Abstract. We provide an expressive framework that allows analyzing and generating provably secure, state-of-the-art Byzantine fault-tolerant (BFT) protocols. Our framework is hierarchical, including three layers. The top layer is used to model the message pattern and abstract key functions on which BFT algorithms can be built. The intermediate layer provides the core functions with high-level properties sufficient to prove the security of the top-layer algorithms. The bottom layer carefully defines predicates according to which we offer operational realizations for the core functions. All three layers in our framework are extensible and enable innovation. One may modify or extend any layer to theoretically cover all BFT protocols, known and unknown. Indeed, unlike prior BFT frameworks, our framework can analyze and recast BFT protocols in an exceedingly fine-grained manner. More importantly, our framework can readily generate new BFT protocols by simply enumerating the parameters in the framework. In this paper, we show that the framework allows us to fully specify and formally prove the security for 23 BFT protocols, including protocols matching HotStuff, Fast-HotStuff, Jolteon, and Marlin, and among these protocols, seven new protocols outperforming existing ones or achieving meaningful trade-offs among various performance metrics.

1 Introduction

Byzantine fault tolerance (BFT) is the only generic software approach that tolerates arbitrary failures and malicious attacks. BFT is now known as the core building block for permissioned blockchains and is increasingly used in permissionless blockchains. As a classic primitive that regained its prominence in recent years, a myriad of BFT protocols has been proposed. The situation, together with the common belief that there is no one-size-fits-all BFT, unfortunately, quickly turns into a nightmare for scientists, practitioners, and especially for new learners. Indeed, people would have to compare and implement many protocols to convince others their protocols are superior. Reviewers, for instance, would have a hard time telling if a BFT protocol is both valid and novel. Meanwhile, practitioners are easily overwhelmed by the increasing number of protocols and implementations. The situation is only exacerbated by various other issues, such
as the re-invention of existing techniques, subtle design errors (e.g., [1, 11, 31]), and insecure extensions and optimizations to existing protocols (see [11]).

At first glance, the problems discussed above appear inevitable: the BFT protocols are notoriously complex and the BFT techniques are inherently versatile. This paper, however, proposes a new, "unified" framework for BFT replication. First, our framework is highly expressive, capturing and recasting several existing protocols. In all cases, we gain in modularity and simplicity. More important, unlike prior frameworks focusing primarily on explaining existing protocols, our framework can systematically generate novel BFT protocols that outperform existing protocols or offer interesting trade-offs among key metrics, all with rigorous proofs of security. In this work, we study leader-based partially synchronous BFT. As illustrated in Fig. 1, our BG framework is hierarchical and includes three layers. Each layer is extensible.

**Layer 1.** The top layer models the message pattern (e.g., all-to-all communication, one-to-all communication, chained communication) and abstracts core functions on which BFT protocol can be built. The layer also models some key parameters, such as thresholds. As in prior works, our algorithms have a normal-case protocol and a view change protocol. We define two core functions: the \( \text{fsb}(\cdot) \) and \( \text{vv}(\cdot) \) functions for the epicenter of partially synchronous BFT—the view change protocol. As we show in layer 2 and layer 3, the two functions are crucial to the correctness of the protocols and enabling innovation.

The syntax we use to describe Layer 1 algorithms has a circumscribed focus: BFT replication over graph of nodes [31] formalized in HotStuff, where safety is specified through voting and commit graph rules, and liveness is modeled through extending graph with new nodes. HotStuff is a 3-phase (7-step) BFT protocol with optimal linear communication complexity even during view changes. As in HotStuff, all the instantiations in our framework support the chained (pipelined) optimization and the leader rotation strategy.

**Layer 2.** The intermediate layer specifies the core functions and security properties which are *sufficient* to prove the security of our Layer 1 algorithms. In this layer, we reduce the safety and liveness of a BFT protocol to the correctness of the properties of the core functions, making it easy to reason about the correctness of the protocol. We introduce a dichotomy for BFT protocols: \( \text{BG}\{x, z\} \) and \( \text{BG}\{x, y, z\} \). \( \text{BG}\{x, z\} \) models protocols *without lock state*, where the view change rules rely purely on the information collected in the view change messages. \( \text{BG}\{x, y, z\} \) represents protocols *with lock state*, where the view change rules depend on both the information collected during view changes and on the lock *locked* in the \( y \)-th phase of the normal case (for some \( y < z \)).
### Table 1: Representative BG BFT protocols generated using $BG[x,z]$ and $BG[x,y,z]$ for $z \leq 3$. One can have many instantiations for the same parameters (e.g., $BG[1,1,2]$, by using different dominant predicates. $AtoA$ denotes all-to-all communication and $1toA$ represents one-to-all or all-to-one (linear) communication.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
<th>Predicates</th>
<th>Replicas</th>
<th>Message Pattern</th>
<th>Steps</th>
<th>Authenticator Complexity</th>
<th>Message Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Normal-case</td>
<td>View Change</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Normal-case</td>
<td>View Change</td>
</tr>
<tr>
<td>1-phase</td>
<td>Fast-BFT [25]</td>
<td>—</td>
<td>$5f+1$</td>
<td>AtoA</td>
<td>2</td>
<td>$O(n^2)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1]$</td>
<td>DP1</td>
<td>$5f+1$</td>
<td>AtoA</td>
<td>3</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,2]$</td>
<td>DP2</td>
<td>$4f+1$</td>
<td>AtoA</td>
<td>5</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1,2]$</td>
<td>DP4</td>
<td>$3f+1$</td>
<td>AtoA</td>
<td>5</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1,3]$</td>
<td>DP5</td>
<td>$3f+1$</td>
<td>AtoA</td>
<td>5,7</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>2-phase</td>
<td>Fast-Hotstuff [20]</td>
<td>—</td>
<td>$5f+1$</td>
<td>AtoA</td>
<td>3</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1,2]$</td>
<td>DP1</td>
<td>$5f+1$</td>
<td>AtoA</td>
<td>3</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1,2]$</td>
<td>DP2</td>
<td>$4f+1$</td>
<td>AtoA</td>
<td>5</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1,2]$</td>
<td>DP5</td>
<td>$3f+1$</td>
<td>AtoA</td>
<td>5</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,1,2]$</td>
<td>DP5</td>
<td>$3f+1$</td>
<td>AtoA</td>
<td>5,9</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>3-phase</td>
<td>Hotstuff[31]</td>
<td>—</td>
<td>$3f+1$</td>
<td>1toA</td>
<td>7</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$BG[1,2,3]$</td>
<td>DP3</td>
<td>$3f+1$</td>
<td>1toA</td>
<td>7</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

**Layer 3.** The bottom layer carefully defines dominant predicates of the core functions, according to which we provide operational realizations. In this layer, we show that by defining different concrete rules of the core functions (i.e., the dominant predicates) that satisfy the security properties defined in layer 2, one could realize the core functions in a novel way, enabling the generation of new protocols. With the above paradigm, we provide five such dominant predicates (DP1 to DP5), each of which can lead to novel BFT protocols. With our framework specified and key theorems proved, we can generate BFT protocols by simply enumerating the parameters.

**Instantiations.** For each dominant predicate, one could enumerate the parameters $x$, $y$, and $z$ for $BG[x,z]$ and $BG[x,y,z]$ ($z \leq 3$) to generate BFT protocols. In total, we obtain from our framework 23 candidate BFT protocols. Among them, seven are strictly better than others, improving some existing protocols in at least one aspect, as illustrated in Table 3.

For 3-phase protocols, $BG[1,2,3]$ with the predicate DP3 is similar to HotStuff (with only minor differences) and achieves the same complexity as HotStuff.

For 2-phase protocols, $BG[1,2]$ with DP3 is similar to Fast-Hotstuff [20] and Jolteon [16], both of which can be viewed as a 2-phase version of HotStuff. Besides, we generate three novel protocols. $BG[1,1,2]$ with DP1 requiring $5f+1$ replicas has linear authenticator complexity and message complexity but has one less phase (two steps) than HotStuff. $BG[1,1,2]$ with DP2 requiring $4f+1$ replicas has linear authenticator complexity and message complexity. The other $BG[1,1,2]$ instantiation for DP5 with $n \geq 3f+1$ replicas has optimal complexities for both the normal-case and the view change. The protocol has $O(n)$ authenticator complexity during view changes by default and $O(n^2)$ complexity only in rare cases. The protocol strictly outperforms Fast-HotStuff, especially when a rotating leader strategy is used. We also show how DP5 leads to Marlin [30] ($BG[1,1,2]^*$), a BFT protocol with linearity and only 2 phases in normal cases.
As a 1-phase protocol, BG[1, 1] with DP1 improves FaB5 [23] in terms of all complexity metrics for both normal case and view change. BG[1, 1] has 3 steps (the minimum number of steps that is derived from our framework). Both BG[1, 1] and FaB5 assume $n \geq 5f + 1$.

Besides, our framework can capture protocols with weak liveness (in DP4), a property achieved by some existing protocols (e.g., Tendermint [8], Casper [9]). The notion is known as a "bad practice." We show that, by adopting another dominant predicate (DP5), we could transform protocols with weak liveness to ones with optimistic responsiveness.

**Discussion.** While our work is not the first to propose a generic framework for fault-tolerant distributed computing, our framework goes far beyond previous approaches. The HotStuff framework focuses on syntactically a casting of four existing protocols and one of their own. Regarding Byzantine agreement protocols, a number of generic algorithms have been proposed for both benign failures (e.g., [25, 18, 22]) and Byzantine failures [28, 26, 19, 24]. Our framework has much more fine-grained modeling on various parameters, systematically leading to a large number of interesting and cleanly specified protocols, including the state-of-the-art BFT protocols with linearity. We believe our framework is right in the sweet spot of what is required for an efficient BFT framework.

## System Model

**BFT Model.** We consider a BFT system consisting of $n$ replicas, where $f$ of them may fail arbitrarily (Byzantine failures). Let $C$ be the set of correct replicas in the system. We consider the partially synchronous model [15], where there exists an unknown global stabilization time (GST) such that after GST, messages sent between two correct replicas arrive within a fixed delay.

**Cryptographic building blocks.** We use a $(t, n)$ threshold signature scheme consisting of the following algorithms ($tgen$, $tsign$, $tcombine$, $tverify$) [27, 7]. $tgen$ outputs a system public key and a vector of $n$ private keys. A partial signature signing algorithm $tsign$ takes as input a message $m$ and a private key $sk_i$ and outputs a partial signature $\sigma_i$. A combining algorithm $tcombine$ takes as input $pk$, a message $m$, and a set of $t$ valid partial signatures, and outputs a signature $\sigma$. A signature verification algorithm $tverify$ takes as input $pk$, a message $m$, and a signature $\sigma$, and outputs a bit. We also use a collision-resistant hash function $hash$ mapping a message of arbitrary length to a fixed-length output.

## Syntax and Properties for BFT over Graphs

The first layer in our BG framework to be described extends the syntax of the BFT replication over graphs (trees) of nodes [31]. A leader-based, partially synchronous BFT protocol has a normal-case protocol and a view change protocol (triggered periodically or when the leader appears faulty). The BFT protocols proceed in a succession of views numbered by monotonically increasing view numbers. The view number maintained by a replica is denoted as $cview$. Each view has a unique leader. Every replica can obtain the identity of the leader by calling Leader($cview$). Each replica stores a tree of nodes, as in Fig. 2. Each
node is also known as a block, denoted as \( b \). We use nodes and blocks interchangeably. A block contains a batch of client requests \( req \), a parent link \( pl \), and their metadata. A parent link for \( b \) is a unique identifier to its parent node, e.g., a hash digest of a parent node. A branch led by a given block \( b \) is the path from \( b \) all the way to the root of the tree. The metadata for a block \( b \) include \textit{height} and \textit{view}: \textit{height} for \( b \) is the number of blocks on the branch led by \( b \), while \textit{view} for \( b \) is the view during which \( b \) is added to the tree. The \textit{view} for \( b \) is equal to or greater than that of the parent block of \( b \). Two branches are \textit{conflicting}, if neither one is an extension of the other. Two nodes are conflicting if the branches led by them are conflicting. In BFT, a monotonically growing branch becomes committed. Each time a block extends the branch led by its parent block. A block \( b' \) is an extension of \( b \) if \( b \) is on the branch led by \( b' \).

An example is illustrated in Fig. 2. \( b_1 \) is committed in view 1, while \( b_2 \) and \( b_3 \) are committed in view 2. A branch led by \( b_2 \) is the path from \( b_2 \) to \( b_0 \). \( b_3 \) is an extension of \( b_2 \) and also an extension of \( b_1 \). The height of \( b_3 \) is 4, equal to the depth of the tree. The parent link for a block \( b_2 \) is a hash of its parent block \( b_1 \), \( b_3 \) and \( d_3 \) are conflicting, as the branches led by them are conflicting.

Note that our syntax is more general than that of HotStuff. In HotStuff, the leader rotates and each replica proposes only one block in its turn. In our syntax, a leader may propose one block before it is replaced (if the leader rotation strategy is used) or many blocks (as in conventional BFT protocols).

We recast the safety definition for BFT replication in the language of graph of blocks. For liveness, we adopt the notion of optimistic responsiveness [31].

- **Safety I**: If \( b \) and \( d \) are conflicting blocks, then they cannot be both committed in the same view, each by at least a correct replica.
- **Safety II**: If \( b \) and \( d \) are conflicting blocks, then they cannot be committed in different views, each by at least a correct replica.
- **Optimistic responsiveness**: After GST, any correct leader needs just to wait for at most \( n - f \) responses to guarantee that it can create a proposal that will make progress.

4 BG Framework: Layer 1

4.1 High-Level Overview

As illustrated in Fig. 3, the normal-case protocol in our framework consists of \( z \) successive phases, where \( z \leq 3 \) and each phase involves only linear communication. After the \( z \) phases, there is a \textit{commit} step that involves only a best-effort broadcast. Among the \( z \) phases, the 1st phase deserves a careful specification,
as replicas need to decide whether to vote for a block. The view change protocol is triggered periodically or when the current leader is suspected to be faulty. As shown in Fig. 3, the new leader collects new-view messages signaling a view change, selects a safe branch using a fsb() function, extends the branch, and broadcasts the new block in a view-update message. Each replica verifies the new block by executing the vv() function. Replicas then switch to the normal-case operation.

4.2 Data Structures

Messages. We use \(\langle \text{type}, \text{view}, \text{height}, \text{block}, \text{justify} \rangle\) to denote the messages sent among replicas. Some fields can be set as \(\bot\). For the normal-case operation, \(m.\text{type} \in \{\text{msg}-j, \text{vote}-j, \text{commit}\}\), where \(j \in [1..z]\) is the phase number. As shown in Fig. 3, the \(j\)-th phase involves two steps: the leader broadcasts a msg-\(j\) message and replicas votes for it with a vote-\(j\) message. \(m.\text{height}\) is set as the height of \(m.\text{block}\). \(m.\text{justify}\) is an optional field that the leader uses to carry a quorum certificate \(QC\) (to be described shortly) in msg-\(j\) messages. For the view change protocol, \(m.\text{type} \in \{\text{new-view}, \text{view-update}\}\). When \(m.\text{type} = \text{new-view}\), \(m.\text{block}\) and \(m.\text{height}\) are set as \(\bot\). When \(m.\text{type} = \text{view-update}\), \(m.\text{block}\) is an associated block and \(m.\text{height}\) is the height of the block. In both cases, \(m.\text{justify}\) contains information needed for a correct view change.

Quorum certificates. A quorum certificate for a message (defined above) is a data type that combines a collection of signatures for the same tuple signed by \(t < n\) signatures. We use a \((t, n)\) threshold signature to reduce the authenticator complexity. In this setting, replicas run the \(t\text{sign}\) algorithm to generate partial signatures for a message. One can run the \(t\text{combine}\) algorithm to combine the \(t\) partial signatures into a threshold signature. A quorum certificate for a message \(m\) is valid if the threshold signature is valid. To distinguish the authenticator using partial signatures, we let \(qc\text{Vote}(m)\) denote the output of the \(t\text{sign}\) algorithm for \(m\). Otherwise, we implicitly use a signature or a threshold signature for authentication. To hide the implementation detail, let \(qc\text{Create}(m)\) be a quorum certificate generated for \(m\).

For different messages, we may use different thresholds for their quorum certificates. The quorum certificate for a vote-\(j\) message \(m\) is denoted by \(QC_j\) and also by \(b.QC_j\) where \(m.\text{block} = b\). \(b.QC_j\) is also called a \(QC_j\) for \(b\). The threshold for \(QC_j\) is set as \(T_j\). For any quorum certificate \(qc\), if \(qc\) is a QC for block \(b\), we let \(qc\text{Block}(qc)\) denote \(b\).

Rank of QCs and blocks. We introduce a notion of rank for QCs and blocks, which is similar to that defined in [16]. For each block \(b\), \(\text{rank}(b)\) depends on \(b.\text{view}\) and \(b.\text{height}\). We only care if the rank of a block is higher than that of another one. Blocks are compared lexicographically by rank (i.e., first by the view number, then by the height). In addition, the rank of \(b.QC_j\) is defined the same as that of block \(b\).

Local state at replicas. Each replica maintains several state parameters, including the current view number \(c\text{view}\), the highest quorum certificates (received in different phases) \(QC_1, \cdots, QC_z\), and last voted block \(vb\). In protocols with
lock state, replica needs to store locked block $lb$, criticalState of a replica contains several parameters in local state, which contains information replicas send in the NEW-VIEW messages.

Note that the data structures for messages and quorum certificates are more general than those defined in HotStuff. In our framework, we introduce a system parameter flag. The flag is used to specify whether $vb$ should be taken into account in selecting the safe branch during view changes.

4.3 Normal-Case Protocol (Algorithm 1)

In the normal-case protocol, there are $z$ phases. Each phase includes two steps with linear communication.

- In phase 1, the leader extends a branch in the tree it maintains with a new block $b$, and broadcasts a MSG-1 message $m$ for $b$ to all replicas, where $m.justify$ is set to $QC_x$. Upon receiving $m$, each replica verifies whether the rank of $b$ is larger than that of the last voted block $vb$. If so, the replica updates its $QC_x$ with $m.justify$ and then sends the leader a signed VOTE-1 message for $b$.
- In phase 2 to phase $z$, replicas repeat the same procedure. In particular, in the $j$-th phase, after collecting $T_{j-1}$ matching threshold signature shares for $b$, the leader combines them into a $b.QC_{j-1}$, broadcasts $b.QC_{j-1}$ in a MSG-1 to all replicas and enters the next phase. Upon receiving a valid MSG-1 message, a replica updates its $QC_{j-1}$ and sends the leader a signed VOTE-j message for $b$. In BG[$x, y, z$], if a replica voted for a block in the $(y + 1)$-th phase, it sets its $lb$ to the block at the same time.
- In the commit step, the leader broadcasts $b.QC_{z}$ in a COMMIT message. Upon receiving a valid COMMIT message, each replica commits the corresponding block.

