Byzantine Consensus in Wireless Networks

HAO LU, JIAN LIU*, and KUI REN

A Byzantine consensus protocol is essential in decentralized systems as the protocol ensures system consistency despite node failures. Research on consensus in wireless networks receives relatively less attention, while significant advancements in wired networks. However, consensus in wireless networks has equal significance as in wired networks.

In this paper, we propose a new reliable broadcast protocol that can achieve reliability with *high fault tolerance* over than the SOTA (PODC '05). With the new protocol, we further develop the first wireless network Byzantine consensus protocol under the assumption of *partial synchrony*. Notably, this consensus protocol removes the requirement of leaders and fail-over mechanism in prior works. We formally prove the correctness of both our new broadcast protocol and consensus protocol.

1 INTRODUCTION

The enduring challenge of Byzantine consensus has experienced a resurgence in recent years, gaining renewed attention as the recognized cornerstone of blockchain technology. While noteworthy breakthroughs have been achieved, these endeavors predominantly concentrate on wired networks, overlooking the crucial aspect of wireless networks. Conversely, Byzantine consensus holds equal significance for applications in wireless (or radio) networks. For instance, in a swarm of unmanned aerial vehicles (UAVs), achieving consensus on a common destination is crucial [18]. We underscore two pivotal distinctions in achieving Byzantine consensus in a wireless network, with one being advantageous and the other detrimental:

- Each node has a broadcast radius r, and any message transmitted by a source node is uniformly received by all its neighbors (i.e., those within a distance of r from the source node). As a result, a node *cannot* equivocate to its neighbors.
- When two nodes are situated beyond a distance of r, their communication relies on the assistance of intermediary nodes for forwarding. Consequently, a source node colluding with an intermediary node retains the potential for equivocation.

Reliable broadcast serves as a vital building block for Byzantine consensus. The process begins with a source node intending to disseminate messages to all other nodes in the system; once a correct node delivers a message, all other correct nodes will eventually deliver the same message, even if the source node equivocates. When all nodes exchange their messages by calling reliable broadcast, all correct nodes will eventually receive the same set of messages (maybe in different orders). Byzantine consensus takes this step further by imposing a specific order on these messages. To our knowledge, there is currently no asynchronous or partially synchronous Byzantine consensus established for wireless networks. Koo [14] proved that achieving reliable broadcast in an asynchronous wireless network is possible only when $f < \frac{1}{2}r(2r+1)$ in L_∞ or $f < 0.23\pi r^2$ in L_2,[1] where f denotes the number of Byzantine nodes in any single neighborhood. Bhandari et al. [1] further introduced a reliable broadcast protocol that aligns with these established bounds[2]. However, these bounds are not tight as signatures are not considered in the proof.

Our contribution. In this paper, we revisit the upper-bounds presented by Bhandari et al., revealing

*Corresponding author.

[1]The distance between two nodes $\mathcal{N}_{(x_1,y_1)}$ and $\mathcal{N}_{(x_2,y_2)}$ is defined to be: $\max\{|x_1 - x_2|, |y_1 - y_2|\}$ in L_∞, and $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ in L_2.

[2]A simplified version was presented in [2].

Authors' address: Hao Lu, luhao@zju.edu.cn; Jian Liu, liujian2411@zju.edu.cn; Kui Ren, kuiren@zju.edu.cn.

that these bounds could, in fact, be higher. Specifically, we propose a new reliable broadcast protocol for asynchronous wireless networks, which can tolerate more Byzantine nodes:

- $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$ in L_∞, and
- $f < \lfloor 0.3\pi r^2 \rfloor$ in L_2.

It also reduces the message complexity in a single neighborhood from $O(n^2)$ to $O(n)$ compared to the state-of-the-art[1], where n denotes the number of nodes in a single neighborhood. Based on this reliable broadcast protocol, we propose the *first* Byzantine consensus for wireless networks under the assumption of partial synchrony.

2 PRELIMINARIES

2.1 Model

Grid wireless network. In a grid wireless network, nodes are located on a grid consisting of 1×1 square units. Each node can be uniquely identified by its grid location (x, y), denoted as $\mathcal{N}_{(x,y)}$. All nodes have a broadcast radius of r, where $r \in \mathbb{Z}$. The messages broadcast by $\mathcal{N}_{(x,y)}$ can be received by all the nodes within a distance r from it. We refer to these nodes as the neighbors of $\mathcal{N}_{(x,y)}$. We remark that $\mathcal{N}_{(x,y)}$ and all its neighbors are termed as the *neighborhood* of (x, y). For brevity, we may use $\mathcal{N}_i$ to denote $\mathcal{N}_{(x,y)}$ (i.e., use i to denote (x, y)). In this paper, we consider two distance metrics, namely L_∞ and L_2.

For reliable broadcast, we consider an *infinite* grid characterized by an unbounded number of nodes (each neighborhood has n nodes). For Byzantine consensus, we extend it to a *finite* grid that comprises N nodes. Specifically, nodes are located on a $P \times Q$ grid, hence $N = P \cdot Q$ with $P = p \cdot r$, $Q = q \cdot r$ (p and q are positive integers).

Communication assumptions. For reliable broadcast, we assume an asynchronous communication, where the sent message will eventually be received but there is no time bound on message delay. For Byzantine consensus, we assume the communication is partially synchronous. There exists a known finite time bound Δ and a special event called Global Stabilization Time (GST) under this environment. The adversary must eventually cause the GST event to occur after some unknown finite time. Any message sent at time t must be received by time $max(t, GST) + \Delta$. Informally, the system behaves asynchronously till GST and synchronously after GST.

Following [1], we assume that message collision does not exist, and the channel is an *idealized shared* wireless channel where the messages broadcast by a source node will eventually received by its neighbors in the sent order.

Adversary. In this paper, we use the term "faulty node" to denote a Byzantine node that can behave arbitrarily but cannot cause collisions. In an infinite grid, we assume up to f faulty nodes can exist in any neighborhood; in a finite grid, we impose an additional assumption that the total number of faulty nodes is bounded by F.

2.2 Multisignatures

In a naive approach to signing a message, each node has a public/private pair of keys. All nodes know the public keys of others, while they only know their own private key. Nodes use private keys to sign a message and output a signature. A signature can be verified by using the corresponding public key. The multisignatures scheme allows multiple nodes to sign a common message and output a single aggregate signature. They can verify the aggregate signature via an aggregate public key. This scheme significantly reduces the space and computation needed for verification compared to the naive approach when multiple signatures are required as certificates. We can implement this approach based on BLS [4], a well-known pairing-based signature scheme. The key operations of the multisignatures scheme are as follows:

- $\text{Sign}(m)$: outputs a signature, where m is the input message.
- $\text{Aggre}(\sigma_1, ..., \sigma_n)$: outputs an aggregate signature, where $\sigma_1, ..., \sigma_n$ are the signatures of a common message.
- $\text{Verify}(\sigma, m)$: outputs $true$ or $false$: judges whether the σ is a valid signature of message m.

In fact, the Aggre operation also outputs a bitmap that indicates which node has the contribution to the aggregate signature. This bitmap enables the Verify operation to compute the aggregate public key correspondingly. For the sake of brevity, we omit the bitmap in subsequent sections. Each node only processes the messages with valid signatures. We also omit the signature verification subsequent sections for brevity.

2.3 Problem Definition

Reliable Broadcast. Here we assume a source node reliably broadcasts messages by calling $r\_bcast(m, \gamma)$, where m is a message, γ is a round number. Every node reliably delivers messages by outputting $r\_deliver(m, \gamma, \mathcal{N}_i)$, where m is a message, γ is a round number and $\mathcal{N}_i$ is the identity of the source node. The round number is used to differentiate the messages broadcast by a single node. A reliable broadcast protocol satisfies the following properties:

- **Validity:** if a correct node $\mathcal{N}_i$ calls $r\_bcast(m, \gamma)$, then all correct nodes will eventually output $r\_deliver(m, \gamma, \mathcal{N}_i)$.
- **Agreement:** if a correct node $\mathcal{N}_i$ outputs $r\_deliver(m, \gamma, \mathcal{N}_j)$, then all other correct nodes will output $r\_deliver(m, \gamma, \mathcal{N}_j)$.

