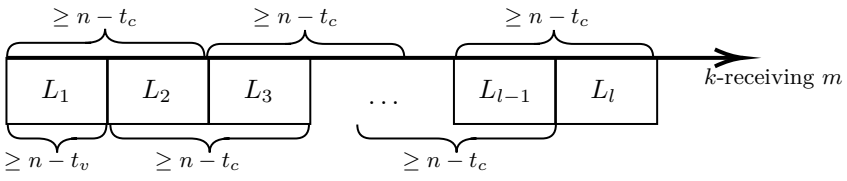

Figure 1 A visual representation of the LEVELS predicate.

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

$$\begin{aligned}\texttt{LEVELS}_{n,t_v,t_c}(L_1,\ldots,L_l) = {} & (\forall\ 1 \leq k < k' \leq l : L_k \cap L_{k'} = \varnothing) \wedge \\ & \left(\left|L_1\right| \geq n-t_v\right) \wedge \left(\forall\ 2 \leq k \leq l : \left|L_k\right| \geq 1\right) \wedge \\ & \left(\forall\ 1 \leq k \leq l-1 : \left|L_k\right| + \left|L_{k+1}\right| \geq n-t_c\right)\end{aligned}$$

For $l = 0$, the predicate is `true` by default.

We add a few properties. Their proofs are enclosed in the full version of the paper.

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

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

▶ **Lemma 10.** *Given* $0 \leq t_c \leq t_v < n$ *and* $l > 0$, $\lambda(l)$ *can be computed as follows.*

If $t_v = t_c = t$*:*

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

Otherwise, if $t_v > t_c$*:*

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

5 Asymptotically Optimal Reliable Broadcast Protocol

We present a protocol for any finite input space achieving reliable broadcast in $\mathcal{N}_b$ 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 3. 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 $(\texttt{READY}, m)$ via all b-cast channels to notify the other recipients. It outputs m when in addition it 1-receives $(\texttt{READY}, m)$ 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 $(\texttt{READY}, m)$ 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 $(\texttt{READY}, m)$ from the $n - t - 1$ corrupted recipients and 1-receives $(\texttt{READY}, m)$ from R_i. Once R_j receives these messages, it sends $(\texttt{READY}, m)$ 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 $(\texttt{READY}, m)$ 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 $(\texttt{READY}, m)$. Additionally, we set the threshold so that it ensures that honest recipients cannot send `READY` for different messages. Moreover, if all honest recipients send $(\texttt{READY}, m)$, all honest recipients eventually output m.

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

$$\texttt{NOTIFY}^i(m,1) = \texttt{true}$$

$$\begin{aligned}\texttt{NOTIFY}^i(m,l) = \exists\ & L_1^i \subseteq \mathbf{R}^i(m,l), ..., L_k^i \subseteq \mathbf{R}^i(m,l-k+1), ..., L_{l-1}^i \subseteq \mathbf{R}^i(m,2) \text{ s.t.} \\ & \big(\forall\ 1 \leq k \leq l-1 : L_k^i \cap \mathbf{R}^i(m,l-k) \neq \varnothing\big) \wedge \\ & \texttt{LEVELS}_{n,t,t}(L_1^i, ..., L_{l-1}^i) \text{ holds.}\end{aligned}$$

$$\begin{aligned}\texttt{DONE}^i(m,l) = \exists\ & L_1^i \subseteq \mathbf{R}^i(m,l), ..., L_k^i \subseteq \mathbf{R}^i(m,l-k+1), ..., L_l^i \subseteq \mathbf{R}^i(m,1) \text{ s.t.} \\ & \big(\forall\ 1 \leq k \leq l-1 : L_k^i \cap \mathbf{R}^i(m,l-k) \neq \varnothing\big) \wedge \\ & \texttt{LEVELS}_{n,t,t}(L_1^i, ..., L_l^i) \text{ holds.}\end{aligned}$$

