E2C— Energy Efficient Byzantine Fault Tolerance for Cyber-Physical Systems

Adithya Bhat[*] Manish Nagaraj[*] Michael K. Reiter[†] Saurabh Bagchi[*]
Aniket Kate[*]

June 21, 2020

Abstract

Motivated by CPS settings, this work explores energy efficiency of Byzantine fault-tolerant state machine replication (SMR) protocols. We develop a partially connected system model that leverages synchronous multicasts among neighbors. We generalize previous fault tolerance results in partial connectivity settings to this model, and propose a novel SMR protocol (E2C) for this model that is best-case optimal. We demonstrate the energy efficiency of E2C through experiments on CPS nodes and analysis. We extend the analysis to demonstrate the parameter ranges in which E2C will be more energy-efficient than relying on a trusted control node to overcome Byzantine sensor behavior.

Contents

[*]Purdue University
[†]UNC-Chapel Hill

1 Introduction

Cyber-Physical Systems (CPS) provide integrated computational, mechatronic and communication components that interact with each other and exhibit multiple modalities. CPS systems can vary from a network of temperature sensors deployed in power plants [33] to unmanned aerial vehicles or drones deployed to monitor traffic [34]. In many of these situations, the various nodes or devices of such systems need to agree on a global state such as the temperature to be maintained in the case of temperature sensors in power plants. An external monitoring agency must be able to both access this information and control these sensor nodes.

These CPS are often generalized to form a two-tier architecture [25,35]. The energy-constrained, lower-tier nodes such as drones or temperature sensors collect readings. These readings are employed in making a control decision, such as maintaining the operating temperature in a power plant; thus, the lower tier nodes must all maintain the same state that is a value or set of values. This forms a feedback control loop, with the upper tier nodes, such as server-class devices, being able to access this provenance information (such as an information log which can later be used for diagnostics and security [3]) that act as control centers.

A common threat to such systems is the presence of Byzantine nodes, which may communicate incorrect values, possibly colluding among multiple Byzantine nodes, to prevent the nodes in the system from maintaining the same value. Consensus algorithms in the presence of Byzantine faults, such as *SMR (State Machine Replication)* protocols [2, 7, 11, 16, 24, 36] ensure that the correct (or non-faulty) nodes agree on the same sequence of values in the presence of Byzantine nodes. These protocols also ensure that correct senders' values will be included in this sequence.

Naïvely employing a protocol that is not optimized for energy efficiency reduces the lifetime of the nodes taking part in the protocol. For example, the state-of-the-art SMR protocol Sync HotStuff [2] is not the most energy-efficient solution for SMR due to its high communication complexity and use of expensive cryptographic primitives, such as certificates. As we show in Section 5, it can be improved significantly to be more suitable for the CPS setting.

Although a single multicast is more energy-efficient than a group of unicasts used for communication in CPS settings, trivially substituting unicasts with multicasts in a Byzantine fault-tolerant SMR protocol alone is not sufficient for energy efficiency. This is because multicasts reduce the number of messages sent and hence the energy used in doing so, but complex cryptography to reduce the communication footprint can overwhelm those energy savings for e.g., BLE communication costs a few mJ of energy, while signing through RSA or ECDSA costs a few 100 mJs. Substituting complex cryptography with simpler primitives, however, increases the communication footprint in either the size of the messages sent or in their number. Hence, a direct application of a single technique such as multicast to existing works is insufficient to provide the most energy-efficient solution. This motivates protocols that make optimal use of multicasts and cryptographic primitives.

Contributions. This work makes the following main contributions.

1. We build a hypergraph model to take advantage of multicasts. We say that the system has k-casts, if the size of every hyper-edge is at least k. Then, each hyper-edge is called a k-cast. We show that in a system where every node has at least d_{in} incoming k-casts and at least d_{out} outgoing k-casts, the number of tolerable faults f satisfies $f < \min(d_{in}, d_{out}) + k - 1$. Sending k-casts instead of k unicasts to each neighbor improves the energy-efficiency of the protocol.

2. We present E2C, an energy-efficient best-case optimal leader based SMR protocol in this hypergraph model. In E2C, we employ energy-efficient cryptographic algorithms while ensuring security of the designed protocol. Motivated from Sync HotStuff [2], our protocol is best-case optimal since we do not use certificates and votes unless the current protocol leader is faulty.

3. We analyze the choices of cryptographic schemes along with parameters and offer options suitable for CPS settings. We show that in a setting with no trusted control nodes, we are always more energy-efficient than Sync HotStuff [2]. In settings where a trusted control node exists, we show how to arrive at feasible regions where E2C is more energy-efficient than a baseline protocol which leverages the trust from the control nodes.

4. We implement the best-case optimal E2C protocol and measure the energy consumption in the various stages of the protocol and show agreement with that estimated by our theoretical analysis.

5. We model the energy cost of protocols in terms of system parameters such as number of nodes n, message size m, etc. We use this to show the need for best-case optimal protocols. We also extend this technique to aid protocol design decisions by providing a comparative energy analysis, i.e., the number of faults a protocol A can tolerate while being more energy-efficient than protocol B. Concretely, we derive a bound for energy analysis for our proposed protocol E2C with respect to a baseline for the two-tiered CPS model when applicable.

Paper Organization. The rest of the paper is organized as follows. In Section 2, we go over the problem statement and some of the approaches to solve it. In Section 3, we describe the notation and some other preliminary materials. In Section 4, we provide a hypergraph model and restate the standard connectivity result for fault tolerance in this model. In Section 5, we describe E2C protocol and prove security of the protocol. In Section 6 we describe our experiments to compare and analyze E2C and the baseline. In Section 7, we analyze protocols and show the need to make protocols best case optimal. We also derive energy bounds, apart from the standard bounds. In Section 8 we describe works that are related to our work and describe how we differ from them. In Section 9 we conclude and point to possible future works.

2 Problem Statement and Solution Approaches

2.1 Two-Tiered CPS System Model

In typical CPS settings, the nodes are classified into two types: Tier 1 and Tier 2 nodes, based on the resources available to the nodes.

For our setting, (as shown in Figure 1) we assume the existence of one Tier 1 node (a server-class machine) that is not resource constrained. Tier 1 nodes can employ any wired/wireless communication medium. The Tier 1 nodes are trusted, i.e., assumed to be correct and serve as control nodes for Tier 2 nodes.

There are $\mathcal{P} = \{p_1, \ldots, p_n\}$ Tier 2 nodes of which f can be compromised by an adaptive adversary. Our adaptive adversary may pick its f faulty nodes as the system progresses; however, it is not mobile and cannot jump from one node to another. These are typically sensors that are resource-constrained. These nodes have limited functionality and run on batteries. Their main function is to collect, aggregate, and furnish sensed values and to update state based on other node's values. These nodes form a cluster and may report to Tier 1 nodes for provenance purposes. The nodes in the cluster are typically physically closer to each other and can communicate with each other directly or through multiple hops using energy-efficient communication mediums such as Bluetooth or BLE.

There are two communication media. One media is energy-efficient and is used to communicate among Tier 2 nodes. The second is an expensive medium used by Tier 1 nodes to communicate with Tier 2 nodes.

We assume that the network is bounded synchronous [27]. There is a public upper bound on the time taken to deliver a message between any two nodes. Moreover, our system is strongly

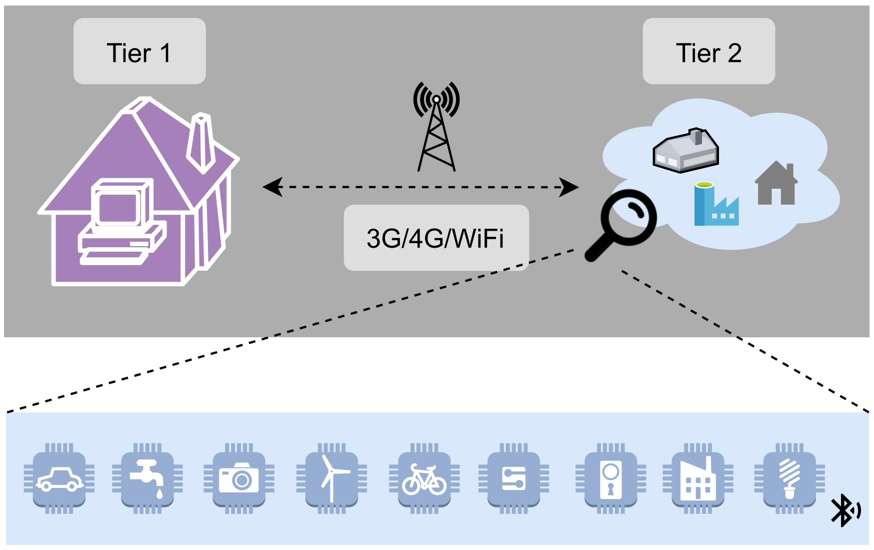

Figure 1: This schematic shows the target CPS setting with two tiers of nodes. The Tier 2 nodes are various sensor nodes, which are constrained in their energy, computational, and wireless range. The Tier 1 node is a server node in a control center and has no resource constraints. A Tier 1 and a Tier 2 node can communicate using a relatively expensive protocol, such as 4G links, while two Tier 2 nodes can communicate in an ad-hoc network setting through a more energy-constrained protocol, such as BLE.

connected (i.e., there is only one component) but not fully connected.

2.2 State Machine Replication—SMR

Nodes in the CPS setting often need to agree on a shared global state. Therefore, an SMR protocol provides clients (or users) an abstraction of a distributed fault-tolerant log is needed. This log is generally used to build a global state such as blockchains or databases. The formal definition of SMR is as follows:

Definition 1 (State Machine Replication (SMR) [2]). *Assume a system of nodes $\mathcal{P} = \{p_1, \ldots, p_n\}$ of which f are faulty.*

1. ***Safety.*** *If two correct nodes p_i and p_j commit to blocks B_i and B_j, respectively, at the same log position, then $B_i = B_j$.*

2. ***Liveness.*** *Each client request is eventually committed by all correct nodes.*

Our focus is on SMR protocols which deal with blocks at height i denoted by B_i. Every block $B_i := \langle b_i, h_{i-1} \rangle$ contains a hash pointer $h_{i-1} = H(b_{i-1})$ to the previous block B_{i-1}, where H is a cryptographic hash function. b_i is the set of signed requests from clients such as SENSE, DO, REPORT, etc. We do not consider the validity of the requests in b_i. As long as the all the correct nodes commit the same series of blocks, any state generated from the sequence of blocks will ensure a common global state among all the correct nodes.