At first glance, these two properties are identical, which is true only when the source node is correct. When the source node is faulty, it could broadcast two different messages by calling r_bcast twice with m and m' respectively. Due to the nature of an idealized shared channel, the neighbors of the source node will receive m and m' in the same order (i.e., receiving m before m'); correct neighbors will forward m and drop m'. However, faulty neighbors could forward m' instead. If the protocol is not carefully designed, different correct nodes might deliver different messages with the same round number, violating the agreement property.

We remark that when we describe the reliable broadcast protocol, we consider only one run. When we need to consider multiple runs of the reliable broadcast (e.g., being used as a building block for Byzantine consensus), we could have the source node attach a sequence number to the message. Then the above example could be considered as attaching the same sequence number to two different messages.

Byzantine consensus. In this paper, we focus on the Byzantine Atomic Broadcast problem. Here, we suppose each node can call $a\_bcast(m, \gamma)$ and output $a\_deliver(m, \gamma, \mathcal{N}_i)$, where m is a message, γ is a round number, and $\mathcal{N}_i$ is the identity of the corresponding source node. A Byzantine Atomic Broadcast protocol satisfies all the following properties:

- **Validity:** if a correct node $\mathcal{N}_i$ calls $a\_bcast(m, \gamma)$, then all correct nodes will eventually output $a\_deliver(m, \gamma, \mathcal{N}_i)$.
- **Agreement:** if a correct node $\mathcal{N}_i$ outputs $a\_deliver(m, \gamma, \mathcal{N}_j)$, then all other correct nodes will output $a\_deliver(m, \gamma, \mathcal{N}_j)$.
- **Total order:** if a correct node $\mathcal{N}_i$ outputs $a\_deliver(m, \gamma, \mathcal{N}_j)$ before $a\_deliver(m', \gamma', \mathcal{N}_{j'})$, then all correct nodes output $a\_deliver(m, \gamma, \mathcal{N}_j)$ before $a\_deliver(m', \gamma', \mathcal{N}_{j'})$.

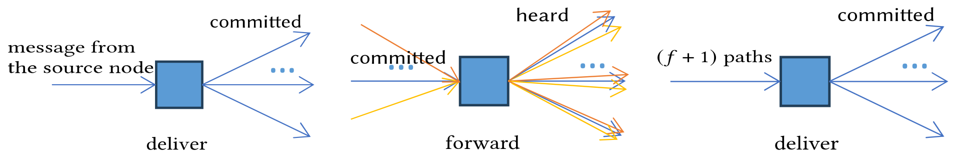
2.4 Revisiting reliable broadcast in wireless networks

The protocol proposed in [1] is the only reliable broadcast protocol that works in our considered model (i.e., in an asynchronous and infinite grid wireless network under Byzantine faults but no collision). In that protocol, when a node commits a message, it notifies its neighbors about this by broadcasting a committed message; when a node receives a committed message, it also notifies its neighbors about this by broadcasting a heard message. We use "node-disjoint" to indicate that paths do not share the same nodes.

- *Nodes that are neighbor to the source node.* They can directly deliver the message because the source node cannot equivocate to its neighbors.
- *Nodes that are not neighbor to the source node.* They deliver a message when they learn the message delivery within a neighborhood through $(f + 1)$ node-disjoint paths. It is for sure that at least one correct node has delivered the corresponding message because up to f nodes can be faulty in any single neighborhood.

Figure 1 shows the communication pattern of this protocol. The blue square represents a node. The arrow line on the left side of the node represents the received messages, while the one on the right represents the sent messages.

(1) Upon receiving the first message from the source node, a neighbor of the source node delivers the message and broadcasts a committed message (Figure 1a).
(2) Upon receiving a committed message, a node broadcasts a heard message. (Figure 1b).
(3) Upon receiving $(f + 1)$ node-disjoint committed or heard messages that signal a message delivery within a neighborhood, a node delivers the corresponding message and broadcasts a committed (Figure 1c).



(a) Neighbors of the source node. (b) All nodes, including the source node, its neighbors, and other nodes. (c) The nodes that are not the neighbors of the source node.

Fig. 1. Communication pattern of the reliable broadcast protocol in [1].

The message complexity of a single neighborhood with n nodes is $O(n^2)$ as each node needs to broadcast n messages (one committed message and $n - 1$ heard messages).

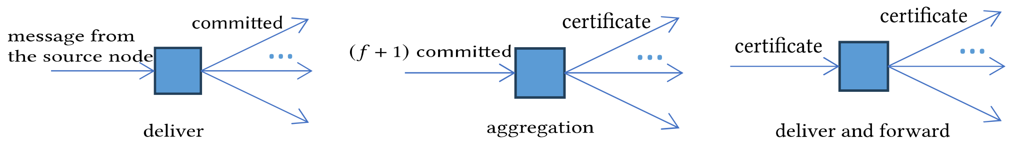
3 RELIABLE BROADCAST IN AN INFINITE GRID

Recall that reliable broadcast enables a source node to disseminate messages to all other nodes in the system, s.t. once a correct node delivers a message, all other correct nodes will eventually deliver the same message, even if the source node equivocates. Similar to [1], we consider two types of nodes:

- *Nodes that are neighbor to the source node.* They can directly deliver the message because the source node cannot equivocate to its neighbors.
- *Nodes that are not neighbor to the source node.* We allow them to deliver a message upon receiving a *certificate* that is an aggregate signature generated by at least $(f + 1)$ neighbors of the source node. Recall that at most f nodes can be malicious in any single neighborhood, a node receiving this certificate is convinced that at least one correct node has delivered the corresponding message.

Figure 2 shows the communication pattern of our reliable broadcast.

(1) Upon receiving a message from the source node, a neighbor of the source node delivers the message and broadcasts a signed committed message (Figure 2a).
(2) Upon receiving a committed message from a neighbor of the source node, (instead of broadcasting a heard message as in Figure 1b), a node waits for $(f+1)$ committed messages from other neighbors of the source node, aggregates them into a certificate, and broadcasts the certificate (Figure 2b).
(3) Upon receiving a certificate, if the node has not delivered any message in this round, it delivers the corresponding message and forwards the certificate to others (Figure 2c).



(a) Neighbors of the source node. (b) Neighbors of the source node. (c) All nodes, including the source node, its neighbors, and other nodes.

Fig. 2. The communication pattern of our reliable broadcast protocol.

The message complexity in any neighborhood is $O(n)$ as each node in the system only needs to broadcast a certificate (neighbors of the source node also broadcast a committed message). In order to validate a signed committed message, a node must know the sender's public key in advance. That means each node should store the public keys of all the source nodes and their neighbors beforehand. Therefore, in an infinite grid, we assume a finite number of nodes can be the source node of reliable broadcast. The details of our reliable broadcast protocol are shown in Figure 3.

Reliable broadcast

By calling $r\_bcast(m, \gamma)$, the source node $\mathcal{N}_s$ broadcasts a $\langle \text{propose}, s, m, \gamma \rangle$ message. Then, nodes run as the following:

(1) **Neighbors of the source node.**
- Upon receiving the first $\langle \text{propose}, s, m, \gamma \rangle$ message in round γ, $\mathcal{N}_i$ computes $\sigma_{\gamma,i} := \text{Sign}(\text{committed}||s||m||\gamma)$ and broadcasts $\left\langle \text{committed}, s, m, \gamma, \sigma_{\gamma,i} \right\rangle$.
- Upon receiving $f+1$ committed messages for m in round γ, $\mathcal{N}_i$ computes $C_\gamma(m) := \text{Aggre}(\{\sigma_*\}_{f+1})$ and broadcasts $\left\langle \text{certificate}, s, m, \gamma, C_\gamma(m) \right\rangle$.
- Upon receiving a $\left\langle \text{certificate}, s, m, \gamma, C_\gamma(m) \right\rangle$, if $\mathcal{N}_i$ has not forwarded a certificate message in round γ, it forwards this one.

(2) **Non-Neighbors of the source node.**
Upon receiving a $\left\langle \text{certificate}, s, m, \gamma, C_\gamma(m) \right\rangle$, if $\mathcal{N}_i$ has not delivered any message in round γ, it delivers m and forwards this certificate message.