---

Algorithm 1: Normal-case protocol for $p_i$

1. **localstate**: the current view $cview$, the last voted block $vb$, the locked block $lb$ in BG[$x, y, z$], certificates $QC_1, QC_2, \ldots, QC_z$.
2. **parameters**: thresholds for different voting phases $T_1, T_2, \ldots, T_z$.
3. As a leader:
   4. //Leader(cview) = $p_i$
   5. - propose a block $b$ extending $qcblock(QC_2)$, broadcast (MSG-1) message $m$ for $b$, where $m.justify \in QC_z$.
   6. - upon receiving $T_j$ signed (VOTE-j) messages for block $b$: // $j \in [1, z]$,
     7. - propose $b.QC_j$, update $QC_j \leftarrow b.QC_j$.
     8. - broadcast a (MSG-(j+1)) message $m$ for $b$, where $m.justify \in QC_z$.
   9. - upon receiving $T_z$ signed (VOTE-z) messages for block $b'$:
     10. - broadcast a COMMIT message $m$ for $b$, where $m.justify \in QC_z$, propose another block.
   11. As a replica:
     12. - upon receiving a valid (MSG-1) message for a block $b$: //rank($b$) > rank($vb$)
     13. - update $vb \leftarrow b$, $QC_x \leftarrow m.justify$ //vote for a new block
     14. - send a (VOTE-1) message for $b$ to Leader(cview).
     15. - upon receiving a valid (MSG-2) message $m$ for $b$: // $j \in [1, z]$
     16. - update $QC_{j+1} \leftarrow m.justify$, send a (VOTE-j) message for $b$ to Leader(cview).
     17. - upon receiving a valid (MSG-(y+1)) for $b$ in BG[$x, y, z$]: update $lb \leftarrow b$
     18. - upon receiving a valid COMMIT message $m$ for $b$:
     19. - update $QC_{j+1} \leftarrow m.justify$, commit $b$ //execute the requests in $b$ in order
     20. - wait for a (MSG-1) message for another block.
#### Algorithm 2: View change protocol for $p_i$

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$\text{criticalState}$: contains many variables in localstate specified in layer 3.</td>
</tr>
<tr>
<td>2.</td>
<td>$\text{parameters}$: thresholds $T_1$, $T_2$, $\ldots$, and $T_4$, and the view change threshold $T$.</td>
</tr>
<tr>
<td>3.</td>
<td>$\bullet \text{ upon timeout}$:</td>
</tr>
<tr>
<td>4.</td>
<td>- update $\text{cview} \leftarrow \text{cview} + 1$.</td>
</tr>
<tr>
<td>5.</td>
<td>- send the $\text{criticalState}$ in a new-view message to $\text{Leader}(\text{cview})$.</td>
</tr>
<tr>
<td>6.</td>
<td>$\text{As a new leader:}$</td>
</tr>
<tr>
<td>7.</td>
<td>$\bullet \text{ upon receiving a set } M \text{ of } T \text{ new-view messages for view } \text{cview}$:</td>
</tr>
<tr>
<td>8.</td>
<td>- $(b', \pi) \leftarrow \text{fsb}(M)$, propose a block $b'$ extending $b'$.</td>
</tr>
<tr>
<td>9.</td>
<td>- broadcast a $\text{view-update}$ message $m$ for $b'$, where $m.\text{justifying is } \pi$.</td>
</tr>
<tr>
<td>10.</td>
<td>- $(\text{flag} = 1)$: vote for $b$ but do not update localstate.</td>
</tr>
<tr>
<td>11.</td>
<td>$\text{wait until } b.QC_b \text{ is collected:}$</td>
</tr>
<tr>
<td>12.</td>
<td>- update $QC_p \leftarrow b.QC_p$, propose a block $b_i$ extending $b$.</td>
</tr>
<tr>
<td>13.</td>
<td>- broadcast $b_i$ in a $(\text{MSG}_1)$ message $m$, where $m.\text{verify is } b.QC_x$.</td>
</tr>
<tr>
<td>14.</td>
<td>- switch to normal case operation.</td>
</tr>
<tr>
<td>15.</td>
<td>$\text{As a replica:}$</td>
</tr>
<tr>
<td>16.</td>
<td>$\bullet \text{ upon receiving a } \text{view-update}$ message $m$ from $\text{Leader}(\text{cview})$ for a block $b$:</td>
</tr>
<tr>
<td>17.</td>
<td>- if $\text{vv}(m) = 0$ in $\text{BG}[x, y, z]$ or $\text{vv}(m, lb) = 0$ in $\text{BG}[x, y, z]$, discard $m$.</td>
</tr>
<tr>
<td>18.</td>
<td>- broadcast a $(\text{VOTE}_1)$ message for $b$ to $\text{Leader}(\text{cview})$.</td>
</tr>
<tr>
<td>19.</td>
<td>$(\text{flag} = 1)$: vote for $b$ but do not update localstate.</td>
</tr>
<tr>
<td>20.</td>
<td>$\text{wait until } a(\text{MSG}_1)$ message $m$ for a block $b'$ extending $b$ is received:</td>
</tr>
<tr>
<td>21.</td>
<td>- update $\text{vote } b \leftarrow b$, $\text{QC}_a \leftarrow m.\text{justifying}$, send a $(\text{VOTE}_1)$ message for $b'$ to $\text{Leader}(\text{cview})$.</td>
</tr>
<tr>
<td>22.</td>
<td>- switch to normal-case protocol.</td>
</tr>
</tbody>
</table>

### 4.4 View Change Protocol (Algorithm 2)

We now present the view change protocol. Crucially, we introduce two key functions for the view change protocol, FSB() and VV(). Intuitively, FSB() is used for the leader to obtain a safe branch to extend during view changes. VV() is used for replicas to decide whether to accept a VIEW-UPDATE message.

Similar to prior works, a timer is started when a replica enters a new view or when a replica waits for a message from the current leader. When the timer of replica $p_i$ expires, a $p_i$ triggers view change by incrementing $\text{cview}$ by one. Then $p_i$ sends $\text{criticalState}$ in a new-view message to the next leader.

The new leader collects a set of $T$ new-view messages, denoted as $M$. It then executes $\text{fsb}(M)$ to obtain $(b', \pi)$, where $b'$ is a block and $\pi$ is a proof that $b'$ is a safe block to extend. Then the leader extends the branch led by $b'$ with a new block $b$ and broadcasts $b$ in a VIEW-UPDATE message $m$. There are two cases depending on the parameter $\text{flag}$ (specified in Sec. 6). If $\text{flag} = 0$, the leader directly switches to normal-case protocol. If $\text{flag} = 1$, the leader still switches to phase 2 but does not update its $\text{lockState}$ until $b.QC_x$ is collected.

A replica accepts a VIEW-UPDATE message $m$ in view $v$ from the new leader only if $\text{vv}(m)$ outputs 1 in $\text{BG}[x, y, z]$ or $\text{vv}(m, lb)$ outputs 1 in $\text{BG}[x, y, z]$. Then the replica sends a VOTE−1 message for $m.\text{block}$. Similar to the two cases for the leader, according to the $\text{flag}$ parameter, the replica may take different actions. If $\text{flag} = 0$, the replica switches to normal-case protocol. If $\text{flag} = 1$, the replica still votes for the first block $b$ proposed by the new leader but does not update localstate or commit $b$. If later the replica receives a $\text{MSG}_1$ message for $b'$ that extends $b$, the replica then votes for $b'$ and switches to normal-case protocol.

We present pseudocode and discuss the details of layer one in Appendix B.
5 BG Framework: Layer 2

Overview of layer 2 in BG. This section specifies properties for the core functions—\texttt{FSB()} and \texttt{VV()}. These properties are \textit{sufficient} to prove the security of our Layer 1 algorithms. However, one can define other appropriate properties that may lead to secure BFT protocols.

When defining properties for our functions, we explicitly distinguish safety properties and liveness properties, which correspond to the safety and liveness of the BFT protocols. The fine-grained characterization of function properties allows us to better understand our framework and generate novel BFT protocols.

We then introduce a dichotomy for BFT protocols: \text{BG}^x, z and \text{BG}^x, y, z. The two kinds of BFT protocols are different in terms of the information each replica maintains in its local state, which could lead to novel BFT protocols. We then define properties of \texttt{FSB()} and \texttt{VV()} for \text{BG}^x, z and \text{BG}^x, y, z, respectively. Finally, we show that with these properties properly defined in Layer 2 of our framework, any protocol generated by our framework is safe and live.

\text{BG}^x, z and \text{BG}^x, y, z. As mentioned earlier, we introduce both \text{BG}^x, z (x \leq z) and \text{BG}^x, y, z (where x \leq y < z). The properties of the core functions are different, as \text{BG}^x, z and \text{BG}^x, y, z have rather different features.

Both \text{BG}^x, z and \text{BG}^x, y, z are z-phase protocols. \text{BG}^x, z models protocols \texttt{without lock state}, where the view change rules rely purely on the information provided by replicas during the view change. \text{BG}^x, y, z represents protocols \texttt{with lock state}, where the view change rules rely on the information collected during the view change and the information on the block \textit{locked} in the y-th phase (for some y < z).

The parameter x applies to both \text{BG}^x, z and \text{BG}^x, y, z. Let b be a block and m be a \texttt{msg-1} message for b. \texttt{m.justify} is set as \texttt{b}'.QC_x, where \texttt{b}' is the parent block of \texttt{b} and QC_x is the QC formed in the x-th phase for \texttt{b}'.

The parameter y in \text{BG}^x, y, z represents the phase where a replica updates its state parameter \texttt{lockState} in the normal-case protocol. We say \texttt{b} a \textit{locked block}, if at least one correct replica has updated its \texttt{lockState} to \texttt{b}. Intuitively, \texttt{lockState} is crucial for a replica in \text{BG}^x, y, z to decide whether to accept a \texttt{view-update} message.

\textbf{A key lemma and defining b'.} We now present Lemma 1, a key lemma to specify the properties of \texttt{FSB()} and \texttt{VV()}. The lemma essentially claims that before a view \texttt{v}, there exists a unique block \texttt{b'} committed with the highest rank. The existence of block \texttt{b'} is essential in defining function properties. Intuitively, from the protocol perspective, the core functions should ensure that block \texttt{b'} stills remain committed after the view change.

\textbf{Lemma 1.} Let \texttt{B' = \{b | block b has been committed before view v\}}. If \texttt{T_j > f} for \texttt{j \in [1..z]}, and \texttt{T_j \geq \left\lceil \frac{\texttt{n-1}}{2}\right\rceil}, then there exists \texttt{b' \in B'} such that for all \texttt{b' \in B'} and \texttt{b' \neq b'}, we have \texttt{mnk(b')} > \texttt{mnk(b')}.\texttt{FSB()} and \texttt{VV()} for \text{BG}^x, z. We now elaborate the properties of \texttt{FSB()} and \texttt{VV()} functions for \text{BG}^x, z. There is flexibility in defining the properties of the functions. These properties are just some sufficient conditions to prove the algorithms in the Layer 1 framework. Given a set \texttt{M_v} of \textit{T new-view} messages (where we call the set \texttt{M_v a view change snapshot}), \texttt{FSB()} takes as input
Given a view-update message $m$, let $b$ denote the parent block of $m$\(_{\text{block}}\)$, and let $v$ denote $m$\(_{\text{view}}\)$. $\text{VV}(\cdot)$ takes as input $m$ and outputs a binary value, representing whether a replica will accept $m$ in view $v$. We consider VV-safety and VV-liveness properties for $\text{VV}(m)$. Intuitively, VV-safety requires that a correct replica will not vote for a conflicting block of $b^v$. Our definition is actually stronger: it requires there exists a set $M_v$ that is the input of $\text{FSB}()$. Essentially, VV-safety ensures that a committed block by any correct replica will remain committed after the view change.

- **FSB-safety**: If $\text{FSB}(M_v)$ outputs $(b, \pi)$, then $\pi$ is a proof that $b$ is either $b^v$ or an extension of $b^v$.
- **FSB-liveness**: Let $T \leq n - f$. $\text{FSB}(M_v)$ outputs some $(b, \pi)$.
- **VV-safety**: $\text{VV}(m)$ outputs 1 by a correct replica only if there exists a set $M_v$ such that $(b, m$\(_{\text{justify}}\))$ is the output of $\text{FSB}(M_v)$.
- **VV-liveness**: If $\text{FSB}(M_v)$ outputs $(b, m$\(_{\text{justify}}\)), \text{VV}(m)$ outputs 1.

Both $\text{FSB}()$ properties and $\text{VV}()$ properties are carefully defined. Several alternative approaches to defining properties fail to "work." All these properties are defined in a way that is neither too restricted nor too broad. We comment, however, that one can define other appropriate properties for the two functions.

**FSB() and VV() for BG\([x, y, z]\).** We now turn to BG\([x, y, z]\). Compared to BG\([x, z]\), $\text{VV}(\cdot)$ in BG\([x, y, z]\) takes as input an additional value lockState locally maintained by each replica. To distinguish the properties in BG\([x, y, z]\) from those in BG\([x, z]\), we add an "L" (standing for "Lock state") for all the properties.

$\text{FSB}()$ takes as input $M_v$ and outputs some $(b, \pi)$. We find that we do not have to define the safety property, as BG\([x, y, z]\) has the lockState that is crucial to ensure safety. Accordingly, we only define a liveness property that happens to be identical to that of BG\([x, z]\), i.e., FSBL-liveness. Given a view-update message $m$, let $b$ denote the parent block of $m$\(_{\text{block}}\)$ and $v$ denote $m$\(_{\text{view}}\)\(_{\text{view}}\). $\text{VV}(\cdot)$ takes as an input $m$ together with lockState of a replica and outputs a binary value. We define both safety and liveness properties for $\text{VV}(m, \text{lockState})$ in VVL-safety and VVL-liveness.

- **FSBL-liveness**: Let $T \leq n - f$. $\text{FSB}(M_v)$ outputs some $(b, \pi)$.
- **VVL-safety**: Let $P = \{p_i\mid p_i \in C$ (the set of correct replicas), $\text{VV}(m, \text{lockState})$ outputs 1 by $p_i$ in view $v\}$. If $b$ is conflicting with $b^v$ or $b$\(_{\text{height}}\) is lower than $b^v$\(_{\text{height}}\), then $|P| < T_1 - f$.
- **VVL-liveness**: If $(b, m$\(_{\text{justify}}\))$ is the output of $\text{FSB}(M_v)$ function on some $M_v$, $\text{VV}(m, \text{lockState})$ outputs 1 at all correct replicas.

Above, VVL-safety intuitively requires that not so many correct replicas will vote for a conflicting block of $b^v$. VVL-safety ensures that a block $b^v$ committed by any correct replica will remain committed after the view change. On the other
hand, VVL-diveness intuitively ensures that all correct replicas will move to the new view after receiving the view-update message from a correct leader.

**Correctness for BG\([x, z]\) and BG\([x, y, z]\).** We now present the following core theorems showing how the framework parameters and the properties of the core functions affect the safety and liveness of the BG\([x, z]\) or BG\([x, y, z]\) protocols.

**Theorem 1.** BG\([x, z]\) achieves safety-I, if \(T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil\) and \(T_j > f\) for all \(j \in [1..z]\).

**Theorem 2.** BG\([x, z]\) achieves safety-II, if \(T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil, T_j > f\) for all \(j \in [1..z]\), and FSB-safety and VV-safety hold.

**Theorem 3.** BG\([x, z]\) achieves optimistic responsiveness, if \(T_1 \leq n - f\) for all \(j \in [1..z]\), \(T \leq n - f\), and FSB-liveness, and VV-liveness hold.

**Theorem 4.** BG\([x, y, z]\) achieves safety-I, if \(T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil\) and \(T_j > f\) for all \(j \in [1..z]\).

**Theorem 5.** BG\([x, y, z]\) achieves safety-II, if \(T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil, T_j > f\) for all \(j \in [1..z]\), and VVL-safety holds.

**Theorem 6.** BG\([x, y, z]\) achieves responsiveness, if \(T_j \leq n - f\) for all \(j \in [1..z]\), \(T \leq n - f\), and FSB-liveness and VV-liveness hold.

These theorems are fine-grained: we pinpoint the conditions "needed" for safety-I, safety-II, and optimistic responsiveness of BFT. The results provide important insights on BFT protocols in our framework and facilitate the design of new BFT protocols. For instance, with Theorem 1 and Theorem 2, while designing a BFT protocol using our framework, one may set \(T_1 = \left\lfloor \frac{n+f+1}{2} \right\rfloor\) and try to set some \(T_j\) for \(j \in [2..z]\) as \(f+1\).

### 6 BG Framework: Layer 3

This section provides the most technical and innovative part of our framework: the realizations of the \(\text{fsb}()\) and \(\text{vv}()\) functions satisfying the properties we define in our Layer 2 framework.

We find that directly working on \(\text{fsb}()\) and \(\text{vv}()\) for the set \(M_v\) (the view change snapshot for view \(v\)) is difficult. Intuitively, there are just many ways of forming \(M_v\), and it is hard to enumerate all meaningful choices. Thus, we take a detour and propose a real vs. virtual paradigm with the following steps:

- Rather than directly defining \(\text{fsb}()\) and \(\text{vv}()\) for \(M_v\) (a real view change snapshot), we introduce the concept of virtual view change snapshot \(M^b\) for a block \(b\). Intuitively, the information included in a virtual snapshot \(M^b\) is mainly decided by the state (committed or locked) of \(b\).

- We define for \(M^b\) dominant predicates that specify criticalState and the \(\text{fsb}()\) and \(\text{vv}()\) functions. We can show that realizations of \(\text{fsb}()\) when taking \(M^b\) as input—according to these dominant predicates—can satisfy the properties defined in layer 2.
We are finally able to prove that using our framework, as long as all committed blocks in $BG[x, z]$ (resp., locked blocks in $BG[x, y, z]$) satisfy a dominant predicate, then our realizations for $FSB()$ and $VV()$ when replacing $M^b$ (virtual) using $M_v$ (real) would also satisfy the properties defined in layer 2.

Under this paradigm, we design five interesting dominant predicates that realize $FSB()$ and $VV()$. In our framework, these realizations lead to useful BFT protocols achieving safety and optimistic responsiveness. One can, however, define new predicates, enabling novel BFT constructions.

### 6.1 Defining Virtual Snapshot $M^b$ for a Block $b$

We begin by defining virtual (view change) snapshot on which our predicates and realizations of $FSB()$ and $VV()$ can be built.

Like a real view change snapshot, a virtual snapshot $M^b$ also contains a set of $T$ NEW-VIEW messages from replicas. The difference is that $M^b$ is associated with a block $b$. In particular, $M^b$ contains the information stored at replicas regarding $b$. Let $v$ denote $b$’s view. We show an example in Fig. 4, where blocks $b'$, $b$, and $b''$ are all proposed in view $v$, $b'$ is on the branch led by $b$, and $b''$ extends $b$. We distinguish three cases for the criticalState contained in each $m^v$ from a correct replica $p_i$:

- **Case 1 (Fig. 4 (a))**: Block $b$ is the highest block $p_i$ has voted for in view $v$. Then the criticalState in $m$ contains the block (and certificates) stored by $p_i$ with the highest rank in view $v$. Specifically, criticalState could contain $b$, $b.QC_1$, ..., $b.QC_{z-1}$, and $b'.QC_z$.

- **Case 2 (Fig. 4 (b))**: Replica $p_i$ has voted for $b'$ but not $b''$, and not $b$. Then the criticalState in $m$ contains the block (and certificates) stored by $p_i$ with a lower height than that of $b$ in view $v$. In this case, while $p_i$ has stored information for $b''$, criticalState could contain only $b'$ and certificates for $b'$.

- **Case 3 (Fig. 4(c))**: Replica $p_i$ has voted for $b'$, $b$, and $b''$. Then the criticalState in $m$ contains the block (and certificates) stored by $p_i$ with the same rank with $b$ in view $v$. Specifically, criticalState could only contain $b$ and certificates for $b$.

We consider the real state of $b$ at a correct replica before the view change (i.e., locked, committed, neither locked nor committed). For a block $b^v$ defined in layer 2, a correct protocol should ensure that the real state of $b^v$ can be obtained from $M_v$. If there exists a correct $FSB()$ function satisfying the properties defined in layer 2, then the output of $FSB(M_v)$ should be $b^v$ or an extension of $b^v$. Hence, $b^v$ still remains committed after the view change.

$M^b$, emphasizing the information for block $b^v$, focuses on whether the real state of $b^v$ can be obtained. Therefore, $M^b$ can be viewed as a special case of $M_v$ but includes more information about $b^v$. Indeed, criticalState in $m$ is $b^v$.

$A$ branch including $b$:

\begin{center}
\begin{tabular}{ccc}
(a) & & \\
(b) & & \\
(c) & & \\
\end{tabular}
\end{center}

**Fig. 4**: Examples of virtual view change snapshots.
sent by a correct replica always contains information the replica stored for block \( b^v \), while the replica may have changed its localstate when view change occurs. If \( m \in M \) is sent at another moment, \( m \) may not directly include any evidence about the real state of \( b^v \). Accordingly, a correct \( \text{FSB}() \) function should also output \( b^v \) or an extension of \( b^v \) taking \( M^{b^v} \) as input. Later on in this section, we define dominant predicates for virtual snapshots and show how they can be used to construct \( \text{FSB}() \) and \( \text{VV}() \) accordingly.