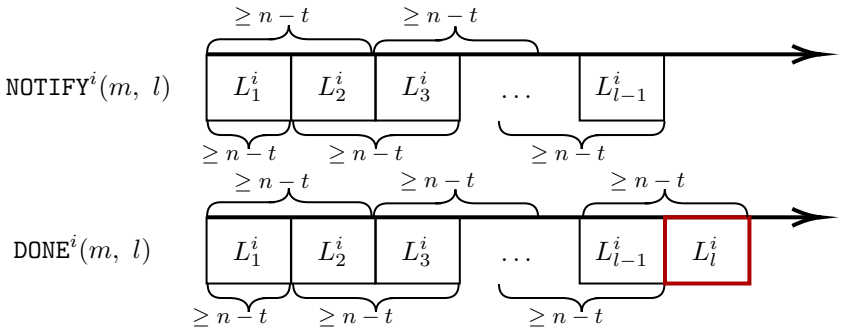

Figure 2 The condition $\texttt{NOTIFY}^i(m,l)$, in contrast to the condition $\texttt{DONE}^i(m,l)$.

Protocol $\Pi_{\texttt{RBC}}^{n,b}$

Code for the sender S

1: On input m, send m to every subset of $b-1$ recipients via b-cast and terminate with output m.

Code for recipient R_i

Initialize $\mathbf{R}^i(m,k) = \varnothing$ for any m and $1 \leq k \leq b$.

1: Upon k-receiving $(\texttt{READY}, m)$ from R_j, add R_j to $\mathbf{R}^i(m,k)$.
2: As soon as $\texttt{NOTIFY}^i(m,l)$ holds and m was l-received from S (for a message m), send $(\texttt{READY}, m)$ to every set of $b-1$ recipients via b-cast and add R_i to $\mathbf{R}^i(m,k)$ for $1 \leq k \leq b$.
3: As soon as $\texttt{DONE}^i(m,l)$ holds and m was l-received from S (for a message m), output m and terminate.

5.2 Resilience Proof

We divide the proof in two cases: when $3 \leq b \leq 4$, and when $b > 4$. We first state a number of intermediate lemmas that help in the proofs of both cases. Note that in our protocol, corrupted recipients can send inconsistent READY messages. However, they are limited by the following property implied by Lemmas 7 and 8.

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

Intuitively, the following property ensures that if no honest recipient can place itself on level $b-2$ for a message m, then the honest recipient cannot send $(\texttt{READY}, m)$ or output m. The proof is enclosed in the full version of the paper.

▶ **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 $(\texttt{READY}, m)$ or complete the protocol with output m.*

The next property ensures that if an honest recipient outputs m on level l, then all the honest recipients can place themselves on level at most $l+1$ for m and send $(\texttt{READY}, m)$. It is proven in the full version of the paper.

▶ **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 $(\texttt{READY}, m)$.*

We now show that once an honest recipient has terminated with output m on level l, every honest recipient can output m, intuitively by placing itself on level at most $l+1$.

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

Proof. Let R_i denote the first honest recipient that sends $(\texttt{READY}, m)$. Then, R_i l-receives m and $\texttt{NOTIFY}^i(m,l)$ holds for $l < b-1$, according to Lemma 12. According to the definition of $\texttt{NOTIFY}^i(m,l)$, $\exists\ L_1^i \subseteq \mathbf{R}^i(m,l)$, ..., $L_k^i \subseteq \mathbf{R}^i(m,l-k+1)$, ..., $L_{l-1}^i \subseteq \mathbf{R}^i(m,2)$ such that $L_k^i \cap \mathbf{R}^i(m,l-k) \neq \varnothing$ for every $1 \leq k \leq l-1$ and $\texttt{LEVELS}_{n,t,t}(L_1^i, ..., L_{l-1}^i)$ holds. Since R_i is the first honest recipient to send $(\texttt{READY}, m)$, every recipient in $\bigcup_{k=1}^{l-1} L_k^i$ is corrupted.

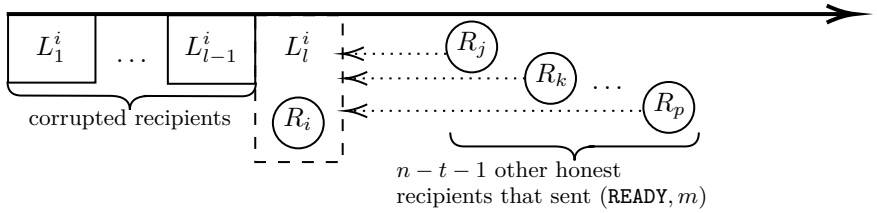

Figure 3 All the honest recipients have sent $(\texttt{READY}, m)$.