2.3 Baseline Solution

The two-tiered architecture brings with it an elegant solution of its own for SMR, which we call the baseline solution. To motivate the need for energy-efficient protocols, we present this baseline

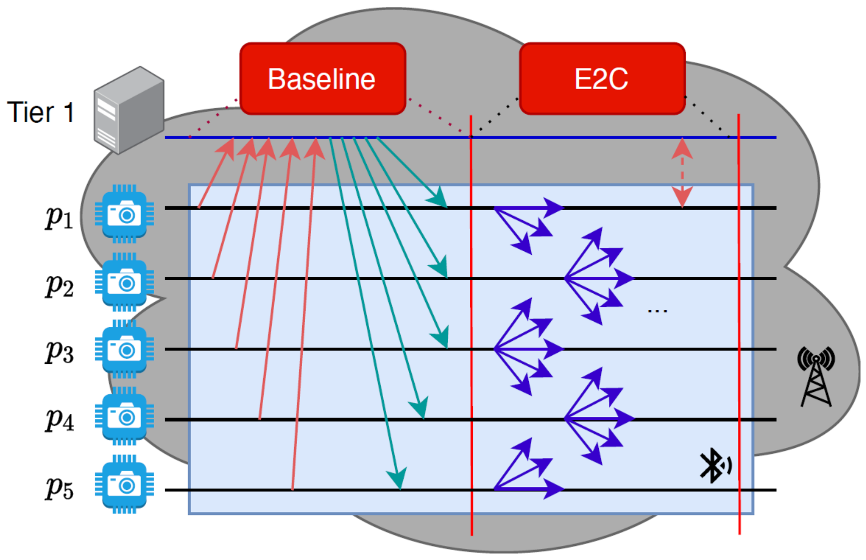

Figure 2: Illustration of the baseline and E2C protocol. In the baseline protocol, the Tier 2 nodes act as clients and send their requests to the Tier 1 node. The Tier 1 node responds to Tier 2 nodes with the latest state after applying all the updates. In E2C protocol, the Tier 2 nodes perform SMR locally by communicating among themselves. The Tier 1 node pulls the state whenever it needs it.

solution for the CPS setting and show how we can often do better than the baseline solution in terms of energy efficiency.

As shown in the first part of Figure 2, in the baseline solution for the CPS setting, all Tier 2 nodes send their requests directly to the Tier 1 node. The Tier 1 node sends the new state obtained by applying the requests, to all Tier 2 nodes. Since the Tier 1 node is trusted, this SMR protocol is Byzantine fault-tolerant.

However, this is not always the best approach in terms of energy efficiency. There are scenarios where it is not possible for the Tier 1 nodes to use the energy-efficient media such as BLE to communicate with the Tier 2 nodes. These media are usually short ranged. This would mean that the Tier 1 nodes are physically proximal to the Tier 2 nodes. This is often not the case in CPS settings. For example, in traffic monitoring, the Tier 1 nodes are placed in monitor stations far away (typically a few miles) from the deployed Tier 2 drones. In such scenarios it is impractical to use BLE, which has a short range (a few hundred meters) as a communication medium between Tier 1 and Tier 2 nodes. Now, if the Tier 1 node uses 4G to communicate with the Tier 2 nodes, the communication with the trusted control node becomes expensive, motivating a local SMR protocol.

2.4 Our Approach

We propose an approach in which the Tier 2 nodes perform SMR locally using an energy-efficient communication medium such as BLE to communicate among themselves. The Tier 1 node pulls the state whenever it sees fit as shown in Figure 2. The steps in E2C can be summarized as follows:

1. *Collect*: In this step, the nodes sense the values and prepare requests for the consensus phase.

2. *Consensus*: In this step, the nodes act as clients and send requests to the SMR protocol. We then use E2C SMR to build a global state.

3. *Report*: This is an optional step that occurs when requested by a Tier 1 node. The Tier 2 nodes report the state (possibly some form of digest of the state). The Tier 1 node collects reports

from Tier 2 nodes and use it for its control decisions.

3 Preliminaries

We assume that there exists one Tier 1 (command and control) node and it is trusted. All the Tier 2 nodes know the E2C public parameters such as the graph parameters and the public keys of the other Tier 2 nodes. The nodes start at the same time $t = 0$.

Hypergraph. To incorporate the multicasts available in the CPS settings, we model the network as a strongly connected hypergraph $\mathcal{H}$. Formally we define a hypergraph as follows:

Definition 2 (Hypergraphs [5]). *$\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ is a hypergraph where $\mathcal{P} = \{p_1, \ldots, p_n\}$ is the set of nodes, and $\mathcal{E} \subseteq \mathcal{P} \times (P(\mathcal{P}) \setminus \{\emptyset\})$ is the set of hyper-edges, where $P(\mathcal{P})$ is the power set of $\mathcal{P}$ and $\emptyset$ is the empty set.*

In $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$, we use p_i as the label for the i^{th} Tier 2 node. For example, if the node p_1 has a multicast with nodes p_2, p_3 and p_4, this can be represented as a hyper-edge $e = \{p_1, \{p_2, p_3, p_4\}\} \in \mathcal{E}$. In our definition of hypergraphs $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$, we do not allow self-loops (e.g., $\{p_i, \{p_i\}\}$). We denote the sender of a hyper-edge $e \in \mathcal{E}$, as $S(e)$ and the receivers of the hyper-edge as $R(e)$. So, $S(e) = p_1$ and $R(e) = \{p_2, p_3, p_4\}$ represents a hyper-edge with p_1 as the sender and p_2, p_3, p_4 as the receivers. The size of a hyper-edge e, is the number of vertices in $R(e)$.

Independence of edges. In the unicast setting, every edge (directed or undirected) enables communication with new nodes. However, in the case of hypergraphs, this is not always true. Consider three edges for some node p_i having $R(e_1) = \{p_1, p_2\}$, $R(e_2) = \{p_2, p_3\}$ and $R(e_3) = \{p_1, p_3\}$. All three edges are unique but one of them is redundant and can be removed while still reaching all three nodes p_1, p_2 and p_3. For our hypergraph, we assume the edge set is independent as these extra edges do not add any additional communications.

Definition 3 (Independence of Edges). *A set of edges $\mathcal{E}$ are said to linearly independent if there does not exist any two subsets $\mathcal{E}_1 \subset \mathcal{E}$ and $\mathcal{E}_2 \subset \mathcal{E}$ such that $\bigcup_{e_i \in \mathcal{E}_1} e_i = \bigcup_{e_i \in \mathcal{E}_2} e_i$.*

We assume the following from hyper-edges. For any correct node p_i, there is at least one hyper-edge $e \in \mathcal{E}$ with $S(e) = p_i$ on which sending a message at time t guarantees that at least one other correct node $p_j \in R(e)$ receives the message within time $t + \delta$ for some public finite bounded δ. This property ensures that all correct nodes stay connected to each other.

***k*-cast.** We achieve interesting properties when the size of any d_{out} outgoing hyper-edges is at least k. When $k = 1$, this translates to a standard directed graph. In a directed hypergraph, a node has two degrees: the in-degree (d_{in}) and the out-degree (d_{out}). We formally define the hypergraph degrees as follows:

Definition 4 (In-degree d_{in}). *For a node $p_i \in \mathcal{P}$ of a hypergraph $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$, the in-degree of a node p_i denoted by $d_{in}(p_i)$, is the number of hyper-edges in $\mathcal{E}$ in which the node p_i is a receiver. We denote by d_{in}, the minimum $d_{in}(p_i)$ among all nodes $p_i \in \mathcal{P}$.*

Definition 5 (Out-degree d_{out}). *For a node $p_i \in \mathcal{P}$ of a hypergraph $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$, the out-degree of a node p_i is denoted by $d_{out}(p_i)$, is the number of hyper-edges in $\mathcal{E}$ in which the node p_i is a sender. d_{out} is the minimum $d_{out}(p_i)$ among all nodes $p_i \in \mathcal{P}$.*

Network delay. For a partially connected system, let the diameter d of the hypergraph be the maximum distance between any two correct nodes in the hypergraph. For a hypergraph in our setting, we are interested in the delay across this diameter. So, we use a related parameter Δ_d

called the *maximum network delay.* Δ_d accounts for δ delays among the edges, the computations performed, the re-transmissions performed if there are collisions, verifying the integrity of the message and other factors involved in the underlying communication medium. This also involves the time to process the messages, perform necessary computations (like verifying signatures or MACs) using the longest possible path consisting of correct nodes.

Definition 6 (Maximum Network Delay Δ_d)**.** *Maximum Network Delay Δ_d is the finite upper bound on the message delay between any two correct nodes of the hypergraph $\mathcal{H}$. If a correct node sends a message to another correct node p_i, then p_i receives that message within Δ_d time from which it was sent.*

Communication primitives. In order to simplify the protocol description, we use the communication primitives like $\mathsf{sendAll}\,()$ to send a message over all outward hyper-edges of the node and $\mathsf{recvAll}\,()$ to receive messages from all inward hyper-edges.

Cryptographic primitives. There are two choices for cryptographic primitives: digital signatures which are public key primitives and Message Authentication Code (MAC) schemes which are symmetric key primitives.

A MAC scheme consists of three algorithms: KeyGen, MAC, and Verify. $sk \leftarrow \mathsf{KeyGen}(\kappa)$, where κ is the security parameter. sk is the shared key that must be established between the sender and the receiver. $\sigma \leftarrow \mathsf{MAC}\,(sk, m)$: σ is the tag generated for the message m with the secret sk. $\{0, 1\} \leftarrow \mathsf{Verify}\,(m, \sigma, sk)$ is the verification algorithm that outputs 1 if the MAC σ is correctly computed on m using key sk, and otherwise outputs 0.

A digital signature scheme consists of three algorithms: KeyGen, Sign, and Verify. $(sk, pk) \leftarrow \mathsf{KeyGen}(\kappa)$, where κ is the security parameter. sk is the secret key used by the sender to sign the message and pk is the public key used by others to verify that the message was signed by whoever possesses sk. $\langle m \rangle_i \leftarrow \mathsf{Sign}\,(sk_i, m)$, where $\langle m \rangle_i$ is the signature on the message m signed by node p_i using secret key sk_i. $\{0, 1\} \leftarrow \mathsf{Verify}\,(m, \sigma, pk)$ is the verification algorithm that outputs 1 if the signature σ signed by sk on the message m is valid, and otherwise outputs 0.

In MAC based protocols, every message includes a vector of tags for every node in the system. These tags could be valid for some nodes while invalid for others. A digital signature scheme guarantees unforgeability i.e., any node cannot generate a signature[1] without possessing the secret key sk and non-repudiation, i.e., if a signature verifies at one node, it will verify at all nodes.

This is useful in detecting equivocation. If there are two signed messages, all nodes can confirm that there was equivocation. Whereas in a MAC scheme due to the usage of vector of tags, it is not trivial to detect equivocation.