### 6.2 Dominant Predicates and Realizations of \( \text{FSB}() \) and \( \text{VV}() \)

We now present the five predicates we introduce in the paper. Depending on the information (criticalState) each replica provides in the new-view message, we could capture the properties of the virtual snapshots to define the predicates. We focus on two critical information: each replica's last voted block \( vb \) and the highest QC. In DP1, the criticalState contains \( vb \) and \( QC_x \). Via DP1, BFT protocols achieving optimal complexity with \( 5f + 1 \) replicas can be derived. DP2 modifies DP1 and allows us to cover BFT protocols with \( 4f + 1 \) replicas. In contrast, in DP3 and DP4, the criticalState only contains \( QC_x \) and the corresponding \( \text{FSB}() \) and \( \text{VV}() \) functions are also simpler. Using DP3, we can obtain many interesting BFT protocols with \( 3f + 1 \) replicas, including \([20]\) and \([31]\). In DP4, we aim at formalizing protocols with weak liveness such as Tendermint \([8, 9]\) and Casper \([9]\). DP5 is based on DP4 and the criticalState contains \( vb \) and \( QC_x \). Compared with DP4, we elaborate the additional information (\( vb \)) contained in criticalState in DP5 and essentially turn protocols with weak liveness property into ones with optimistic responsiveness. Below we describe DP1 in detail and briefly describe the rest of them. Here, for any snapshot, we let \( M.vb \) and \( M.QC_x \) denote all \( vb \)'s and all \( QC_x \)'s contained in \( M \), respectively.

The pseudocode for all the core functions (i.e., \( \text{FSB}() \) and \( \text{VV}() \)) are summarized in Table 2.

**Dominant predicate DP1.** Given a block \( b \) where the real state of \( b \) is committed, we consider the following situation in DP1: "enough" correct replicas have already voted for \( b \) but not "enough" correct replicas have received the quorum certificate. To simplify the description, we define \( \text{Votes}(b, T, k) \) and \( \text{Certs}(b, T, x, y) \). In \( \text{BG}[x, z] \) and \( \text{BG}[x, y, z] \), \( \text{Votes}(b, T, k) \) represents the lower bound on the number of \( b \) contained in \( M^{b.vb} \) when a correct replica has received \( b.QC_k \) in normal-case operations in view \( b.view \). Similarly, \( \text{Certs}(b, T, x, y) \) represents the lower bound on the number of \( b.QC_x \) contained in \( M^{b.vb} \) when a correct replica has received \( b.QC_y \) in normal-case operations in view \( b.view \).

We now define DP1 for \( \text{BG}[x, z] \) and \( \text{BG}[x, y, z] \). \( \text{BG}[x, z] \) satisfies DP1 iff \( \text{Votes}(b, T, z) > T/2 \) (i.e., more than a half of elements in \( M^{b.vb} \) are \( b \) for a committed block \( b \)); \( \text{BG}[x, y, z] \) satisfies DP1 iff \( \text{Votes}(b, T, y) > T/2 \) (i.e., more than a half of elements in \( M^{b.vb} \) are \( b \) for a locked block \( b \)).

We show an example in Fig. 5 for a \( \text{BG}[x, y, z] \) satisfying DP1 with 6 replicas in total (i.e., \( f = 1 \) and \( n = 5f + 1 \) ) and we set \( T \) to 5 (i.e., \( 4f + 1 \) ). If a correct replica has committed \( b \), any virtual snapshot contains messages from at least 3 (i.e., \( 2f + 1 \) and more than \( T/2 \) ) correct replicas whose \( vb \) is \( b \). Accordingly, a
<table>
<thead>
<tr>
<th>pred. protocol</th>
<th>FSM()</th>
<th>VV()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DP1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BG[x, z] (vb, QC_x)</strong></td>
<td>flag = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{func}(M_x)$</td>
<td>return $(b_1, \pi_1)$ // $\text{BG}[x, z]$</td>
</tr>
<tr>
<td></td>
<td>01 $b_1 \leftarrow \text{null}$, $b_2 \leftarrow \text{null}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02 for $b \in M_x$, vb</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03 if num$(b, M_x, vb) \geq T/2$ then $b_2 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00-1 (\pi \leftarrow \text{M}_x)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>06-2 $\pi \leftarrow M_x$, vb</td>
<td></td>
</tr>
<tr>
<td></td>
<td>return $(b_2, \pi_2)$ // $\text{BG}[x, y, z]$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{func}_v((\text{view-update}, v, b', \text{height}, b', \pi), lb)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12 $b \leftarrow \text{the parent block of } b'$, $v_0 \leftarrow b, \text{view}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>09 if $M_x \in \pi$ and $(b, \pi) = \text{num}(M_x)$ and $v_0 &lt; v$ then return 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 then return 0</td>
<td></td>
</tr>
<tr>
<td><strong>DP2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BG[x, y, z] (vb, QC_x)</strong></td>
<td>flag = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{func}(M_x)$</td>
<td>return $(b_1, \pi_1)$ // $\text{BG}[x, z]$</td>
</tr>
<tr>
<td></td>
<td>18 $b_0$, $b_1$, $b_2$, $b_3 \leftarrow \text{null}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19 for $b \in M_x, vb$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 if num$(b, M_x, vb) \geq f + 1$ and rank$(b) &gt; \text{rank}(b_1)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21 then $b_1 \leftarrow b$, $b_0 \leftarrow b_1$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>22 else num$(b, M_x, vb) \geq f + 1$ and rank$(b) &gt; \text{rank}(b_2)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23 then $b_2 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24 for $b : b \in M_x, QC_x$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25 if rank$(b) &gt; \text{rank}(b_3)$ then $b_3 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>26 if rank$(b_0) &gt; \text{rank}(b_3)$ then</td>
<td></td>
</tr>
<tr>
<td></td>
<td>27 $b_0 \leftarrow b$, $b_1 \leftarrow b$, $b_2 \leftarrow b$, $b_3 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28 $\pi \leftarrow (b_0, \text{QC}_x, M_x, QC_x)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>29 return $(b_0, \pi)$ // $\text{BG}[x, y, z]$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30 return $(b_0, \pi)$ // $\text{BG}[x, y, z]$</td>
<td></td>
</tr>
<tr>
<td><strong>DP3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BG[x, z] (QC_x)</strong></td>
<td>flag = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{func}(M_x)$</td>
<td>return $(b_2, \pi)$ // $\text{BG}[x, z]$</td>
</tr>
<tr>
<td></td>
<td>40 $b_0 \leftarrow \text{null}$, $b_2 \leftarrow \text{null}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>41 for $b \in M_x, vb$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>42 if rank$(b) &gt; \text{rank}(b_3)$ then $b_2 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>44-1 $\pi \leftarrow (b_2, \text{QC}_x, M_x)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>44-2 $\pi \leftarrow (b_2, \text{QC}_x, M_x)$</td>
<td></td>
</tr>
<tr>
<td><strong>DP5</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BG[x, y, z] (vb, QC_x)</strong></td>
<td>flag = 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{func}(M_x)$</td>
<td>return $(b_2, \pi)$ // $\text{BG}[x, y, z]$</td>
</tr>
<tr>
<td></td>
<td>55 if rank$(b) &gt; \text{rank}(b_3)$ then $b_2 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>56 for $b : b \in M_x, QC_x$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>57 if rank$(b) &gt; \text{rank}(b_3)$ then $b_2 \leftarrow b$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>58 if rank$(b) &gt; \text{rank}(b_3)$ then</td>
<td></td>
</tr>
<tr>
<td></td>
<td>59 $\pi \leftarrow (b_2, \text{QC}_x, M_x, QC_x)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60 $\pi \leftarrow (b_2, \text{QC}_x, M_x, QC_x)$</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Realization of FSB(), VV() and criticalState according to different dominant predicates. We use num$(d, D)$ to denote the number of d's in a set D.
correct \( \text{FSB}(M^b) \) function should output block \( b \) if the number of \( b \) in \( M^b \cdot \text{vb} \) is larger than \( T/2 \), i.e., at least \( T/2 \) replicas have voted for \( b \).

We now specify \( \text{FSB}() \) and \( \text{VV}() \) functions. The pseudocode of core functions for \( \text{DP1} \) is presented in lines 1-17 in Table 2. For \( \text{DP1} \), we set \( \text{flag} \) as 1.

\( \text{FSB}() \) takes as input a snapshot \( M_v \) for view \( v \) and outputs \((b, \pi)\). Based on \( M_v \), we can obtain two intermediate blocks \( b_1 \) and \( b_2 \). If there exist a \( b \) such that \( \text{num}(b, M_v, \text{vb}) > T/2 \), \( b_1 \) is set as \( b \) (lines 02-03). Then \( \text{FSB}() \) outputs \((b_1, M_v)\) in \( \text{BG}[x, z] \) and \((b_1, M_v, \text{vb})\) in \( \text{BG}[x, y, z] \) (lines 06-1 and 06-2). Otherwise, we have \( b_1 = \text{null} \). Block \( b_2 \) (lines 04-05) is the block with the highest rank such that \( b_2, \text{QC}_x \) is included in \( M_v \cdot \text{QC}_x \). Then \( \text{FSB}() \) returns \((b_2, b_2, \text{QC}_x, M_v)\) in \( \text{BG}[x, z] \) and \((b_2, b_2, \text{QC}_x)\) in \( \text{BG}[x, y, z] \) (lines 07-1 and 07-2).

As the output of \( \text{FSB}() \) is different for \( \text{BG}[x, z] \) and \( \text{BG}[x, y, z] \), the \( \text{VV}() \) function is also different. For \( \text{BG}[x, z] \), \( \text{VV}() \) takes as input a \textit{view-update} message \( m \) and outputs a binary value. According to the output of \( \text{FSB}() \) function for \( \text{BG}[x, z] \), \( m, \text{justify} \) should be \( M_v \). Let \( b \) denote the parent block of \( m, \text{block} \) and \( v \) be the view of the replica. \( \text{VV}(m) \) outputs 1 if \((b, \pi) = \text{FSB}(M_v) \) and \( b, \text{view} < v \), i.e., the replica has to verify whether the leader indeed extends a safe branch given \( M_v \). Otherwise, \( \text{VV}() \) outputs 0.

For \( \text{BG}[x, y, z] \), the \( \text{VV}() \) function additionally takes as input \textit{lockState} (i.e., \( \text{lb} \)). \( \text{VV}(m, \text{lb}) \) outputs 1 if one of the following conditions holds: 1) \( m, \text{justify} \) is \( M_v \), more than \( T/2 \) elements in \( M_v \cdot \text{vb} \) are \( b \), \( b, \text{view} < v \), and \( \text{rank}(b) \geq \text{rank}(lb) \), i.e., \( T/2 \) replicas have voted for a higher block than \( lb \); 2) \( m, \text{justify} \) is \( b, \text{QC}_x \), \( b, \text{view} < v \), and \( \text{rank}(b) \geq \text{rank}(lb) \), i.e., a \( \text{QC}_x \) has been formed for \( b \) and the rank of \( b \) is no less than that of \( lb \). Otherwise, \( \text{VV}() \) outputs 0.

For \( \text{DP1} \), we obtain the following theorems.

**Lemma 2.** If \( T - (n - T_1 + f) > T/2 \), then \( \text{BG}[x, z] \) or \( \text{BG}[x, y, z] \) satisfies \( \text{DP1} \).

**Theorem 7.** \( \text{BG}[x, z] \) (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) \( \text{BG}[x, z] \) satisfies \( \text{DP1} \); 2) \( 2f < T \leq n - f \); 3) \( \left\lfloor \frac{n - f + 1}{2} \right\rfloor \leq T_1 \leq n - f \); and 4) \( f < T_j \leq n - f \) for \( j \in [1..z] \).

**Theorem 8.** \( \text{BG}[x, y, z] \) (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) \( \text{BG}[x, y, z] \) satisfies \( \text{DP1} \); 2) \( 2f < T \leq n - f \); 3) \( \left\lfloor \frac{n + f + 1}{4} \right\rfloor \leq T_1 \leq n - f \); 4) \( f < T_j \leq n - f \) for \( j \in [1..z] \); and 5) \( n - T_1 + f + 1 \leq T_{y+1} \).

We present the proofs of these theorems in Appendix F and Appendix G.

**Dominant predicate \( \text{DP2} \).** \( \text{DP1} \) has restricted on the number of replicas that vote for certain block, i.e., \( \text{VOTES}(b, T, y) > T/2 \). An interesting question to answer is whether we could relax the \( T/2 \) threshold for a meaningful predicate. So we relax the threshold from \( T/2 \) to \( f + 1 \) in \( \text{DP2} \). For \( \text{BG}[x, y, z] \), it satisfies
DP2 iff Votes(b, T, y) ≥ f + 1 and Certs(b, T, x, z) ≥ T − (2f + 1). DP2 allows us to cover protocols with 4f + 1 replicas. (We did not find interesting DP2 constructions for BG[x, z] though.)

The constructions of core functions for DP2 are shown in lines 18-40 in Table 2. We set flag as 1. $\text{fsb()}$ takes as input $M_v$ and outputs $(b, \pi)$, where $b$ is a block and $\pi$ is a proof showing that $b$ is a safe block to extend. Based on $M_v$, we can obtain four intermediate variables (block $b_0$, $b_1$, $b_2$ and $b_3$). We first use $b_1$, $b_2$ to store (at most) two different blocks, each appears more than $f$ times in $M_v.vb$. If such a block does not exist, $b_1$ or $b_2$ (or both) is set as null. If $b_1$ or $b_2$ (or both) exists, block $b_0$ is set as the block with a higher rank (lines 20-23). Furthermore, block $b_3$ (see lines 24-25) is the block with the highest rank such that $b_3.QC_x$ is included in $M_v.QC_x$. Then there are three possible outputs for $\text{fsb()}$.

1) If $\text{rank}(b_0) > \text{rank}(b_3)$ and $\text{rank}(b_1) = \text{rank}(b_2)$, $\text{fsb()}$ outputs $(b_3, \pi)$ where $\pi = (b_3.QC_x, M_v.QC_x)$ (see lines 26-28).
2) If $\text{rank}(b_0) > \text{rank}(b_3)$ and $\text{rank}(b_1) \neq \text{rank}(b_2)$, $\text{fsb()}$ outputs $(b_0, \pi)$ where $\pi = (M_v.vb)$ (see lines 26-29).
3) Otherwise, $\text{fsb()}$ outputs $b_3$ and a proof $\pi$ where $\pi = b_3.QC_x$ (see line 30).

In case 1), the parent block of $b_1$ and $b_2$ must be the same block $b$. More than 2f + 1 replicas set their QC$_x$ to the QC$_x$ for $b$. Therefore, neither $b_1$ nor $b_2$ was committed and $M_v.QC_x$ is a proof that $b$ is a safe block to extend. In case 2), $b_0$ is the locked block with the highest rank or $b_0$ has a higher rank than any locked block by a correct replica. In case 3) $b_3$ has the same or a higher rank than any locked block by a correct replica.

$\text{vv()}$ takes as input a VIEW-UPDATE message $m$ and lockState (i.e., lb). Let $b'$ denote the parent block of $m.block$, then $\text{vv()}$ outputs 1 if one of the following four conditions is satisfied: 1) $b'.QC_x \in \pi$ and $\text{rank}(b') \geq \text{rank}(lb)$ (lines 32-33); 2) $M_v.vb \in \pi$ and $\text{num}(b', M_v.vb) \geq f + 1$ and $\text{rank}(b') > \text{rank}(lb)$ (lines 34-35); 3) $M_v.vb \in \pi$ and $\text{num}(b', M_v.vb) \geq f + 1$ and $b' = lb$ (lines 36-37); 4) $M_v.QC_x \in \pi$ and $b'.QC_x \in \pi$ and $\text{num}(b', QC_x, M_v.QC_x) > 2f + 1$ (lines 38-39).

For DP2, we obtain the following theorems:

**Lemma 3.** If $T − (n − T_1 + f) \geq f + 1$ and $T − (n − T_{x+1} + f) \geq T − (2f + 1)$, then BG[$x, y, z$] satisfies DP2.

**Theorem 9.** BG[$x, y, z$] (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) BG[$x, y, z$] satisfies DP2; 2) $f < T \leq n − f$; 3) $\left\lfloor \frac{n + f + 1}{2} \right\rfloor \leq T_1 \leq n − f$; 4) $f < T_j \leq n − f$ for $j \in \{1, \ldots, n\}$; and 5) $x \leq y < z$, $n − T_1 + f + 1 \leq T_{y+1}$.

We present proofs for the above theorems in Appendix H.

**Dominant predicate** DP3. Both DP1 and DP2 require each replica to maintain its last vote vb and the QCs as part of the criticalState. In DP3, criticalState contains only the highest QC$_x$. In particular, we consider the following situation for DP3: "enough" correct replicas have received the quorum certificate for $b$. BG[$x, y, z$] satisfies DP3 iff Certs($b, T, x, z$) ≥ 1; BG[$x, y, z$] satisfies DP2 iff Certs($b, T, x, y$) ≥ 1.
The constructions of core functions for DP3 are shown in lines 41-52 in Table 2. Block $b_2$ (lines 42-43) is the block with the highest rank such that $b_2.QC_v$ is included in $M_v.QC_v$. $FSB(M_v)$ outputs $(b_2, (b_2.QC_x, M_v))$ (line 44-41) for $BG[x, z]$ and $(b_2, b_2.QC_x)$ (line 44-2) for $BG[x, y, z]$

Let $m$ denote a VIEW-UPDATE message for view $v$, let $b$ denote the parent block of $m.block$ and $\pi$ denote $m.justify$. Then $VV(m)$ for $BG[x, z]$ returns 1 in view $v$ if $m.justify$ is $M_v$, $(b, \pi) = \text{FSB}(M_v)$, and $b.view$ is lower than $v$ (see lines 45-48). For $BG[x, y, z]$, $VV()$ has an additional input $lb$. $VV(m, lb)$ outputs 1 if $b.QC_x \in \pi$, $b.view < v$ and $\text{rank}(b) \geq \text{rank}(lb)$ (see lines 49-52). Otherwise, $VV()$ outputs 0.

We obtain the following theorems for $BG[x, z]$ and $BG[x, y, z]$:

**Lemma 4.** If $x < z$ and $T - (n - T_{x+1} + f) > 0$ or if $x \geq z$ and $T - (n - 1) > 0$, then $BG[x, z]$ satisfies DP3.

**Lemma 5.** If $x < y$ and $T - (n - T_{x+1} + f) > 0$ or if $x = y$ and $T - (n - 1) > 0$, then $BG[x, y, z]$ satisfies DP3.

**Theorem 10.** $BG[x, z]$ (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) $BG[x, z]$ satisfies DP3; 2) $f < T \leq n - f$; 3) \( \left\lceil \frac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f \); and 4) $f < T_j \leq n - f$ for $j \in [1..z]$.

**Theorem 11.** $BG[x, y, z]$ (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) $BG[x, y, z]$ satisfies DP3; 2) $f < T \leq n - f$; 3) \( \left\lceil \frac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f \); 4) $f < T_j \leq n - f$ for $j \in [1..z]$; and 5) $n - T_1 + f + 1 \leq T_{y+1}$.

We present the proofs for these theorems in Appendix I and Appendix J.

**Dominant predicate DP4.** DP4 aims at capturing protocols with weak liveness. In particular, during the view changes, any correct leader needs to wait for NEW- VIEW messages from all correct replicas. While protocols with weak liveness are considered a bad practice [31], we use DP4 to cover well-known protocols of this kind such as Tendermint and Casper [8, 9]. We discuss DP4 in detail in Appendix D.

**Dominant predicate DP5.** Since DP4 captures protocols with weak liveness property, a natural question is whether we can define a dominant predicate that can transform a protocol with weak liveness to one with optimistic responsiveness. We answer this question affirmatively in DP5. In DP5, criticalState is set to $QC_x$ and $vb$. Given a block $b$, DP5 is based on two situations. The first situation is that $b$ has been committed by at least one correct replica and "enough" correct replicas have locked $b$. The second situation is more subtle: "enough" correct replicas may have already voted for a block $b$ but not "enough" correct replicas have locked $b$. In this situation, $M_v.QC_x$ may include no information about $b$.

Then correct replicas having locked $b$ may reject a VIEW-UPDATE message from a correct new leader as in DP4. $BG[x, y, z]$ satisfies DP5 iff $\text{CERTS}(b, T, x, z) \geq 1$ and $\text{VOTES}(b, T, y) \geq 1$. In our concrete constructions of $\text{FSB}()$ and $\text{VV}()$, a VIEW-UPDATE message may need to optionally include $M_v$.