As shown in Figure 3, since every honest recipient sends $(\texttt{READY}, m)$, eventually $\left|\mathbf{R}^j(m,1) \setminus \left(\bigcup_{k=1}^{l-1} L_k^i\right)\right| \geq n-t$ for any honest recipient R_j. We can assign $n-t-\left|L_{l-1}^i\right|$ honest recipients to L_l^i. Then, $\texttt{DONE}^i(m,l)$ holds and R_i can output m. Every honest recipient R_j can achieve $\texttt{DONE}^j(m,l+1)$ by assigning $L_k^j = L_k^i$, according to Lemma 11, and by assigning to L_{l+1}^j the honest recipients that are not in L_l^i. Then, $\left|L_l^j \cup L_{l+1}^j\right| \geq n-t$ and $\texttt{DONE}^j(m,l+1)$ holds. Additionally, from Lemma 7, R_j $(l+1)$-receives m and therefore it can output m. ◀

5.2.1 Resilience for $3 \leq b \leq 4$

▶ **Lemma 15.** *If $t < \lambda(1)$, validity is satisfied.*

Proof. Let m denote the input of the honest sender and let $m' \neq m$. Since the recipients only b-receive m and $t < \lambda(1) \leq \lambda(b-2)$, no honest recipient sends $(\texttt{READY}, m')$ or outputs

m', according to Lemma 12. Every recipient can 1-receive m from S, hence every honest recipient eventually sends $(\texttt{READY}, m)$. From Lemma 14, we obtain that every honest recipient eventually outputs m. ◀

▶ **Lemma 16.** *If $t < \lambda(1)$, consistency is satisfied.*

Proof. Assume that an honest recipient R_i completes the protocol with output m. Since $t < \lambda(1)$, we can assume without loss of generality that R_i has 1-received m.

According to Lemma 8, no honest recipient can $(b-2)$-receive $m' \neq m$ and since $t < \lambda(1) \leq \lambda(b-2)$, we obtain from Lemma 12 that no honest recipient sends $(\texttt{READY}, m')$ or outputs m'. According to Lemma 13, every honest recipient sends $(\texttt{READY}, m)$. It follows from Lemma 14 that every honest recipient outputs m. ◀

We immediately obtain the following result from lemmas 15 and 16.

▶ **Theorem 17.** *Let $3 \leq b \leq 4$. $\Pi^{n,b}_{\texttt{RBC}}$ is a reliable broadcast protocol secure up to $t < \lambda(1)$ corruptions in $\mathcal{N}_b$.*

From Lemma 10, which states that $t < \lambda(1) = n - t$, and Theorem 17 we obtain:

▶ **Corollary 18.** *If $3 \leq b \leq 4$, $\Pi^{n,b}_{\texttt{RBC}}$ is a reliable broadcast protocol secure up to $t < \frac{n}{2}$ corruptions in $\mathcal{N}_b$.*

5.2.2 Resilience for $b > 4$

We firstly add a few properties that will be useful in proving that our protocol achieves asynchronous broadcast. For certain thresholds, the READY messages the honest recipients send are unique and *consistent* with respect to the messages that the honest parties output upon termination. That is, if an honest recipient outputs m, then it sent $(\texttt{READY}, m)$, while if an honest party sends $(\texttt{READY}, m)$, no honest party can output $m' \neq m$.

The proofs of the following lemmas are enclosed in the full version of the paper.

▶ **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 $\texttt{NOTIFY}^i(m, l)$ holds for an honest recipient R_i, $\texttt{NOTIFY}^j(m', l')$ is false for any honest recipient R_j, where $m \neq m'$.*

▶ **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 $(\texttt{READY}, m)$.*

▶ **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 $(\texttt{READY}, m)$, no honest recipient completes the protocol with output $m' \neq m$.*

We now show the conditions for our protocol to achieve validity and consistency.