Fig. 3. The details of our reliable broadcast.

3.1 Correctness proof when $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$ in L_∞

Next, we prove the validity and agreement of our reliable broadcast when $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$ in L_∞. The intuition of our proof is that "a valid certificate will be *constructed*, and *received by all correct nodes*". We consider the worst case where the "important" nodes that are near the source node are faulty. The derivation of $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$ can be found in Appendix A.

We first prove the validity.

Lemma 3.1. *Suppose $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$, a valid* certificate *will be constructed and broadcast if a source node calls $r\_bcast(m, \gamma)$ before other calls to r_bcast.*

Proof. Due to the nature of an idealized shared wireless channel, all neighbors of the source node will receive m before other messages sent from the source node (if the source node is faulty, it will broadcast other messages after m). Then, each correct neighbor will broadcast a committed message for m.

Each of the nodes within the yellow area of Figure 4a (including the boundary) has at least $(2r+1)(r+1)$ shared neighbors with the source node. For example, node P has the least number of shared neighbors with the source node, i.e., the dark green area in Figure 4b, which locates $(2r+1)(r+1)$ nodes. Notice that the nodes within the yellow area will receive the committed messages for m from the correct shared neighbors. As a result, each of them can receive at least $(2r+1)(r+1) - f \geq f + 1$ committed messages. That means any of them is able to construct a certificate. Then, we only need to prove that there is at least one correct node within this area, which is straightforward because the number of the nodes in this area is $2r(r+1)+1 > f$ (cf. Appendix B). Therefore, a valid certificate will be constructed by at least one correct node. □

Lemma 3.2. *Suppose $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$, if a valid* certificate *is constructed and broadcast by a correct node, all correct nodes will receive it.*

Proof. If a correct node $\mathcal{N}_{(a,b)}$ broadcasts a certificate and at least one *correct* neighbors in each of four directions (i.e., up $\mathcal{N}_{(*,b+y)}$, down $\mathcal{N}_{(*,b-y)}$, left $\mathcal{N}_{(a-x,*)}$, right $\mathcal{N}_{(a+x,*)}$, $0 < x, y \leq r$) forwards it, then all correct nodes will receive the certificate. Next, we prove that the certificate can be propagated in four directions.

We prove this by contradiction. Suppose a correct node broadcasts a certificate, but no neighbor in the right direction forwards the certificate. Then, all these $r(2r+1)$ neighbors (the red area of Figure 4c) should be faulty. However, $r(2r+1) > f$, which leads to a contradiction. Therefore, if a correct node broadcasts a certificate, at least one correct neighbor in the right direction will forward the certificate. The same reasoning applies to the other three directions, enabling the message to spread in four directions. □

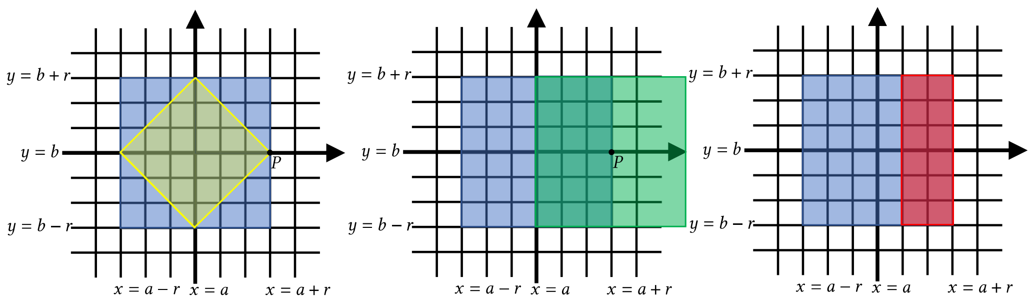
Theorem 3.3 (Validity). *Suppose $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$, if a correct node $\mathcal{N}_i$ calls $r\_bcast(m, \gamma)$, then all correct nodes will output $r\_deliver(m, \gamma, \mathcal{N}_i)$.*

Proof. By Lemma 3.1, a valid certificate will be generated and broadcast. By Lemma 3.2, all correct nodes will receive this certificate and output $r\_deliver(m, \gamma, \mathcal{N}_i)$. □

Next, we prove the agreement.

Lemma 3.4. *Suppose $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$, if a source node calls $r\_bcast(m, \gamma)$ before other calls to r_bcast, no valid* certificate *for m' ($m' \neq m$) can be constructed.*

Proof. We prove this by contradiction. All correct neighbors of the source node will broadcast a committed message for m because they will receive m before other messages sent from the source



(a) The more important nodes. (b) Shared neighbors. (c) Neighbors in the right direction.
🟦 The neighbors of the source node $\mathcal{N}_{(a,b)}$. ◇ The more important nodes. 🟩 The neighbors of the node $\mathcal{N}_P$.
🟥 The neighbors in the right direction of $\mathcal{N}_{(a,b)}$.

Fig. 4. The L_∞ metric in an infinite grid wireless network.

node. Suppose a valid certificate for m' ($m' \neq m$) is constructed. Then, at least one correct neighbor of the source node has broadcast a committed message for m', which leads to a contradiction.

□

Theorem 3.5 (Agreement). *If a correct node $\mathcal{N}_i$ outputs $r\_deliver(m, \gamma, \mathcal{N}_j)$, then all other correct nodes will output $r\_deliver(m, \gamma, \mathcal{N}_j)$.*

Proof. Suppose a correct node $\mathcal{N}_i$ outputs $r\_deliver(m, \gamma, \mathcal{N}_j)$ and consider the following two cases:

- *The node $\mathcal{N}_i$ is the neighbor of the source node $\mathcal{N}_j$.* In this case, the source node must have called $r\_bcast(m, \gamma)$ before other calls. By Lemma 3.1, a valid certificate for m will be constructed. By Lemma 3.4, no valid certificate for m' ($m' \neq m$) can be constructed. Then, by Lemma 3.2, all correct nodes will receive it and output $r\_deliver(m, \gamma, \mathcal{N}_j)$.
- *The node $\mathcal{N}_i$ is not the neighbor of the source node $\mathcal{N}_j$.* In this case, $\mathcal{N}_i$ must have received a valid certificate for m and forwarded it. By Lemma 3.2, all correct nodes will receive it and output $r\_deliver(m, \gamma, \mathcal{N}_j)$.

Therefore, if a correct node outputs $r\_deliver(m, \gamma, \mathcal{N}_j)$, other correct nodes will also output $r\_deliver(m, \gamma, \mathcal{N}_j)$.

□

3.2 Correctness proof when $f < \lfloor 0.3\pi r^2 \rfloor$ in L_2

Next, we prove the validity and agreement of our reliable broadcast when $f < \lfloor 0.3\pi r^2 \rfloor$ and $r \geq 4$ in L_2. The derivation of $f < \lfloor 0.3\pi r^2 \rfloor$ can be found in Appendix C. Notice that the base for the deductions in the L_2 metric is that for sufficiently large r, the area is a good approximation for the number of nodes in these regions.

We first prove the validity.

Lemma 3.6. *Suppose $f < \lfloor 0.3\pi r^2 \rfloor$, a valid* certificate *will be constructed and broadcast if a source node calls $r\_bcast(m, \gamma)$ before other calls to r_bcast.*

Proof. Similar to the proof of Lemma 3.1, each correct neighbor of the source node will broadcast a committed message for m.

