Reliable Broadcast in Critical Applications: Asset Transfer and Smart Home

Yingjian Wu[1], Yicheng Shen[2], Haochen Pan[3], Lewis Tseng[2], Moayad Aloqaily[4]

[1]University of California San Diego, CA, USA
[2]Boston College, MA, USA
[3]The University of Chicago, IL, USA
[4]Mohamed Bin Zayed University of Artificial Intelligence (MBZUAI), UAE
E-mails: yiw079@ucsd.edu, {shenvw; lewis.tseng}@bc.edu, haochenpan@uchicago.edu, maloqaily@ieee.org

***Abstract*—Asynchronous Byzantine reliable broadcast receives renewed attention recently, as it is fundamental to many fault-tolerant critical applications. This paper focuses on the Byzantine Reliable Broadcast protocol, which was first proposed by Bracha in 1987. Several recent protocols have improved the round and bit complexity of these algorithms. Motivated by practical network constraints in modern applications, this paper revisits the problem and reduces both complexity in communication and local computation. State-of-the-arts protocols are evaluated using the developed framework that simulates realistic bandwidth constraints. The evaluation demonstrates that our protocols, which use *cryptographic hash functions* and *erasure coding* in a novel way, have superior performance in critical applications such as asset transfer and smart home.**

***Index Terms*—reliable broadcast, impossibility, evaluation**

I. Introduction

Reliable broadcast (RB) ensures that a message from a source node is delivered to all the fault-free nodes in a cluster/system that is asynchronous and potentially susceptible to node failures. Compared to consensus protocols [1] which satisfies strong properties, reliable broadcast is more flexible, but still robust enough for many fault-tolerant systems. RB is an important primitive for critical applications, such as group mailing lists and payment systems. Many other works have also used RB to improve energy reduction, resilience, and speed in networked devices [2] [3]. In this paper, we particularly focus on the application of asset transfer and smart home applications. We envision that our algorithms can also assist Byzantine Machine Learning (BML) in industry 5.0 [4] due to its practicality for edge devices.

In recent years, the amount of data has increased at an unprecedented speed. Despite the downward impact of COVID-19 on many industries, data generation increased to 64.2ZB in 2020 according to IDC[1]. Many data-centric applications confront new challenges caused by a large amount of data. Moreover, this trend is expected to continue into the future due to the introduction and flourishing of Internet of Things (IoT) and cryptocurrencies. Although network service providers and data centers are constantly upgrading their hardware to meet the increasing demand, bandwidth is still a constraint for critical applications. It is worth noting that data centers, especially wide area network (WAN), require high bandwidth and high link quality, since many services are delay-sensitive; IoT which relies on wireless network is also sensitive to bandwidth since data generated from IoT is becoming increasingly larger and most IoT applications have real-time constraints.

Asset transfer refers to the online settlement of various assets, not just limited to funds or money. In the case of payment systems of both traditional currencies and cryptocurrencies, transfers need to guarantee correctness regardless of failures, including crash and Byzantine faults [5]. IoT devices are equipped with chips and sensors which provide possessing computing and communication capabilities. Both applications require messages to be reliably broadcast.

In 1987, Bracha introduced the notion of RB and used it to solve Byzantine consensus [6]. Subsequently, many Byzantine-tolerant RB (or simply Byzantine RB) protocols have been proposed which improved several different dimensions such as complexity in number of rounds, and number of bits exchanged. Table I summarizes the key related work and compared it to the proposed protocols. Most RB protocols [6] assume unlimited bandwidth and choose flooding-based designs. Recent RB and Byzantine consensus protocols (e.g., [7]–[9]) with proven optimal bit complexity usually incur higher round complexity and/or local computation. None of these RB protocols are designed for a practical and modern setting where the network's bandwidth is constrained and nodes have only limited computation power.

In this paper, we rely on *cryptographic hash functions* [12] and *erasure coding* [13] to devise more bandwidth-friendly Byzantine RB protocols. To evaluate our protocols, we design and implement a general testing platform on top of [14], [15], which allows us to specify practical constraints on network quality, bandwidth and node's computation power.