▶ **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 $(\texttt{READY}, m')$ or outputs m'. Every honest recipient eventually sends $(\texttt{READY}, m)$, as they can 1-receive m from S. 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 that an honest recipient R_i completes the protocol with output m. According to Lemmas 19 and 21, no honest recipient sends $(\texttt{READY}, m')$ or outputs m', where $m' \neq m$. Consequently, it follows from Lemmas 20 and 13, that every honest recipient sends $(\texttt{READY}, m)$. Then, we obtain from Lemma 14 that every honest recipient outputs m. ◀

The next result follows immediately from Lemmas 22 and 23.

▶ **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 $l + l' \geq b$, $\Pi^{n,b}_{\texttt{RBC}}$ achieves reliable broadcast in $\mathcal{N}_b$.*

In the full version of the paper, we show that $t < \frac{b-4}{b-2}n + \frac{8}{b-2}$ (resp. $t < \frac{b-3}{b-1}n + \frac{6}{b-1}$) implies the hypothesis of Theorem 24. Hence, we obtain the following:

▶ **Corollary 25.** *Let $b > 4$. If b is even (resp. odd), $\Pi^{n,b}_{\texttt{RBC}}$ achieves resilient reliable broadcast in $\mathcal{N}_b$ secure against $t < \frac{b-4}{b-2}n + \frac{8}{b-2}$ (resp. $t < \frac{b-3}{b-1}n + \frac{6}{b-1}$) corruptions.*

6 Optimal Nonstop Broadcast

In this section, we present a protocol that achieves nonstop reliable broadcast secure against $t < \frac{b-1}{b+1}n$ corruptions. In the full version of the paper, 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]. The main difference relies on the fact that our construction needs to handle the fact that not all partial channels need to give output. As a consequence, instead of relying on the intermediate abstraction *proxcast*, which provides a fixed level of consistency at a known point in time, parties need to keep track of the received messages, and continue sending messages whenever *the consistency level increases*, which makes the overall combinatorial analysis substantially more complex.

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 $t'_v = \min(t_v, n-2)$ 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 up to t_v corruptions if $t_v < n-1$, and up to $n-2$ corruptions out of the $n-1$ recipients otherwise. On the other hand, if R_i is dishonest, then 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 for 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 $(\texttt{READY}, k, m_k, l_k)$ 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, all the other honest recipients confirm with level $l+1$, and eventually every honest recipient outputs at level $l+1$. Intuitively, the protocol does not allow parties to terminate 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:

$$\texttt{DONE}^i(m,l) = \exists\, L_1 \subseteq \mathbf{R}^i(m,1), \ldots, L_l \subseteq \mathbf{R}^i(m,l) \text{ s.t. } \texttt{LEVELS}_{n,t_v,t_c}(L_1,\ldots,L_l) \text{ holds.}$$

▶ **Definition 26.** *A message* $(\texttt{READY}, k, m_k, l_k)$ *is consistent with respect to the set of messages* $\{(\texttt{READY}, k', m_{k'}, l_{k'}) \mid 1 \leq k' < k\}$ *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 27.** *A recipient* R_i *accepts a message* $(\texttt{READY}, k, m_k, l_k)$ *from a recipient* R_j *if it has received the messages* $\{(\texttt{READY}, k', m_{k'}, l_{k'}) \mid 1 \leq k' \leq k\}$ *from* R_j *and* $(\texttt{READY}, k, m_k, l_k)$ *is consistent with respect to them.*

▶ **Definition 28.** $\mathbf{R}^i(m,l) \subseteq \mathcal{R}$ *denotes the set of recipients that* R_i *has accepted a message* $(\texttt{READY}, \cdot, m, l)$ *from. We use* $\mathbf{R}^i(m, \leq l)$ *to denote* $\bigcup_{j=1}^{l} \mathbf{R}^i(m,j)$.

We now formally present our protocol $\Pi^{n,b}_{\texttt{nRBC}}(t_v, t_c, S, \mathcal{R})$ for n recipients in the communication network $\mathcal{N}_b$, where S is the sender, and $\mathcal{R}$ is the set of recipients.

Protocol $\Pi^{n,b}_{\texttt{nRBC}}(t_v, t_c, S, \mathcal{R})$

Code for the sender S

1: On input m, send m via b-cast to every subset of $b-1$ recipients.

Code for recipient $R_i \in \mathcal{R}$

Initialize $k = 0$ and $\mathbf{R}^i(m,l) = \varnothing$ for every m and $1 \leq l \leq b$.