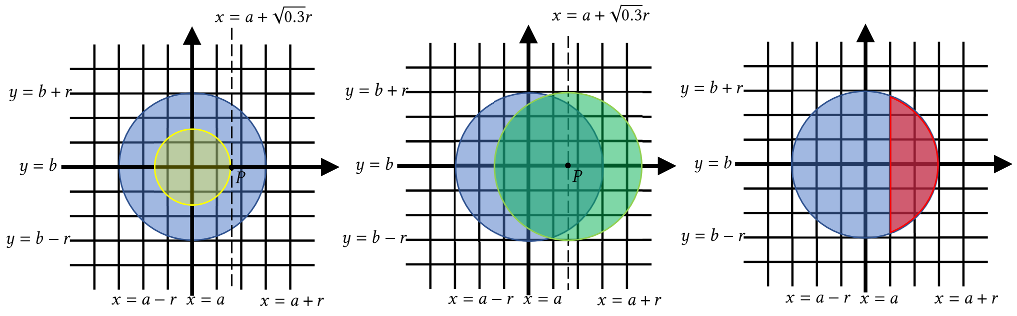
Denote each of the nodes within the yellow area of Figure 5a (including the boundary) has at least N_1 shared neighbors with the source node. For example node P shares the least number of neighbors with the source node. Specifically, their shared neighbors are within the dark green area in Figure 5b, and their number can be approximated as (cf. Appendix D):

$$N_1 = 2r^2 \cos^{-1}\frac{\sqrt{0.3}}{2} - \frac{1}{2}r^2\sqrt{1.11}$$

When $r \geq 4$, each node within the yellow area can receive at least $N_1 - f \geq f + 1$ committed messages because each of them will receive the committed messages for m from the correct shared neighbors. That means any of them is able to construct a certificate. Furthermore, there is at least one correct node within this area because $f < \lfloor 0.3\pi r^2 \rfloor$. Therefore, a valid certificate can be constructed by at least one correct node. □

Lemma 3.7. *Suppose $f < \lfloor 0.3\pi r^2 \rfloor$, if a valid* certificate *is constructed and broadcast by a correct node, all correct nodes will receive it.*

Proof. Similar to the proof of Lemma 3.2, we prove this by proving adversaries can not prevent a certificate from spreading in any direction. For example, to prevent a certificate from spreading to the right, all nodes located on the right should be faulty, i.e., the red area of Figure 5c, which locates $(0.5\pi r^2 - 2r - 1)$ nodes. However, $(0.5\pi r^2 - 2r - 1) - f \geq 1$ when $r \geq 4$, which means at least one correct node on the right. Then, the adversary can not prevent a certificate from spreading to the right. Therefore, by symmetry, the adversary can not prevent the messages from spreading in any direction. □



(a) The more important nodes. (b) Shared neighbors. (c) Neighbors in the right direction.

🟦 The neighbors of the node $\mathcal{N}_{(a,b)}$. 🟨 The more important nodes.

🟩 The neighbors of the node $\mathcal{N}_P$. 🟥 The neighbors in the right direction of $\mathcal{N}_{(a,b)}$.

Fig. 5. Analysis in an infinite grid radio network and L_2 metric.

Theorem 3.8 (Validity). *If a correct node $\mathcal{N}_i$ calls r_bcast(m, γ), then all correct nodes will output r_deliver$(m, \gamma, \mathcal{N}_i)$.*

Proof. Based on the Lemma 3.6, and Lemma 3.7, the proof of this theorem remains identical to the Theorem 3.3. □

Then, we prove the agreement.

Lemma 3.9. *Suppose $f < \lfloor 0.3\pi r^2 \rfloor$, if a source node calls r_bcast(m, γ) before other calls to r_bcast, no valid* certificate *for m' $(m' \neq m)$ can be constructed.*

Proof. The proof remains identical to that of Lemma 3.4. □

Theorem 3.10 (Agreement). *If a correct node $\mathcal{N}_i$ outputs $r\_deliver(m, \gamma, \mathcal{N}_j)$, then all other correct nodes will output $r\_deliver(m, \gamma, \mathcal{N}_j)$.*

Proof. With the Lemma 3.6, Lemma 3.7, and Lemma 3.9, the agreement proof remains identical to that of Theorem 3.5. □

4 RELIABLE BROADCAST IN A FINITE GRID

In a finite grid, a *corner node* has fewer neighbors than other nodes. For example, in Figure 6a, the node $\mathcal{N}_{(a,b)}$ only has neighbors in its right and up directions. Thus, we need a new bound f to achieve reliable broadcast even when the source node is in the corner.

4.1 Correctness proof when $f < \lfloor \frac{1}{2}(r+1)^2 \rfloor$ in L_∞

We prove the validity and agreement of our reliable broadcast when $f < \lfloor \frac{1}{2}(r+1)^2 \rfloor$ in L_∞. The derivation of $f < \lfloor \frac{1}{2}(r+1)^2 \rfloor$ in L_∞ can be found in Appendix E. We first prove the validity.

Lemma 4.1. *Suppose $f < \lfloor \frac{1}{2}(r+1)^2 \rfloor$ in L_∞, a valid* certificate *will be constructed and broadcast if a source node calls $r\_bcast(m, \gamma)$ before other calls to r_bcast.*

Proof. We consider the worst case where the source node is in the corner as shown in Figure 6a. For the same reason as in the proof of the Lemma 3.1, each neighbor of the source node will broadcast a committed message for m. The neighbors of the corner source node $\mathcal{N}_{(a,b)}$ are in the yellow area, which locates $(r+1)^2$ nodes, and they are all within the broadcast radius of each other. Therefore, each of them can receive at least $(r+1)^2 - f \geq f+1$ committed messages for m. Furthermore, at least one neighbor of the source node is correct because $(r+1)^2 > f$. Therefore, a valid certificate will be constructed by at least one correct node. □

Lemma 4.2. *Suppose $f < \lfloor \frac{1}{2}(r+1)^2 \rfloor$ in L_∞, if a valid* certificate *is constructed and broadcast by a correct node, all correct nodes will receive it.*

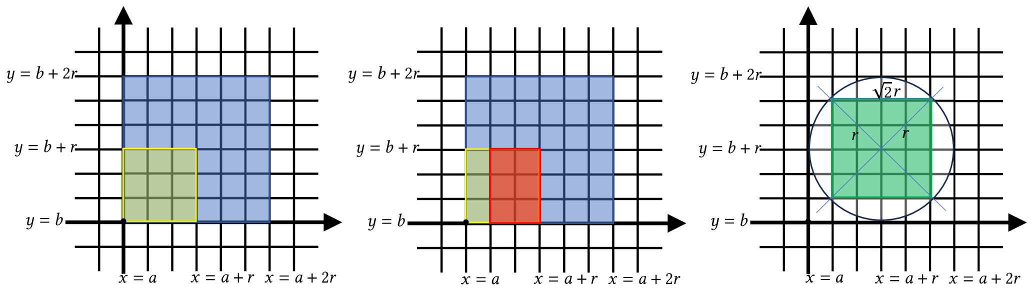
Proof. Again, we consider the worst case where the certificate is broadcast by a corner node $\mathcal{N}_{(a,b)}$. As shown in Figure 6b, $\mathcal{N}_{(a,b)}$ has $r(r+1) > f$ neighbors in the right direction. Therefore, if a correct $\mathcal{N}_{(a,b)}$ broadcasts a certificate, at least one correct node in its right direction will receive and forward the certificate. The same reasoning applies to the up direction, enabling the certificate to spread in the up direction. Thus, the certificate from the corner node $\mathcal{N}_{(a,b)}$ can always propagate to the right and the up directions (if necessary) and be received by all correct nodes. □

Theorem 4.3 (Validity). *If a correct node $\mathcal{N}_i$ calls $r\_bcast(m, \gamma)$, then all correct nodes will output $r\_deliver(m, \gamma, \mathcal{N}_i)$.*

Proof. Based on Lemma 4.1, and Lemma 4.2, the proof of this theorem remains identical to the proof of Theorem 3.3. □

Lemma 4.4. *Suppose $f < \lfloor \frac{1}{2}(r+1)^2 \rfloor$ in L_∞, if a source node calls $r\_bcast(m, \gamma)$ before other calls to r_bcast, no valid* certificate *for m' $(m' \neq m)$ can be constructed.*

Proof. The proof remains identical to that of Lemma 3.4. □



(a) Neighbors of the corner node. (b) Neighbors in the right direction. (c) Range transformation.

🟦 All nodes in a finite grid wireless network. 🟨 The neighbors of the corner node $\mathcal{N}_{(a,b)}$.
🟥 The neighbors of the node $\mathcal{N}_{(a,b)}$ in the right direction. ○ The actual broadcast range of a node.
🟩 The transformed broadcast range of a node.

Fig. 6. The L_∞ and L_2 metric in a finite grid wireless network.