The constructions of core functions for $BG[x, y, z]$ based on DP5 are shown in Table 2. In lines 53-68, $\text{FSB}()$ takes as input $M_v$ and outputs some $(b, \pi)$. From $M_v$,
we can obtain two intermediate variables (block $b_1$ and block $b_2$). In lines 54-55, block $b_1$ is set as (any) one block with the highest rank in $M_v$. The way of obtaining $b_2$ (lines 56-57) is exactly the same as that of DP1 and $b$ is set to $b_2$. Then $\text{fsb()}$ outputs $(b_2, b, QC_x, M_r, QC_x)$, if $\text{rank}(b_1) > \text{rank}(b_2)$. Otherwise, $\text{fsb()}$ outputs $(b_2, b_2, QC_x)$.

Let $m$ denote a \textsc{view-update} message for view $v$, $b$ denote the parent block of $m$-$\text{block}$, and $\pi$ denote $m$-$\text{justify}$. $\text{vv}(m, lb)$ outputs 1 if one of the following conditions is satisfied: 1) $b.QC_x \in \pi$ and $\text{rank}(b) \geq \text{rank}(lb)$ (see lines 62-63); or

2) $b.QC_x \in \pi$ and $M_v, QC_x \in \pi$ and $b$ is the block with the highest rank such that $b.QC_x \in M_v.QC_x$ (see lines 64-67).

We present the main theorems for DP5 as follows:

**Lemma 6.** If $T - (n - T_1 + f) > 0$ and $T - (n - T_{x+1} + f) > 0$, then $\text{BG}(x, y, z)$ satisfies DP5.

**Theorem 12.** $\text{BG}(x, y, z)$ (in Table 2) achieves safety and responsiveness if the following are satisfied: 1) $\text{BG}(x, y, z)$ satisfies DP5; 2) $f < T \leq n - f$; 3) $\frac{n+f+1}{2} - T_1 \leq n - f$; 4) $f < T_j \leq n - f$ for $j \in \{1, \ldots, z\}$; and 5) $n - T_1 + f + 1 \leq T_{y+1}$.

We present the proof for the above theorems in Appendix K. A DP5 variant. According to the construction of $\text{fsb()}$ function for DP5, a new leader may need to include $M_v$ in its \textsc{view-update} message. One can alternatively use one more phase to remove this $M_v$. We discuss the variant in detail in Appendix C.2. The idea is at the core of Marlin [30], one of the state-of-the-art 2-phase BFT protocols with linearity.

### 7 BG Instantiations

The section discusses the selective BFT protocols generated from the BG framework. To generate BFT protocols, we could simply enumerate the parameters $x$, $y$, $z$ for each dominant predicate and generate either a $\text{BG}(x, z)$ or a $\text{BG}(x, y, z)$ protocol. For each $\text{BG}(x, z)$ or $\text{BG}(x, y, z)$, using the realizations for core functions generated in layer 3, we can check whether the system of inequalities with thresholds $(T, T_1, \cdots, T_z)$ being unknown is achievable according to the theorems described in Sec. 6.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
protocol & core realization & thresholds & status \\
\hline
BG[1, 1, 2] & DP1 & $T = T_1 = 4f + 1$ & \text{the most efficient 1-phase protocol} \\
\hline
BG[1, 2] & DP2 & $T = T_1 = 2f + 1$ & \text{same as described in Table 1} \\
\hline
BG[1, 1, 2] & DP3 & $T = T_1 = 4f + 1$ & \text{the most efficient 2-phase protocol without lock state} \\
\hline
BG[1, 1, 2] & DP4 & $T = T_1 = 2f + 1$ & \text{the first update protocol which achieves $O(n)$ message complexity and $O(n)$ authenticator complexity for both normal-case and view change} \\
\hline
BG[1, 1, 2] & DP5 & $T = T_1 = 2f + 1$ & \text{the first $1/2$ BFT protocol to the best of our knowledge} \\
\hline
BG[1, 1, 2] & DP6 & $T = T_1 = 2f + 1$ & \text{the protocol can be further optimized to cover Marlin [30]} \\
\hline
BG[1, 1, 2] & DP7 & $T = T_1 = 2f + 1$ & \text{a variant of BG[1, 1, 2] that achieves $O(n)$ authenticator complexity at the cost of two more phases for view change} \\
\hline
BG[1, 1, 2] & DP8 & $T = T_1 = 2f + 1$ & \text{with minor differences in data structure and information carried in the new-view message} \\
\hline
\end{tabular}
\caption{Representative BG BFT protocols generated using our framework for $\leq 3$.}
\end{table}
In light of the existence of efficient 3-phase BFT protocols (e.g., HotStuff, PBFT), we only enumerate $x$, $y$, and $z$ for $z \leq 3$. We obtain 23 candidate protocols in total. In Appendix C, we present all the system inequalities and ranges of the thresholds in Table 5 and Table 7, and a complete list of all 23 protocols in Table 6.

Among the protocols, seven of them compare favorably with existing ones in terms of at least one characteristics, as summarized in Table 3. For each protocol, we specify the dominant predicate and their features. We then present in this section how to generate the protocols using our frameworks and how the protocols outperform prior works. We present in Table 3 a summary of the conditions (which are crucial to determining whether a protocol is achievable) for all the protocols described in this section. The full list is presented in Appendix Sec. C.1, Table 5.

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Conditions</th>
<th>Other conditions</th>
<th>Other conditions</th>
<th>Ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP1: $\text{V}erif(b, T, z) &gt; T/2$</td>
<td>$T - (n - T_1 + f) &gt; T/2$</td>
<td>$n - T_1 + f + 1 \leq T_3 \leq n - f$</td>
<td>$f &lt; T &lt; n - f$</td>
<td>( T \in \mathbb{Z} )</td>
</tr>
<tr>
<td>BG[1, 2, 3]</td>
<td>$\frac{n + f + 1}{2} \leq T_1 \leq n - f$</td>
<td>$f &lt; T_3 \leq n - f$</td>
<td>$f &lt; T_3 \leq n - f$</td>
<td>$f &lt; T_3 \leq n - f$</td>
</tr>
</tbody>
</table>

**Table 4:** Conditions for some BG candidates summarized from the Lemmas and Theorems for the dominant predicates.

**BG[1, 2, 3] with DP3.** We begin with a 3-phase protocol $\text{BG}[1, 2, 3]$, a protocol with $lockState$ based on DP3. To generate $\text{BG}[1, 2, 3]$, we need to determine the values of $n$, $f$, $T_1$, $T_2$, $T_3$, and $T$. According to Table 4, we can obtain a system of inequalities (2) as follows:

\[
\begin{align*}
T - (n - T_2 + f) &> 0 \\
n - T_1 + f + 1 &\leq T_3 \leq n - f \\
\frac{n + f + 1}{2} &\leq T_1 \leq n - f \\
f < T < n - f \\
f < T_3 \leq n - f \\
f < T_3 \leq n - f \\
f < T_3 \leq n - f
\end{align*}
\]

While the inequalities have many solutions, we set $n$ as $3f + 1$, and set $T_1$, $T_2$, $T_3$ and $T$ as $2f + 1$ to achieve optimal resilience. Accordingly, we obtain $\text{BG}[1, 2, 3]$, as shown in Fig. 6. The protocol is similar to HotStuff in performance and message flow except minor differences in data structure and information carried in the New-View message.
**BG[1, 2] with DP3.** To specify the details of the protocol, we need to determine the values of \( n, f, T_1, T_2 \) and \( T \). According to Table 4, we can obtain a system of inequalities (2) as follows:

\[
\begin{aligned}
T - (n - T_2 + f) &> 0 \\
f < T < n - f \\
\frac{n + f + 1}{2} &\leq T_1 \leq n - f \\
f < T_1 \leq n - f \\
f < T_2 \leq n - f
\end{aligned}
\]

The system of inequalities have (many) solutions. To achieve optimal resilience, we set \( n = 3f + 1 \), and set \( T_1, T_2 \) and \( T \) as \( 2f + 1 \). Then using Algorithm 3, Algorithm 4, and the realization of the \( \text{FSB()} \) and \( \text{VV()} \) in Table 2, we obtain \( \text{BG}[1, 2] \) as shown in Fig. 7.

**BG[1, 1] with DP1.** Again, we need to specify the values of \( n, f, T_1, T_2 \) and \( T \). According to Table 4, we obtain the following system of inequalities (3):

\[
\begin{aligned}
T - (n - T_2 + f) &> T/2 \\
n - T_1 + f + 1 &\leq T_2 \leq n - f \\
f < T < n - f \\
\frac{n + f + 1}{2} &\leq T_1 \leq n - f \\
f < T_1 \leq n - f \\
f < T_2 \leq n - f
\end{aligned}
\]

The above system of inequalities has solution only if \( n > 5f \). We thus set \( n = 5f + 1 \), and set \( T_1, T_2 \) and \( T \) as \( 4f + 1 \). According to Algorithm 3, Algorithm 4, and the realization of the \( \text{FSB()} \) and \( \text{VV()} \) in Table 2, we obtain \( \text{BG}[1, 1] \) as shown in Fig. 8.

**BG[1, 1, 2] with DP2.** According to Table 2, we have the following system of inequalities (4):
The above system of inequalities has solutions only if $n > 4f$. This time, we set $n$ as $4f + 1$, and set $T_1$, $T_2$ and $T$ as $3f + 1$. From Algorithm 3, Algorithm 4, and the realization of the FSB() and VV() in Table 2, we obtain BG[1,1] as depicted in Fig. 9.
According to Table 4, we obtain the following system of inequalities (5):

$$\begin{cases} 
T - (n - T_1 + f) > 0 \\
T - (n - T_2 + f) > 0 \\
n - T_1 + f < T < n - f \\
f < T < n - f \\
\left\lfloor \frac{n + f + 1}{2} \right\rfloor \leq T_1 \leq n - f \\
f < T_1 \leq n - f \\
f < T_2 \leq n - f 
\end{cases}$$

The system of inequalities has solutions only if $n > 3f$. We then set $n$ to $3f + 1$, and $T_1$, $T_2$ and $T$ to $2f + 1$ and obtain $\text{BG}[1, 1, 2]$ as shown in Fig. 10.

The system of inequalities has solutions only if $n > 5f$. We thus set $n$ as $5f + 1$, and $T_1$ and $T$ as $4f + 1$ and the protocol $\text{BG}[1, 1]$ is shown in Fig. 11.

**References**


A Additional Related Work

We have discussed the most relevant work throughout the paper. This section now discusses additional related work.

First of all, the idea of using the local state variables stems from the HotStuff technique [31], a novel technique that originally used by HotStuff BFT protocol [31] and implicitly used in AMS validated Byzantine agreement (VBA) [4]. Such a technique is later used to build various other Byzantine fault-tolerant protocols [29, 3, 2]. In the partial synchronous setting, a number of HotStuff BFT variants have been proposed, including [20, 16, 17, 5, 30].

Some of our protocols admit a threshold of $f+1$ for some (but not all) phases. A threshold signature with $f+1$ as the threshold, in general, compared to a high-threshold $(2f+1)$ signature, has many benefits. First, a low-threshold signature is more efficient, as one now only waits for $f+1$ partial threshold signatures to form a combined signature [14]. Second, without assuming trusted setup, it is computationally less efficient to build high-threshold signature [21, 2, 13, 6]. Also, some prior protocols explore using low thresholds in their constructions (e.g., [10]).

B Algorithms for Layer 1

In this section, we provide the deferred formal description of our framework defined in layer 1. The normal-case protocol is presented in Algorithm 3, and the view change protocol is presented in Algorithm 4. Both protocols use the data structures defined in Sec. 4.2.

B.1 Formal Description for Normal-Case Protocol

We present normal-case protocol for $BG[x, z]$ and $BG[x, y, z]$ in Algorithm 3. In the normal-case protocol, there are $z$ phases. Each phase includes two steps with linear communication. Note that $x, y,$ and $z$ are integers such that $x \leq z$ in $BG[x, z]$ and $x \leq y \leq z$ in $BG[x, y, z]$.

▷ In lines 6-14 (phase 1), the leader extends a branch in the tree it maintains, creates a new block $b$, and broadcasts a message $m = \langle \text{msg-1, cview, b.height, b, QC}_x \rangle$ to all replicas. Upon receiving a $\text{msg-1}$ message $m$, each replica verifies $m$.block.

▷ In lines 15-24, starting from phase 2, replicas repeat the same procedure until the $z$-th phase completes. In particular, in the $j$-th phase, after collecting $T_j - 1$ matching threshold signature shares, the leader runs $\text{qCCreate}()$ to obtain a $QC_{j-1}$ for block $b$, broadcasts $\langle \text{msg-2, cview, b.height, b, QC}_j \rangle$ to all replicas.
and enters the next phase. Upon receiving a valid MSG-\(j\) message, a replica sends the leader QC\(\text{Vote}(m)\) using a \texttt{VOTE-}\(j\) message. In \(\text{BG}[x, y, z]\), if a replica voted for a block in the \((y+1)\)-th phase of normal-case protocol, it sets its \(lb\) to the block at the same time.

\[\text{\(\triangleright\) In lines 25-32 (the commit step), the leader broadcasts QC}\_z\text{ to all replicas in a COMMIT message. Upon receiving a valid QC}\_z, each replica commits the corresponding block.}\]

\begin{algorithm}
\caption{Normal-case protocol for \(\text{BG}[x, z]\) and \(\text{BG}[x, y, z]\)}
\begin{algorithmic}[1]
\State \textbf{Initialization:}
\State \textit{localState}: \texttt{cview} \leftarrow 1, \texttt{vb} \leftarrow \bot, QC_1, QC_2, \ldots, QC_{z} are initialized to \texttt{\bot}.
\State \textit{criticalState}: set by layer 3 of the framework; contain variables in \textit{localState}.
\State \textit{lockState}: QC_\(y\) (in \(\text{BG}[x, y, z]\)) or \texttt{\bot} (in \(\text{BG}[x, z]\))
\State \textit{flag}: a system parameter set by layer 3 of the framework.
\State \textbf{Phase 1:}
\State \(\triangleright\) as a leader
\State \(b' \leftarrow \text{qcBlock}(QC}_z\), \(b \leftarrow (\texttt{cview}, b'.height + 1, \texttt{req}, \texttt{hash}(b'))\)
\State broadcast (MSG-1, \texttt{cview}, b.height, b, b'.QC\_z)
\State \(\triangleright\) as a replica
\State \textbf{wait for message} (MSG-1, \texttt{cview}, b.height, b, b'.QC\_z) from \texttt{leader}(\texttt{cview})
\State \textbf{if} \(\texttt{cview} = \texttt{cview}\) and \(b'.height = b'.height + 1\text{ and }\texttt{hash}(b')\text{ and }\texttt{rank}(b')\geq \text{\texttt{rank}(\texttt{vb}}}\)
\State QC\_z \leftarrow b'.QC\_z, \(vb \leftarrow b\), \texttt{msg} \leftarrow (\texttt{VOTE-1}, cview, b.height, b, \bot)
\State \texttt{send qcVote}(\texttt{msg}) to \texttt{leader}(cview)
\State \textbf{Phase 2} to Phase \(z\) (for \(2 \leq j \leq z\))
\State \(\triangleright\) as a leader
\State \textbf{wait for message} (MSG-\(j\), cview, b.height, b, b.QC\_\(j-1\)) from \texttt{leader}(\texttt{cview})
\State \textbf{broadcast} (MSG-\(j\), cview, b.height, b, qc\texttt{Create}(M))
\State \(\triangleright\) as a replica
\State \textbf{wait for message} (MSG-\(j\), cview, b.height, b, b.QC\_\(j-1\)) from \texttt{leader}(\texttt{cview})
\State \textbf{if} \(\texttt{b.view} = \texttt{cview}\) and \(\text{\texttt{rank}(b)} > \text{\texttt{rank}(qc\texttt{Block}(QC\_j-1)})\)
\State QC\_\(j-1\) \(\leftarrow b.QC\_j-1\), \(m \leftarrow (\text{\texttt{VOTE-j}}, cview, b.height, b, \bot)\)
\State \textbf{if} \(j = y + 1\) then \(lb \leftarrow b\)
\State \texttt{send qcVote}(\texttt{msg}) to \texttt{leader}(cview)
\State \textbf{Commit}
\State \(\triangleright\) as a leader
\State \textbf{wait for message} (COMM\(IT\), \texttt{cview}, b.height, b, qc\texttt{Create}(M))
\State \(\triangleright\) as a replica
\State \textbf{wait for message} (COMM\(IT\), \texttt{cview}, b.height, b, b.QC\_\(y\)) from \texttt{leader}(\texttt{cview})
\State \textbf{if} \(\texttt{b.view} = \texttt{cview}\) and \(\text{\texttt{rank}(b)} > \text{\texttt{rank}(qc\texttt{Block}(QC\_z))}\)
\State QC\_\(z\) \(\leftarrow b.QC\_z\), execute the requests in \(b\) in order
\State \textbf{Finally}
\State switch to New-view phase of view change protocol if \texttt{timeout} occurs in any phase
\end{algorithmic}
\end{algorithm}

\subsection{View Change Protocol}

We now present the view change protocol using the core functions \texttt{FSB(\(\cdot\))} and \texttt{VV(\(\cdot\))} in a black-box manner. As is described in Sec. 4.4, view change is triggered by \(p_i\) when \texttt{timeout} occurs.

\(\triangleright\) In lines 1-3, a replica triggers view change by incrementing \texttt{cview} by one. Then \(p_i\) sends \texttt{criticalState} in a \texttt{NEW-VIEW} message to the next leader.

\(\triangleright\) In lines 4-14, the new leader collects a set of \(T\) \texttt{NEW-VIEW} messages, denoted as \(M_{i}.\) It then executes \texttt{FSB}(\(M_{i}\)) to obtain \((b', \pi)\), where \(b'\) is a block and \(\pi\) is
Algorithm 4: View change protocol for BG\([x, z]\) and BG\([x, y, z]\)

1.▷ New-view:
2. as a replica
3. \(cview \leftarrow cview + 1\), send \(((\text{NEW-VIEW}, cview, \bot, \bot, \text{criticalState})\) to \text{Leader}(cview)
4.▷ View-update
5. as a new leader
6. \((b', \pi)\leftarrow \text{fsb}(M_v)\) // \(M_v\) is a set of \(T\) \text{NEW-VIEW} messages collected in \(v\)
7. \(b \leftarrow (b'.\text{height} + 1, cview, \text{req}, \text{hash}(b'))\)
8. broadcast \(m = (\text{VIEW-UPDATE}, cview, b.\text{height}, b, \pi)\)
9. if \(\text{flag} = 0\) then switch to Phase 2 of normal-case operation
10. if \(\text{flag} = 1\) then
11. wait for \(T_1\) matching votes:
12. \(M \leftarrow \{m \mid m = (\text{VOTE}-1, cview, b.\text{height}, b, \bot)\}\)
13. \(QC_x \leftarrow \text{qcCreate}(M)\)
14. execute Phase 2 to Phase \((x+1)\) without updating \(\text{lockState}\) until \(b/QC_x\) is generated
15. then update \(QC_x\) with \(b/QC_x\) and switch to phase 1 of normal-case operation
16. as a replica
17. wait for \((\text{VIEW-UPDATE}, cview, b.\text{height}, b, \pi)\) from \text{Leader}(cview)
18. if \((\text{vv}(m) = 0 \text{ in } \text{BG}[x, z] \text{ or } \text{vv}(m, \text{lb}) = 0 \text{ in } \text{BG}[x, y, z])\) then discard \(m\)
19. if \(\text{flag} = 0\) then switch to Line 13 of normal-case without updating \(QC_x\)
20. if \(\text{flag} = 1\) then
21. execute Phase 1 to Phase \(x\) of normal-case operation without updating \(\text{lockState}\)
22. if a \((\text{MSG}-1)\) message for a block \(b^*\) extending \(b\) is received
23. switch to Phase 1 of normal-case operation
24.▷ Finally
25. switch to New-view phase of view change protocol if \text{timeout} occurs in any phase

A proof that \(b'\) is a safe block to extend. Then the leader extends the branch led by \(b'\) with a new block \(b\) and broadcasts \(b\) in a \text{VIEW-UPDATE} message \(m\). Depending on the parameter \(\text{flag}\), there are two cases. If \(\text{flag} = 0\), the leader directly switches to phase 2 of normal-case operation. If \(\text{flag} = 1\), the leader still switches to phase 2 but does not update its \(\text{lockState}\) until \(b/QC_x\) is generated.