Public Parameters. pp represents the set of public parameters defined as the set $\mathsf{pp} := \{\mathcal{H}, f, \Delta_d\}$. Here, $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ is the hypergraph, k is the minimum size of a hyper-edge in $\mathcal{H}$, and Δ_d is the maximum network delay in the system. We clarify that when a node p_i knows p_j, we mean it has the public key of node p_j. Additionally, the MAC keys between a Tier 2 node and the Tier 1 node have been set up (or we can use TLS for this, but we ignore how it is done).

4 Fault Tolerance in Hypergraphs

For regular undirected graphs, if d is the minimum degree of the node, then the number of faults tolerated is bounded by d [13] given by $f < d$. For directed graphs, if d_i and d_o are the minimum

[1]The probability that an adversary can generate a signature for a particular message is negligible, parameterized by the security parameter κ.

in-degree and out-degree among all the nodes, then the standard results tolerate faults f by the smaller of the two values given by $f < \min(d_i, d_o)$.

For our hypergraph model with k-casts using system parameters $\texttt{pp} := \{\mathcal{H}, f, \Delta_d\}$, let $\texttt{neigh}(p_i)$ be the number of distinct nodes to which the node p_i can send messages. $\texttt{neigh}(p_i)$ is at least $min(d_{out}) + k - 1$. To understand the intuition for this bound, we argue that every k-cast adds one new node that p_i can communicate to. If this were not the case, then $\mathcal{E}$ is not an independent set of edges and there exist some redundant edges. The first d_{out} link adds at least k nodes that p_i can reach. After this, every additional d_{out} link extends the reachability of p_i by at-least one node, leading to our lower bound. Similar arguments hold for the incoming links.

We state this formally in the following lemma.

Lemma 1. *In the system $\texttt{pp} := \{\mathcal{H}, f, \Delta_d\}$, let the number of nodes who hear the messages sent by a node p_i be denoted by $\texttt{neigh}(p_i)$. Then*

$$\texttt{neigh}(p_i) \geq \min(d_{out}) + k - 1$$

To achieve Byzantine fault-tolerance, we require any node to be able to communicate with a correct node. This implies $f < \texttt{neigh}(p_i)$ for any node p_i. As long as one correct node is reachable, we can ensure that the faulty nodes cannot partition the network. In other words, the faulty nodes must not be able to prevent correct nodes from hearing each other. This ensures that all correct nodes can talk to each other even if most of the neighbors of a node are Byzantine. Extending similar arguments to ensure that a correct node must be able to hear from at least one correct node, gives us $f < \min(d_{in}, d_{out}) + k - 1$.

For example, if a leader equivocates, then the above property ensures that either all correct nodes hear it or none of them hear it. In our case, in the event of an equivocation, it is important that a correct node is able to communicate this to the other correct nodes. We show that there always exists a path between any two correct nodes in $\mathcal{H}$. The intuition follows from Lemma 1. Every node can reach at least one other correct node.

Lemma 2 (Honest Path Lemma)**.** *In a hypergraph $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ with $f < \min(d_{in}, d_{out}) + k - 1$, there always exists a path between two nodes in $\mathcal{H}$, consisting solely of correct nodes.*

Proof. For a node $p_i \in \mathcal{P}$, a set of vertices that p_i is connected to through its d_{in} k-casts is (say) $V_{in}(p_i)$. Let $V_{out}(p_i)$ be the set of vertices p_i is connected to, through d_{out} k-casts. Let $F \subset \mathcal{P}$ be the set of faulty nodes with $|F| = f$. We require

$$\forall\ p_i \in \mathcal{P},\ (V_{in}(p_i) \cup V_{out}(p_i)) \cap (\mathcal{P} - F) \neq \emptyset$$

This implies that the number of faulty nodes must satisfy

$$\begin{aligned} \forall\ p_i \in \mathcal{P},\ f &< |V_{in}(p_i) \cup V_{out}(p_i)| \\ f &< \min_{p_i \in \mathcal{P}} |V_{in}(p_i) \cup V_{out}(p_i)| \end{aligned}$$

This can be rewritten as

$$\begin{aligned} |V_{in}(p_i)| \geq k + d_{in} - 1\ \vee\ |V_{out}(p_i)| &\geq k + d_{out} - 1 \\ f &< \min(d_{in}, d_{out}) + k - 1 \end{aligned}$$

The hypergraph $\mathcal{H}$ remains strongly connected even after removing the f faulty nodes as long as $f < \min(d_{in}, d_{out}) + k - 1$. This ensures that the faulty nodes cannot partition the network, and there exists at least one path consisting of only correct nodes between any two nodes. □

Let $\mathsf{pp} := \{\mathcal{H}, f, \Delta_d\}$ be the parameters of the system. Let S be the source with a correctly formatted message m.
i **Source Send.** The source S broadcasts m, using its k-casts by invoking $\mathsf{sendAll}\,(m)$.
ii **Retransmit.** On receiving messages $m \leftarrow \mathsf{recvAll}\,()$: if the message received m is new and the message m is correct, then the other nodes forward it along their k-casts by invoking $\mathsf{sendAll}\,(M)$ and add it to the set of received messages $msgs$.

Figure 3: E2C Diffusion Protocol

Theorem 1. *Consider a hypergraph $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ with $\mathcal{P} = \{p_1, \ldots, p_n\}$ nodes. Each node is a receiver in at least d_{in} k-casts and a sender in at least d_{out} k-casts. If $\mathcal{H}$ has less than f Byzantine nodes where $f < \min(d_{in}, d_{out}) + k - 1$ then the hypergraph $\mathcal{H}$ cannot be partitioned.*

Proof. For a partially connected graph, the fault tolerance is determined by the number of nodes a node p_i is connected to, which in our case is `neigh`(p_i). From Lemma 2, with $f < N_d$ we can ensure that $\mathcal{H}$ is not partitioned. □

With the guarantees from Lemma 1 and Lemma 2, Theorem 1 implies that $f < \min(d_{in}, d_{out}) + k - 1$ faults cannot partition the network. This result corresponds exactly to the results shown in literature [13] if the hypergraph is treated as a regular graph using unicasts described above by setting $k = 1$, $d_{out} \leftarrow d_o$ and $d_{in} \leftarrow d_i$.

5 E2C Protocol

Diffusion Protocol. We begin with the diffusion protocol described in Figure 3, the sender sends to all its neighbors a valid message m and every node repeats this process. We say a message m is *correct*, if m is one of the correctly formatted protocol messages. From Lemma 2, we know that there exists a path from a sender to a receiver consisting solely of correct nodes. It guarantees that the sender's message is delivered to every correct node via a path consisting of correct nodes within time Δ_d. As an optimization, to conserve energy, the diffusion protocol involves the nodes keeping a buffer of received messages $msgs$, and forwarding only new messages to the others.

The diffusion protocol achieves the reliable broadcast property within Δ_d. We show this formally in Theorem 2. When the sender is faulty, there can be multiple messages from the sender. The sender can send a message close to the end of Δ_d. In this scenario, a message exists in the buffers of some correct nodes, but not all of them.

The diffusion protocol can be thought of as a sendAll or broadcast primitive that is used in fully connected systems, extended to our hypergraph setting. We use this a building block for the E2C protocol.

5.1 E2C- Steady State Protocol(s)

The E2C protocol runs in rounds called views. In each view, there is a leader $L \in \mathcal{P}$. This leader is chosen using a leader selection algorithm `getLeader` (possibly round robin) that outputs a leader for the current view. If the leader does not make progress or if the leader equivocates, the view change protocol is triggered.

Blocking vs. Non Blocking. We consider two types of commit strategies: *Blocking* commit where nodes only focus on extending the latest committed block and buffer all proposals for future

Let $\mathsf{pp} := \{\mathcal{H}, f, \Delta_d\}$ be the parameters of the system. Let L be the leader for the current round and let the current view number be v. Let B_{i-1} be the head of the latest committed block.

Steady State Protocol. In the steady state protocol, the nodes do the following:

i **Propose.** The leader L diffuses $\langle \mathsf{propose}, B_i, v\rangle_L$ where $B_i := \langle b_i, h_{i-1}\rangle$, b_i is the proposal of new commands, and B_{i-1} is the highest committed block according to L.

ii **Relay.** Upon receiving the first valid proposal for height-i block B_i from L or through a relay by some other replica, if it is extending the replica's previously committed or locked block, continue diffusing the proposal to all other replicas. Set $B_{lck} \leftarrow B_i$. Set $\mathsf{commit-timer}_i$ to 2Δ and start counting down.

iii **(Blocking) Commit.** When $\mathsf{commit-timer}_i$ reaches 0, commit B_i. Process next proposal from buffer if available.

Figure 4: E2C Steady-State SMR protocol with blocking commits

Let $\mathsf{pp} := \{\mathcal{H}, f, \Delta_d\}$ be the parameters of the system. Let L be the leader for the current round and let the current view number be v.

Steady State Protocol. In the steady state protocol, the nodes do the following:

i **Propose.** The leader L diffuses $\langle \mathsf{propose}, B_i, v\rangle_L$ where $B_i := \langle b_i, h_{i-1}\rangle$, b_i is the proposal of new commands, and B_{i-1} is the highest committed block according to L.

ii **Relay.** Upon receiving the first valid proposal for any height-i block B_i from L or through a relay by some other replica, if it is extending the replica's previously committed or locked block, continue diffusing the proposal to all other replicas. Set $B_{lck} \leftarrow B_i$. Set $\mathsf{commit-timer}_i$ to 2Δ and start counting down.

iii **(Non-Blocking) Commit.** When $\mathsf{commit-timer}_i$ reaches 0, commit B_i. Start $\mathsf{commit-timer}_{i+1}$ and all its ancestors.

Figure 5: E2C Steady-State SMR protocol with non-blocking commits

blocks, and *Non-Blocking* commit where nodes can commit to blocks at arbitrary heights. The Blocking commit requires a buffer of only one block and is friendly to memory constrained devices. The Non-Blocking commit strategy allows the protocol to proceed at network speed and not the maximum network delay Δ_d while still enjoying the benefits of our protocol but comes at the cost of an extra round in the view change.

Steady State with correct leader. In the steady-state, when the leader is correct, E2C protocol guarantees safety. All correct nodes commit to the blocks sent by the leader. The leader creates valid blocks and uses the diffusion protocol (Figure 3) to send the block to all the nodes. When the leader is correct, by the properties guaranteed from the diffusion protocol, the protocol trivially enjoys safety. This is formalized in Lemma 3.

When the leader is correct, it is not possible for the faulty nodes to break safety of the protocol (from Lemma 3). However, if the leader equivocates, the hypergraph connectivity result in Theorem 1 ensures that the correct nodes detect the equivocation, abort the round, and start a fresh round with a new leader by initiating a view change protocol. In Lemma 4, we formally argue the commit safety for correct nodes.