Theoretically speaking, if an algorithm assumes a perfect cryptographic hash function [16] for ensuring correctness, then the algorithm cannot be claimed error-free. This is because that such an algorithm cannot prevent from the adversary with *unlimited* computation power. However, this paper is targeting practical systems, and as we have observed in many real-world systems, cryptographic hash functions are useful, e.g., Bitcoin [17] and PBFT [18]. Our EC-based algorithms rely on $[n, k]$ MDS erasure codes. The bit complexity analysis assumes that

[1]https://www.idc.com/getdoc.jsp?containerId=prUS47560321

TABLE I:
Comparison between our algorithms and prior works. The proposed algorithms are H-BRB, and EC-BRB.

Algorithm	Bit complexity	System Size (Resilience)	Round Complexity (Common Case)	Error -free	MDS codes	Bottleneck
CRB [10]	$O(n^2L)$	$\geq f+1$	1	Yes	No	-
Bracha RB [6]	$O(n^2L)$	$\geq 3f+1$	3	Yes	No	Flooding
Raynal RB [11]	$O(n^2L)$	$\geq 3f+1$	2	Yes	No	Flooding
Patra RB [8]	$O(nL)$	$\geq 3f+1$	9	Yes	Yes	Polynomial time local computation with large constants
Nayak et al. [9]	$O(nL)$	$\geq 3f+1$	10	Yes	Yes	Polynomial time local computation with large constants
H-BRB[3f+1]	$O(nL)+O(nfL)$	$\geq 3f+1$	3	No	No	Hash Function
H-BRB[5f+1]	$O(nL)+O(nfL)$	$\geq 5f+1$	2	No	No	Hash Function
EC-BRB[3f+1]	$O(nL)+O(nfL)$	$\geq 3f+1$	3	No	Yes	Hash Function + MDS code
EC-BRB[4f+1]	$O(nL)+O(nfL)$	$\geq 4f+1$	3	No	Yes	Hash Function + MDS code

L is sufficiently large. For our algorithms, L is assumed to be at least n^2, same as the state-of-arts protocols [9].

The bit complexity of our RB protocol is $O(nL+nfL)$: (i) when the source is fault-free, the nodes only need to exchange $O(nL)$ bits; and (ii) when source is faulty (which is uncommon case in practical applications), our RB protocols need $O(fL)$ extra bits per node for recovering the original message. Such a recovery design is appropriate for practical applications. This is because f is usually assumed to be small in real-world cases. Our erasure-coding-based RB protocols have another advantage: a fault-free source only needs to transmit $O(nL/k)$ bit to another node. As we can see in the asset transfer application, this gives it a clear performance advantage. The round complexity assumes the common case when the source is fault-free.

The main contributions of this paper are: We devise a family of algorithms that rely on hash function and erasure coding to improve three dimensions: bit complexity, round complexity, and computation complexity. (i) ***H-BRB***, a Byzantine-tolerant hash-based RB; and (iii) ***EC-BRB***, a Byzantine-tolerant RB that uses erasure coding.

The rest of the paper is organized as follows. In Section II, we introduce preliminaries. In section III, we discuss Byzantine reliable broadcast. In section IV, we present the evaluations, and conclude in Section VI.

II. Preliminaries

The system is modeled as a static asynchronous fully connected message-passing system composed of n nodes. We assume $\leq f$ nodes, including the source, may be *Byzantine* faulty. Nodes are fully connected by asynchronous, authenticated and reliable point-to-point channels [10], [19]. "Asynchronous" means that nodes do not know the wall-clock time or global clock, and messages maybe be delayed arbitrarily. Reliable channels assume that (i) messages are eventually delivered with fault-free sender and receiver, and (ii) a fault-free receiver receives a message iff a sender sent the message. Authenticated channel prevents from Sybil attacks.

A Byzantine node may choose to behave arbitrarily, and may *equivocate*, i.e., send arbitrary messages to different sets of nodes. We also call a Byzantine node as a faulty node. If a node does not have such behavior, then we call it fault-free.