In lines 15-22, a replica accepts a \text{VIEW-UPDATE} message \(m\) in view \(v\) from the new leader only if \(\text{vv}(m)\) outputs 1 in \(\text{BG}[x, y, z]\) or \(\text{vv}(m, \text{lb})\) outputs 1 in \(\text{BG}[x, y, z]\). Similar to the two cases for the leader, according to the \(\text{flag}\) parameter, the replica may take different actions upon receiving \(m\). If \(\text{flag} = 0\), the replica switches to Line 13 of normal-case without updating \(QC_x\). If \(\text{flag} = 1\), the replica still votes for \(m\) but does not update \(\text{lockState}\) or commit any block until a \((\text{MSG}-1)\) message for a block \(b^*\) extending \(b\) is received. Then the replica switch to phase 1 of normal-case operation.

C Protocols and Inequalities

In this section, we discuss all the 23 BFT protocols generated in our framework.

C.1 Enumerating the BG Protocols

While protocols presented in Sec. 7 outperform existing BFT protocols, we could enumerate \(x, y, z\) for the predicates, creating 23 protocols in total. Table 5 presents the system of inequalities for these protocols, and Table 6 presents the system features of all the protocols. We also present the ranges of framework parameters (e.g., the thresholds) of these protocols in Table 7 (as summarized from the theorems presented in layer 3).
Table 5: Solvability for each BG candidate summarized from the Lemmas and Theorems for the dominant predicates. To see if a specific BG candidate protocol achieves safety and optimistic responsiveness, one needs to see if a system of inequalities for conditions is solvable. For instance, the system of inequalities for BG[1,1] includes the condition for DP1 (row 1, column 3), \(x, y\)-dependent conditions (which are null here), and other general conditions (row 1, column 5). If a system of inequalities is solvable for a specific BG candidate, the range for its thresholds is illustrated in Table 7.

C.2 The DP5 Variant

We present the constructions for core functions for the variant of DP5. In this variant, we set \(f_{\text{flag}}\) as 0 and the core functions remains the same as those in DP5. However, the view change protocol of the DP5 variant is slightly different from that mentioned in the framework. In this variant, a new leader uses one more phase to achieve linear communication while satisfying VVL-safety. Specifically, basing on a set \(M_v\) of \(n-f\) New-View messages, a new leader \(p_i\) decides whether to start an additional phase. If the rank of the highest block in \(M_v\) is larger than that of the highest \(QC_z\) \(q_{c}\) in \(M_v\), then \(p_i\) broadcasts \(q_{c}\) to ask replicas whether \(q_{c}\) is indeed the highest \(QC_z\) they receive.

If \(T\) replicas respond "yes", then according to DP5, \(p_i\) knows that a block with a higher rank than that of \(q_{c}\) cannot be committed. These responses form a certificate, and \(p_i\) can use this certificate to validate its new block that extends \(q_{c}\).
<table>
<thead>
<tr>
<th>Protocol</th>
<th>Predicate</th>
<th>Replicas</th>
<th>Message Pattern</th>
<th>Steps</th>
<th>Authentication Complexity</th>
<th>Message Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFT [2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BG[1,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>AnhA</td>
<td>2</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[2,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>4</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>Transverse[11]</td>
<td>5f + 1</td>
<td>Ack</td>
<td>3</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>Consensus [9]</td>
<td>5f + 1</td>
<td>Ack</td>
<td>3</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>Fisch-Homemoff[20]</td>
<td>5f + 1</td>
<td>Ack</td>
<td>5</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[1,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>5</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[2,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>5</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>2-phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BG[1,1]</td>
<td>DP3</td>
<td>3f + 1</td>
<td>Ack</td>
<td>5</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[1,1']</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>5</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[1,1']</td>
<td>DP3</td>
<td>3f + 1</td>
<td>Ack</td>
<td>5</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>3-phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BFT [2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BG[1,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>7</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[2,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>7</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[3,1]</td>
<td>DP1</td>
<td>5f + 1</td>
<td>Ack</td>
<td>7</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[2,2]</td>
<td>DP3</td>
<td>3f + 1</td>
<td>Ack</td>
<td>7</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
<tr>
<td>BG[2,2]</td>
<td>DP3</td>
<td>3f + 1</td>
<td>Ack</td>
<td>7</td>
<td>(O(n^*))</td>
<td>(O(n^*))</td>
</tr>
</tbody>
</table>

Table 6: BFT protocols generated using \(\text{BG}[x,z]\) and \(\text{BG}[x,y,z]\) for \(z \leq 3\). For instance, \(\text{BG}[1,1]\) with a predicate \(\text{DP1}\) is a 1-phase protocol. One can have many instantiations for the same parameters (e.g., \(\text{BG}[1,1,2]\)) by using different predicates. AtOA denotes all-to-all communication and 1toA represents one-to-all or all-to-one (linear) communication.
When a replica responds no, the replica is also required to send its $QC_x$ $qc'$ in this response, where $\text{rank}(qc') > \text{rank}(qc)$. Then $p_i$ will propose a block $b$ extending $qc'$. Note that the rank of $qc'$ is no less than the locked block of any replica, and all correct replicas will vote for $b$. In particular, if a correct replica $p_j$ locked at $b$, then the parent block of $b$ (denoted $b'$) satisfies that $b'.\text{view} = b.\text{view}$. Meanwhile, if $p_j$ sets its $vb$ as $b$, $p_j$ must have stored a $QC_x$ for $b'$. In DP5, $\text{Votes}(b, T, y) \geq 1$, so the rank of the highest $QC_x$ $qc$ contained in a view change snapshot is at most one less than that of the highest locked block.

This additional ask-respond phase ensures that the new block proposed by $p_i$ will be voted by all correct replicas without being sent with $n - f$ new-view messages.

We use $BG[1,1,2]^*$ to denote the protocol generated using the variant of DP5. As shown in Fig. 12, $BG[1,1,2]^*$ achieves $O(n)$ authenticator complexity at the cost of two more (optional) phases for view change.

The above mechanism is at the core of Marlin [30], Marlin, however, does further optimizations, including one where the ask-response phase is combined with the broadcast-vote phase of two new blocks.

<table>
<thead>
<tr>
<th>protocol</th>
<th>$p_0$</th>
<th>$T$</th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$BG[1,1]$</td>
<td>$DP1$</td>
<td>$(f, n-f)$</td>
<td>$\left(\min\left{\frac{n+f}{3}, n+f - \frac{n-f}{3}\right}, n-f\right)$</td>
<td>$[f+1, n-f]$</td>
</tr>
<tr>
<td>$BG[2,2]$</td>
<td>$DP1$</td>
<td>$(f, n-f)$</td>
<td>$\left(\min\left{\frac{n+f}{3}, n+f - \frac{n-f}{3}\right}, n-f\right)$</td>
<td>$[f+1, n-f]$</td>
</tr>
<tr>
<td>$BG[1,1]$</td>
<td>$DP3$</td>
<td>$(f, n-f)$</td>
<td>$\left[\frac{n+f}{3}, n-f\right]$</td>
<td>$[n+f - T + 1, n-f]$</td>
</tr>
<tr>
<td>$BG[1,1]$</td>
<td>$DP1$</td>
<td>$(f, n-f)$</td>
<td>$\left[\min\left(n+T, n+f - \frac{n-f}{3}\right), n-f\right]$</td>
<td>$[\max(n-f + 1, n-f + 1), n-f]$</td>
</tr>
<tr>
<td>$BG[1,1]$</td>
<td>$DP5$</td>
<td>$(f, n-f)$</td>
<td>$\left(\max(n-T+f+1, \frac{n+f}{3}), n-f\right)$</td>
<td>$[\max(n-T+f+1, n-T+f+1), n-f]$</td>
</tr>
</tbody>
</table>

**Table 7:** Ranges for the thresholds of BG protocols that are achievable given the system of inequalities presented in Table 5. All $DP1$ protocols use $5f + 1$ replicas, while protocols with other predicates have optimal resilience.
**C.3 Optimizations of The Protocols**

Here we present optimizations for our protocols with DP1 to reduce the number of steps for view changes. Recall that flag is set to 1 in DP1. According to Algorithm 4, after the view change, the new leader proposes its first block b. Replicas can vote for b but do not update their lockState or commit b. After replicas receive another proposed block extending b from the leader, they then switch to normal case operation. Accordingly, the first block proposed by a new leader needs one more phase to be committed, i.e., the block can be committed only after its extension is committed.

Fortunately, we can make some modification to our view-update protocol to reduce this additional phase for view change. In this section, we provide the optimized implementation of protocols with DP1. We follow the notations defined in Sec. 4.2 despite some minor modification.

**Message.** For the view change protocol, we extend the definitions of the new-view message. There are two types of new-view messages. One remains the same as that defined in our main framework. The new one we define sets the justify field as ⊥ but other fields are used to store information for a block.

**View-change certificate (VC).** We introduce a certificate called view-change certificate (VC). Recall that during the view change, each replica sends a signed new-view message containing its criticalState. A view-change certificate for a message m is a collection of signatures for a new-view message m, where m.justify is ⊥. For a VC vc for m, vc.view is m.view and we also called vc a view-change certificate for m.block. The threshold for a VC is set as ⌈(T+1)/2⌉, where T is the threshold of new-view messages the new leader needs to collect.

**Local state.** In the modified protocol, each replica needs to maintain the latest view-change certificate received during the view change in its lockState, denoted as QC_vc. The criticalState of a replica is set to vb, QC_x, and QC_vc of the replica.

In the modified protocol, we ask replicas to create partial signatures for the block proposed in the new-view messages so QCs can be formed during the view change as well. These combined signature (if any) naturally becomes a certificate for the first block proposed by a new leader. Besides, replicas need
to locally stores the certificate for such a block. This certificate ensures that no votes for a committed block can be overwritten and should be included in the \texttt{NEW-VIEW} message if another view change occurs.

We make modification to view change protocol in layer 2 and provide new realizations of \texttt{FSB()} and \texttt{VV()} in layer 3.

\textbf{View change protocol.} We present the modified view change protocol in Algorithm 5. Similar to Algorithm 4, we present the modified protocol using \texttt{FSB()} and \texttt{VV()} functions in a black-box manner. When the timer of replica \( p_i \) expires in \texttt{cview}, view change is triggered.

▷ In lines 1-6, a replica starting view change by incrementing \texttt{cview} by one. Then \( p_i \) sends \texttt{criticalState} in a \texttt{NEW-VIEW} message to the next leader. If the last voted block \( \texttt{vb} \) of \( p_i \) has a higher rank than \( QC_x \) of \( p_i \), \( p_i \) also creates a partial signature for \( \texttt{vb} \) and include the partial signature in the \texttt{NEW-VIEW} message.

▷ In lines 7-14, the new leader collects a set of \texttt{T NEW-VIEW} messages, denoted as \( M_v \). It then executes \texttt{FSB}(\( M_v \)) to obtain \((b', \pi)\), extends the branch led by \( b' \) with a new block \( b \), and broadcasts \( b \) in a \texttt{VIEW-UPDATE} message \( m \). Then the leader waits for \( T_1 \) matching votes to form a \( QC_1 \) for \( b \). After receiving \( b, QC_2 \), the leader set its \( QC_x \) to \( b, QC_x \) and directly switches to phase 2 of normal-case operation.

▷ In lines 15-20, a replica accepts a \texttt{VIEW-UPDATE} message \( m \) in view \( v \) from the new leader only if \( \texttt{VV}(m, \cdot) \) outputs 1. If \( m, \texttt{justify} \) contains a view change certificate or a \( QC_x \), then the replica sets its \( QC_x \) to the certificate and switches to Line 13 of our algorithm but does not update its \( QC_x \).

\textbf{Realization of \texttt{FSB()} and \texttt{VV()} for DP1.} We present a modified realization of the \texttt{FSB()} and \texttt{VV()} functions for \texttt{BG}[x, z] and \texttt{BG}[x, y, z] in Table 8. \texttt{FSB()} takes as input \( M_v \) and outputs \((b, \pi)\). Based on \( M_v \), we can obtain three intermediate blocks \( b_1, b_2, \) and \( b_3 \). Block \( b_1 \) represents the block more than \( T/2 \) replicas have voted for, if any. Block \( b_2 \) represents the highest block with a QC. If there exist a \( b \) such that \( \text{num}(b, M_v, vb) > T/2 \), then these votes for \( b_1 \) can form a \texttt{VC} and \( b_1 \) is set as \( b \) (lines 02-04). Then \texttt{FSB()} outputs \((b_1, (M_v, vc))\) in \texttt{BG}[x, z] and \((b_1, (\bot, vc))\) in \texttt{BG}[x, y, z] (lines 05-1 and 05-2). Otherwise, we have \( b_1 = \text{null} \).

Block \( b_2 \) (lines 06-07) is the block with the highest rank such that \( b_2, QC_x \) is included in \( M_v, QC_x \). Block \( b_3 \) (lines 08-10) is the block such that a \texttt{VC} \( v \) for \( b_1 \) is included in \( M_v, QC_x \) and \( v \) is the view-change certificate with the highest view contained in \( M_v, QC_x \). If the view of \( v \) is larger than \( b_2 \), then \texttt{FSB()} returns \((b_2, (M_v, vc))\) in \texttt{BG}[x, z] and \((b_2, (\bot, vc))\) in \texttt{BG}[x, y, z] (lines 12-1 and 12-2). Otherwise, \texttt{FSB()} returns \((b_2, (M_v, b_2, QC_x))\) in \texttt{BG}[x, z] and \((b_2, (\bot, b_2, QC_x))\) in \texttt{BG}[x, y, z] (lines 13-1 and 13-2).

In \texttt{BG}[x, z], \texttt{VV()} is the same with that shown in Table 2. For \texttt{BG}[x, y, z], besides \( m \), the function additionally takes as input \( lb \). \texttt{VV()} outputs 1 in view \( v \) if one of the following two conditions is satisfied: 1) a VC for \( b \) is included in \( m, \texttt{justify} \), \( b, \texttt{view} < v \), and \( \text{rank}(b) \geq \text{rank}(lb) \) (see lines 19-20); 2) \( b, QC_x \) is included in \( m, \texttt{justify} \), \( b, \texttt{view} < v \), and \( \text{rank}(b) \geq \text{rank}(lb) \) (see lines 21-22). The first condition proves that a VC is formed during the view change while
Algorithm 5: View change protocol

1. \(\triangleright\) New-view:
2. as a replica
3. \(cview \leftarrow cview + 1, m \leftarrow \bot, m_1 \leftarrow \langle \text{new-view}, cview, \bot, \bot, \text{criticalState} \rangle\)
4. if \(\text{rank}(vb) > \text{rank}(\text{ QC BLOCK}(QC_x))\)
5. \(m \leftarrow \langle \text{new-view}, cview, vb, vb.height, \bot \rangle\)
6. send \(m_1 \) and \(\text{QC V OTE}(m)\) to \(\text{LEADER}(cview)\)
7. \(\triangleright\) View-update
8. as a new leader
9. \((b', \pi) \leftarrow \text{fsb}(M_v)\) // \(M_v\) is a set of \(T\) new-view messages collected in \(v\)
10. \(b \leftarrow \langle b'.height + 1, cview, \text{req}, \text{hash}(b') \rangle\)
11. broadcast \(m = \langle \text{view-update}, cview, b.height, b, \pi \rangle\)
12. wait for \(T_1\) matching votes:
13. \(M \leftarrow \{ m \mid m = \langle \text{vote-1}, cview, b.height, b, \bot \rangle \}\): \(QC_1 \leftarrow \text{QC CREATE}(M)\)
14. \(\triangleright\) switch to Phase 2 of normal-case operation
15. as a replica
16. wait for \(\langle \text{view-update}, cview, b.height, b, \pi \rangle\) from \(\text{LEADER}(cview)\)
17. if \(\text{vv}(m, \cdot) = 1 \) and \(\pi = (\pi_1, \pi_2)\)
18. if \(\pi_2\) is a view change certificate then \(QC_{vc} \leftarrow \pi_2\)
19. if \(\pi_2\) is a \(QC_x\) and \(\text{rank}(\pi_2) > \text{rank}(QC_x)\) then \(QC_x \leftarrow \pi_2\)
20. // switch to line 13 of normal-case without updating \(QC_x\)
21. \(\triangleright\) Finally
22. switch to New-view phase of view change protocol if \(\text{timeout}\) occurs in any phase

the second condition proves that a block formed before the view change has potentially been committed by at least one correct replica.

D. BG Protocols with Weak Liveness

Weak liveness is used to capture the liveness property of some existing protocols (e.g., Tendermint, Casper), where a correct leader needs to wait for the messages from all correct replicas. Protocols achieving the notion would rely on synchrony for liveness. This notion is defined in HotStuff [31], as shown below.

- **Weak liveness:** After GST, any correct leader needs to wait for responses from all correct replicas to guarantee that it can create a proposal that will make progress.

D.1 Weak Liveness: Layer 2

Our Layer 2 framework can easily and formally capture protocols achieving weak liveness rather than optimistic responsiveness. Take BG[\(x, y, z\)] for example. Let \(S(M_v)\) denote the set of senders of snapshot \(M_v\). We can define a weak liveness property for the \(\text{fsb}(\cdot)\) function:

- **FSBL-wliveliness:** If \(C \subseteq S(M_v)\), then \(\text{fsb}(M_v)\) outputs \((b, \pi)\).
To capture weak liveness, we propose predicate DP4 in layer 3. DP4 is similar to DP3 except that any correct leader needs to wait for NEW-VIEW messages from all correct replicas to guarantee that it can received QCx for the locked block with the highest rank. Therefore, the leader can create a proposal that will make progress. We let such a set of NEW-VIEW messages for view v be Mv(C). We also define M3i(C) such that M3i(C) is identical to M3, except that M3i(C) contains messages from all correct replicas.

While DP3 makes sense for both BG[x, z] and BG[x, y, z], we focus on BG[x, y, z] to capture existing protocols also with weak liveness property (e.g., Tendemint and Casper): flag is set to 0, BG[x, y, z] satisfies DP4 iff b.QCx ∈ M3i(C) for any locked block b.

We present a realization of the fsb() and vv() for BG[x, y, z] in Table 9. These constructions of core functions are identical to that for DP3 except for the different inputs of fsb().

We have the following theorem for BG[x, y, z] with DP4:

**Theorem 14.** BG[x, y, z] (in Table 2) achieves safety and weak liveness if the following are satisfied: 1) BG[x, y, z] satisfies DP4 for locked blocks; 2) \( \left\lfloor \frac{n+f+1}{2} \right\rfloor \leq T_1 \leq n - f \); 3) \( f < T_j \leq n - f \) for \( 1 \leq j \leq z \); and 4) \( x \leq y < z \), \( n - T_1 + f + 1 \leq T_{y+1} \).
Table 9: Realizing \texttt{FSB()} and \texttt{VV()} according to \texttt{DP4}.

<table>
<thead>
<tr>
<th>Event</th>
<th>Critical state</th>
<th>FSB()</th>
<th>VV()</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{DP4}</td>
<td>$\mathbb{E}[x, y, \Delta]$ (QC)</td>
<td>$\text{func}, \text{vis}(M_b)$ \quad $b_v \leftarrow \text{null}$ \quad for $b : b.QC_v \in M_b, QC_x$ \quad if $\text{rank}(b) &gt; \text{rank}(b_v)$ then $b_v \leftarrow b$ \quad return $(b_v, \pi)$ s.t. $\pi = (b_2.QC_x)$</td>
<td>$\text{func}, \text{vv}(\text{VIEW-UPDATE}, cview, k', \text{height}, b', \sigma, \Delta, b)$ \quad $b_v$ = the parent block of $b'$ \quad if $b.QC_v \in x$ and $\text{rank}(b) \geq \text{rank}(b)$ then \quad if $b_v &gt; cview$ then return $1$ \quad return $0$</td>
</tr>
</tbody>
</table>

The proof of Theorem 14 is the same as that for Theorem 9 except that the threshold $T$ is replaced by a requirement that a \texttt{VIEW-UPDATE} message should include $M^b(C)$ in contrast to $M^b$.

By adopting the notion, a communication-optimal 2-phase BFT (WBG[1, 1, 2], resembling 2-phase HotStuff) can be directly obtained from our framework and it outperforms Tendemint and Casper.

### E Proofs of Theorems for Layer 2 Algorithms

**Lemma 7.** Given a block $b$, if $T_j > f$ for all $j \in [1..z]$ and if $b.QC_k$ has been formed in view $b.v$iew for some $k \in [1..z]$, then $b.QC_1, \ldots, b.QC_k$ were formed in the same view.