1: **if** $n = b$ **then**
2: Upon receiving a value from the b-cast with sender S and recipient set $\mathcal{R}$, output m.
3: **end if**
4: When receiving $(\texttt{READY}, p, m_p, l_p)$ from R_j, wait until receiving $M_j = \{(\texttt{READY}, p', m_{p'}, l_{p'}) \mid 1 \leq p' < p\}$ from R_j. If $(\texttt{READY}, p, m_p, l_p)$ is consistent with respect to M_j, add R_j to $\mathbf{R}^i(m_p, l_p)$.
5: When l-receiving m from the sender such that $(\texttt{READY}, k+1, m, l)$ is consistent with respect to your previously sent messages, send $(\texttt{READY}, k+1, m, l)$ to the other recipients by invoking $\Pi^{n-1,b}_{\texttt{nRBC}}(\min(t_v, |\mathcal{R} \setminus \{R_i\}| - 1), t_c - 1, R_i, \mathcal{R} \setminus \{R_i\})$, add R_i to $\mathbf{R}^i(m,l)$, and increment k.
6: When observing for the first time that $\texttt{DONE}^i(m,l)$ holds and you have l-received m from the sender, output m.

6.2 Resilience Proof

In this section, we define a predicate $\mathtt{Q}^b_n(t_v, t_c)$ which reflects the conditions that t_v and t_c must satisfy such that $\Pi^{n,b}_{\texttt{nRBC}}$ achieves (t_v, t_c)-nonstop reliable broadcast.

$\mathtt{Q}^b_b(t_v, t_c) = \texttt{true}$ since $\Pi^{n,b}_{\texttt{nRBC}}$ implements an ideal b-cast. Then, for $n > b$, $\mathtt{Q}^b_n(t_v, t_c) = \bigwedge_{k=0}^{n-b-1} \mathtt{P}^b_{n-k}(\min(t_v, n-k-1), t_c)$, where $\mathtt{P}^b_n(t_v, t_c)$ denotes a *local* predicate enclosing the conditions that t_v and t_c must satisfy assuming that $\Pi^{n-1,b}_{\texttt{nRBC}}$ achieves (t_v, t_c)-nonstop reliable

broadcast. We prove that $\mathtt{P}_n^b(t_v, t_c)$ can be defined as follows.

$$\mathtt{P}_n^b(t_v, t_c) = [t_v < \lambda(b-1) \vee n < \lambda(b)] \wedge \left[\forall 1 \leq l \leq b : (t_c < \lambda(l-1) + \lambda(b-l-1) \vee n < \lambda(l) + \lambda(b-l))\right]$$

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 $\mathtt{Q}_{n-1}^b(\min(t_v, |\mathcal{R}|-2), t_c-1)$ holds, i.e., that $\Pi_{\mathtt{nRBC}}^{n-1,b}(\min(t_v, |\mathcal{R}\setminus\{R_i\}|-1), t_c-1, R_i, \mathcal{R}\setminus\{R_i\})$ achieves validity (resp. consistency) up to $\min(t_v, |\mathcal{R}\setminus\{R_i\}|-1)$ (resp. t_c-1) corruptions. We show that satisfying $\mathtt{P}_n^b(t_v, t_c)$ suffices for $\Pi_{\mathtt{nRBC}}^{n,b}$ to achieve validity.

▶ **Lemma 29.** *If an honest recipient R_i sends $(\mathtt{READY}, i, m, l)$, then eventually $R_i \in \mathbf{R}^j(m, l)$ for any honest recipient R_j.*

Proof. R_i invokes $\Pi_{\mathtt{nRBC}}^{n-1,b}(\min(t_v, |\mathcal{R}\setminus\{R_i\}|-1), t_c-1, R_i, \mathcal{R}\setminus\{R_i\})$ with at most $\min(t_v, |\mathcal{R}|-2)$ corrupted recipients. It follows that $\Pi_{\mathtt{nRBC}}^{n-1,b}$ 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 $\mathbf{R}^j(m, l)$. ◀

▶ **Lemma 30.** *If m is the input of the honest sender, then every honest recipient can output m.*