Commit Safety. When a correct node commits to a block B_i, E2C protocol does not guarantee that all the other correct nodes will commit to B_i in the same view v. If the leader is Byzantine and decides to equivocate, then the correct nodes abort on detection of equivocation. The Byzantine leader can send an equivocating message near the end of $2\Delta_d$. Some correct nodes start blaming, while the other correct nodes commit to B_i in the view v. We recover from this scenario using a view change protocol which ensures that the remaining correct nodes also commit to B_i in the next successful view. We note that we mention successful view because the next leader can also be Byzantine.

5.2 View Change Protocol(s)

As shown in Figure 6 in E2C, a view change can occur if (i) the leader stops making progress, i.e does not send valid blocks every $2\Delta_d$, (ii) equivocates in a view with two blocks in the same height, and (iii) does not correctly extend the highest block. These only occur when the leader is faulty.

No Progress Blame. If a node does not receive sufficient blocks (one block every $2\Delta_d$), then the node sends a blame message. Note that it is possible for some correct nodes to send this. But it

Let L and L' be the leader of view v and $v+1$ respectively.

View Change. The nodes on receiving messages ensure the following:

i **(No-progress) Blame.** If less than p blocks are received from L in $(2p+1)$ time in view v, diffuse $\langle \mathsf{blame}, v \rangle_j$.

ii **(Extension) Blame.** If the proposed block B_i does not extend currently committed block B_{i-1}, then diffuse $\left\langle \langle \mathsf{blame}, v \rangle_j, \mathsf{ext}, B_{i-1} \right\rangle$.

iii **(Equivocation) Blame.** If a leader L proposes equivocating blocks, diffuse $\left\langle \langle \mathsf{blame}, v \rangle_j, b_i, b'_i \right\rangle$.

iv **Quit View.** Upon gathering $f+1$ $\langle \mathsf{blame}, v \rangle$ messages, diffuse them, and quit view v, i.e. abort all $\mathsf{commit} - \mathsf{timer}$ (s).

v **Recovery and Change View.** Perform a recovery and change view by using one of the algorithms in Figure 7.

Figure 6: E2C View-Change Protocol

does not violate safety to do so. This is because of the existence of an honest path. If one correct node hears a proposal for that round, then it will forward it to the other correct nodes within $2\Delta_d$. However, if none of the correct nodes hear a block then there will be $f+1$ blames, and the view change is guaranteed.

Equivocation Blame. A leader L can equivocate by sending B_i and B_i' for any block (not just the latest block). In this case, from Theorem 1, we can guarantee that all correct nodes hear this. Therefore, $f+1$ blames are guaranteed.

Extension Blame. A correct leader L must extend the previously proposed block B_i with B_{i+1} such that h_i is embedded in $B_{i+1} := \langle b_{i+1}, h_i \rangle$. By the commit safety for B_i, all $n-f > f$ nodes detect this and send blame messages and change the view.

In E2C, depending on the availability of trusted control nodes as Tier 1 nodes, there are two mechanisms to perform a view change. The first method *View Change with Tier 1*, involves reporting the blame directly to Tier 1 nodes and perform recovery. In the second method *Local View Change*, we can use the energy efficient communication medium between the Tier 2 nodes and perform a view change protocol.

Both the methods are efficient in certain hypergraph settings and are presented in Figure 7. Depending on the deployment conditions such as how much more expensive is the link from Tier 2 nodes to Tier 1 nodes, and the cost of signature generation and verification, it can sometimes be more energy-efficient to report the blame messages directly to the Tier 1 nodes. The general flow of a view change is specified in Figure 6.

Changing view and Recovery. After receiving $f+1$ blame messages, the nodes collect all $f+1$ blame messages and send this to all the other nodes. This indicates that all the correct nodes are quitting the current view and if any correct node is not yet sure then it must also do the same. After sending they wait for Δ_d to ensure that all correct nodes have quit the old view v. After quitting the old view, the nodes perform recovery using one of the two recovery strategies described previously: *View Change with Tier 1* or *Local View Change*.

View Change with Tier 1. In this mechanism (shown in Figure 7) of view change, a node that collects $f+1$ blame messages immediately notifies the Tier 1 node of a fault by the Tier 2 leader. Then the Tier 1 node collects status messages from the other Tier 2 nodes and takes over the role of the leader for that round to ensure commit safety.

The Tier 2 nodes can safely use the blocks sent by the Tier 1 node and commit it. After committing, the Tier 2 nodes change their leader. Since the link between Tier 1 and Tier 2 links are expensive, we can further optimize the view change protocol by making the Tier 1 node send the newly committed blocks only to the new leader and the new leader diffusing the Tier 1 node's signed message. This ensures instant commitment to the new block B_i which the leader chooses appropriately based on the status of all Tier 2 nodes.

Local View Change for Blocking commit. In this mechanism (shown in Figure 7), the Tier 2 nodes change the leader locally using the energy-efficient medium and resolve the commit. The first node to detect equivocation at any point, diffuses the blame message. The new leader then collects signed vote messages from every node. The nodes send the B_{lck} they have committed to. There can only be the state where some nodes have committed to B_{lck} in view v and the others aborted. But even if the other correct nodes aborted, by protocol B_{lck} will still mark the first block the node received as B_{lck}. This ensures that $f+1$ vote messages are available for B_{lck} during view change if at least one correct node committed to B_{lck}.

Local View Change for Non-Blocking commit. In this mechanism (shown in Figure 7), the first node to detect equivocation at any point, diffuses the blame message and stops updating B_{lck} and all the timers are cancelled. All the nodes send their latest committed block and all the nodes

Let $\mathsf{pp} := \{\mathcal{H}, f, \Delta_d\}$ be the system. Let L be the leader for the current view v and L' be the leader for the next view $v+1$.

Assisted recovery using Tier 1 nodes. The Tier 1 and Tier 2 nodes do the following:
1. **Status.** The Tier 1 node after receiving $f+1$ blame messages, sends a Change-View message to all the nodes in $\mathcal{P}$.
2. **Recovery.** The Tier 1 node collects the latest commit status B_{lck} from all the Tier 2 nodes, and sets the block to the highest committed block among all the Tier 2 nodes. If B_{lck} is different, i.e there is no B_{lck} which $f+1$ nodes attest to, then the Tier 1 node picks one of the blocks and sets it as the block for the current height.
3. **Change View.** After receiving updates from the Tier 1 node, the nodes update their states and change the leader L to L' and set $t=0$ and continue the protocol.

Local Recovery and Change of view for Blocking commit. The Tier 2 nodes do the following:
1. **Wait.** After diffusing the $f+1$ blame messages, the Tier 2 nodes diffuse a vote message $\langle \mathsf{vote}, B_{lck}\rangle_j$ for the latest locked block B_{lck} and set a timer for Δ_d.
2. **Status.** If $f+1$ vote messages are received for any B_{lck}, then forward this to the new leader L'. Otherwise, forward vote message for $\langle \mathsf{vote}, B_{lck-1}\rangle_i$.
3. **New Leader.** The new leader L' collects vote messages from all the nodes. On collecting $f+1$ vote messages, it attaches them to the first proposal in the new view $v+1$.
4. **Validate New Leader.** Finally, the nodes verify that the new view is correct by ensuring that the attached vote messages are valid; otherwise the nodes send $\left\langle \langle \mathsf{blame}, v+1\rangle_j, \mathsf{ext}\right\rangle$

Local Recovery and Change of view for Non-Blocking commit. The Tier 2 nodes do the following:
1. **Wait.** After diffusing the $f+1$ blame messages, the Tier 2 nodes diffuse a vote message $\langle \mathsf{vote}, B_{lck}\rangle_j$ for the latest committed block B_i and set a timer for Δ_d.
2. **Status.** For every vote message on block B_i, send a vote for B_i if B_{lck} correctly extends B_i. If $f+1$ votes are collected for B_{lck}, send it to the new leader L'.
3. **New Leader.** The new leader L' collects certificates from all the nodes. It attaches them to the first proposal in the new view $v+1$.
4. **Validate New Leader.** Finally, the nodes verify that the new view is correct by ensuring that the attached certificates are valid; otherwise the nodes send $\left\langle \langle \mathsf{blame}, v+1\rangle_j, \mathsf{ext}\right\rangle$

Figure 7: Recovery Strategies for E2C protocol

vote for these blocks if their respective B_{lck} correctly extends it. The nodes build certificates (vector of signatures attesting this block) and send them to the new leader. For a correctly committed block, it is guaranteed that all correct nodes will vote for it (Lemma 7).

5.3 An example run

In Figure 8, we give an example run of the protocol. If the leader is correct, then the initial block B_i proposed by L will be propagated through k-casts using the diffusion protocol. Since faulty nodes can not forge messages, all correct nodes will set B_{lck} to B_i. After $2\Delta_d$ hearing no equivocation, they commit to B_i. And the protocol continues to the next $\langle \mathsf{propose}, B_i, v\rangle$ step skipping the view change.

If the leader is Byzantine, then it may send an equivocating block as shown in the first message in the blame phase. Since there is a correct node, the equivocation is detected and both the contradicting blocks are diffused in the system. After receiving $f+1$ blame messages, all the correct nodes (nodes p_2,p_3,p_4 and p_5) trigger the **Quit View** step.

Now, the nodes need to generate a certificate so that the next leader can correctly extend the right block. Depending on when the leader equivocates some of the nodes may have already committed to B_i. If there is one correct node that has committed to B_i then all correct nodes must have B_{lck} as B_i. Therefore, in the vote certificate step, the next leader obtains $f+1$ signatures on B_i and can extend it.

If no node has committed to B_i and the leader equivocated in a way that there does not exists $f+1$ votes on any B_i, then as per protocol, the nodes send votes for B_{k-1}. The leader is now forced to extend B_{k-1} as there do not exist $f+1$ votes for any block whose height is larger than $k-1$.

5.4 E2C Security Analysis

We first prove that nodes can reliably communicate with each other using the diffusion protocol as a primitive in Theorem 2.