To facilitate the discussion, we introduce the following notations. Every message m sent by a fault-free source s is associated with a sequence number or index h. Thus m can be uniquely indexed through a tuple (s,h). In all of our algorithms, we use $\texttt{SetMsg}_i[s,h]$ to denote the set of messages that the node i collects, in which are candidates that can be identified with (s,h). When the context is clear, we omit the subscript i. We use $\texttt{Count}[*]$ to denote a local counter of certain type of messages that is initialized to 0. We use $\mathbb{H}(*)$ to denote the cryptographic hash function.

What is Reliable Broadcast?

The reliable broadcast properties adopted in this paper are from [6], [20]. The source s calls "Reliable-Broadcast(m,h)" to broadcast a message m with sequence number h reliably, which then executes the RB protocol by exchanging messages with other nodes. For each fault-free node, under suitable conditions, the RB protocol notifies the application layer so that "Reliable-Accept(m',h)" is invoked, and (m',h) is delivered. A correct RB protocol needs to satisfy these properties:

Property 1 (Termination)**.** • *If a fault-free source s performs Reliable-Broadcast(m,h), with a message m having index h then all fault-free nodes will eventually Reliable-Accept(s,m,h).*

- *If a fault-free node performs Reliable-Accept(s, m, h), then all fault-free nodes eventually perform Reliable-Accept(s, m, h).*

Property 2 (Validity)**.** *If a fault-free source s does not perform Reliable-Broadcast(m, h) then no fault-free node will ever perform Reliable-Accept(s, m, h).*

Property 3 (Agreement)**.** *If a fault-free node performs Reliable-Accept(s, m, h) and another fault-free node will eventually perform Reliable-Accept(s, m', h) then $m = m'$.*

Property 4 (Integrity)**.** *A fault-free node reliably accepts at most one message of index h from a source s.*

III. Byzantine Reliable Broadcast

This section considers Byzantine-tolerant Reliable Broadcast protocols. Our H-BRB algorithms can be viewed as a simplified version of PBFT [18] or an extension of Bracha's RB protocol [6] with the usage of cartographic hash; hence, is not presented here due to page constraints. To reduce communication complexity, H-BRB[3f+1] relies on the usage of cryptographic hash function to check whether a message received by non-sources are the same. The name contains"3f+1", because it requires $n \geq 3f+1$. We also design H-BRB[5f+1], which works given $n \geq 5f+1$. H-BRB[5f+1] has less number of rounds and messages, compared to H-BRB[3f+1].

H-BRB is bottle-necked at the source, because similar to PBFT [18], the source still needs to send $O(nL)$ bits. This section presents two ideas of using erasure coding to reduce the number of bits that need to be sent by the source. In our EC-based protocols, the source only needs to transmit a small coded element.

A. MDS Erasure Code: Preliminaries

Our algorithms use a linear $[n, k]$ MDS (Maximum Distance Separable) erasure code [13] over a finite field $\mathbb{F}_q$ to encode the message m. An $[n, k]$ MDS erasure code has the property that any k out of the n coded elements, computed by encoding m, can be used to recover (decode) the original message m.

For encoding, m is divided into k elements $m_1, m_2, \ldots, m_k$ with each element having size L/k (assuming size of m is L). The encoder takes the k elements as input and produces n coded elements $c_1, c_2, \ldots, c_n$ as output, i.e., $[c_1, \ldots, c_n] = ENC([m_1, \ldots, m_k])$, where ENC denotes the encoder. For brevity, we simply use $ENC(m)$ to represent $[c_1, \ldots, c_n]$.

The vector $[c_1, \ldots, c_n]$ is referred to as the *codeword* corresponding to the message m. Each coded element c_i also has size $\frac{L}{k}$. In our algorithms, the source disseminates a **single** coded element to each node. We use ENC_i to denote the projection of ENC on to the i^{th} output component, i.e., $c_i = ENC_i(v)$. Without loss of generality, the coded element c_i is associated with node i, for $1 \leq i \leq n$.

B. EC-BRB[3f+1]

The first algorithm, EC-BRB[3f+1], is based on H-BRB[3f+1], in which each node i not only forwards $\mathbb{H}(m)$, but also a coded element c_i. This design reduces bit complexity. EC-BRB[3f+1] uses $[n, f+1]$ MDS erasure code, and does not rely on detection or correction capability. That is, the decoder function DEC can correctly decode the original message if the input contains at least $f+1$ *uncorrupted* coded elements. EC-BRB[3f+1] does not need the correction/detection, since each node directly uses $\mathbb{H}(m)$ to verify the validity of the decoded message.