Proof. Let m denote the input of the honest sender. Hence, the recipients eventually 1-receive m from S and never l-receive $m' \neq m$ for $l < b$. Then, the honest recipients can send $(\mathtt{READY}, \cdot, m, 1)$ as it is consistent with respect to any messages they sent beforehand. Eventually, $\left|\mathbf{R}^i(m, 1)\right| \geq n - t_v$ and therefore $\mathtt{DONE}^i(m, 1)$ holds according to Lemma 29 for every honest recipient R_i. It follows that every honest recipient R_i eventually outputs m. ◀

▶ **Lemma 31.** *If $\mathtt{P}_n^b(t_v, t_c)$ holds and m is the input of the honest sender, then no honest recipient outputs $m' \neq m$.*

Proof. Assume that an honest recipient R_i outputs m'. $\mathtt{DONE}^i(m', b)$ must hold since no recipient l-receives m' from the sender for $l < b$, implying that $n \geq \lambda(b)$. Additionally, $\mathbf{R}^i(m', \leq b-1)$ consists entirely of corrupted recipients, and since $\mathtt{DONE}^i(m', b)$ implies $\mathtt{DONE}^i(m', b-1)$, it follows that $t_v \geq \lambda(b-1)$. Hence, $t_v \geq \lambda(b-1) \wedge n \geq \lambda(b)$, which contradicts $\mathtt{P}_n^b(t_v, t_c)$. ◀

Finally, using Lemmas 30 and 31, we conclude the following.

▶ **Lemma 32.** *If $\mathtt{P}_n^b(t_v, t_c)$ holds, then $\Pi_{\mathtt{nRBC}}^{n,b}(t_v, t_c, S, \mathcal{R})$ achieves validity.*

6.2.2 Consistency

In each result enclosed in this section, we assume that there are at most t_c corrupted recipients, and that $\mathtt{Q}_{n-1}^b(\min(t_v, |\mathcal{R}|-2), t_c-1)$ holds. We show that $\mathtt{P}_n^b(t_v, t_c)$ suffices for $\Pi_{\mathtt{nRBC}}^{n,b}$ to achieve consistency.

Properties of `READY` Messages

Intuitively, the following lemma shows that the honest recipients eventually receive the same messages after an invocation of the subprotocol.

▶ **Lemma 33.** *If an honest recipient receives m from R_i, all the other honest recipients eventually receive m. Additionally, if R_i is honest, m is the message that R_i sent.*

Proof. R_i invokes $\Pi^{n-1,b}_{\texttt{nRBC}}(\min(t_v, |\mathcal{R}\setminus\{R_i\}|-1), t_c-1, R_i, \mathcal{R}\setminus\{R_i\})$. There are at most $t_c \leq \min(t_v, |\mathcal{R}|-2)$ corrupted recipients in $\mathcal{R}\setminus\{R_i\}$ if R_i is honest, and at most t_c-1 otherwise. Since $\mathtt{Q}^b_{n-1}(\min(t_v, |\mathcal{R}|-2), t_c-1)$ holds, $\Pi^{n-1,b}_{\texttt{nRBC}}$ achieves (t_v, t_c)-nonstop broadcast. ◀

Lemma 33 immediately implies that the honest parties eventually accept the same messages from the other recipients. The proof is enclosed in the full version of the paper.

▶ **Lemma 34.** *If $R_j \in \mathbf{R}^i(m,l)$ for an honest recipient R_i, then eventually $R_j \in \mathbf{R}^k(m,l)$ for any honest recipient R_k.*

The following lemma provides a range of levels on which the honest recipients can place themselves for a message m. The result is proven in the full version of the paper.

▶ **Lemma 35.** *Let $U \subseteq \mathcal{R}$ be the smallest set containing an honest recipient such that S U-sends m. Then, for any honest $R_i \in \mathcal{R}$, there is no honest party in $\mathbf{R}^i(m, \leq l-1) \cup \mathbf{R}^i(m', \leq b-l-1)$, where $l = \big|U\big|$.*

If l is the smallest level on which an honest recipient can place himself based on the messages of the sender, then eventually the $n-t_c$ honest recipients will place themselves on levels l and $l+1$, as they continue to run the protocol even after they obtain an output.