Theorem 2 (Diffusion)**.** *If the sender is correct and sends a correct protocol message M, then the diffusion protocol ensures that all correct nodes receive M by the end of Δ_d.*

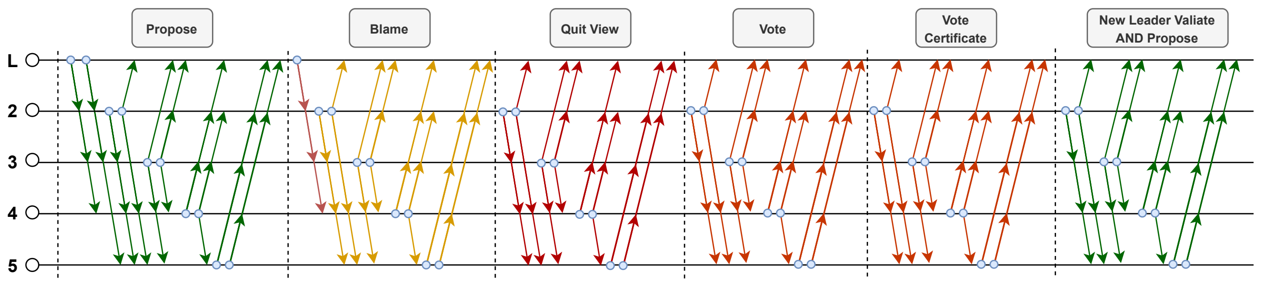

Figure 8: We illustrate the E2C protocol for Blocking commit with an example hypergraph $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ which has $\mathcal{P} = \{p_1, \ldots, p_5\}$, $k = 3$, $d_{in} = 5$, $d_{out} = 2$. Here, the system of 5 nodes can tolerate up to 2 faults (from $n > 2f$ rather than $f < \min(d_{in}, d_{out}) + k - 1$). The node p_1 is the leader L. Each horizontal line represents the node. Each vertical line is a hyper-edge. The blue dot represents the transmitter of each k-cast while the arrow heads represent the receivers in each k-cast.

Proof. Assuming that the sender sends a correct protocol message M. From the honest path lemma from Lemma 2, all the correct nodes will receive this correct message M within Δ_d from some correct node in `neigh`. ☐

Now, we show that when the leader is correct, the faulty nodes cannot change the leader and the nodes will commit to the block proposed by the leader in Lemma 3.

Lemma 3. *If the leader is correct and the leader proposes B_i, then all correct nodes commit to B_i.*

Proof. By the honest path assumption (from Lemma 2) and the properties guaranteed by the diffusion protocol, all the correct nodes receive the block sent by the leader. Since the digital signature scheme used is secure, we can guarantee that a faulty node cannot create alternate signed messages from the leader. Therefore, all correct nodes must agree on B_i. The f faulty nodes cannot generate view change messages to trigger view change. A correct node will send a blame message only if one of the conditions are satisfied, which are guaranteed not to occur if the leader is correct, i.e. proposes every $2\Delta_d$ and does not equivocate. ☐

We show that even if the leader is Byzantine, no two Tier 2 nodes can commit to different blocks in Lemma 4.

Lemma 4. *In E2C protocol, no two correct nodes can commit to different blocks B_i and B'_i in the same view v.*

Proof. We prove this Lemma by contradiction. Let $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ be a hypergraph with $\mathcal{P} = \{p_1, \ldots, p_n\}$. Say two correct nodes $p_i, p_j \in \mathcal{P}$ commit to B_i and B'_i respectively. From Lemma 2, we know that there exists an honest path between p_i and p_j. Since the correct path consists of correct nodes, all the nodes execute the *Relay* step in Figure 4. This implies that at least one of the nodes, will detect the equivocation within Δ_d and trigger a view change before $2\Delta_d$. Therefore, it is not possible for two correct nodes to commit to two different blocks in the same view v. ☐

Now, we show that using Tier 1 nodes, we can ensure that all nodes start the next view at the same height with the same block.

Lemma 5. *The view change protocol with Tier 1 ensures that all correct nodes commit to the same block B_i if $f < \min(\min(d_{in}, d_{out}) + k - 1, n/2)$.*

Proof. We need $f < \min(d_{in}, d_{out}) + k - 1$ to ensure that there is no partitioning in the system. This ensures that $f + 1$ nodes detect the blame condition. We need $f < n/2$ to ensure that the correct nodes are in majority. Otherwise, the faulty nodes can always send No-Progress blame messages. After this, the proof follows trivially from the design and the assumption that the Tier 1 node is correct. ☐

In Lemma 6 we show that the local view change protocol for Blocking commit is secure.

Lemma 6. *Let $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ be a hypergraph with $\mathcal{P} = \{p_1, \ldots, p_n\}$ as the nodes. The local view change protocol for blocking commit ensures that all correct nodes commit to the same block B_i when $f < \min(\min(d_{in}, d_{out}) + k - 1, n/2)$.*

Proof. Lemma 3 with $f < \min(d_{in}, d_{out}) + k - 1$ guarantees that all correct nodes hear the blame message with $f+1$ nodes requesting a change in view. The nodes that committed to a block B_i will also have set B_{lck} to B_i. If the leader equivocates at a previous height $k' < k$, then there will be sufficient votes for B_i and therefore B_{lck}. The next correct leader will extend this block correctly.

If the leader equivocates with many B_i to different nodes, then the nodes will try getting $f+1$ votes for B_{lck}. When this fails, it could only mean that no node has committed to any of the B_{lck} (from Lemma 4). Then all the nodes will send vote messages for B_{k-1}. Therefore, the next correct leader extends B_{k-1} ensuring commit safety.

We require both the honest path and commit safety to hold. Therefore the fault tolerance f is the minimum of $n > 2f$ and $f < \min(d_{in}, d_{out}) + k - 1$. □

In Lemma 7 we show that the local view change protocol for Non-Blocking commit is secure.

Lemma 7. *Let $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ be a hypergraph with $\mathcal{P} = \{p_1, \ldots, p_n\}$ as the nodes. The local view change protocol for Non-Blocking commit ensures that all correct nodes commit to the same block B_i when $f < \min(\min(d_{in}, d_{out}) + k - 1, n/2)$.*

Proof. The main difference from Lemma 6 is that we need to ensure that all committed blocks of correct nodes will be committed in the next view. In other words, we need to ensure that there are certificates for every correctly committed block and the highest such certificate must be extended by the new leader.

If a block B is committed correctly by a node p_i at time $t + 2\Delta_d$, then p_i relayed the message to all correct nodes before time $t + \Delta_d$. All correct nodes observed that this correctly extends their latest B_{lck}, and therefore did not send any blame messages. This means that they will vote for B during a view change ensuring the existence of certificates for the latest correctly committed block for all correct nodes.

Trivially, using the extension blame, we can ensure that the new leader must choose the highest certified block and not the other certificates. □

We prove the security of the E2C protocol by showing that it satisfies the *safety* and *liveness* requirements stated in Definition 1 with Theorem 5 and Theorem 6, we prove E2C protocol is secure.

Theorem 3. *Let $\mathcal{H} := \{\mathcal{P}, \mathcal{E}\}$ with nodes $\mathcal{P} = \{p_1, \ldots, p_n\}$ and edges $\mathcal{E}$. If $f < \min(\min(d_{in}, d_{out}) + k - 1, n/2)$, E2C protocol achieves SMR as specified in Definition 1.*

To prove this, we first prove that the individual view changes are secure in the following Theorem 4

Theorem 4 (Secure View Change)**.** *The view change protocols ensure that all correct nodes commit to the same blocks B when the sender is faulty and the view change resilience condition is satisfied.*

Proof. For the view change algorithms described in Figure 7, the proofs follow from Lemma 5 and Lemma 6 with $f < \min(\min(d_{in}, d_{out}) + k - 1, n/2)$. □

We now prove that E2C protocol achieves safety in Theorem 5.

Theorem 5 (SMR Safety)**.** *In E2C protocol, all correct nodes commit to the same block B_i for any height k.*

Proof. Assume that the leader is correct. By the honest path assumption (from Lemma 2) and the properties guaranteed by the diffusion protocol in Lemma 3, all the correct nodes receive the block sent by the leader and commit to it.

Now, consider the case where the leader is Byzantine. If the leader does not send anything, all correct nodes send blame messages. If the sender equivocates, then by the guarantees from Theorem 1 the equivocation is detected by correct nodes. The view change protocol is triggered which ensures that all correct nodes commit to the same blocks. From the commit safety properties guaranteed from Lemma 4, Lemma 6 and Lemma 5, the view change ensures that all the correct nodes resolve the same block B_i for the current height i. Therefore, E2C protocol is secure. □

Theorem 6 (SMR Liveness)**.** *In E2C protocol, the protocol continues to make progress (commit blocks) as long as* $f < \min(\min(d_{in}, d_{out}) + k - 1, n/2)$.

Proof. Assume that the leader is correct. Then f faulty nodes cannot generate sufficient blame messages to trigger a view change. Since there exists only one signed message $M = \langle m, \sigma \rangle$ and the leader does not equivocate, by the security properties of the signature scheme, the E2C protocol guarantees liveness when the leader is correct. When the leader is Byzantine, the view change (which terminates in finite number of rounds) terminates within f views. Therefore, eventual progress is guaranteed for E2C protocol. □

5.5 Optimizations

We propose four orthogonal optimization techniques to E2C protocol to further improve the energy efficiency.

Removal of Extension Blame. We can remove the extension blame and resort to just two types of blame messages: No progress and Equivocation by simply discarding blocks that do not extend the latest committed block. We do not mark them in B_{lck} but store it in our buffer. This will eventually lead to $f + 1$ No progress blames or equivocation if the leader tries to propose a second block for the given height.

Equivocation scenario speedups. We can speed up the view change in the event of an equivocation by skipping the Quit View phase. This follows from the fact that all correct nodes will vote for the equivocation as the signatures must match for all the correct nodes. We can thus save one round in this phase.

Signature Verification Optimization. Cryptographic operations are expensive. Observe that, the protocol in the *Blame* phase, can work without all nodes verifying the equivocation from the leader. For systems with more than $2f$ nodes, it is sufficient for $2f + 1$ nodes to verify the signature and diffuse a blame if the leader equivocates. If a node receives $f + 1$ such messages, then the node knows that at least one node has detected the equivocation by the leader and can send a blame message.

Batching Optimization. Batching techniques often aid in amortizing the cost of consensus. In E2C protocol, the nodes can optimistically pre-commit to the block received from the neighbors without signature checks. After c such rounds, the nodes can initiate a checkpoint protocol where a full SMR protocol is initiated. If the leader was correct, this saves a significant amount of energy for the system. If the leader was faulty for all c rounds, the nodes need to recover blocks for those rounds. The worst case scenario is the same as the standard E2C protocol. However, we have a significant energy improvements in the best case.

Note on optimizations and security. It is easy to show that these optimizations do not violate the safety and liveness of the system. We can easily extend the proofs provided in Section 5.4 to show that the protocols with the optimizations are still secure.

6 Experiments and Performance Analysis

System Setup. The energy measurement was performed using `NUCLEO-F401RE` as the primary boards. This board is based on an ARM Cortex M4 84 MHz processor along with `NUCLEO-IDB05A1`

Table 1: Fitting energy costs (in J) into equations dependent on the message size m (in bytes) of the form $f(m) = Am + B$, with mean squared error ε. μ_S and μ_V represent HMAC signing and verifying respectively.