**Proof.** For $k = 1$, correctness is trivial. For $k \in [2..z]$, since $b.QC_k$ exists, at least $T_k$ replicas have sent \texttt{VOTE-k} messages for $b$. As $T_k > f$, at least one correct replica $p_i$ has sent \texttt{VOTE-k} messages for $b$. Hence, $p_i$ must have received $b.QC_{k-1}$ contained in a \texttt{MSG-k} message for the same view. This completes the proof of the lemma.

**Lemma 8.** Let $b$ and $d$ be two blocks proposed in view $v$ such that the view of the parent block of $b$ (denoted $b'$) and the view of the parent block of $d$ (denoted $d'$) are lower than $v$. If $T_j > f$ for all $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, and the $QC_x$ for $b$ and $d$ are both formed in view $v$, then $b = d$ and $QC_x$ for $b$ is the $QC_x$ with the lowest rank formed in view $v$.

**Proof.** Let $b_v$ be the block with the lowest height for which a $QC_x$ was formed in view $v$. Let $b_v'$ denote the parent block of $b_v$. According to Lemma 7, at least a correct replica $p_i$ has sent a \texttt{VOTE-1} message for $b_v$ to form the $b_v.QC_1$ in view $v$. If $b_v'.v$iew $= v$, then $p_i$ must have received a $QC_x$ for $b_v'$ and $\text{rank}(b_v') < \text{rank}(b_v)$. This causes a contradiction, because the $QC_x$ for $b_v$ is defined to be the $QC_x$ with the lowest height formed in view $v$. Thus, $b_v'.v$iew $< v$. As $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$ and the $QC_x$'s for $b, d$, and $b_v$ are all formed in view $v$, at least one correct replica has sent a \texttt{VOTE-1} message for $b, d$, and $b_v$ to form $QC_1$ for these blocks. Note that in view $v$, a correct replica sends a \texttt{VOTE-1} message for at most one block whose parent block is proposed before view $v$. Thus, it must hold that $b = d = b_v$. 


Lemma 9. Let $T_j > f$ for any $j \in [1..z]$ and $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$. Let $b_v$ and $b_1$ be blocks such that $QC_x$ for both $b_v$ and $b_1$ have been formed in view $v$ and the view of the parent block of $b_v$ is lower than $v$. Then $b_1 = b_v$ or $b_1$ is an extension of $b_v$.

Proof. According to Lemma 8, $b_v, QC_x$ is the $QC_x$ with the lowest rank formed in view $v$. Let $b_0$ denote the block with the lowest rank on the branch led by $b$ such that $b_0, view = v$.

For any block $b$, according to Lemma 7, the formation of $b, QC_x$ implies that $b, QC_1$ has also been formed in view $b, view$. Then, at least a correct replica has sent a vote message for $b$ since $T_1 > f$. Besides, if the view of the parent block $b'$ of $b$ is equal to $b, view$, a correct replica will send vote message for $b$ only after receiving $b', QC_x$. Then the existence of $b_1, QC_x$ implies that $QC_x$'s for $b'$ has been formed in view $v$. Furthermore, $QC_x$ for $b_0$ and any block that is an extension of $b_0$ on the branch led by $b_1$ have been formed in view $v$. Then according to Lemma 8, $b_0$ equals $b_v$ and $b_1 = b_0 = b_v$ or $b_1$ is an extension of $b_v$.

Lemma 10. If $T_j > f$ for all $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $b$ and $d$ are two conflicting blocks and $b, view = d, view = v$, then $b, QC_x$ and $d, QC_x$ cannot be formed in view $v$.

Proof. Let $b_v$ be the block with the lowest rank such that a $QC_x$ for $b_v$ was formed in view $v$. If there exists $QC_x$ for two conflicting blocks $b$ and $d$ formed in view $v$, then according to Lemma 9, $b$ and $d$ should be the same as $b_v$ or both $b$ and $d$ are extension of $b_v$. Additionally, according to Lemma 7, the $QC_x$ has been formed for both $b$ and $d$ in view $v$. Accordingly, $b, height \geq b_v, height$ and $d, height \geq b_v, height$. Then, we distinguish two cases:

1) $b, height = d, height$. If $b, height = b_v, height$, Obviously, we have $b = d = b_v$, contradicting to the assumption that $b$ and $d$ are conflicting blocks. If $b, height > b_v, height$, then $b$ and $d$ are blocks with the same rank and the view of the parent blocks of $b$ and $d$ are $v$. Then vote message for $b$ or $d$ are sent during normal-case protocol. Note that each correct replica sends a vote message only once for blocks with a specific height in a view during the normal case. Since $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $QC_1$ for $b$ and $d$ cannot be formed in the same view $v$.

2) $b, height \neq d, height$. We assume w.l.o.g., $b, height > d, height$. If $d, height = b_v, height$, then $d = b_v$ and $b$ is an extension of $b_v$, contradicting to the assumption. If $d, height > b_v, height$, $b$ and $d$ are also extension block of $b_v$. Let $b_0$ denote the block on the branch led by $b$ such that $b_0, height = d, height$, then $b_0$ is also an extension of $b_v$, $b_0, view = v$ and $b_0 \neq d$. Note the existence of $b, QC_x$ implies that $b_0, QC_x$ has been formed in view $v$. However, according to the discussion in Case 1), $QC_x$'s for $b_0$ and $d$ cannot be formed in view $v$. That’s a contradiction.

This completes the proof of the lemma.

Lemma 1 Let $B^v = \{ b | \text{block } b \text{ has been committed before view } v \}$. If $T_j > f$ for $j \in [1..z]$ and $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, then there exists $b^v \in B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$ and rank($b^v$) > rank($b'$).
Inductive case: Assume extension of Lemma 9, and Base case: Suppose \( v^* \) such that \( (\text{qcBlock}_{v^*}) \). Proof. For any block \( b \in B^v \), according to Lemma 7, a \( QC_x \) for \( b \) is formed in \( b.\text{view} \). We can find a set of blocks with the highest view by traversing all blocks in \( B^v \). According to Lemma 10, any two blocks contained in the set have different heights. Therefore, we can find a block \( b^v \) with the highest height in the set. Obviously, \( b^v \) satisfies the conditions described in the lemma.

**Theorem 1** \( BG[x, z] \) achieves safety-I, if \( T_1 \geq \left\lceil \frac{n+1}{2} \right\rceil \) and \( T_j > f \) for all \( j \in [1..z] \).

**Proof.** If two conflicting blocks \( b \) and \( d \) are committed in the same view \( v \), each by a correct replica, then there must exist \( QC_x \) for \( b \) and \( d \) formed in view \( v \). This is a contradiction with Lemma 7 and Lemma 10.

**Theorem 4** \( BG[x, y, z] \) achieves safety-I, if \( T_1 \geq \left\lceil \frac{n+1}{2} \right\rceil \) and \( T_j > f \) for all \( j \in [1..z] \).

**Proof.** The proof follows from Theorem 1.

**Lemma 11.** In \( BG[x, z] \), if a block \( b \) has been committed by at least one correct replica in view \( v \) and FSB-safety and VV-safety hold in view \( v + 1, \ldots, v + k \) (\( k \geq 1 \)), then for any \( QC_x \) \( qc \) formed in view \( v' \) (\( v + 1 \leq v' \leq k \)), \( \text{qcBlock}(qc) \) is an extension of \( b \).

**Proof.** For a \( QC_x \) \( qc \) formed in view \( v' \) (\( v + 1 \leq v' \leq v + k \)), let \( b' \) denote \( \text{qcBlock}(qc) \). We need to prove that \( b' \) is an extension of \( b \).

Let \( b_{v'} \) denote the block with the lowest height for which a \( QC_x \) has been formed in view \( v' \). According to Lemma 1, we know \( b_{v'} \) exists for any \( v' \) such that \( v + 1 \leq v' \leq k \). Then we prove the lemma by induction over the view \( v' \), starting from view \( v + 1 \).

**Base case:** Suppose \( v' = v + 1 \). In this case, we know \( b_{v'} \) \( .\text{height} \geq b.\text{height} \) and \( b_{v'} .\text{view} = b.\text{view} = v \). From Theorem 1, we have either \( b_{v'} \) equals \( b \) or \( b_{v'} \) extends \( b \). According to Lemma 7 and Lemma 8, the view of the parent block of \( b_{v'} \) is lower than \( v' \) and a \( QC_1 \) has been formed in view \( v' \). Since \( T > f + 1 \), at least one correct replica has sent \( \text{vote} \) for \( b_{v'} \) during view change. According to FSB-safety and VV-safety, \( b_{v'} \) must be an extension of \( b_{v'} \). According to Lemma 9, \( b' \) is equal to \( b_{v'} \) or \( b' \) is an extension of \( b_{v'} \). Then \( b' \) must be an extension of \( b \).

**Inductive case:** Assume \( v' = v + k_0 + 1 \) (\( 1 \leq k_0 < k \)) and for any \( QC_x \) \( qc \) formed in view \( v + 1, \ldots, v + k_0 \), \( \text{qcBlock}(qc) \) is an extension of \( b \). We prove that \( b' \) is an extension of \( b \). Note correct replicas will commit a block only after receiving a \( QC_x \) for the block and \( x \leq z \). According to Lemma 7 and the inductive hypothesis, we know \( b_{v'} = b \) or \( b_{v'} \) extends \( b \). According to Lemma 7 and Lemma 8, the view of the parent block of \( b_{v'} \) is lower than \( v' \) and a \( QC_1 \) has been formed in view \( v' \). Since \( T > f + 1 \), at least one correct replica has sent \( \text{vote} \) for \( b_{v'} \) during view change. According to FSB-safety and VV-safety, \( b_{v'} \) must be an extension of \( b_{v'} \). According to Lemma 9, \( b' \) is equal to \( b_{v'} \) or \( b' \) is an extension of \( b_{v'} \). Then \( b' \) must be an extension of \( b \).
Thus, for any $qc$ formed in view $v'$ ($v + 1 \leq v' \leq v + k$), QCBlock($qc$) is an extension of $b$.

**Lemma 12.** In BG$_{x,y,z}$, if a block $b$ has been committed by at least one correct replica in view $v$ and VVL-safety hold in view $v, \ldots, v + k$ ($k \geq 1$), then for any QC$_x$ $qc$ formed in view $v, \ldots, v + k$, QCBlock($qc$) is an extension of $b$.

**Proof.** For a QC$_x$ $qc$ formed in view $v'$ ($v + 1 \leq v' \leq v + k$), let $b'$ denote QCBlock($qc$). We need to prove that $b'$ is an extension of $b$.

Let $b_{v'}$ denote the block with the lowest height for which a QC$_x$ has been formed in view $v'$. According to Lemma 1, we know $b^{v'}$ exists for any view $v'$. Then we prove the lemma by induction over the view $v'$, starting from view $v + 1$.

**Base case:** Suppose $v' = v + 1$. In this case, we know $b^{v'}.height \geq b.height$ and $b^{v'}.view = b.view = v$. From Theorem 1, we have either $b^{v'}$ equals $b$ or $b^{v'}$ extends $b$. According to Lemma 7 and Lemma 8, the view of the parent block of $b_{v'}$ is lower than $v'$ and a QC$_1$ for $b_{v'}$ has been formed in view $v'$. Thus, more than $T_1 - f$ correct replica has sent VOTE-1 for $b_{v'}$ during view change. According to VVL-safety, $b_{v'}$ must be an extension of $b^{v'}$. According to Lemma 9, $b'$ is equal to $b_{v'}$ or $b'$ is an extension of $b_{v'}$. Then $b'$ must be an extension of $b$.

**Inductive case:** Assume $v' = v + k_0 + 1$ ($1 \leq k_0 < k$) and any QC$_x$ formed in view $v + 1, \ldots, v + k_0$ is a QC$_x$ for an extension of $b$. We prove that $b'$ is an extension of $b$. Note correct replicas will commit a block only after receiving a QC$_z$ for the block and $x \leq z$. According to Lemma 7 and the inductive hypothesis, we know $b^{v'}.height \geq b.height$ and $b^{v'}.view = b.view = v$. From Theorem 1, we have either $b^{v'}$ equals $b$ or $b^{v'}$ extends $b$. According to Lemma 7 and Lemma 8, the view of the parent block of $b_{v'}$ is lower than $v'$ and a QC$_1$ for $b_{v'}$ has been formed in view $v'$. Thus, more than $T_1 - f$ correct replica has sent VOTE-1 for $b_{v'}$ during view change. According to VVL-safety, $b_{v'}$ must be an extension of $b^{v'}$. According to Lemma 9, $b'$ is equal to $b_{v'}$ or $b'$ is an extension of $b_{v'}$. Then $b'$ must be an extension of $b$.

Thus, for any $qc$ formed in view $v'$ ($v + 1 \leq v' \leq v + k$), QCBlock($qc$) is an extension of $b$.

**Theorem 2.** BG$_{x,y,z}$ achieves safety-II, if $T_1 \geq \left\lceil \frac{n + f + 1}{2} \right\rceil$, $T_j > f$ for all $j \in [1..z]$, and FSB-safety and VV-safety hold.

**Proof.** According to Lemma 1, we know $b^{v'}$ exists for any view $v$. To prove BG$_{x,y,z}$ satisfies safety II, we need to show the following: if a block $b$ has been committed by at least one correct replica in view $v$, then any blocks committed after view $v$ is an extension of $b$.

Assume that there exist a block $b'$ committed in view $v'$ ($v' > v$) such that $b'$ is not an extension of $b$. According to Lemma 7, $b'.QC_x$ is formed in view $v'$. Note that FSB-safety and VV-safety hold in view $v, \ldots, v'$. By Lemma 11, QCBlock($qc$) must be an extension of $b$, a contradiction.

Now we can conclude that for any block $b$ that has been committed by at least one correct replica proposed in view $v$, any blocks committed after view $v$ should be an extension of $b$. Hence, BG$_{x,y,z}$ satisfies safety II.
Theorem 5 \(BG[x, y, z]\) achieves safety-II, if \(T_i \geq \left\lceil \frac{n+f+1}{2} \right\rceil\), \(T_j > f\) for all \(j \in [1..z]\), and VVL-safety holds.

Proof. According to Lemma 1, we know \(b^v\) exists for any view \(v\). To prove \(BG[x, y, z]\) satisfies safety II, we need to show the following: if a block \(b\) has been committed by at least one correct replica in view \(v\), then any blocks committed after view \(v\) is an extension of \(b\).

Assume that there exist a block \(b'\) committed in view \(v' (v' > v)\) such that \(b'\) is not an extension of \(b\). According to Lemma 7, \(b'.QC_x\) \(qc\) is formed in view \(v'\). By Lemma 12, \(qcBlock(qc)\) must be an extension of \(b\), a contradiction. This completes the proof of the lemma.

Theorem 3 \(BG[x, z]\) achieves optimistic responsiveness, if \(T_j \leq n – f\) for all \(j \in [1..z]\), \(T \leq n – f\), and FSB-liveness and VV-liveness hold.

Proof. Suppose after GST, in a new view, the leader \(p_i\) is correct. Since \(T \leq n – f\), \(p_i\) can collect \(T\) new-view messages from correct replicas. By FSB-liveness, \(p_i\) can run \(FSB()\) and obtain some \((b, \pi)\). Then \(p_i\) sends a view-update message \(m\) such that \(m.block = b_v\). We distinguish two cases:

1) \(flag = 1\). According to VV-liveness, \(b_v\) can be voted by enough replicas to form \(b_v.QC_x\). Then \(b_v\) will be voted by all the correct replicas to form \(b_v.QC_{y}. QC_{z}\). As \(T_j \leq n – f\) for all \(j \in [1..z]\), \(b_v.QC_x\) can be formed and \(p_i\) will propose a block \(b'\) extending \(b_v\). Since \(b_v.QC_x\) is the first \(QC_x\) formed in view \(v\), the condition on Line 11 of Algorithm 1 is satisfied. Since \(T_j \leq n – f\) for all \(j \in [1..z]\), it is clear that \(b'.QC_1, \ldots, b'.QC_z\) can be formed by \(p_i\). Hence, \(b_v\) can be committed.

2) \(flag = 0\). Since \(T_j \leq n – f\) for all \(j \in [1..z]\), it is clear that \(b_v.QC_1, \ldots, b_v.QC_z\) can be formed by \(p_i\). Hence, \(b_v\) can be committed.

\(BG[x, z]\) achieves optimistic responsiveness, because there is no step that requires a specific timeout in both cases.

Theorem 6 \(BG[x, y, z]\) achieves optimistic responsiveness, if \(T_j \leq n – f\) for all \(j \in [1..z]\), \(T \leq n – f\), and FSB-liveness and VVL-liveness hold.

Proof. Suppose after GST, in a new view, the leader \(p_i\) is correct. Since \(T \leq n – f\), \(p_i\) can collect \(T\) new-view messages from correct replicas. Then by FSB-liveness, \(p_i\) can run \(FSB()\) and obtain some \((b, \pi)\). Then \(p_i\) sends a view-update message \(m\) such that \(m.block = b_v\). Note \(T_j \leq n – f\) for all \(j \in [1..z]\). According to VVL-liveness, \(p_i\) can received enough vote-1 messages to form a \(QC_x\) for \(b_v\). Then no matter \(flag = 1\) or \(flag = 0\), a block can be committed. Therefore, \(BG[x, y, z]\) achieves optimistic responsiveness.

F Proofs of Theorems for \(BG[x, z]\) with \(\mathcal{DP}_1\)

Lemma 2. If \(T – (n – T_1 + f) > T/2\), then \(BG[x, z]\) or \(BG[x, y, z]\) satisfies \(\mathcal{DP}_1\).
Proof. In a BG\([x, z]\) or BG\([x, y, z]\) protocol, for any block \(b\), if \(b.QC\) is received by a correct replica \(p_i\) and \(p_i\) set its QC\(_z\) to \(b.QC\) in view \(v\), then \(b.view = v\).

According to Lemma 7, \(b.QC\) is also formed by the leader in view \(v\). Accordingly, at least \(T_1 - f\) correct replicas have sent \(\text{vote} - 1\) messages for \(b\) such that \(b.QC\) is formed. As \(p_i\) set its QC\(_z\) to \(b.QC\) in view \(v\), \(b\) is block proposed in normal case and the \(T_1 - f\) replicas set their \(vb\) to \(b\) in view \(v\). Thus, fewer than \(n - T_1\) correct replicas have not yet set their \(vb\) to \(b\) in view \(v\). Therefore, for any \(M^b\), at most \(n - T_1 + f\) messages are sent by replicas who have not set their \(vb\) to \(b\), i.e., there are at least \(T - (n - T_1 + f)\) \(b\) in \(M^b.vb\). Since \(T - (n - T_1 + f) > T/2\), more than \(T/2\) elements in \(M^b.vb\) are \(b\). That means that \(\text{Votes}(b, T, z) > T/2\) and the BG\([x, z]\) or BG\([x, y, z]\) satisfies \(\text{DP1}\).

**Lemma 13.** If \(T_j > f\) for all \(j \in [1..z]\), and \(T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil\), then FSB-liveness holds for BG\([x, z]\).

Proof. Given a \(M_v\), we need to prove that \(\text{FSB}(M_v)\) outputs some \((b, \pi)\). Based on \(M_v\), we can obtain two intermediate variables, block \(b_1\) and block \(b_2\). Since \(\text{num}(b_1, M_v.vb) > T/2\), \(b_1\) is a unique block or \(\text{null}\). By Lemma 10, \(b_2\) is also a unique block. Therefore, after comparing ranks of \(b_1\) and \(b_2\), \(\text{FSB}(M_v)\) will output a block together with a proof \(\pi\).

**Lemma 14.** If BG\([x, z]\) satisfies \(\text{DP1}\), \(T_j > f\) for \(j \in [1..z]\), \(T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil\), and \(T > 2f\), and there exists a block committed by at least one correct replica in view \(v - 1\), then FSB-safety holds in view \(v\) for BG\([x, z]\).