The pseudo-code of EC-BRB[3f+1] is presented in Algorithm 1, 2, and 3. Note that Line 13 in Algorithm 2 requires exponential computation. This is because that each

Algorithm 1 EC-BRB[3f+1]: source s on msg m, index h

1: **function** RELIABLE-BROADCAST(m, h)
2: $\{c_1, c_2, \ldots, c_n\} = \text{ENC}(m)$
3: SEND(MSG-TAG, $s, \mathbb{H}(m), c_k, h$) to node k

Algorithm 2 EC-BRB[3f+1]: i's event handler

1: **function** RECEIVING(MSG-TAG, s, H, c, h)
2: **if** $j = s$ **and** first (MSG-TAG, $s, *, *, h$) **then**
3: $\texttt{SetCode}[s, H, h] \leftarrow \texttt{SetCode}[s, h, H] \cup \{c\}$
4: $\texttt{Count}[\text{ECHO-TAG}, s, H, h] + +$
5: **if** never sent (ECHO-TAG, $s, *, h$) **then**
6: SEND(ECHO-TAG, s, H, c, h) to all nodes
7: **function** RECEIVING(ECHO-TAG, s, H, c, h)
8: **if** first (ECHO-TAG, $s, *, *, h$) from j **then**
9: $\texttt{Count}[\text{ECHO-TAG}, s, H, h] + +$
10: $\texttt{SetCode}[s, H, h] \leftarrow \texttt{SetCode}[s, h, H] \cup \{c\}$
11: **if** $\texttt{Count}[\text{ECHO-TAG}, s, H, h] \geq f+1$ **then**
12: **if** $\nexists m \in \texttt{SetMsg}[s, h]$ s.t. $\mathbb{H}(m) = H$ **then**
13: **for** each $C \subseteq \texttt{SetCode}[s, H, j]$, $|C| = f+1$ **do**
14: $m \leftarrow DEC(C)$
15: **if** $\mathbb{H}(m) = H$ **then**
16: $\texttt{SetMsg}[\text{s, h}] \leftarrow \texttt{SetMsg}[\text{s, h}] \cup \{m\}$
17: CHECK(s, H, h)
18: **function** RECEIVING(ACC-TAG, s, H, h)
19: **if** first (ACC-TAG, $s, *, h$) from j **then**
20: $\texttt{Count}[\text{ACC-TAG}, s, H, h] + +$
21: **if** $\texttt{Count}[\text{ACC-TAG}, s, H, h] \geq f+1$ **then**
22: **if** $\nexists m' \in \texttt{SetMsg}[s, h]$ s.t. $\mathbb{H}(m') = H$ **then**
23: SEND (REQ-TAG, s, H, h) to nodes if have not sent (REQ-TAG, s, H, h) to them before
24: CHECK(j, H, h)
25: **function** RECEIVING(REQ-TAG, s, H, h)
26: **if** first (REQ-TAG, s, h) from j **then**
27: **if** $\exists m' \in \texttt{SetMsg}[s, h]$ s.t. $\mathbb{H}(m') = H$ **then**
28: SEND (FWD-TAG, s, m', h) to j
29: **function** RECEIVING(FWD-TAG, s, m, h)
30: **if** have sent (REQ-TAG, $s, \mathbb{H}(m), h$) to j **then**
31: **if** first (FWD-TAG, s, m, h) from j **then**
32: $\texttt{SetMsg}[s, h] \leftarrow \texttt{SetMsg}[s, h] \cup \{m\}$
33: CHECK$(s, \mathbb{H}(m), h)$