Constants	A	B	Error ε
SHA-256	$2.13e{-6}$	$1.19e{-4}$	$1.58e{-9}$
$\mu_S = \mu_V$	$2.13e{-6}$	$6.93e{-4}$	$1.59e{-9}$
BLE_S	$2.27e{-6}$	$4.47e{-7}$	$6.60e{-6}$
$WiFi_S$	$3.09e{-4}$	$1.81e{-3}$	$1.46e{-2}$
$4G_S$	$1.93e{-3}$	N/A	N/A
BLE_R	$2.15e{-6}$	$-1.89e{-5}$	$1.57e{-5}$
$WiFi_R$	$2.60e{-4}$	$2.91e{-4}$	$1.10e{-3}$
$4G_R$	$2.72e{-4}$	N/A	N/A

Bluetooth LE expansion boards and `NUCLEO-IDW04A1` WiFi expansion boards. The main board has 512 kB of flash memory and 96 kB of SRAM. The energy measurement device used was `Saleae Logic-Pro 8` signal analyzer along with `INA169` current sensor.

6.1 Instantiating Primitives

We measure the energy costs of the communication and cryptographic primitives described in Section 3. As analyzed in [19], we fit the empirical results into lines of the form $f(m) = Am + B$, where m is the message size in bytes and summarize them in Table 1 and Table 2.

Communication Primitives. We measure the energy cost of sending and receiving messages of different size over WiFi and BLE. We measure this for 100 points of message sizes chosen uniformly from $[1, 10000]$ bytes and fit them into lines. Table 1 shows the best fit parameters and the mean squared error for the linear regression. We observe that the energy measured for sending and receiving messages over WiFi using the nodes are of the same magnitude as that indicated by prior work [19] after calculating the throughput using the testbed. Subsequently, we estimate the energy consumed by cellular communication using the evaluation from prior work [19].

We use BLE advertisement packets as multicasts. We observe that these packets have user controlled payloads limited to 20 bytes per packet. For large message sizes, we use multiple invocations of advertisements.

Symmetric Key Primitives. We take a look at the cost of hashing messages and generating and verifying MAC tags. We take a look at the hash costs because of several reasons. As messages are generally shortened into a digest using a cryptographic hash function before being input into public key algorithms. For instance, the public key algorithm $\mathsf{Sign}\,()$ takes as input the hash of the message to generate digital signatures. They also play a major role in reducing the message size during the Blame phase.

We instantiate the MAC (Message Authentication Code) algorithm using `SHA-256` as the underlying hash function and measure the hashing cost for different message sizes. We use short keys of recommended size 64 bytes. The cost of signing (μ_S) and verifying (μ_V) is the same as for the HMAC scheme. The major cost in the HMAC scheme was mostly due to the underlying `SHA-256` algorithm. We found that the cost of hashing increased linearly with message size. We present the linear regression and the error in Table 1. As previously stated, they share the same slope ($A = 2.13 \times 10^{-6}$) with respect to m but have different intercepts (B).

Public Key Primitives. To instantiate the public key primitives, we compare the performance

Table 2: Comparing energy costs (in J) for signature generation (σ_S) and verification (σ_V) for ECDSA curves and RSA schemes.

Algorithm	Parameters	Sign (σ_S)	Verify (σ_V)
ECDSA	BP160R1	5.80	11.03
	BP256R1	13.88	27.34
	SECP192R1	0.84	1.50
	SECP192K1	1.16	2.24
	SECP224R1	1.10	2.14
	SECP256R1	1.60	3.04
	SECP256K1	1.72	3.35
RSA	1024-bit modulus	0.40	0.02
	1260-bit modulus	0.79	0.03
	2048-bit modulus	2.41	0.06

of elliptic curve algorithms and RSA with equivalent security moduli (in Table 2). A public key signing operation cost (σ_S) and verification operation cost (σ_V) is measured after computing the `SHA-256` hash of the message. So these are constants and with respect to the linear regression, they have $A = 0$.

Table 3: Energy cost (in mJ) of Implementation of E2C (with Blocking Commit) on Embedded Devices for the various phases in E2C. The theoretical error %s show that the energy consumption was close to the expected values. The bits of security provided by the RSA schemes are denoted in parenthesis. In the steady state, only *Propose* phase contributes to the energy consumption, whereas when the leader is faulty, *all* the phases contribute to the energy consumption.

Protocol Phase	RSA Scheme	1024 (80-bit security)		1260 (93-bit security)		2048 (112-bit security)	
	Role of Node	Exp. Value	(+) Error %	Exp. Value	(+) Error %	Exp. Value	(+) Error %
Propose	Leader	408.42	0.71	803.84	0.37	2427.86	0.20
	Others	30.84	4.55	43.41	7.70	74.34	2.75
Blame	Others	436.80	0.92	837.45	0.69	2492.30	1.21
Quit View	Others	130.54	4.81	181.95	1.67	231.72	2.60
Vote	Others	407.14	0.99	796.65	0.25	2419.14	0.08
Vote Cert	Others	120.38	3.42	174.36	2.38	214.49	0.94
Leader Valid	Leader	423.47	0.95	817.26	0.36	2451.74	0.12
	Others	141.43	0.14	206.26	0.98	400.47	0.50

For ECDSA [20], brainpool curves (`BP-XXX`) [28] are generally expensive. NIST optimized curves (`SECP-XXX`) [14] offer better performance in comparison. We borrow the implementations for both types of curves from MbedTLS [4, 18] (commit: `b6229e304`). We implement BP160R1 using the interface provided by MbedTLS and RFC 5639 [26].

Anecdotally, it took 3 years of efforts using high performance machines in order to factor one 768-bit RSA modulus [22]. Therefore, 80-bits of security provided by RSA 1024-bit modulus, is practical for the CPS settings. RSA provides two benefits over ECDSA: (i) reduced energy costs (σ_S and σ_V), and (ii) benefit of asymmetry in verification cost and the signing cost ($\sigma_S \gg \sigma_V$). The second benefit implies higher energy costs for faulty nodes (providing erroneous signatures) while the impact is significantly less on the correct nodes (that are trying to validate).

A note on Hash Based Signatures. Hash based signatures are energy efficient but the public key sizes are large or the signature sizes are large. For instance, signature schemes like Winternitz-OTS [9, 29], HORS [32] have large key sizes for one time use. This results in large storage requirements for Tier 2 nodes. We then take a look at multi-time hash based signatures such as XMSS [10] and HORST [6]. These algorithms require large RAM sizes (larger than what can be found in our Tier 2 nodes). For example, consider XMSS with the compression parameter $w = 2^8$ and re-usable for 2^{10} signatures. For messages of size 32 bytes (output of `SHA-256` for example), the signature is 102 bytes long with large intermediate memory requirements. It requires 765 hash computations to sign a message, and 785 hash computations to verify. Due to large memory requirements for hash based signature schemes, we do not consider them in this work.

6.2 Implementation on Embedded Devices

E2C protocol was implemented on a system of $n = 10$ nodes. The minimum number of nodes any node could communicate with was $k = 4$, the minimum degrees of the topology were $d_{out} = 1$, $d_{in} = 4$ and the Δ_d was set to $0.1s$. With this topology, all the nodes can communicate with every other node (through hops) using BLE.

We use the block structure for E2C as shown below, where H is a cryptographic hash function, i is the block height, L is the leader. $B_i := \langle i, H(b_i), H(b_{i-1}), \langle i, H(b_i)\rangle_L, b_i\rangle$.

In CPS settings, these blocks contain data b_i that is sensed by the nodes or requests that the nodes need to process. We skip validation of the requests in b_i in our implementation as it is common to all SMR solutions.

The energy consumed was recorded for the various phases of the protocol. The energy consumed during idle scanning was subtracted from the energy measured and summarized in Table 3. Only the Leader node was made malicious and sent a contradicting message in the blame phase. The number of messages exchanged and the number of cryptographic operations performed was calculated and using values in Table 1 and Table 2 a theoretical energy cost was computed. We found that these values were close to the practically measured energy consumption, proving that the biggest sources of energy consumption are from communication and cryptographic operations. Table 3 summarizes the energy consumed during the various phases of the protocol as well as the percentage error with respect to the theoretical costs.

7 Protocol Analysis

Since we assume a bounded synchronous network with static faults, the network will always be reliable and the faults if any, must arise from the Byzantine faults.

Notation. We define a vector $\vec{X}$ which consists of the system parameters. For example, $\vec{X} = [n, f, m, S, R, \sigma_s, \sigma_v]^T$, where n is the total number of nodes tolerating f faults; m is the maximum supported payload size; S and R are the costs to send and receive per byte; and σ_s and σ_v are the costs to sign and verify digital signatures.

Let ψ denote the energy cost function of a leader based protocol. The function takes as input $\vec{X}$ and represents the function that computes the energy cost of the protocol. For some constants c_1, c_2, c_3, c_4, c_5, c_6 and c_7, an example $\psi(\vec{X}) = c_1 m + c_2 n + c_3 mn + c_4 mnS + c_5 mnR + c_6\sigma_s + c_7 n\sigma_v$. We drop the vector $\vec{X}$ when comparing two different protocols and just represent it as ψ when obvious.

We denote a particular protocol by placing the name of the protocol in the superscript (e.g., $\psi^{Baseline}$). We denote the best case scenario (without faults) of the protocol by placing a subscript B (for example ψ_B). Similarly, we denote the worst case (with faulty leader and faults) and the view change cost (cost to change the leader) by using the subscripts W and V, respectively (for example, ψ_W and ψ_V). We assume[2] $\psi_V = \psi_W - \psi_B$.

7.1 The need for Best-Case Optimal Protocols

Let $\psi^\star$ represent a given protocol and we wish to design a protocol ψ that is more energy-efficient than $\psi^\star$. Trivially, if $\psi \leq \psi^\star$ always (i.e., $\psi_B \leq \psi_B^\star$ and $\psi_W \leq \psi_W^\star$), then we have already achieved our goal. Therefore, let us consider when this is not true. Let N be the total number of rounds in which V view changes occur. Then using $(N - V)\,\psi_B + V\,(\psi_W) \leq (N - V)\,\psi_B^\star + V\,(\psi_W^\star)$, we get

$$\nu_f = \frac{V}{N} \leq \frac{\psi_B^\star - \psi_B}{\psi_V - \psi_V^\star}$$

Let $\nu_f = \frac{V}{N}$ be the ratio of view changes to total number of consensus rounds. The fraction of best case runs are $1 - \nu_f$ with $0 \leq \nu_f \leq 1$.