Theorem 4.5 (Agreement). *If a correct node $\mathcal{N}_i$ outputs $r\_deliver(m, \gamma, \mathcal{N}_j)$, then all other correct nodes will output $r\_deliver(m, \gamma, \mathcal{N}_j)$.*

Proof. With the Lemma 4.1, Lemma 4.2 and Lemma 4.4, the agreement proof of Theorem 3.5 is applicable for this case. □

4.2 Correctness proof when $f < \lfloor \frac{1}{2}(\lfloor \frac{\sqrt{2}}{2} r \rfloor + 1)^2 \rfloor$ in L_2

We transform the problem from the L_2 metric to the L_∞ metric at the cost of reducing nodes' broadcast range. In more detail, we consider the nodes with a radius distance r in the L_2 metric as having a radius distance $r' = \lfloor \frac{\sqrt{2}}{2} r \rfloor$ in the L_∞ metric. As is shown in Figure 6c, we consider the green area to be the neighborhood of the node $\mathcal{N}_{(a+r,b+r)}$ whose real neighborhood is within the black circle area.

5 BYZANTINE CONSENSUS IN A FINITE GRID

The famous FLP result [10] states the impossibility of achieving deterministic consensus in an asynchronous network where at least one replica may crash. In this paper, we consider the partial synchrony to circumvent the FLP impossibility result. Specifically, the proposed consensus protocol ensures both safety and liveness in a synchronous network but does not guarantee liveness when the network behaves asynchronously. Furthermore, consensus among infinite nodes can be achieved by PoW or PoS protocols. PoW is notorious for its high costs, whereas PoS offers a solution to this issue. The core of PoS is to select a finite number of nodes based on some rules, then reach a consensus among them, and finally synchronize with other nodes. Thus, we focus on the Byzantine consensus with finite nodes in a finite grid. In this section, we present a Byzantine consensus protocol for wireless networks under the assumption of partial synchrony. We consider a finite grid, which locates $N = P \cdot Q$ nodes where $P = p \cdot r$, $Q = q \cdot r$. This consensus protocol can tolerant $F \leq \lfloor \lceil \frac{p}{2} \rceil \lceil \frac{q}{2} \rceil f \rfloor$ (cf. Appendix F) faulty nodes among N nodes.

In this protocol, nodes use the protocol proposed in Section 4 to reliably broadcast proposals round by round. Then, they construct their Directed Acyclic Graph (DAG) locally based on the delivered proposals. Then, each of them orders the DAG's vertices to make all nodes agree on the same sequence of vertices (i.e., proposals). Next, we introduce the DAG construction and vertices ordering more specifically.

Byzantine Consensus In Details

- **Local Data Structure**
  $\gamma := 0$ ▷ The round number.
  $\text{DAG}_i[]$ ▷ The local DAG, $\text{DAG}_i[0]$ is harded coded and $|\text{DAG}_i[0]| \geq N - F$.

A node $\mathcal{N}_i$ executes the following operations in background:

- **Proposal Construct.** When constructing a proposal in round γ, $\mathcal{N}_i$ puts $\text{DAG}_i[\gamma - 1]$ to a set p_1 and puts the DAG_i' proposals that there is no path to them to a set p_2. Then, $\mathcal{N}_i$ constructs a proposal $M = \langle \text{propose}, b, p_1, p_2 \rangle$, where b is a batch of requests.
- **Propose.** Upon $|\text{DAG}_i[\gamma]| \geq N - F$, and $\mathcal{N}_i$ meets either of the following conditions:
  - $\gamma \ mod\ 2 = 0$.
  - $\gamma \ mod\ 2 = 1$. The propose-timer$_{\gamma,i}$ expires or the $\mathcal{N}_i$ has added the leader proposal in round γ to DAG_i.

  $\mathcal{N}_i$ sets $\gamma = \gamma + 1$, constructs a proposal M in round γ and calls $a\_bcast(M.b, \gamma)$ by calling $r\_bcast(M, \gamma)$.
- **Jumping Propose.** Upon $|\text{DAG}_i[\gamma']| \geq N - F$ and $\gamma' > \gamma$, $\mathcal{N}_i$ sets $\gamma = \gamma'$, constructs a proposal M in round γ and calls $a\_bcast(M.b, \gamma)$ by calling $r\_bcast(M, \gamma)$. It also starts propose-timer$_{\gamma,i}$ when γ mod 2 = 1.
- **DAG Update.** Upon $r\_deliver(M, \gamma'', \mathcal{N}_j)$, where $M = \langle \text{propose}, b, p_1, p_2 \rangle$, $\mathcal{N}_i$ waits for that DAG_i contains all the proposals pointed by p_1 and p_2. Then, it adds M to $\text{DAG}_i[\gamma'']$.
- **Proposal Commit.** Upon $\text{DAG}_i[\gamma^+]$ contains $F + 1$ proposals that point to the leader proposal in round $\gamma^+ - 1$, $\mathcal{N}_i$ commits all the uncommitted leader proposals in its causal histories (include itself) in the order of round number.
  When committing a leader proposal, $\mathcal{N}_i$ also commits all its uncommitted causal histories in a deterministic order.
  $\mathcal{N}_i$ commits a proposal M from $\mathcal{N}_j$ in round γ by calling $a\_deliver(M.b, \gamma, \mathcal{N}_j)$.

Fig. 7. The details of our Byzantine consensus protocol.

DAG construction. Nodes reliably broadcast their proposals with some metadata in an infinite sequence of rounds to help them form a DAG. Specifically, each proposal has references pointing to the proposals from previous rounds. The vertices and edges in a DAG represent the proposals and references, respectively. Notice that only when the causal history of a vertex has been added to the DAG, the replica can add that vertex to the DAG, thereby avoiding the DAG's vertices pointing to a fabricated vertex. When a node adds $(N - F)$ vertices in the current round to the DAG, it enters the next round and proposes a new vertex, which points to these $(N - F)$ vertices and the vertices that there is no path to them in the previous rounds.

Vertices ordering. We interpret the structure of the DAG as a consensus protocol by considering the edges as votes. We introduce a leader in each odd round and refer to the vertex proposed by the leader as the leader vertex. We say the leader vertex has one vote if the vertex in the next round points to it. If a leader vertex has $(F + 1)$ votes, nodes commit all the uncommitted leader vertices in this leader vertex's causal histories (including this leader vertex) in the order of the round number. Recall that a replica enters the next round and proposes a new vertex when it has added $(N - F)$ vertices in the current round to the DAG. Then, if replicas always add the leader vertex to the DAG after adding $(N - F)$ other vertices, the leader vertex will have no votes. As a result, adversaries can prevent nodes from committing vertices by controlling the order in which nodes deliver vertices. To tackle this problem, we have nodes to wait for some time in each odd round to deliver the leader vertex and add it to the DAG. Surprisingly, when the leader is correct, the protocol progresses at network speed without extra waiting.

The core of our consensus protocol is to make replicas to agree on the same sequence of leader vertices. Due to the reliable broadcast, all nodes agree on the causal histories of the leader vertices. Thus, they can order the causal histories by some pre-defined deterministic rule to agree on the total order of the DAG's vertices (i.e., proposals). The details of our proposed consensus protocol are shown in Figure 7. A formal proof is available in Appendix G.

6 RELATED WORK

6.1 Reliable broadcast in grid wireless networks

Kranakis et al. propose a time-efficient broadcasting algorithm [16] for finite radio networks with a regular grid pattern and crash-stop failures. Koo [14] assumes that there are no collisions and address spoofing. Koo provides the first analysis of broadcasting algorithms for an infinite grid radio network with Byzantine failures and proves the impossibility of achieving reliable broadcast when the adversary corrupts $\lceil \frac{1}{2} r(2r + 1) \rceil$ neighbors of any honest node. Vaikuntanathan [22] proves that the protocol proposed in [14] indeed tolerates $\frac{1}{\sqrt{2}} r^2$ Byzantine faults. Bhandari et al. present a reliable broadcast algorithm [1] that matches the impossibility bound. Then they present a simpler characterization [2] and proofs for results proved earlier in [1]. In [15], Koo et al. present a reliable broadcast protocol up to the maximum tolerable Byzantine fault threshold when known bounded number of collisions and spoofs are allowed. Gilbert et al. [11] examine a model characterized by single-hop and collision-bounded, where a faulty node and a non-faulty node are allowed to send a maximum of β and β' messages, respectively. Notably, the value of β is assumed to be unknown to non-faulty nodes beforehand, while the source is assumed to be good. The authors establish the maximum ratio between the disruption caused by the adversary and the cost associated with causing that disruption. Additionally, they investigate the adversary's ability to delay the protocol without executing a single broadcast.