Algorithm 3 EC-BRB[3f+1]: helper sub-routine

```
1: function CHECK(s, H, h)
2:     if m ∈ SetMsg[s, h] s.t. ℍ(m) = H then
3:         if Count[ECHO-TAG, s, H, h] ≥ f + 1 then
4:             if never sent (ECHO-TAG, s, ∗, ∗, h) then
5:                 {c_1, ..., c_n} ← ENC(m)
6:                 SEND (ECHO-TAG, s, H, c_i, h) to all nodes
7:         if Count[ECHO-TAG, s, H, h] ≥ n − f then
8:             if never sent (ACC-TAG, s, ∗, h) then
9:                 SEND (ACC-TAG, s, H, h) to all nodes
10:        if Count[ACC-TAG, s, H, h] ≥ f + 1 then
11:            if never sent (ACC-TAG, s, ∗, h) then
12:                SEND (ACC-TAG, s, H, h) to all nodes
13:        if Count[ACC-TAG, s, H, h] ≥ n − f then
14:            RELIABLE-ACCEPT(s, m, h)
```

node needs to identify correct $f+1$ coded elements to decode and recover the original message. This requires $O(\binom{n}{f+1})$ computation complexity. With a small f, the local computation is negligible. We improve the scalability in the next Byzantine RB protocol.

C. EC-BRB[4f+1]

To address the scalability issue, we rely on the correction capability of MDS code. By doing so, the resilience has to be reduced in the sense that our new algorithm requires $n \geq 4f+1$. This trade-off turns out is necessary, as identified in our technical report.

Error-correction in MDS Codes: We use $[n, k]$ MDS code where $k = n - 3f$. The distance between different codewords is $d = n - k + 1 = 3f + 1$. The correctness of our algorithm use the following theorems from coding theory.

Theorem 1. *The decoder function DEC can correctly decode the original message if the input contains at least $n-f$ coded elements. Among the elements used, up to f may be erroneous.*

Theorem 2. *Assume $n \geq 4f+1$. Consider codeword $C = \{c_1, c_2, \ldots, c_n\}$ and codeword $C' = \{c'_1, c'_2, \ldots, c'_n\}$ such that (i) C has at most f erasures, (ii) C' has at most f erasures,[2] and (iii) at most f of the remaining coded elements are different between the two codewords. If $DEC(C) = m$, then $DEC(C')$ either returns m or detects an error.*

Theorem 2 needs $n \leq 4f$, because by construction, each pair of codewords has distance $3f+1$. Even with the Byzantine source, it is possible to find a scenario that $DEC(C) = m$ and $DEC(C') = m'$ for $m' \neq m$ when $n \leq 4f$.

EC-BRB[4f+1]: Algorithm: The structure is similar to EC-BRB[4f+1]; hence, the pseudo-code is omitted due to page constraints. The key characteristic of EC-BRB[4f+1] is that upon receiving the ECHO-TAG messages, each node directly uses decoder function to recover the original message m. If the source is fault-free, then the error-correcting feature of MDS code trivially handles the corrupted coded element forwarded by Byzantine nodes. Therefore, unlike EC-BRB[3f+1], nodes do not perform the exponential computation. One key design is to have codeword distance at least $3f$. This allows all fault-free nodes to correctly construct a message, even if a faulty source colludes with other faulty non-source nodes.

In addition to transmitting coded elements, the source also uses Bracha's RB [6] to reliably broadcast $\mathbb{H}(m)$ in parallel. This ensures that even with a faulty source, fault-free nodes must decode the same value; otherwise, the $\mathbb{H}(m)$ would not match. Since $\mathbb{H}(m)$ can be viewed as a constant for a large enough message, this design has no impact on the bit complexity.

IV. EVALUATION

We evaluate the performance of RB protocols through simulations over realistic settings with the focus on two critical applications: asset transfer and smart home. We first implement a configurable and extensible benchmarking platform, *Reliability Mininet Benchmark (RMB)*, for distributed reliability protocols over asynchronous message-passing networks.

RMB is built on top of Mininet [14], [15]. It is integrated with a workload generator that generates reliable broadcast requests. The RMB framework is implemented using Go. Python scripts are provided to configure and launch RMB.

Network parameters can be configured within YAML files. Benchmark managers and protocols are invoked by a Python script after the network is initialized. Advanced users can also configure a topology and try various link parameters. A key benefit of RMB is to free user from tedious work of configuring environments (regarding networks and computation power) and faulty behaviors.