Proof. Let \(B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}\). According to Lemma 1, we let \(b^v\) denotes a block in \(B^v\) such that for all \(b' \in B^v\) and \(b' \neq b^v\), we have \(\text{rank}(b^v) > \text{rank}(b')\). Since there exists a committed block in view \(v - 1\), we know \(b^v.view = v - 1\). From Lemma 7, there must exist QC\(_x\) for \(b^v\), which is formed in view \(v - 1\). By Lemma 13, \(\text{FSB}(M_v)\) will output some \((b, \pi)\). Note \(b\) is equal to either \(b_1\) or \(b_2\), where \(b_1\) and \(b_2\) are two intermediate variables based on \(M_v\). We now prove that \(b\) is either \(b^v\) or an extension of \(b^v\). Since \(\text{DP1}\) holds in BG\([x, z]\), we consider two cases:

1) \(\text{num}(b^v, M^v.vb) > T/2\). Then both \(b\) and \(b_1\) equal to \(b^v\). Hence, \(b\) is either \(b^v\) or an extension of \(b^v\).

2) \(\text{num}(b^v, M^v.vb) \leq T/2\). Since \(\text{num}(b^v, M^v.vb) > T/2\) and \(T > 2f\), it is clear that at least one correct sender \(p_i\) of a message in \(M_v\) has changed its \(vb\) from \(b^v\) to some other block \(b'\) in view \(v - 1\). According to the condition on Line 11 of Algorithm 1, we have \(\text{rank}(b') > \text{rank}(b^v)\) and \(p_i\) has received a QC\(_x\) for \(b'\), the parent block of \(b\). We further know that \(b'.view = v - 1\) and \(b'.height \geq b^v.height\). Then according to Lemma 10, \(b'\) is either \(b^v\) or an extension of \(b^v\) and \(b'\) must be an extension of \(b^v\). Similarly, the \(vb\) of \(p_i\) contained in its \text{NEW-VIEW} message must be an extension of \(b^v\). So \(b_1\) must be an extension of \(b^v\) or \(\text{null}\). In addition, \(p_i\) will send its QC\(_z\) \(qc'\) in a \text{NEW-VIEW} message, QC\(_{\text{VIEW}}(qc') = v - 1\), and QC\(_{\text{HEIGHT}}(qc') \geq b'.height\). Then \(\text{rank}(b_2) \geq \text{rank}(qc')\). According to Lemma 10, we now know \(b_2\) is \(b^v\) or an
extension of $b'$, Hence, $b$ must be $b''$ or an extension of $b''$ no matter $b$ equals $b_1$ or $b$ equals $b_2$.

This completes the proof of the lemma.

**Lemma 15.** If $BG[x, z]$ satisfies DP1, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n + f + 1}{2} \right\rceil$, and $2f < T \leq n - f$, then FSB-safety holds for $BG[x, z]$.

**Proof.** For any view $v$, let $B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}$. According to Lemma 1, we let $b^v$ denote a block in $B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$, we have $\text{rank}(b^v) > \text{rank}(b')$.

We prove that FSB-safety holds for $BG[x, z]$ by proving that FSB-safety holds in every view. For a specific view $v'$, let $w$ denote $b^v \cdot \text{view}$. We prove that FSB-safety holds in view $v'$ iteratively. First, we prove that FSB-safety holds in view $w + 1$. Then we prove that if FSB-safety holds in view $w + 1, \ldots, w + k$ (for any integer $k$ such that $1 \leq k \leq v' - w - 1$), FSB-safety also holds in view $w + k + 1$. When $k = v' - w - 1$, we know that FSB-safety holds in view $v'$.

According to Lemma 14, we know that FSB-safety holds in view $w + 1$.

Then, assume that FSB-safety holds in view $w + 1, \ldots, w + k$ (for any integer $k$ such that $1 \leq k \leq v' - w - 1$), we need to show that FSB-safety holds in view $w + k + 1$. By Lemma 13, FSB($M_{w+k+1}$) will output some $(b, \pi)$. Note $b$ is either $b_1$ or $b_2$, where $b_1$ and $b_2$ are two intermediate variables obtained from $M_{w+k+1}$. We now prove that $b$ is either $b''$ or an extension of $b''$. Since DP1 holds in $BG[x, z]$, we consider two cases:

1) $\text{num}(b^v, M^v) > T/2$. Then both $b$ and $b_1$ equal to $b''$. Hence, $b$ is either $b''$ or an extension of $b''$.

2) $\text{num}(b^v, M_{w+k+1}) \leq T/2$. Since $\text{num}(b^v, M^v) > T/2$ and $T > 2f$, it is clear that at least one correct sender $p_i$ of a message in $M_w$ has changed its $vb$ from $b''$ to some other block $b'$ during view $w, w + 1, \ldots, w + k$. Since $\text{flag} = 1$, $p_i$ changed its $vb$ only if $\text{rank}(b') > \text{rank}(b'')$. According to Algorithm 1, $p_i$ has received a $QC_x$ $qc$ for the parent block $b''$ of $b'$ and $\text{rank}(b'') \geq \text{rank}(b')$.

By Lemma 10, Lemma 11 and the hypothesis, we know that $b''$ is either $b''$ or an extension of $b''$ and $b'$ must be an extension of $b''$. Similarly, the $vb$ of $p_i$ contained in its NEW-VIEW message must be an extension of $b''$. So $b_1$ must be an extension of $b''$ or null. In addition, $p_i$ will send its $QC_x$ $qc$ in a NEW-VIEW message. Since correct replicas only change its $QC_x$ $qc$ to $QC_x$ with the same or a higher rank, we have $\text{rank}(QC_BLOC\text{K}(qc)) \geq \text{rank}(QC_BLOC\text{K}(qc))$. Therefore, $\text{rank}(b_2) \geq \text{rank}(b'')$ and $b_2$ is included in $M_{w+k+1}$. If $b_2, \text{view} = w$, then according to Lemma 10, $b_2$ is either $b''$ or an extension of $b''$. If $b_2, \text{view} > w$, then according to Lemma 11 and the inductive hypothesis, $b_2$ is either $b''$ or an extension of $b''$. Hence, $b$ is either $b''$ or an extension of $b''$ no matter $b$ equals $b_1$ or $b$ equals $b_2$.

In both cases $b$ is either $b''$ or an extension of $b''$, then FSB-safety holds in view $w + k + 1$. When $k = v' - w - 1$, we know that FSB-safety holds in view $v'$. This completes the proof of the lemma.

**Lemma 16.** If $T_1 > f$, VV-safety holds in $BG[x, z]$.
Lemma 19. If $BG[\text{output} \in \{2\}$, block $b$, $Pr$ of. The proof resembles the proof of Lemma 13. In any view $v$, the leader can obtain two intermediate variables, block $b_1$ and block $b_2$ based on a $M_v$ and output $(b, \pi)$ after comparing ranks of $b_1$ and $b_2$.

Lemma 18. If $BG[\text{output} \leq 2$, block $b$, $Pr$ of. The proof resembles the proof of Lemma 13. In any view $v$, the leader can obtain two intermediate variables, block $b_1$ and block $b_2$ based on a $M_v$ and output $(b, \pi)$ after comparing ranks of $b_1$ and $b_2$.

Lemma 17. If $T_j \leq n - f$ for $1 \leq j \leq z$, VV-liveness holds in $BG[x, z]$.

Proof. Given any VIEW-UPDATE message $m$, let $b$ denote the parent block of $m$ block. According to the instantiation of $VV()$, $VV(m)$ outputs 1 only if $M_v \in m$ justify and FSB$(M_v) = (b, m$ justify $)$. Hence, $VV()$ outputs 1 by a correct replica in view $v$ only if there exists a set $M_v$ such that $(b, m$ justify $)$ is the output of FSB$(M_v)$.

Theorem 7. $BG[x, z]$ (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) $BG[x, z]$ satisfies DP1 2) $f < T \leq n - f$; 3) $|n+f+1/2| \leq T_1 \leq n - f$; and 4) $2f < T_j \leq n - f$ for $j \in [1..z]$.

Proof. Correctness follows from Theorem 1, Theorem 2, Theorem 3, Lemma 13, Lemma 15, Lemma 16 and Lemma 17.

G Proofs of Theorems for $BG[x, y, z]$ with DP1

Lemma 18. If $BG[x, y, z]$ satisfies DP1, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \lceil n+f+1 \rceil$, and $f < T \leq n - f$, then FSB-liveness holds for $BG[x, y, z]$.

Proof. The proof resembles the proof of Lemma 13. In any view $v$, the leader can obtain two intermediate variables, block $b_1$ and block $b_2$ based on a $M_v$ and output $(b, \pi)$ after comparing ranks of $b_1$ and $b_2$.

Lemma 19. If $BG[x, y, z]$ satisfies DP1, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \lceil n+f+1 \rceil$, and $f < T \leq n - f$, then $F_S_B_l_i_v_e_n_e$ holds in $BG[x, y, z]$.

Proof. Let $B^n = \{ b \mid b$ block $b$ has been committed before view $v \}$. According to Lemma 1, we can find $b^v \in B^n$ such that for all $b' \in B^n$ and $b' \neq b^v$, we have rank$(b^v) > \text{rank}(b')$. Since there exists a committed block in view $v - 1$, we know $b^v$.view = $v - 1$.

As there exists $QC_x$ formed for $b^v$ in view $v - 1$ and $y < z$, at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^v$ and sent VOTE$(y+1)$ for $b^v$ such that $b^v$.QC$_{y+1}$ is formed in view $v - 1$. Let $m$ denote a VIEW-UPDATE message such that $m$.view = $v$ and the parent block $b_v$ of $m$.block is conflicting with $b^v$ or $\text{rank}(b_v) < \text{rank}(b^v)$. Let $P = \{ p_i \mid p_i \in C \text{ (the set of correct replicas), }$ $VV(m, lockState) \text{ outputs in view } v \text{ by } p_i \}$.

For any one correct replica $p_i$ who has locked $b^v$, let $qc$ be its lockState when receiving $m$. Let $b_1$ denote QCLOCK$(qc)$. Since a correct replica only change its $QC_x$ to a $QC_x$ with the same or a higher rank, $\text{rank}(qc) \geq \text{rank}(b^v)$. $VV(m, lockState)$ returns true by $p_i$ if one of the following two conditions is satisfied:
1) \textit{m.justify} contains $b_v, \text{QC}_x, b_v, \text{view} < v$ and $\text{rank}(b_v) \geq \text{rank}(b_1)$ (lines 15-16 in Table 2).

2) \textit{m.justify} contains $M_v, vb, \text{num}(b_v, M_v, vb) > T/2$, $b_v, \text{view} < v$ and $\text{rank}(b_v) \geq \text{rank}(b)$ (lines 13-14 in Table 2).

Suppose that $\text{rank}(b_v) < \text{rank}(b')$. In this case, $\text{VV}(m, \text{lockState})$ outputs 0 since none of the above conditions is satisfied. Suppose that $\text{rank}(b_v) \geq \text{rank}(b')$. If case 1) is satisfied, then according to Lemma 10, $b_v$ must be equal to $b'$ or an extension of $b'$. If case 2) is satisfied, then at least one correct sender of a message in $M_v$ has changed its $vb$ from $b''$ to $b_v$ in view $v - 1$ since $\text{BG}[x, y, z]$ satisfies $\text{DP1}$. According to Algorithm 3, $\text{rank}(b_v) > \text{rank}(b'')$ and the $\text{QC}_x$ for the parent block of $b_v$ is received by the replica in view $v - 1$. By Lemma 10, $b_v$ must be an extension of $b''$. According to the assumption that either $b''$ is conflicting with $b'$ or $\text{rank}(b_v) < \text{rank}(b'')$, for all the correct replicas who have locked $b''$, $\text{VV}(m, \text{lockState})$ return false. Since $T_{y+1} \geq n - T_1 + f + 1$ and at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b''$, we know that $|P| < T_1 - f$ and VVL-safety holds in view $v$.

**Lemma 20.** If $\text{BG}[x, y, z]$ satisfies $\text{DP1}$, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $T > 2f$, $x \leq y < z$, and $T_{y+1} \geq n - T_1 + f + 1$, then VVL-safety holds in $\text{BG}[x, y, z]$.

Proof. For any view $v$, let $B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}$. According to Lemma 1, we can find $b'' \in B^v$ such that for all $b' \in B^v$ and $b' \neq b''$, we have $\text{rank}(b'') > \text{rank}(b')$.

We prove that VVL-safety holds for $\text{BG}[x, y, z]$ by proving that VVL-safety holds in every view. For a specific view $v'$, let $w$ denote $b'''.\text{view}$. We need to prove that VVL-safety holds in view $v'$. To do this, we need to do the following: First, we prove that VVL-safety holds in view $w + 1$. Then we prove that if VVL-safety holds in view $w + 1, \cdots, w + k$ (for any integer $k$ such that $1 \leq k \leq v' - w - 1$), VVL-safety also holds in view $w + k + 1$. Then for $k = v' - w - 1$, we know that VVL-safety holds in view $v'$.

According to Lemma 19, we know that VVL-safety holds in view $w + 1$.

Then, assume that VV-safety holds in view $w + 1, \cdots, w + k$ (for any integer $1 \leq k \leq v' - w - 1$). We need to show that VV-safety also holds in view $w + k + 1$. As $b''$ has been committed in view $w$ and $y < z$, at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b''$ and sent $\text{VOTE-} (y+1)$ for $b''$ to form $b'''.\text{QC}_{y+1}$ in view $w$. Let $m$ denote a $\text{VIEW-UPDATE}$ message such that $m.\text{view} = w + k + 1$ and the parent block $b'$ of $m.\text{block}$ is conflicting with $b''$ or $\text{rank}(b') < \text{rank}(b'')$. Let $P = \{p_i \mid p_i \in C \text{ (the set of correct replicas)}, \text{VV}(m, \text{lockState}) \text{ returns true in view } w + k + 1 \text{ by } p_i\}$.

For any one correct replica $p_i$ who has locked $b''$, let $qc$ be its $\text{lockState}$ when receiving $m$. Let $b_i$ denote $\text{QC}^{\text{Block}}(qc)$. Since a correct replica only change its $\text{QC}_x$ to a $\text{QC}_x$ with the same or a higher rank, $\text{rank}(qc) \geq \text{rank}(b'')$. $\text{VV}(m, \text{lockState})$ returns true by $p_i$ if one of the following two conditions is satisfied:
1) \( m.justify \) contains \( b'.QC_x, b'.view < w+k+1 \) and \( \text{rank}(b') \geq \text{rank}(b_i) \) (lines 15-16 in Table 2).

2) \( m.justify \) contains \( M_{w+k+1}, vb, b'.view < w+k+1 \) and \( \text{num}(b', M_{w+k+1}, vb) > T/2 \) and \( \text{rank}(b') \geq \text{rank}(b_i) \) (lines 13-14 in Table 2).

Suppose that \( \text{rank}(b_i) < \text{rank}(b'') \). In this case, \( \text{Vv}(m, \text{lockState}) \) outputs 0 since both conditions are not satisfied. Suppose that \( \text{rank}(b') \geq \text{rank}(b'') \). If case 1) is satisfied, then according to Lemma 7, Lemma 10, Lemma 12 and the inductive hypothesis, \( b' \) must be equal to \( b'' \) or an extension of \( b'' \). If case 2) is satisfied, then at least one correct sender of a message in \( M_{w+k+1} \) has changed its \( vb \) from \( b'' \) to \( b' \) in view \( w, \ldots, w+k \) since \( BG[x, y, z] \) satisfies \( \text{DP1} \) and \( T > 2f \). According to Algorithm 3, \( \text{rank}(b') > \text{rank}(b'') \), the \( QC_x \) for the parent block \( b'' \) of \( b' \) is received by the replica, \( b'.view = b'.view \) and \( \text{rank}(b'') \geq \text{rank}(b'') \). By Lemma 10, Lemma 12 and the inductive hypothesis, \( b' \) must be an extension of \( b'' \). According to the assumption that either \( b' \) is conflicting with \( b'' \) or \( \text{rank}(b') < \text{rank}(b'') \), for all the correct replicas who have locked \( b'' \), \( \text{Vv}(m, \text{lockState}) \) outputs 0. Since \( T_{y+1} \geq n-T_1+f+1 \) and at least \( T_{y+1}-f \geq n-T_1+1 \) correct replicas have locked \( b'' \), we know that \( |P| < T_1-f \) and VVL-safety holds in view \( w+k+1 \).

Then for \( k = \nu' - w - 1 \), we know that VVL-safety holds in view \( \nu' \).

**Lemma 21.** If \( BG[x, y, z] \) satisfies \( \text{DP1} \), \( f < T_j \leq n-f \) for \( j \in [1..n] \), and \( T_1 \geq \left\lfloor \frac{n+f+1}{2} \right\rfloor \), then VVL-liveness holds in \( BG[x, y, z] \).

**Proof.** For any view \( v \), let \( B_i^v = \{ b | \text{block } b \text{ has been locked before view } v \} \). According to Lemma 7 and Lemma 10, we can fine \( b_i^v \in B_i^v \) such that for all \( b_i^v \in B_i^v \) and \( b_i^v \neq b_i^v \), \( \text{rank}(b_i^v) > \text{rank}(b_i^v) \). Given a \text{VIEW-UPDATE} message \( m \) in view \( v \), let \( b \) denote the parent block of \( m.\text{block} \). If \( (b, m.\text{justify}) \) is the output of \( \text{FSB}(M_v) \) on some \( M_v \), there are two cases to consider:

1) \( b_i^v.\text{QC}_x \in M_v.\text{QC}_x \). In this case, \( b \) is a block such that \( \text{rank}(b) \geq b_i^v \). That’s because \( b \) is either equal to \( b_1 \) or equal to \( b_2 \), where \( b_1 \) and \( b_2 \) are two intermediate variables obtained from \( M_v \). Since \( b_i^v.\text{QC}_x \in M_v.\text{QC}_x \), we have that \( \text{rank}(b_1) \geq b_i^v \). If \( b_1 \) exists, we can also have that \( \text{rank}(b_1) \geq b_i^v \) since \( BG[x, y, z] \) satisfies \( \text{DP1} \). Therefore, we know that \( \text{Vv}(m, \text{lockState}) \) outputs 1 by all correct replicas since either condition 1) (lines 15-16 in Table 2) or condition 2) (lines 13-14 in Table 2) is satisfied for them.

2) \( b_i^v \notin M_v.\text{QC}_x \). We distinguish two cases. If there exists a block \( b_i \) such that \( b_i.\text{QC}_x \in M_v.\text{QC}_x \) and \( \text{rank}(b_i) \geq \text{rank}(b_i^v) \), then block \( b \) should satisfies that \( \text{rank}(b) \geq \text{rank}(b_i) \). Therefore, we know that \( \text{Vv}(m, \text{lockState}) \) will output 1 by all correct replicas. If any \( qc \in M_v.\text{QC}_x \) satisfies that \( \text{rank}(\text{QC}(qc)) < \text{rank}(b) \), we prove that the output of \( \text{FSB}(M_v) \) is \( (b_i^v, M_v, vb) \). Since \( BG[x, y, z] \) satisfies \( \text{DP1} \), we have \( \text{num}(b_i^v, M_{b_i^v}, vb) > T/2 \). Any correct replica changes its \( vb \) and \( QC_x \) only in the case that it has received a \( QC_x \) \( qc \) such that \( \text{rank}(\text{QC}(qc)) \geq \text{rank}(vb) \). Therefore, no correct replica of any message in \( M_v \) has changed its \( vb \) from \( b_i^v \) to some other block and \( \text{num}(b_i^v, M_{b_i^v}, vb) > T/2 \). Then the output of \( \text{FSB}(M_v) \) is \( (b_i^v, M_v, vb) \) and \( \text{Vv}(m, \text{lockState}) \) out-
puts 1 by all correct replicas since condition 2) (lines 13-14 in Table 2) is satisfied for them.

**Theorem 8.** $BG[x, y, z]$ (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) $BG[x, y, z]$ satisfies $\text{DP}1$; 2) $2f < T \leq n - f$; 3) $\left\lceil \frac{n+f+1}{2} \right\rceil \leq T_i \leq n - f$; 4) $f < T_j \leq n - f$ for $j \in [1..z]$; and 5) $x \leq y < z$, $n - T_1 + f + 1 \leq T_y + 1$.

**Proof.** Correctness follows from Theorem 4, Theorem 5, Theorem 6, Lemma 18, Lemma 20 and Lemma 21.

**H Proofs of Theorems for $BG[x, y, z]$ with $\text{DP}2$.**

**Lemma 3.** If $T - (n - T_1 + f) \geq f + 1$ and $T - (n - T_x+1 + f) \geq T - (2f + 1)$, then $BG[x, y, z]$ satisfies $\text{DP}2$.