6.2 Reliable broadcast in general graph wireless networks

Pelc et al. [19] investigate the possibility of achieving reliable broadcast in a general graph under Byzantine failures. While they do not provide specific thresholds, the research presents upper and lower bounds that establish the feasibility of reliable broadcast. In [20], they investigate the possibility of achieving reliable broadcast under random transient failures. Each node may fail at each step with a constant probability $p < 1$. They provide the time complexity of almost-safe broadcasting and the tight bounds on p. Bhandari et al. [3] investigate the impact of random transient failures in a wireless grid network and provide necessary and sufficient conditions for achieving reliable broadcast in this model. In [9], they explore a scenario where Byzantine nodes have the capability to disrupt communication and interfere with the airwaves in an unrestricted manner. They assume that devices have access to multiple communication channels and propose an ϵ-gossip algorithm to mitigate the impact of Byzantine behavior.

6.3 Consensus in wireless networks

Chockler et al.[5] investigate the solvability of consensus for single-hop wireless networks under crash-stop failures and a realistic collision-prone model with an unknown number of participants. Considine et al. [7] consider the model that "multicast" channels of bounded size are available to the parties. They assume the existence of a multicast channel among every subset of players of some size b and give solutions for reliable broadcast and consensus problems. Clement et al. [6] investigate a non-equivocation model and show how to obtain a generic transformation from a consensus protocol that works under the crash-stop model into a protocol that provides the same guarantees under the Byzantine failure without increasing the number of participants. Some previous works [17, 23] aim to achieve an approximate Byzantine consensus under the wireless network model. These algorithms follow an iterative approach, where the state variable at each node is updated in each iteration as a linear interpolation of the states of selected neighbors. Khan et al. [13] explore the problem of Byzantine consensus with binary inputs under local broadcast model. They provide a comprehensive analysis and establish the necessary and sufficient conditions for achieving Byzantine consensus in this specific model.

6.4 DAG-based consensus

Keidar et al. [12] propose a consensus protocol called DAG-Rider based on a directed acyclic graph (DAG), which utilizes reliable broadcast as a fundamental component to construct the DAG. In this protocol, replicas order proposals solely based on the local DAG without further communication. However, DAG-Rider exhibits a long tail latency of approximately six rounds of reliable broadcast. In contrast, Tusk [8] and Bullshark [21] aim to improve the number of rounds required for reliable broadcast. Tusk reduces the rounds to five, while Bullshark further enhances it to only two rounds by leveraging synchronous periods. However, they are all designed for the classical point-to-point wired networks, not optimized for wireless networks.

7 CONCLUSION

In this paper, we first design a protocol that achieves reliable broadcast when $f < \lfloor \frac{1}{2}(r+1)(2r+1) \rfloor$ in the L_∞ metric and $f < \lfloor 0.3\pi r^2 \rfloor$ in the L_2 metric. Then, we propose the first Byzantine consensus protocol for wireless networks under the assumption of partial synchrony, which is leaderless and eliminates the fail-over mechanism. Finally, we provide the detailed descriptions and formal proofs for these two protocols.

REFERENCES

[1] Vartika Bhandari and Nitin H Vaidya. 2005. On reliable broadcast in a radio network. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. 138–147.

[2] Vartika Bhandari and Nitin H Vaidya. 2005. On reliable broadcast in a radio network: A simplified characterization. *CSL, UIUC, Tech. Rep* (2005).

[3] Vartika Bhandari and Nitin H Vaidya. 2007. Reliable broadcast in wireless networks with probabilistic failures. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 715–723.

[4] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short signatures from the Weil pairing. In *International conference on the theory and application of cryptology and information security*. Springer, 514–532.

[5] Gregory Chockler, Murat Demirbas, Seth Gilbert, Calvin Newport, and Tina Nolte. 2005. Consensus and collision detectors in wireless ad hoc networks. In *Proceedings of the twenty-fourth annual ACM Symposium on Principles of Distributed Computing*. 197–206.

[6] Allen Clement, Flavio Junqueira, Aniket Kate, and Rodrigo Rodrigues. 2012. On the (limited) power of non-equivocation. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*. 301–308.

[7] Jeffrey Considine, Matthias Fitzi, Matthew Franklin, Leonid A Levin, Ueli Maurer, and David Metcalf. 2005. Byzantine agreement given partial broadcast. *Journal of Cryptology* 18 (2005), 191–217.

[8] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. 2022. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*. 34–50.

[9] Shlomi Dolev, Seth Gilbert, Rachid Guerraoui, and Calvin Newport. 2007. Gossiping in a multi-channel radio network: An oblivious approach to coping with malicious interference. In *International Symposium on Distributed Computing*. Springer, 208–222.

[10] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM* 32, 2 (apr 1985), 374–382. https://doi.org/10.1145/3149.214121

[11] Seth Gilbert, Rachid Guerraoui, and Calvin Newport. 2006. Of malicious motes and suspicious sensors: On the efficiency of malicious interference in wireless networks. In *International Conference On Principles Of Distributed Systems*. Springer, 215–229.

[12] Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. 2021. All you need is dag. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 165–175.

[13] Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H Vaidya. 2019. Exact byzantine consensus on undirected graphs under local broadcast model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 327–336.

[14] Chiu-Yuen Koo. 2004. Broadcast in radio networks tolerating byzantine adversarial behavior. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*. 275–282.

[15] Chiu-Yuen Koo, Vartika Bhandari, Jonathan Katz, and Nitin H Vaidya. 2006. Reliable broadcast in radio networks: The bounded collision case. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*. 258–264.

[16] Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. 2001. Fault-tolerant broadcasting in radio networks. *Journal of Algorithms* 39, 1 (2001), 47–67.

[17] Heath J LeBlanc, Haotian Zhang, Xenofon Koutsoukos, and Shreyas Sundaram. 2013. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications* 31, 4 (2013), 766–781.

[18] Iván Maza, Fernando Caballero, Jesús Capitán, José Ramiro Martínez-de Dios, and Aníbal Ollero. 2011. Experimental results in multi-UAV coordination for disaster management and civil security applications. *Journal of intelligent & robotic systems* 61 (2011), 563–585.

[19] Andrzej Pelc and David Peleg. 2005. Broadcasting with locally bounded Byzantine faults. *Inform. Process. Lett.* 93, 3 (2005), 109–115. https://doi.org/10.1016/j.ipl.2004.10.007

[20] Andrzej Pelc and David Peleg. 2005. Feasibility and complexity of broadcasting with random transmission failures. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. 334–341.

[21] Alexander Spiegelman, Neil Giridharan, Alberto Sonnino, and Lefteris Kokoris-Kogias. 2022. Bullshark: Dag bft protocols made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2705–2718.

[22] Vinod Vaikuntanathan. 2005. Brief announcement: broadcast in radio networks in the presence of byzantine adversaries. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. 167–167.

[23] Haotian Zhang and Shreyas Sundaram. 2012. Robustness of information diffusion algorithms to locally bounded adversaries. In *2012 American Control Conference (ACC)*. IEEE, 5855–5861.

A DERIVATION OF $f < \lfloor \frac{1}{2}(2r+1)(r+1) \rfloor$

The intuition of this derivation is that "a valid certificate will be *constructed*, and *received by all correct nodes*".