We use a virtual machine (VM) on Google Cloud Platform (GCP). The VM is a custom N2 instance with 96 vCPUs and 86.5 GB memory. The VM runs Ubuntu 18.04 LTS and is located in us-east1-b. We evaluate the RB algorithms in two different scenarios: a payment system for asset transfer and a setting of a smart home. We create simulated network topology with traffic controlled links. Constraints, such as delay, jitter, limited bandwidth, and loss, are set for links between nodes to simulate the real network conditions.

The numbers presented below do not include the result for Patra's algorithm [21] nor Nayak's algorithm [9]. Both of these algorithms have optimal complexity $O(nL)$ in the theoretical setting. However, we found them with sub-par performance in our target scenarios. The reason is Patra's algorithm induces high computation complexity. It requires roughly 400 ms to complete for a network of 32 nodes. On the other hand, algorithm in [9] has higher round complexity, which takes more than 500 ms to complete on average. Both algorithms are slower than other algorithms we have tested.

Scenario 1: Asset Transfer We use a three-layer topology to simulate the Border Gateway Protocol (BGP): the top layer is a core switch; the middle layer consists of edge switches; the bottom layer consists of nodes that are individually connected

[2]Erasures at C and C' may occur at different positions.

with the edge switches. In our evaluation, we set 1 node per edge switch and there are 10 nodes in total.

The message size is set to 0.1 MB. In payment systems, each message that corresponds to a payment is small, but batching is normally used for improved throughput. Therefore, our message size can be considered as the size of a batch of messages (or a block in a blockchain-based system). We send 2000 rounds of messages from the source node sequentially. That is, no pipelining is enabled, because we want to focus on the pure performance of the RB algorithms.

In a real setting, the agents, such as banks, might not be geographically close to each other. Due to long distances between agents, we need to consider the latency factor. We set a 50 ms delay and 25 ms jitter for the links. Moreover, the agents communicate through wired networks connected with cables, so we set the maximum bandwidth to be 50 Mbit/s and the loss to be 0.

Scenario 2: Smart Home We use a single switch topology to simulate the smart home setting. In this topology, a single switch is connected with multiple nodes, simulating the scenario where a router is connected with many IoT devices through Wi-Fi. We report the result of 40 nodes.

In smart home applications, other than special cases like video surveillance, most signals, such as control messages, temperature measurements, and images exchanged between devices are small. Therefore, we set the message size to 1 KB. Same as the previous scenario, we send 2000 rounds of messages from the source node. Although many current routers show high bandwidth from their specifications, we set the maximum bandwidth to 50 MB because bandwidth significantly suffer from longer-distance communication and obstacles (e.g., walls) blocking Wi-Fi signal. There also exists a small latency during communication. Thus, we set a 10 ms delay and a 3 ms jitter. Another common issue for wireless communication is the packet loss, say due to signal collision. Therefore, we test both 0% and 2% message loss rates.

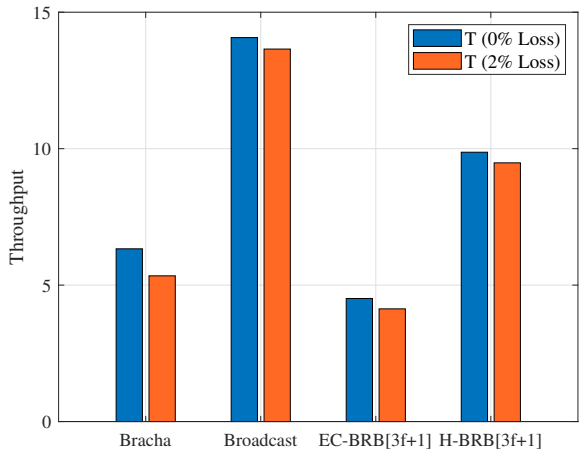

Fig. 1: Payment System (Asset Transfer)

Figure 1 presents the throughput of RB algorithms in the asset transfer setting. Broadcast is the non-fault-tolerant algorithm where the source simply multi-casts the messages to each node and waits for an acknowledgement from each node. EC-BRB[4f+1] has the best throughput, even higher than the non-fault-tolerant version. This is because the source only needs to send $O(L)$ bits in the common case, and the local computation is efficient.