Unfavorable conditions. There are two regions of solutions for the above inequality. (i) $\psi_B > \psi_B^\star$ with $\psi_V < \psi_V^\star$, which we call the *worst case optimal* solution set, and (ii) $\psi_B < \psi_B^\star$ with $\psi_V > \psi_V^\star$ which we call the *best case optimal* solution set. This means that we want to either make the protocol best case efficient or worst-case efficient in order for ψ to be more energy efficient than $\psi^\star$. Since the number of static faults are bounded by f, we need a best case optimal ψ, i.e. the *best case optimal* solution set. We also want $\psi_V - \psi_V^\star$ to be as close as possible. In other words, if ψ_V is much better than $\psi_V^\star$, then the number of best case rounds $N - V$ we need to run the protocol will be large, in which case a worst-case optimal solution set may be better.

$$N \geq V\left(\frac{\psi_V - \psi_V^\star}{\psi_B^\star - \psi_B}\right)$$

Dolev & Strong [13] gave a lower bound on the number of rounds required to achieve any agreement:

Theorem 7 (Lower Bound on BA [13])**.** *Byzantine Agreement with authentication can be achieved for n processors with at most f faults within $f + 1$ phases, assuming $n > f + 1$.*

This suggests that in the worst case it is impossible to avoid the $f + 1$ round lower bound. For our synchronous setting, the network is always reliable within the Δ_d parameter. Therefore, in a leader based protocol we will have at most f Byzantine leaders. Therefore, intuitively, protocols in our setting must be designed to be best-case optimal.

Energy Efficiency. Let f_s be Byzantine nodes that attack safety. These nodes may send incorrect messages, but send messages when they must. Let f_c be Byzantine nodes that attack liveness (they do not send messages, but if they do they will send the correct messages). We then have the following safety and connectivity bounds for E2C protocol:

$$n > 2f_s \tag{SB}$$

$$f_c < d_{in} + k - 1 \tag{CB-I}$$

$$f_c < d_{out} + k - 1 \tag{CB-II}$$

Now, we wish to obtain a similar relationship for energy faults f_e. If we want a protocol to be always energy-efficient, then the number of faults $f_e \leftarrow f$ should be such that f_e of the worst cases

[2]Note that we do not assume that $\psi_V > 0$. There exist protocols where the protocol aborts early in case of Byzantine failures such as Tendermint [8], which results in $\psi_V < 0$.

for ψ should still be better than f_e worst cases for $\psi^\star$. Solutions for f that satisfy $\psi(f) \leq \psi^\star(f)$ dictate the energy bound. This is true for example, when ψ and $\psi^\star$ are monotonic functions of f, i.e. $\frac{\partial \psi}{\partial f} \leq \frac{\partial \psi^\star}{\partial f}$.

For our setting, let f_e Byzantine nodes cause the worst case scenario to occur f_e times after which finally a best case scenario occurs. Then using $f_e \cdot \psi_W^{E2C} + \psi_B^{E2C} \leq \psi^{Baseline}$, we obtain

$$f_e \leq \frac{\psi^{Baseline} - \psi_B^{E2C}}{\left(\psi_B^{E2C} + \psi_V^{E2C}\right)} \tag{EB}$$

This bound reiterates that we need to make protocols energy efficient in the best case scenario with lower view-change costs compared to the baseline justifying the fit of E2C when compared to other protocols.

7.2 E2C vs. other SMR protocols

In the absence of Tier 1 nodes, the current best SMR solution is Sync HotStuff [2]. In E2C steady-state, the leader signs once, and all the nodes verify the signature once. As a system there is 1 signature generation and $n-1$ signature verifications. In terms of communication every node receives the message and forwards it only once. Therefore the communication complexity is nd_{out} or nd_{in} for the whole system.

In contrast with Sync HotStuff [2], it has certificates in every block and a voting step where all the nodes vote for the block proposed by the leader. Therefore, they have n signature generations and $2n^2 - n$ signature verifications to verify all the votes and the signatures in the certificate. They also have to send every vote along with the leader's proposal which leads to a communication complexity of $n^2 d_{out}$.

For any $f < \min\left(\min\left(d_{in}, d_{out}\right) + k - 1, n/2\right)$, we can show that E2C protocol is always better than Sync HotStuff [2] using Lemma 8 and Lemma 9.

Lemma 8. *In the steady state,* $\psi_B^{E2C} < \psi_B^{Sync-HotStuff}$

Proof. This follows from the design of E2C. In the steady state, the leader does not include certificates. Secondly in the propose phase, the leader's proposal is voted by all the nodes. We avoid this entirely by marking blocks and making certificates on demand (i.e a view change). □

Lemma 9. $\psi_W^{E2C} > \psi_W^{Sync-HotStuff}$*, when the leader is faulty.*

Proof. Both Sync HotStuff [2] and E2C have a voting phase. blame phase and quit view phase. In both the protocols, the new leader has to send certificate in the next propose phase. The vote certificate phase (in both Blocking and Non-Blocking commit) is the same as the one in Sync HotStuff [2]. Similarly, the phase where all the replicas send their highest certificate to the new leader in a partially connected system is also the same. With optimizations (of protocol simplification and others) we can further reduce the energy costs. E2C for Blocking commit performs slightly worse because in Sync HotStuff [2], there is one voting step, whereas in our protocol we have them separately in relay step and vote step. This leads to slightly bad performance of E2C when compared to Sync HotStuff [2] in the worst case. Similarly, for the Non-Blocking commit, we have two extra rounds: relay and sending the latest committed block to all the nodes. □

Even if our protocol has more rounds than Sync HotStuff [2] (1 or 2 respectively for Blocking and Non-Blocking commit), the loss is easily recoverable from the best case rounds due to significant savings from signature generations and verifications.

Table 4: Analyzing the relationship between m and n by computing solutions for $\psi^{E2C} \leq \psi^{Baseline}$ using an equation of the form $c_1 mn \geq c_2 n + c_3$.

Link Comb.	Public Key Scheme	c_1	c_2	c_3
BLE-WiFi	RSA-1024	$1.66e22$	$5.17e22$	$1.12e25$
	RSA-1260		$3.54e23$	$2.24e25$
	RSA-2048		$1.26e24$	$6.94e25$
	`SECP192R1`		$4.37e25$	$-1.95e25$
	`SECP224R1`		$6.26e25$	$-3.07e25$
	`SECP256R1`		$8.92e25$	$-4.25e25$
BLE-4G	RSA-1024	$3.24e23$	$-7.38e24$	$-5.61e25$
	RSA-1260		$-5.87e24$	$-1.12e26$
	RSA-2048		$-1.32e24$	$-3.47e26$
	`SECP192R1`		$2.11e26$	$-9.74e25$
	`SECP224R1`		$3.05e26$	$-1.53e26$
	`SECP256R1`		$4.38e26$	$-2.12e26$
WiFi-4G	RSA-1024	$1.03e22$	$1.49e24$	$3.50e24$
	RSA-1260		$1.88e24$	$7.01e24$
	RSA-2048		$3.15e24$	$2.17e25$
	`SECP192R1`		$1.43e25$	$-6.09e24$
	`SECP224R1`		$2.03e25$	$-9.59e24$
	`SECP256R1`		$2.87e25$	$-1.33e25$

7.3 Feasible Regions

Let σ_S and σ_V denote the cost of public key signature generation and verification for Tier 2 nodes with σ_m as the signature size. T_S^1 and T_R^1 denote the cost of sending and receiving messages of size m from the link between Tier 1 and Tier 2. Similarly, T_S^2 and T_R^2 denote the cost of sending and receiving messages of size m from the link between Tier 2 nodes. The (A) and (B) notation denote the constants from our linear regression $f(x) = Ax + B$. The constants we obtain are from the hashing costs and HMAC schemes we use. Additionally, for all the analysis, we use $d_{out} = 1$ and $k = 3$.

We analytically count the operations (and cross checked with an NS3 simulation [31]) and present the following equations for $\psi^{Baseline}$, ψ_B^{E2C} and ψ_V^{E2C}.

$$\psi^{Baseline} = \begin{array}{c} 4.26e{-}6mn + 1.52e{-}3n \; + \\ n\left(T_S^1(A) + T_R^1(A)\right) + \\ n\,(m+32)\left(T_S^1(B) + T_R^1(B)\right) \end{array}$$

$$\psi_B^{E2C} = \begin{array}{c} 3.08e{-}4n + \sigma_S + 2.13e{-}6mn \; + \\ (n-1)\,\sigma_V + 3n\left(T_R^2(B) + T_R^2(A)\,(m+\sigma_m+66)\right) + \\ n\left(T_S^2(B) + T_S^2(A)\,(m+\sigma_m+66)\right) \end{array}$$

$$\psi_V^{E2C} = \begin{array}{c} 2.59e{-}4 \; + \; 1.19e{-}4n + 2.13 \times e{-}6mn \; + \\ (n-1)\,\sigma_V + (2n-1)\,\sigma_S + \\ 3n\left(T_R^2(B) + T_R^2(A)\,(m+15\sigma_m+330)\right) + \\ n\left(T_S^2(B) + T_S^2(A)\,(m+15\sigma_m+330)\right) \end{array}$$

Message size and Nodes. We explore the minimum message size, given that we use various public key schemes and using various communication protocols between Tier 2 and Tier 1 nodes.

Modeling protocols as equations we can generalize the comparison between E2C and baseline using a general equation of the form $c_1 mn \geq c_2 n + c_3$ For our concrete setting, we describe the feasibility in Table 4. It can be interpreted by examining the signs of the coefficients. If c_2 and c_3 are negative, then for any $m, n > 0$ E2C is more efficient than the baseline solution. If the $c_2 > 0$ and $c_3 < 0$, then for smaller n, we need a larger payload size m for E2C to be more energy-efficient.

Cryptography. We look at possible cryptographic schemes that RSA and ECDSA can be replaced with. For CPS, this also has applications in designing cryptographic accelerator hardware as they can help reduce the cost of public key primitives. Equating $\psi^{E2C} = \psi^{Baseline}$, we get $c_1\sigma_S + c_2\sigma_V + c_3\sigma_m \leq c_4\mu_S + c_5\mu_V + c_6$. For our concrete setting, with system size n being 1000 and message size m being 2048, using HMAC scheme, we get the following planar region that satisfies $\psi^{E2C} \leq \psi^{Baseline}$.

$$\sigma_S + 999\sigma_V + 8.72e{-}3\sigma_m \leq 1171.18 \tag{BLE-4G Crypto}$$

$$\sigma_S + 999\sigma_V + 8.72e{-}3\sigma_m \leq 4567.35 \tag{WiFi-4G Crypto}$$

$$\sigma_S + 999\sigma_V + 1.09\sigma_m \leq 2281.45 \tag{BLE-WiFi Crypto}$$

Any public key cryptography scheme that satisfies these relationships ensures that E2C is more energy-efficient than the baseline solution.