We first consider how the neighborhood of a source node ensures that "a valid certificate will be *constructed*". As shown in Figure 8a, the node P among the nodes within the yellow area has the least shared neighbors with the source node. Specifically, their shared neighbors are within the dark green area in Figure 8b, which locates $(2r+1)((2-\beta)r+1)$ nodes. Recall that the neighbors of the source node broadcast the committed messages, which implies that nodes may receive committed messages from their shared neighbors with the source node. As a result, each node within the yellow area can receive at least $f+1$ committed messages and constructs a valid certificate when $(2r+1)((2-\beta)r+1) - f \geq f+1$. Then, a valid certificate will be constructed when there is at least one correct node within the yellow area in Figure 6a, which locates $2\beta r(\beta r+1)+1$ nodes. Thus, the number f of the faulty nodes in each neighborhood of a source node should meet the following two rules:

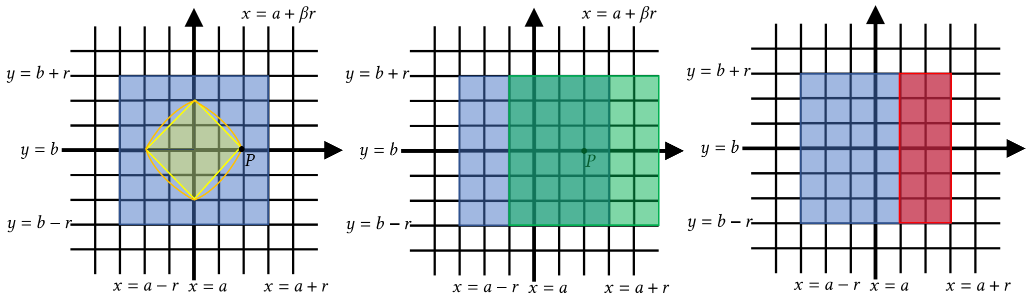
(1) $(2r+1)((2-\beta)r+1) - f \geq f+1$
(2) $f < 2\beta r(\beta r+1)+1$

Then, we consider how every neighborhood ensures that "a valid certificate will be *received by all correct nodes*". As shown in Figure 8c, all neighbors of the node $\mathcal{N}_{(a,b)}$ in the right are within the red area, which locates $(2r+1)r$ nodes. Then, by symmetry, each node has $(2r+1)r$ nodes in every direction. In order to allow a certificate to spread to all directions (i.e., up, down, left, and right), nodes should have at least one correct node in each direction, which requires $f < (2r+1)r$.

In summary, the number f of the faulty node in each neighborhood should meet:

(1) $(2r+1)((2-\beta)r+1) - f \geq f+1$
(2) $f < 2\beta r(\beta r+1)+1$
(3) $f < (2r+1)r$

Actually, when the $\beta = 0.8$, we get the maximum $f < \frac{1}{2}(2r+1)(1.2r+1)$. Notice that the node P should located on the grid. We make $\beta = 1$ conservatively and get $f < \frac{1}{2}(2r+1)(r+1)$ because the node P can not located on the grid when $r \bmod 5 \neq 0$. Therefore, in our protocol, each single neighborhood should tolerate $f < \frac{1}{2}(2r+1)(r+1)$ faulty nodes.



(a) The more important nodes. (b) Shared neighbors. (c) Neighbors in the right direction.

🟦 The neighbors of the node $\mathcal{N}_{(a,b)}$. ◇ The more important nodes. 🟩 The neighbors of the node $\mathcal{N}_P$.
🟥 The neighbors in the right direction of $\mathcal{N}_{(a,b)}$.

Fig. 8. The L_∞ metric in an infinite grid wireless network.

B DERIVATION OF $2r(r+1)+1$

The yellow area in Figure 4a locates $2r(r+1)+1$ nodes. Specifically, it locates

$$1+3+5+\cdots+(2r-1)+(2r+1)+(2r-1)+\cdots+5+3+1$$
$$= 2[1+3+5+\cdots+(2r-1)]+(2r+1)$$
$$= 2r^2+(2r+1) = 2r(r+1)+1$$

nodes.

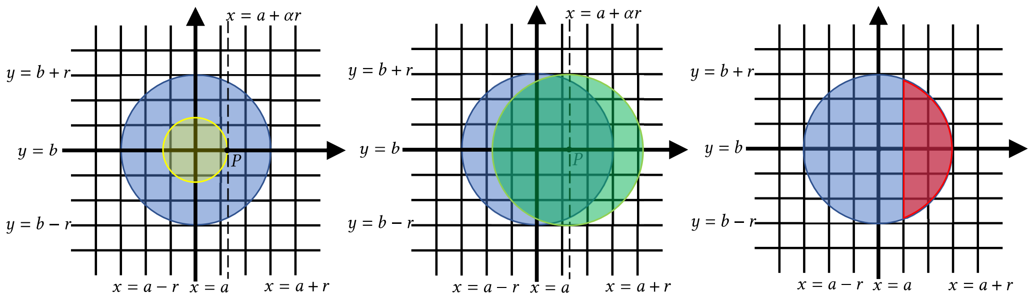
C DERIVATION OF $f < \lfloor 0.3\pi r^2 \rfloor$

The intuition of this derivation is that "a valid certificate will be *constructed*, and *received by all correct nodes*".

Among the nodes within the yellow area in Figure 9a, which locates $\pi(\alpha r)^2$ nodes, the node P has the least shared neighbors with the source node (i.e., the dark green area in Figure 9b, which locates $2r^2 cos^{-1}\frac{\alpha}{2} - \frac{1}{2}r^2\alpha\sqrt{4-\alpha^2}$ nodes). All neighbors of the node $\mathcal{N}_{(a,b)}$ in the right are within the red area in Figure 9c, which locates $0.5\pi r^2 - 2r - 1$ nodes. Similar to the derivation in Appendix A, the number f of the faulty nodes in each neighborhood should meet:

(1) $f < \pi(\alpha r)^2$
(2) $2f + 1 \leq 2r^2 cos^{-1}\frac{\alpha}{2} - \frac{1}{2}r^2\alpha\sqrt{4-\alpha^2}$
(3) $f < 0.5\pi r^2 - 2r - 1$

When the $\alpha = \sqrt{0.3}$, we get the maximum $f < 0.3\pi r^2$. Therefore, in our protocol, each single neighborhood should tolerate $f < 0.3\pi r^2$ faulty nodes. We do not select a strictly α to allow the node P to locate on a grid because the bound in this metric is approximate (but it is increasingly accurate for large r).



(a) The more important nodes. (b) Shared neighbors. (c) Neighbors in the right direction.

🔵 The neighbors of the node $\mathcal{N}_{(a,b)}$. 🟡 The more important nodes. 🟢 The neighbors of the node $\mathcal{N}_P$.
🔴 The neighbors of the node $\mathcal{N}_{(a,b)}$ in the right direction.

Fig. 9. The L_2 metric in an infinite grid wireless network.

D DERIVATION OF $2r^2 \cos^{-1}\frac{\sqrt{0.3}}{2} - \frac{1}{2}r^2\sqrt{1.11}$

Let two circles of radius R and r and centered at $(0,0)$ and $(d,0)$ intersect in a region shaped like an asymmetric lens. The area of this lens is[3]:

$$r^2cos^{-1}\left(\frac{d^2+r^2-R^2}{2dr}\right) + R^2cos^{-1}\left(\frac{d^2+R^2-r^2}{2dR}\right)$$
$$-\frac{1}{2}\sqrt{(-d+r+R)(d+r-R)(d-r+R)(d+r+R)}$$

Thus, we can calculate the area of the dark green area in Figure 5b by making $R = r$ and $d = \sqrt{0.3}r$, and get the area:

$$2r^2\cos^{-1}\frac{\sqrt{0.3}}{2} - \frac{1}{2}r^2\sqrt{1.11}$$

E DERIVATION OF $f < \lfloor\frac{1}{2}(r+1)^2\rfloor$

The intuition of this derivation is that "a valid certificate will be *constructed*, and *received by all correct nodes*". In the finite grid, there exist corner nodes that have the least neighbors. Thus, we consider the worst case, i.e., when a corner node becomes a source node, to decide the f.