Figure 2 reports the throughput for our BRB algorithms and Bracha's algorithm. Table II reports also the percentage drop in throughput due to higher message loss rate. As expected, H-BRB[3f+1] has the best performance, due to its more efficient computation and lower bit complexity. EC-BRB[3f+1] does not have great performance, but its % drop in throughput is much lower than Bracha's algorithm.

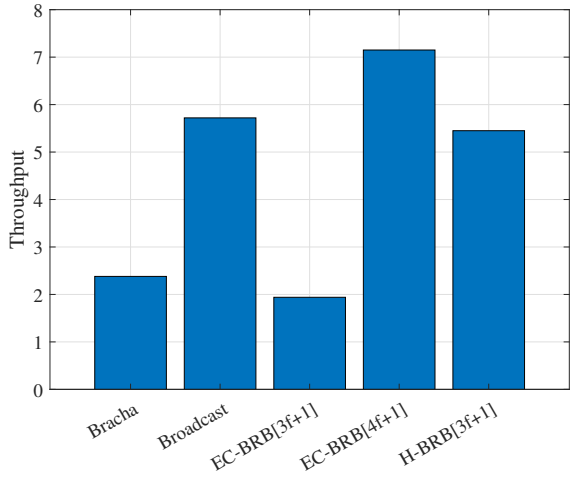

Fig. 2: Smart Home (IoT) Application

TABLE II: Smart Home (IoT) Application

Algorithm	T (0% Loss)	T (2% Loss)	% drop
H-BRB[3f+1]	9.87	9.48	3.95%
EC-BRB[3f+1]	4.51	4.13	8.43%
Broadcast	14.07	13.65	2.99%
Bracha	6.33	5.34	15.64%

V. Related Work

Byzantine Reliable broadcast has been well studied since 1980's [6]. Various properties have been considered, e.g., different topologies [22]–[25], probabilistic properties with logarithmic per-node communication and computation complexity [26], and Mobile ad hoc networks (MANETs) [27]. [10] presents comprehensive surveys on existing reliable broadcast protocols. Closest algorithms are compared in Table I.

Recently, reliable broadcast protocols have been made more efficiently [7], [8], [21], [28]. Liang and Vaidya [28] use erasure-coding for designing an error-free synchronous Byzantine agreement algorithm with optimal bit complexity. Liang

et al. [7] and Fitzi and Hirt [29] present Byzantine agreement and RB algorithms in synchronous settings. Choudhury [30] proposes Byzantine RB algorithm that requires only majority correctness, unlike $n \geq 3f + 1$ in prior works. Choudhury's algorithm achieves optimal bit complexity (i.e., $O(nL)$). The algorithm relies on the hardware to ensure that faulty nodes do not equivocate. Our algorithms work for asynchronous systems and do not assume hardware support.

VI. Conclusion

This paper presents a family of Byzantine reliable broadcast algorithms designed for bandwidth constrained networks. We experimentally demonstrate that our algorithms have superior performance over the state-of-arts algorithms using our evaluation platform RMB. Particularly, using RMB, we show that our algorithms perform very well in the settings of asset transfer and smart home applications.

Acknowledgements

Yingjian and Haochen worked on the project while affiliated with Boston College. Authors from Boston College were partially supported by National Science Foundation award CNS-1816487. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government. The authors would also like to acknowledge Saptaparni Kumar's comment on earlier version of this work.

References

[1] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. on Programming Languages and Systems*, 1982.

[2] Lewis Tseng and et. al. Reliable broadcast with trusted nodes: Energy reduction, resilience, and speed. *Computer Networks*, 182, 2020.

[3] Lewis Tseng, Yingjian Wu, Haochen Pan, Moayad Aloqaily, and Azzedine Boukerche. Reliable broadcast in networks with trusted nodes. In *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019.

[4] Anran Du, Yicheng Shen, Qinzi Zhang, Lewis Tseng, and Moayad Aloqaily. Cracau: Byzantine machine learning meets industrial edge computing in industry 5.0. *IEEE Transactions on Industrial Informatics*, 2021.