▶ **Lemma 36.** *Let $U \subseteq \mathcal{R}$ be the smallest set containing an honest recipient such that S U-sends m and let $l = \big|U\big|$. Then, it eventually holds that $\big|\mathbf{R}^i(m, \leq l+1) \setminus \mathbf{R}^i(m, \leq l-1)\big| \geq n-t_c$ for any honest recipient R_i.*

Proof. Let R_j denote an arbitrary honest recipient. From Lemma 35, $R_j \notin \mathbf{R}^i(m, \leq l-1) \cup \mathbf{R}^i(m', \leq b-l-1)$. R_j eventually $(l+1)$-receives m according to Lemma 7, and it sends $(\texttt{READY}, \cdot, m, l+1)$ since it is consistent with respect to any messages it previously sent. Consequently, $R_j \in \mathbf{R}^i(m, \leq l+1) \setminus \mathbf{R}^i(m, \leq l-1)$ since according to Lemma 33, R_i eventually receives each message that R_j sends. ◀

Our protocol ensures that the notifications of the recipients satisfy the following condition. The proof is enclosed in the full version of the paper.

▶ **Lemma 37.** *If $m \neq m'$ and $l + l' \leq b$, $\mathbf{R}^i(m, \leq l) \cap \mathbf{R}^i(m', \leq l') = \varnothing$ for any honest R_i.*

Properties of the DONE Predicate

▶ **Lemma 38.** *If $\texttt{DONE}^i(m,l)$ holds for an honest recipient R_i, then $\texttt{DONE}^j(m,l)$ eventually holds for any other honest R_j.*

Proof. Follows from Lemma 34, as R_j eventually receives the same messages as R_i. ◀

▶ **Lemma 39.** *Let $U \subseteq \mathcal{R}$ be the smallest set containing an honest recipient such that S U-sends m and let $l = \big|U\big|$. Assume that $\texttt{DONE}^i(m,l)$ holds for an honest R_i.*

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

Proof. Assume that $\texttt{DONE}^i(m', b-l)$ holds. Note that, according to Lemma 37, $\mathbf{R}^i(m, \leq l) \cap \mathbf{R}^i(m', \leq b-l) = \varnothing$. Then, $n \geq \big|\mathbf{R}^i(m, \leq l) \cup \mathbf{R}^i(m', \leq b-l)\big| \geq \lambda(l) + \lambda(b-l)$, as shown in Figure 4. Additionally, using Lemma 35, we obtain that every recipient in $\mathbf{R}^i(m, \leq l-1) \cup \mathbf{R}^i(m', \leq b-l-1)$ is corrupted. It follows that $t \geq \big|\mathbf{R}^i(m, \leq l-1) \cup \mathbf{R}^i(m', \leq b-l-1)\big| \geq \lambda(l-1) + \lambda(b-l-1)$, contradicting $\mathtt{P}^b_n(t_v, t_c)$.

◀

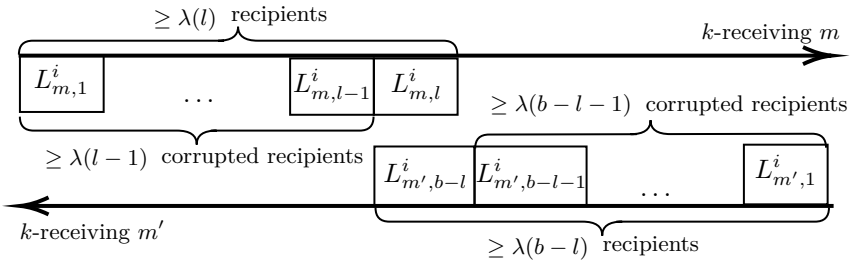

Figure 4 Levels if both $\texttt{DONE}^i(m,l)$ and $\texttt{DONE}^i(m',b-l)$ hold.

Achieving Consistency

▶ **Lemma 40.** *If an honest recipient R_i outputs m, every honest recipient eventually outputs m.*

Proof. Since R_i has output m, it l-received m such that $\texttt{DONE}^i(m,l)$ holds. If every honest recipient can l-receive m, $\texttt{DONE}^i(m,l)$ is enough to guarantee their output, by Lemma 38.