[3]https://mathworld.wolfram.com/Circle-CircleIntersection.html

We first consider how the neighborhood of a corner source node ensures "a valid certificate will be *constructed*". As shown in Figure 6a, all neighbors of the corner source node are within the yellow area, which locates $(r+1)^2$ nodes. Interestingly, they are all within each other's broadcasting range, which means that they have the same number (i.e., $(r+1)^2$) of shared neighbors with the source node. Similar to the derivation in Appendix A, the number f of the faulty nodes in the neighborhood of a source node should meet:

(1) $f < (r+1)^2$
(2) $2f + 1 \leq (r+1)^2$

Then, we consider how every neighborhood ensures that "a valid certificate will be *received by all correct nodes*". As shown in Figure 6b, all neighbors of the node $\mathcal{N}_{(a,b)}$ in the right are within the red area, which locates $(r+1)r$ nodes. Similar to the derivation in Appendix A, the number f of the faulty nodes in each neighborhood should meet:

(1) $f < (r+1)r$

In summary, the number f of the faulty node in each neighborhood should meet:

(1) $f < (r+1)^2$
(2) $2f + 1 \leq (r+1)^2$
(3) $f < (r+1)r$

Then, we get the maximum $f < \frac{1}{2}(r+1)^2$. Therefore, in our protocol, each single neighborhood should tolerate $f < \frac{1}{2}(r+1)^2$ faulty nodes.

F DERIVATION OF $F \leq \lceil \frac{p}{2} \rceil \lceil \frac{q}{2} \rceil f$

Recall that each neighborhood tolerates at most f faulty nodes. The $p \cdot r \times q \cdot r$ finite grid consists of $\lceil \frac{p}{2} \rceil \lceil \frac{q}{2} \rceil$ neighborhoods, including incomplete neighborhoods on the periphery of the system. Thus, it can tolerate $\lceil \frac{p}{2} \rceil \lceil \frac{q}{2} \rceil f$ faulty nodes, i.e.,$F \leq \lceil \frac{p}{2} \rceil \lceil \frac{q}{2} \rceil f$.

G CORRECTNESS OF OUR BYZANTINE CONSENSUS PROTOCOL

We first prove the validity.

LEMMA G.1. *After GST, if the predefined leader in round γ is correct, all correct nodes will commit this leader proposal.*

PROOF. After GST, the network becomes synchronous, where all correct nodes can *r_deliver* propose messages from others in a known finite time. The exact time is related to the size of the $N \times M$ finite grid wireless network. In this proof, we assume the time bound is Δ. Specifically, if a correct replica *r_bcast*s or *r_deliver*s a message at time t, all correct replicas *r_deliver* it by time $t + \Delta$. Then, we prove this lemma to have all correct nodes to set their propose-timer to 2Δ.

First, we prove that all correct nodes enter the round γ and *r_bcast* proposals within Δ. We suppose the fastest node $\mathcal{N}_i$ *r_bcast*s a proposal in round γ at time t, which implies that the $\text{DAG}_i[\gamma - 1] \geq N - F$ at time t. As a result, all correct nodes can *r_deliver* these proposals corresponding to the $\text{DAG}_i[\gamma - 1]$ and all their dependencies by time $t + \Delta$, thus adding them to the local $\text{DAG}_*[\gamma - 1]$. Then, the most sluggish nodes can enter the round $\gamma - 1$ and *r_bcast* proposals because they meet the Jumping Propose rule. Then, they can immediately propose new proposals in round γ by time $t + \Delta$ because they meet the Propose rule (i.e., they are in round $\gamma - 1$, $|\text{DAG}_*[\gamma - 1]| \geq N - F$ and $(\gamma - 1) \bmod 2 = 0$). Therefore, all correct nodes enter the round γ and *r_bcast* proposals within Δ.

Next, we show that all proposals in round $\gamma + 1$ from correct nodes point to the leader proposal in round γ. Recall that each correct node enters round γ and *r_bcast*s proposals within Δ. As a result, after entering the round γ, each of them can *r_deliver* proposals from each other in the round γ within 2Δ (i.e., before the timeout of propose-timer). Before entering the round $\gamma + 1$, the $\mathrm{DAG}_*[\gamma]$ must contain the leader proposal. Thus, when they enter the round $\gamma + 1$ and *r_bcast* proposals, their proposals must point to the leader proposal in round γ.

Finally, we prove that all correct nodes can commit the leader proposal in round γ. When $|\mathrm{DAG}_*[\gamma + 1]| \geq N - F$, there are at least $N - 2F \geq F + 1$ proposals that from the correct nodes. Then, at least $F + 1$ proposals in $\mathrm{DAG}_*[\gamma + 1]$ point to the leader proposal in round γ, which meets the commit rule. Thus, all correct nodes can commit the leader proposal in round γ.

Therefore, after *GST*, all correct nodes can commit the leader proposal in round γ when this leader is correct.

□

Theorem G.2 (Validity). *If a correct node $\mathcal{N}_i$ calls a_bcast(b, γ), then all correct nodes will output a_deliver$(b, \gamma, \mathcal{N}_i)$.*

Proof. If a correct node $\mathcal{N}_i$ calls *a_bcast*(b, γ), it reliably broadcasts a proposal M with b to others. By the validity of the reliable broadcast, the proposal M will eventually be added to the $\mathrm{DAG}[\gamma]$. After that, all correct nodes create the new proposals that must point to the M. By Lemma G.1, after *GST*, all correct nodes will commit the leader proposal in round γ' when this leader is correct. Recall that when nodes commit a proposal, they also commit its causal histories, including the M. Thus, all correct nodes will output *a_deliver*$(b, \gamma, \mathcal{N}_i)$. □

Then, we prove the agreement.

Lemma G.3. *If a correct node $\mathcal{N}_i$ directly commits a leader proposal M in round γ, all future committed leader proposals have paths to the M.*

Proof. Suppose a node $\mathcal{N}_i$ directly commits M in round γ, which requires at least $F+1$ proposals in $\mathrm{DAG}_i[\gamma+1]$ (denotes V) point to the M. Then, we prove that all future committed leader proposals have paths to at least one proposal in the V. Recall that each proposal points to at least $N - F$ proposals in the previous round. Then, each proposal from correct nodes in the round $\gamma + 2$ points to at least one proposal in V because any quorum of $N - F$ proposals intersects with any quorum of $F + 1$ proposals. When $|\mathrm{DAG}[\gamma + 2]| \geq N - F$, it must include proposals from the correct nodes in the round $\gamma + 2$. As a result, the proposals from correct nodes in the round $\gamma + 3$ must have paths to at least one proposal in V because they point to the proposals in $\mathrm{DAG}[\gamma + 2]$. For the same reason, all proposals in the round $\gamma' > \gamma + 3$, have paths to at least one proposal in V. Thus, all future committed leader proposals have paths to the leader proposal M. □

Theorem G.4 (Agreement). *If a correct node $\mathcal{N}_i$ outputs a_deliver$(b, \gamma, \mathcal{N}_j)$, then all other correct nodes will output a_deliver$(b, \gamma, \mathcal{N}_j)$.*

Proof. Suppose $\mathcal{N}_i$ outputs *a_deliver*$(b, \gamma, \mathcal{N}_j)$ by committing the proposal M. In more detail, the M is in the causal histories of the M' in the round $\gamma' > \gamma$, and $F + 1$ proposals in $\mathrm{DAG}_i[\gamma' + 1]$ points to the M'. By Lemma G.1, after *GST*, all correct nodes will commit a leader proposal M'' in round $\gamma'' > \gamma'$ when this leader is correct. By Lemma G.3, the M'' has a path to the M'. Recall that before committing M'', they first commit the M'. By the agreement property of reliable broadcast, all correct nodes *r_deliver* the same proposal in a round from the same sender. Thus, when they commit the causal histories of M' including the M, they output *a_deliver*$(b, \gamma, \mathcal{N}_j)$. Therefore, all other correct nodes will output *a_deliver*$(b, \gamma, \mathcal{N}_j)$. □

Next, we prove the total order.

Theorem G.5 (Total order). *If a correct node $\mathcal{N}_i$ outputs $a\_deliver(b, \gamma, \mathcal{N}_j)$ before outputting $a\_deliver(b', \gamma', \mathcal{N}_{j'})$, then every correct node must output $a\_deliver(b, \gamma, \mathcal{N}_j)$ before outputting $a\_deliver(b', \gamma', \mathcal{N}_{j'})$.*

Proof. In our protocol, before committing a leader proposal, each node first commits the uncommitted leader proposals in its causal histories in the order of the round number. Thus, each node commits the leader proposals in the order of the round number. When committing a leader proposal, each node commits its causal histories by a deterministic rule. By the agreement property of the reliable broadcast, the leader proposals in a round across different nodes have the same causal histories. Thus, all correct nodes *a_deliver* proposals (blocks) in the same order. □