[5] Daniel Collins, Rachid Guerraoui, Jovan Komatovic, Matteo Monti, Athanasios Xygkis, Matej Pavlovic, Petr Kuznetsov, Yvonne-Anne Pignolet, Dragos-Adrian Seredinschi, and Andrei Tonkikh. Online payments by merely broadcasting messages (extended version), 2020.

[6] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, November 1987.

[7] Guanfeng Liang, Benjamin Sommer, and Nitin Vaidya. Experimental performance comparison of byzantine fault-tolerant protocols for data centers. In *2012 Proceedings IEEE INFOCOM*. IEEE, March 2012.

[8] Arpita Patra and C. Pandu Rangan. Communication optimal multi-valued asynchronous byzantine agreement with optimal resilience. In *Information Theoretic Security - 5th International Conference, ICITS 2011, Amsterdam, The Netherlands, May 21-24, 2011. Proceedings*, pages 206–226, 2011.

[9] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 28:1–28:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[10] Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems - An Algorithmic Approach*. Springer, 2018.

[11] Damien Imbs and Michel Raynal. Trading off t-resilience for efficiency in asynchronous byzantine reliable broadcast. *Parallel Processing Letters*, 26(04):1650017, 2016.

[12] Balaji Srinivasan Babu, M. Nikhil Krishnan, Myna Vajha, Vinayak Ramkumar, Birenjith Sasidharan, and P. Vijay Kumar. Erasure coding for distributed storage: An overview. *CoRR*, abs/1806.04437, 2018.

[13] W. C. Huffman and V. Pless. *Fundamentals of error-correcting codes*. Cambridge university press, 2003.

[14] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, New York, NY, USA, 2010. Association for Computing Machinery.

[15] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible network experiments using container-based emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, page 253–264, New York, NY, USA, 2012. Association for Computing Machinery.

[16] Bart Preneel. 1 cryptographic hash functions : An overview.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[18] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In Margo I. Seltzer and Paul J. Leach, editors, *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, pages 173–186. USENIX Association, 1999.

[19] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[20] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In Teruo Higashino, editor, *Principles of Distributed Systems*, pages 229–239, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[21] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, pages 34–49, 2011.

[22] Flaviu Cristian, Houtan Aghili, H. Raymond Strong, and Danny Dolev. Atomic broadcast: From simple message diffusion to byzantine agreement. *Inf. Comput.*, 118(1):158–179, 1995.

[23] Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. Reliable broadcast with respect to topology knowledge. *Distributed Computing*, 30(2):87–102, 2017.

[24] Lewis Tseng, Nitin H. Vaidya, and Vartika Bhandari. Broadcast using certified propagation algorithm in presence of byzantine faults. *Inf. Process. Lett.*, 115(4):512–514, 2015.

[25] Silvia Bonomi, Giovanni Farina, and Sébastien Tixeuil. Multi-hop byzantine reliable broadcast with honest dealer made practical. *Journal of the Brazilian Computer Society*, 25(1):9, Sep 2019.

[26] Rachid Guerraoui, Petr Kuznetsov, Matteo Monti, Matej Pavlovic, and Dragos-Adrian Seredinschi. Scalable byzantine reliable broadcast. In *33rd International Symposium on Distributed Computing, DISC 2019, October 14-18, 2019, Budapest, Hungary*, pages 22:1–22:16, 2019.

[27] Wei Lou and Jie Wu. Double-covered broadcast (DCB): A simple reliable broadcast algorithm in manets. In *Proceedings IEEE INFOCOM 2004, The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, March 7-11, 2004*, pages 2084–2095, 2004.

[28] Guanfeng Liang and Nitin H. Vaidya. Error-free multi-valued consensus with byzantine failures. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 11–20, 2011.

[29] Matthias Fitzi and Martin Hirt. Optimally efficient multi-valued byzantine agreement. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 163–168, 2006.

[30] Ashish Choudhury. Multi-valued asynchronous reliable broadcast with a strict honest majority. In *Proceedings of the 18th International Conference on Distributed Computing and Networking, Hyderabad, India, January 5-7, 2017*, page 1, 2017.