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

According to Lemma 36, $\left|\mathbf{R}^j(m, \leq l+1) \setminus \mathbf{R}^j(m, \leq l-1)\right| \geq n-t$ eventually holds for any honest R_j, and, since $\left|\mathbf{R}^j(m,l+1)\right| \geq 1$, $\texttt{DONE}^j(m,l+1)$ holds. It follows that every honest recipient eventually outputs m. ◀

▶ **Lemma 41.** *If $\mathtt{P}^b_n(t_v,t_c)$ and an honest recipient R_i outputs m, then no honest recipient outputs $m' \neq m$.*

Proof. Assume that an honest recipient R_j outputs $m' \neq m$. Let $U \subseteq \mathcal{R}$ be the smallest set containing an honest recipient such that S U-sends m and let $l = \left|U\right|$. Since R_i completed the protocol, and according to Lemma 9, $\texttt{DONE}^i(m,l)$ holds. It follows from Lemma 38 that $\texttt{DONE}^j(m,l)$ eventually holds as well. According to Lemma 39, $\texttt{DONE}^j(m,b-l)$ must be false. Then, R_j has completed the protocol by $(b-l-1)$-receiving m', contradicting Lemma 8. ◀

Using Lemmas 40 and 41, we conclude the following.

▶ **Lemma 42.** *If $\mathtt{P}^b_n(t_v,t_c)$ holds, then $\Pi^{n,b}_{\texttt{nRBC}}(t_v,t_c,S,\mathcal{R})$ achieves consistency.*

6.2.3 Corruption Threshold

We assemble the results proved in the sections 6.2.1 and 6.2.2. The next result follows immediately from Lemmas 32 and 42.

▶ **Theorem 43.** *If $\mathtt{Q}^b_n(t_v,t_c)$ holds, then $\Pi^{n,b}_{\texttt{nRBC}}$ achieves (t_v,t_c)-nonstop reliable broadcast.*

In the full version of the paper, we show the technical lemmas that prove that if $2t_v + (b-1)t_c < (b-1)n$, then predicate $\mathtt{Q}^b_n(t_v,t_c)$ holds, leaving us with the next theorem.

▶ **Theorem 44.** *If $2t_v + (b-1)t_c < (b-1)n$, then $\Pi^{n,b}_{\texttt{nRBC}}(t_v,t_c)$ achieves (t_v,t_c)-nonstop reliable broadcast in $\mathcal{N}_b$. Additionally, $\Pi^{n,b}_{\texttt{nRBC}}(t,t)$ is a t-resilient nonstop reliable broadcast protocol in $\mathcal{N}_b$, for any $t < \left(\frac{b-1}{b+1}n\right)$.*

References

1 Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *25th ACM STOC*, pages 52–61. ACM Press, May 1993. doi:10.1145/167088.167109.

2 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

3 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

4 Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM (JACM)*, 32(4):824–840, 1985.

5 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.

6 Jeffrey Considine, Matthias Fitzi, Matthew K. Franklin, Leonid A. Levin, Ueli M. Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, July 2005. doi:10.1007/s00145-005-0308-x.

7 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

8 Matthias Fitzi and Ueli M. Maurer. From partial consistency to global broadcast. In *32nd ACM STOC*, pages 494–503. ACM Press, May 2000. doi:10.1145/335305.335363.

9 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420.

10 Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, January 2000. doi:10.1007/s001459910003.

11 Alexander Jaffe, Thomas Moscibroda, and Siddhartha Sen. On the price of equivocation in byzantine agreement. In Darek Kowalski and Alessandro Panconesi, editors, *31st ACM PODC*, pages 309–318. ACM, July 2012. doi:10.1145/2332432.2332491.

12 Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

13 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989. doi:10.1145/73007.73014.

14 D. V. S. Ravikant, M. Venkitasubramaniam, V. Srikanth, K. Srinathan, and C. P. Rangan. On byzantine agreement over (2,3)-uniform hypergraphs. In R. Guerraoui, editor, *Distributed Computing, 18th International Conference, DISC 2004, Amsterdam, The Netherlands, October 4-7, 2004, Proceedings*, volume 3274 of *Lecture Notes in Computer Science*, pages 450–464. Springer, 2004.

15 Pavel Raykov. Broadcast from minicast secure against general adversaries. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015, Part II*, volume 9135 of *LNCS*, pages 701–712. Springer, Heidelberg, July 2015. doi:10.1007/978-3-662-47666-6_56.