For RSA, if the modulus size is N, then $\sigma_m = \frac{\log_2 N}{8}$. For ECDSA, if the underlying group is of order 2^p, then $\sigma_m = p + 1$.

Communication. We explore the feasibility of replacing BLE and 4G with different communication media. For our concrete setting, we use RSA-2048 as the cryptographic scheme, system size n is 1000 and message size m is 2048. Using BLE and WiFi as the communication link between Tier 2 nodes, we get Equation (BLE-C) and Equation (WiFi-C) respectively, satisfying which guarantees $\psi^{E2C} \leq \psi^{Baseline}$.

$$2080\left(T_S^1(A) + T_R^1(A)\right) + T_S^1(B) + T_R^1(B) \geq 7.74e{-}2 \tag{BLE-C}$$

$$2080\left(T_S^1(A) + T_R^1(A)\right) + T_S^1(B) + T_R^1(B) \geq 2.46 \tag{WiFi-C}$$

Table 5: Comparison of Best-Case and Worst Case scenarios in related works on SMR. Here, the number of nodes is n; the smallest degree of connectivity of any node is represented by d. Some of the protocols are for fully connected graphs but we compare them in partially connectivity settings where all the protocols leverage multicasts.

Protocol	Correct Leader (Best Case)			Faulty Leader (Worst Case)		
	Communication Complexity	Public Key Operations		Communication Complexity	Public Key Operations	
		Sign	Verify		Sign	Verify
Dolev & Strong [13]	$O(n^2d)$	$O(n)$	$O(n^2)$	$O(n^2d)$	$O(n)$	$O(n^2)$
Abraham *et al.* [1]	$O(n^2d)$	$O(n)$	$O(n^2)$	$O(n^2d)$	$O(n)$	$O(n^2)$
Sync HotStuff [2]	$O(n^2d)$	$O(n)$	$O(n^2)$	$O(n^2d)$	$O(n)$	$O(n^2)$
E2C (This work)	$\mathbf{O(nd)}$	$\mathbf{O(1)}$	$\mathbf{O(n)}$	$\mathbf{O(n^2d)}$	$\mathbf{O(n)}$	$\mathbf{O(n^2)}$

This technique can be used to decide when to perform E2C as opposed to the baseline solution. For example, when the network conditions have time varying energy characteristics such as more energy consumption when the network is noisy, we can simply check the corresponding communication equation to decide. Equation (BLE-C) justifies the result shown in Table 4.

8 Related Work

To the best of our knowledge, there is no work that formally analyzes protocols for energy efficiency and we are the first ones to do so.

SMR and BA. Byzantine agreement (BA) [13] is an agreement problem where all the correct nodes need to agree on a message sent by the sender in the presence of malicious nodes. Informally, a BA protocol satisfies *safety* which requires all correct nodes to commit to the same value and *validity* which requires that if all correct nodes start with value v then all correct nodes must output v.

A key difference between BA and SMR that we consider in this work is the *validity* condition. In BA, if all correct nodes start with a value v, then the output of the protocol for all correct nodes must be v. However, an SMR protocol abstracts the value v as requests from clients, and leaves the validity to the semantics or application layer and not the consensus layer.

We observe that the SMR protocols are not directly applicable to CPS. Some prior works [2, 7, 8, 11, 16, 24, 36] have had a focus on improving the message complexity of SMR. But nodes in CPS tend to be constrained in their energy and computational resources. These protocols do not consider how to optimize the energy consuming aspects which include both cryptography and communication. Protocol designs as of today use votes and cryptographic certificates (aggregation of signatures) while completely ignoring their computational costs. In the absence of signature aggregation techniques as used in [2, 16, 36], these protocols increase the energy requirements for SMR.

Sync HotStuff [2] presents an SMR protocol in an authenticated synchronous fully connected network. It brings down the communication complexity using votes and certificates. Motivated from it, E2C generates certificates *on-demand* when the view changes.

Using Multicasts. Multicasts occur naturally due to the the spatial arrangement of CPS nodes and the omni-directional nature of the wireless communication medium. It does not cost more energy to use a multicast link over a unicast link. There exist works [12,15,17,21,23,30] that make use of multicasts while addressing BA. A common drawback of existing works is that they are not tailored to leverage these multicast properties available in wireless CPS settings.

Koo *et al.* [23] provide a bound on the number of faulty nodes that can be tolerated in the neighborhood of a correct sender while using multicasts to achieve BA. Montemanni *et al.* [30] and Guo *et al.* [17] propose models in determining the optimal transmission power of a sender. These works [17, 23, 30] however, can not be directly used for the CPS settings where the transmission power of the antennae is fixed.

Fitzi *et al.* [15] assume the existence of reliable multicast links between *any* three nodes. Using this assumption they prove that the bound on the number of faulty nodes that can be supported in a fully connected synchronous network can be improved from $n/3$ to $n/2$ to solve BA. Considine *et al.* [12] show that a resilience of $\frac{n}{f} > (k+1)/(k-1)$ can be achieved in a system with n nodes and k-casts, with f of them being faulty. Khan *et al.* [21] consider multicasts as undirected edges in a graph and provide necessary and sufficient conditions for BA in their model.

The main drawbacks in these works [12,15,21] is that there is an assumption that *all* subset of k nodes have a reliable k-cast present. In our work, we consider a weaker assumption in a network where multicasts exist only between neighboring nodes.

9 Conclusion and Future Works

This work aims to underscore the need for energy-efficient consensus protocols in the presence of Byzantine faults. Such energy efficiency is critical in CPS where the nodes are typically energy-constrained. We find that despite the large volume of work in distributed protocols, no existing work has considered the constraints of the CPS environment. We show the need for best-case optimal protocols for efficiency. We also provide a general hypergraph model that can utilize multicasts when available in the system. We propose the E2C protocol, which provides an energy-efficient SMR solution. By leveraging k-casts when available in CPS settings as well as careful use of cryptographic primitives, we achieve energy efficiency. In a system of n nodes with f faulty nodes, we show that in a hypergraph using k-casts, where d_{in} is the minimum in-degree of any node and d_{out} is the minimum out-degree of any node, $f < \min(d_{in}, d_{out}) + k - 1$. Through practical implementations on embedded devices and simulations, we show that E2C protocol consumes less energy than the state-of-the-art SMR protocol Sync HotStuff. Our results motivate further research on best-case optimal protocols for other distributed protocols.

Acknowledgement. This work was supported by grants from ARL. We thank Karthik Nayak for his insightful comments and feedback on the protocol for non-blocking commits. We further thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected $o(1)$ rounds, expected $o(n^2)$ communication, and optimal resilience. *Financial Cryptography and Data Security (FC)*, 2019.

[2] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 654–667.

[3] Ejaz Ahmed, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Imran Khan, Abdelmuttlib Ibrahim Abdalla Ahmed, Muhammad Imran, and Athanasios V Vasilakos. The role of big data analytics in internet of things. *Computer Networks*, 129:459–471, 2017.

[4] ARMmbed. Armmbed/mbedtls, Dec 2019.

[5] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

[6] Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. Sphincs: practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer, 2015.

[7] Alysson Bessani, João Sousa, and Eduardo EP Alchieri. State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE, 2014.

[8] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains.* PhD thesis, 2016.

[9] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the winternitz one-time signature scheme. In *International Conference on Cryptology in Africa*, pages 363–378. Springer, 2011.

[10] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss-a practical forward secure signature scheme based on minimal security assumptions. In *International Workshop on Post-Quantum Cryptography*, pages 117–129. Springer, 2011.

[11] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[12] Jeffrey Considine, Matthias Fitzi, Matthew Franklin, Leonid A Levin, Ueli Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, 2005.

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

[14] PUB Fips. 186-2. digital signature standard (dss). *National Institute of Standards and Technology (NIST)*, 20:13, 2000.

[15] Matthias Fitzi and Ueli Maurer. From partial consistency to global broadcast. In *STOC*, pages 494–503. Citeseer, 2000.

[16] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. Sbft: A scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 568–580. IEEE, 2019.

[17] Song Guo and Oliver WW Yang. Energy-aware multicasting in wireless ad hoc networks: A survey and discussion. *Computer Communications*, 30(9):2129–2148, 2007.

[18] ARM Holdings. Arm mbedtls, 2019.

[19] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 225–238. ACM, 2012.

[20] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[21] Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H Vaidya. Exact byzantine consensus on undirected graphs under local broadcast model. *arXiv preprint arXiv:1903.11677*, 2019.

[22] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K Lenstra, Emmanuel Thomé, Joppe W Bos, Pierrick Gaudry, Alexander Kruppa, Peter L Montgomery, Dag Arne Osvik, et al. Factorization of a 768-bit rsa modulus. In *Annual Cryptology Conference*, pages 333–350. Springer, 2010.

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

[24] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong. Zyzzyva: speculative byzantine fault tolerance. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 45–58. ACM, 2007.

[25] Hyun Jung La and Soo Dong Kim. A service-based approach to designing cyber physical systems. In *2010 IEEE/ACIS 9th International Conference on Computer and Information Science*, pages 895–900. IEEE, 2010.

[26] Manfred Lochter and Johannes Merkle. Rfc 5639-elliptic curve cryptography (ecc) brainpool standard-curves and curve generation. *Federal Office for Information Security of the German Federal Republic, secunet Security Networks, IETF*, 2010.

[27] Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996.

[28] Johannes Merkle and Manfred Lochter. Elliptic curve cryptography (ecc) brainpool standard curves and curve generation. 2010.

[29] Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989.

[30] Roberto Montemanni, Luca Maria Gambardella, and Arindam Kumar Das. The minimum power broadcast problem in wireless networks: a simulated annealing approach. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 4, pages 2057–2062. IEEE, 2005.

[31] Nsnam. nsnam/ns-3-dev-git, Oct 2019.

[32] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy*, pages 144–153. Springer, 2002.

[33] Yanko Sheiretov, Dave Grundy, Vladimir Zilberstein, Neil Goldfine, and Susan Maley. Mwm-array sensors for in situ monitoring of high-temperature components in power plants. *IEEE Sensors Journal*, 9(11):1527–1536, 2009.

[34] Suman Srinivasan, Haniph Latchman, John Shea, Tan Wong, and Janice McNair. Airborne traffic surveillance systems: video surveillance of highway traffic. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 131–135. ACM, 2004.

[35] Ying Tan, Steve Goddard, and Lance C Pérez. A prototype architecture for cyber-physical systems. *ACM Sigbed Review*, 5(1):26, 2008.

[36] Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356. ACM, 2019.