**Proof.** In a $BG[x, y, z]$, for any block $b$, if $b$ has been locked by a correct replica $p_i$ in view $v$, then $b.view = v$ and $p_i$ has also set its $QC_y$ to $b.QC_y$. According to Lemma 7, $b.QC_1$ is also formed by the leader in view $v$. Accordingly, at least $T_1 - f$ correct replicas have sent $\text{VOTE}-1$ messages for $b$ such that $b.QC_1$ is formed. As $p_i$ set its $QC_y$ to $b.QC_y$ in view $v$, $b$ is block proposed in normal case and the $T_1 - f$ replicas set their $vb$ to $b$ in view $v$. Thus, fewer than $n - T_1$ correct replicas have not yet set their $vb$ to $b$ in view $v$. Therefore, for any $M^b$, at most $n - T_1 + f$ messages are sent by replicas who have not set their $vb$ to $b$, i.e., there at least $T - (n - T_1 + f)$ $b$ in $M^b.vb$. Since $T - (n - T_1 + f) \geq f + 1$, we have that $\text{VOTES}(b, T, z) > f + 1$.

Besides, for any block $d$, if $d.QC_z$ is received by a correct replica $p_i$ and $p_i$ set its $QC_z$ to $d.QC_z$ in view $v$, then $d.view = v$. According to Lemma 7 and $x \leq y < z$, $d.QC_z$ is also formed in view $v$. Accordingly, at least $T_{x+1} - f$ correct replicas have received $d.QC_x$ and sent $\text{VOTE}-(x + 1)$ messages for $d$ such that $d.QC_{x+1}$ is formed. As $p_i$ set its $QC_z$ to $d.QC_z$ in view $v$, $d$ is block proposed in normal case and the $T_{x+1} - f$ correct replicas set their $QC_z$’s to $d.QC_z$ in view $v$. Thus, fewer than $n - T_{x+1}$ correct replicas have not yet set their $QC_z$’s to $d.QC_x$, i.e., there at least $T - (n - T_{x+1} + f)$ $b$ in $M^b.vb$. Since $T - (n - T_{x+1} + f) \geq T - (2f + 1)$, we have that $\text{CERTS}(d, T, x, z) \geq T - (2f + 1)$.

Therefore, $BG[x, y, z]$ satisfies $\text{DP}2$.

**Lemma 22.** If $BG[x, y, z]$ satisfies $\text{DP}2$, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, and $f < T \leq n - f$, then $FSBL$-liveness holds for $BG[x, y, z]$.

**Proof.** Given a $M_v$, we need to prove that $\text{FSB}(M_v)$ outputs some $(b, \pi)$. Based on $M_v$, we can obtain four intermediate variables, block $b_0$, $b_1$, $b_2$ and $b_3$. Since $\text{num}(b_1, M_v.vb) > f + 1$ and $\text{num}(b_2, M_v.vb) > f + 1$, $b_1$, $b_2$ and $b_3$ are unique
blocks or null. By Lemma 10, b₃ is also a unique block. Therefore, after comparing ranks of b₀, b₁, b₂ and b₃, \( FSB(M_v) \) will output a block together with a proof \( \pi \).

**Lemma 23.** If \( BG[x, y, z] \) satisfies DP2, \( T_j > f \) for \( j \in [1..z] \), \( T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil \), \( x \leq y < z \), \( T_{y+1} \geq n - T_1 + f + 1 \), and there exists a block committed by at least one correct replica in view \( v-1 \), then VVL-safety holds in view \( v \) for \( BG[x, y, z] \).

**Proof.** Let \( B^v = \{ b \mid \text{block} \ b \ \text{has been committed before view} \ v \} \). According to Lemma 1, we can find \( b^v \in B^v \) such that for all \( b' \in B^v \) and \( b' \neq b^v \), we have \( \text{rank}(b') > \text{rank}(b^v) \). Since there exists a committed block in view \( v-1 \), we know \( b^v.\text{view} = v - 1 \).

As there exists \( QC_x \) formed for \( b^v \) in view \( v-1 \) and \( y < z \), at least \( T_{y+1} - f \geq n - T_1 + f + 1 \) correct replicas have locked \( b^v \) and sent \( \text{VOTE}(y+1) \) for \( b^v \) to form \( b^v.QC_{y+1} \) in view \( v-1 \). Let \( m \) denote a \( \text{VIEW-UPDATE} \) message such that \( m.\text{view} = v \) and the parent block \( b_v \) of \( m.\text{block} \) is conflicting with \( b^v \) or \( \text{rank}(b_v) < \text{rank}(b^v) \). Let \( P = \{ p_i \mid p_i \in C \ \text{(the set of correct replicas)} \} \), \( \text{VVL}(m, lockState) \) outputs 1 in view \( v \) by \( p_i \) if one of the following four conditions is satisfied:

1. \( m.\text{justify} \) contains \( b_v.QC_x \), and \( \text{rank}(b_v) \geq \text{rank}(b^v) \) and \( b_v.\text{view} < v \) (lines 32-33 in Table 2).
2. \( m.\text{justify} \) contains \( M_v.vb, \text{num}(b_v), M_v.vb \geq f + 1 \) and \( \text{rank}(b_v) \geq \text{rank}(b_i) \) and \( b_v.\text{view} < v \) (lines 34-35 in Table 2).
3. \( m.\text{justify} \) contains \( M_v.vb, \text{num}(b_v), M_v.vb \geq f + 1 \) and \( b_v = b_i \) and \( b_v.\text{view} < v \) (lines 36-37 in Table 2).
4. \( m.\text{justify} \) contains \( M_v.QC_x \), and \( \text{num}(b_v), QC_x, M_v, QC_x \) \( > 2f+1 \) and \( b_v.\text{view} < v \) (lines 38-39 in Table 2).

Suppose that \( \text{rank}(b_v) < \text{rank}(b^v) \). In this case, \( \text{VVL}(m, lockState) \) outputs 1 only when condition 4) is satisfied. Since \( BG[x, y, z] \) satisfies DP2 and \( b^v \) is committed in view \( v-1 \), we have \( \text{num}(b^v), QC_x, M^b, QC_x \geq T - (2f + 1) \). As \( \text{rank}(b_v) < \text{rank}(b^v) \) and a correct replicas only updates its \( QC_x \) to a \( QC_x \) for a block with a higher rank, \( \text{num}(b_v), QC_x, M_v, QC_x \) \( \leq 2f + 1 \), a contradiction.

Suppose that \( \text{rank}(b_v) \geq \text{rank}(b^v) \). If condition 1) or condition 4) is satisfied, then according to \( x \geq y \), Lemma 7 and Lemma 10, \( b_v \) must be equal to \( b^v \) or is an extension of \( b^v \). If case 2) is satisfied, then at least one correct replica has received a \( QC_x \) for the parent block \( b' \) of \( b_v \) such that \( b'.\text{view} = b_v.\text{view} \). Since \( \text{rank}(b_v) \geq \text{rank}(b_i) \geq \text{rank}(b^v) \), we have that \( \text{rank}(b') \geq \text{rank}(b^v) \). According to \( x \leq y \), Lemma 7 and Lemma 10, \( b' \) must be equal to \( b^v \) or an extension of \( b^v \). Therefore, \( b_v \) must be an extension of \( b^v \). If case 3) is satisfied, then \( b_v = b_i \).

Therefore, \( b_v \) is equal to \( b^v \) or an extension of \( b^v \).
According to the assumption that either $b_v$ is conflicting with $b^v$ or $\text{rank}(b_v) < \text{rank}(b^v)$, for all the correct replicas who have locked $b^v$, $\text{VV}(m, \text{lockState})$ return false. Since $T_{y+1} \geq n - T_1 + f + 1$ and at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^v$, we know that $|P| < T_1 - f$ and VVL-safety holds in view $v$.

**Lemma 24.** If $BG[x, y, z]$ satisfies $\mathcal{P}2$, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, and $T_{y+1} \geq n - T_1 + f + 1$, then VVL-safety holds in $BG[x, y, z]$.

**Proof.** For any view $v$, let $B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}$. According to Lemma 1, we can find $b^v \in B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$, we have $\text{rank}(b^v) > \text{rank}(b')$.

We prove that VVL-safety holds for $BG[x, y, z]$ by proving that VVL-safety holds in every view. For a specific view $v'$, let $w$ denote $b^{v'} . \text{view}$. We need to prove that VVL-safety holds in view $v'$. Our proof consists of the following steps. First, we prove that VVL-safety holds in view $w + 1$. Then we prove that if VVL-safety holds in view $w + 1, \cdots, w + k$ (for any integer $1 \leq k \leq v' - w - 1$), VVL-safety also holds in view $w + k + 1$. Then for $k = v' - w - 1$, we know that VVL-safety holds in view $v'$.

According to Lemma 23, we know that VVL-safety holds in view $w + 1$.

Then, assume that VVL-safety holds in view $w + 1, \cdots, w + k$ (for any integer $1 \leq k \leq v' - w - 1$). We need to show that VVL-safety also holds in view $w + k + 1$. As there exists $QC_z$ formed for $b^{v'}$ in view $w$ and $y < z$, at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^{v'}$ and sent a $\text{VOTE-(y+1)}$ message for $b^{v'}$ to form $b^{v'} . QC_{y+1}$ in view $w$. Let $m$ denote a $\text{VIEW-UPDATE}$ message such that $m . view = w + k + 1$ and the parent block $b'$ of $m . block$ is conflicting with $b^{v'}$ or $\text{rank}(b') < \text{rank}(b^{v'})$. Let $P = \{p_i \mid p_i \in C \text{ (the set of correct replicas)}, \text{VV}(m, \text{lockState}) \text{ outputs 1 in view } w + k + 1 \text{ by } p_i\}$.

As there exists $QC_z$ formed for $b^{v'}$ in view $w$ and $y < z$, at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^{v'}$ and sent $\text{VOTE-(y+1)}$ for $b^{v'}$ to form $b^{v'} . QC_{y+1}$ in view $w$. For any correct replica $p_i$ who has locked $b^{v'}$, let $qc$ be its $\text{lockState}$ when receiving $m$. Let $b_l$ denote $\text{QC}Block(qc)$. Since a correct replica only change its $QC_x$ to a $QC_x$ with the same or a higher rank, $\text{rank}(qc) \geq \text{rank}(b^{v'})$. According to Lemma 7, Lemma 10, Lemma 12 and the inductive hypothesis, $b_l . view \in \{w, \cdots, w + k\}$ and $b_l$ is equal to $b^{v'}$ or an extension of $b^{v'}$. $\text{VV}(m, \text{lockState})$ returns true by $p_i$ if one of the following four conditions is satisfied:

1. $m . justify$ contains $M_{w+k+1, QC_x}$, and $\text{rank}(b') \geq \text{rank}(b_l)$ and $b_v . view < w + k + 1$ (lines 32-33 in Table 2).
2. $m . justify$ contains $M_{w+k+1, vb}$, $\text{num}(b', M_{w+k+1, vb}) \geq f + 1$ and $\text{rank}(b') > \text{rank}(b_l)$ and $b'_l . view < w + k + 1$ (lines 34-35 in Table 2).
3. $m . justify$ contains $M_{w+k+1, vb}$, $\text{num}(b', M_{w, vb}) \geq f + 1$ and $b_v = b_l$ and $b'_l . view < w + k + 1$ (lines 36-37 in Table 2).
4. $m . justify$ contains $M_{w+k+1, QC_x}$, and $\text{num}(b', QC_x, M_v, QC_x) > 2f + 1$ and $b'_l . view < w + k + 1$ (lines 38-39 in Table 2).
Suppose that \(\text{rank}(b') < \text{rank}(b''')\). In this case, \(\text{VV}(m, \text{lockState})\) returns true only when condition 4) is satisfied. Since \(\text{BG}[x, y, z]\) satisfies DP2 and \(b''\) is committed in view \(w\), we have \(\text{num}(b''', \text{QC}_x, M^{b''}, \text{QC}_x) \geq T - (2f + 1)\). As \(\text{rank}(b') < \text{rank}(b''')\) and correct replicas only change its \(\text{QC}_x\) to a \(\text{QC}_x\) the same or a higher rank, \(\text{num}(b', \text{QC}_x, M_{w+k+1}, \text{QC}_x) \leq 2f + 1\), a contradiction.

Suppose that \(\text{rank}(b') \geq \text{rank}(b''')\). If condition 1) or condition 4) is satisfied, then according to \(x \leq y\), Lemma 10, Lemma 7, Lemma 12 and the inductive hypothesis, \(b'\) must be equal to \(b''\) or an extension of \(b''\). If case 2) is satisfied, then at least one correct replica has received a \(\text{QC}_x\) for the parent block \(b''\) of \(b'\) such that \(b''\).view = \(b'.view\). Since \(\text{rank}(b') > \text{rank}(b_{l+1}) \geq \text{rank}(b''')\), we have that \(\text{rank}(b') \geq \text{rank}(b''')\). According to \(x \leq y\), Lemma 10, Lemma 7, Lemma 12 and the inductive hypothesis, \(b''\) must be equal to \(b''\) or an extension of \(b''\). Therefore, \(b'\) must be an extension of \(b''\). If case 3) is satisfied, then \(b' = b_{l+1}\). Therefore, \(b'\) is equal to \(b''\) or an extension of \(b''\).

According to the assumption that \(b'\) is conflicting with \(b''\), for all the correct replicas who have locked \(b''\), \(\text{VV}(m, \text{lockState})\) will output 0. Since \(T_{y+1} \geq n - T_f + f + 1\) and at least \(T_{y+1} - f \geq n - T_f + 1\) correct replicas have locked \(b''\), we know that \(|P| < T_f - f\) and VVL-safety holds in view \(w + k + 1\).

Then for \(k = v' - w - 1\), we know that VVL-safety holds in view \(v'\).

**Lemma 25.** If \(\text{BG}[x, y, z]\) satisfies DP2, \(f < T_j \leq n - f\) for \(j \in [1..z]\), and \(T_j \geq \left\lfloor \frac{n + f + 1}{2} \right\rfloor\), then VVL-safety holds in \(\text{BG}[x, y, z]\).

**Proof.** For any view \(v\), let \(B^v_i = \{b \mid \text{block } b \text{ has been locked before view } v\}\). According to \(x \leq y\) and Lemma 10, we can find \(b^v_{i+1} \in B^v_i\) such that for all \(b' \in B^v_i\) and \(b' \neq b^v_i\), we have \(\text{rank}(b^v_i) > \text{rank}(b')\). Given a VIEW-UPDATE message \(m\) in view \(v\), let \(b\) denote the parent block of \(m.\text{block}\). If \((b, m, \text{justify})\) is the output of \(\text{FSB}(M_v)\) on some \(M_v\), there are two cases to consider:

1) \(b^v_i.\text{QC}_x \in M_v.\text{QC}_x\). In this case, \(b\) is a block such that \(\text{rank}(b) \geq b^v_i\). Therefore, \(\text{VV}(m, \text{lockState})\) outputs 1 by all correct replicas since either condition 1) (lines 34-35 in Table 2) or condition 2) (lines 36-37 in Table 2) is satisfied for them.

2) \(b^v_i \notin M_v.\text{QC}_x\). Let \(b_3\) denote the block with the highest rank such that \(b_3.\text{QC}_x \in M_v.\text{QC}_x\). We distinguish two sub-cases. If \(\text{rank}(b_3) \geq \text{rank}(b^v_i)\), then we know that the block \(b\) should satisfies that \(\text{rank}(b) \geq \text{rank}(b^v_i)\). Therefore, we know that \(\text{VV}(m, \text{lockState})\) outputs 1 by all correct replicas.

If \(\text{rank}(b_3) < \text{rank}(b^v_i)\), we prove that \(\text{num}(b^v_i, M_v, vb) \geq f + 1\). Since \(\text{BG}[x, y, z]\) satisfies DP2, we have \(\text{num}(b^v_i, M^{b^v_i}, vb) > f + 1\). Any correct replica changes its \(vb\) and \(\text{QC}_x\) only in the case that it has received a \(\text{QC}_x\) \(qc\) such that \(\text{rank}(qc.\text{block}(qc)) \geq \text{rank}(vb)\). Thus, for any correct sender of a message in \(M_v\), if a replica has sent a \text{vote}−1 message for \(b^v_i\), it will not change its \(vb\) before sending \text{NEW-VIEW} message. We have that \(\text{num}(b^v_i, M_v, vb) \geq f + 1\). If there exists another block \(d\) such that \(\text{num}(d, M_v, vb) \geq f + 1\) and \(\text{rank}(d) = \text{rank}(b^v_i)\), then the output of \(\text{FSB}(M_v)\) is \((b_3, b_3.\text{QC}_x, M_v.\text{QC}_x)\). As \(\text{num}(d, M_v, vb) \geq f + 1\) and \(\text{num}(b^v_i, M_v, vb) \geq f + 1\), \(\text{QC}_x\) for the parent block of \(b^v_i\) and \(d\) are both
formed in view $b^v_i\cdot\text{view}$. According to Lemma 10, $b^v_i$ and $d$ have the same parent block $b_i$. Then $\text{num}(b_i, QC_z, M_v, QC_z) \geq 2f + 2$. Therefore, we know that $\text{V}v(m, \text{lockState})$ outputs 1 by all correct replicas since condition 4) (lines 34-35 in Table 2) is satisfied for them. If no such block $d$ exists, then the output of $\text{FSB}(M_v)$ is $(b^v_i, M_v, vb)$. Therefore, according to Lemma 10, we know that $\text{V}v(m, \text{lockState})$ outputs 1 by all correct replicas since either condition 2) (lines 36-37 in Table 2) or condition 3) (lines 38-39 in Table 2) is satisfied. This completes the proof.

**Theorem 9.** $BG[x, y, z]$ (in Table 2) achieves safety and responsiveness if the following are satisfied: 1) $BG[x, y, z]$ satisfies DP2; 2) $f < T \leq n - f$; 3) $\left\lceil \frac{n + f + 1}{2} \right\rceil \leq T_1 \leq n - f$; 4) $f < T_j \leq n - f$ for $j \in [1..2]$; and 5) $n - T_1 + f + 1 \leq T_2 + 1$.

**Proof.** Correctness follows from Theorem 4, Theorem 5, Theorem 6, Lemma 22, Lemma 24 and Lemma 25.

## I Proofs of Theorems for $BG[x, z]$ with DP3

**Lemma 4.** If $x < z$ and $T - (n - T_{x+1} + f) > 0$ or if $x \geq z$ and $T - (n - 1) > 0$, then $BG[x, z]$ satisfies DP3.

**Proof.** We consider two cases in the lemma separately.

In a $BG[x, z]$ ($x < z$), for any block $b$, if $b.QC_z$ is received by a correct replica $p_i$ and $p_i$ set its $QC_z$ to $b.QC_z$ in view $v$, then $b.view = v$. According to $x < z$ and Lemma 7, $b.QC_{z+1}$ is also formed by the leader in view $v$. Accordingly, at least $T_{z+1} - f$ correct replicas have sent $\text{VOTE}-(x+1)$ messages for $b$ such that $b.QC_{z+1}$ is formed. As $p_i$ set its $QC_y$ to $b.QC_y$ in view $v$, $b$ is block proposed in normal case and the $T_{z+1} - f$ correct replicas set their $QC_z$'s to $b.QC_{z+1}$ in view $v$. Thus, fewer than $n - T_{x+1}$ correct replicas have not yet set their $QC_z$'s to $b.QC_z$ in view $v$. Therefore, for any $M^b$, at most $n - T_{z+1} + f$ messages are sent by replicas who have not set their $QC_z$'s to $d.QC_z$, i.e., there at least $T - (n - T_{z+1} + f)$ in $M^b.vb$. Since $T - (n - T_{z+1} + f) > 0$, we have that $\text{CERTS}(b, T, x, z) = 0$.

In a $BG[x, z]$ ($x = z$), for any block $b$, if $b.QC_z$ is received by a correct replica $p_i$ and $p_i$ set its $QC_z$ to $b.QC_z$ in view $v$, then $b.view = v$. In this case, $b.QC_z$ is received by at least only one replica. If $T - (n - 1) > 0$, a new leader needs to collect new-view messages from all the replicas in the system. Then $b.QC_z$ is included in $M^b.QC_z$ and we have that $\text{CERTS}(b, T, x, z) > 0$.

That completes the proof.

**Lemma 26.** If $T_j > f$ for all $j \in [1..z]$, and $T_1 \geq \left\lceil \frac{n + f + 1}{2} \right\rceil$, then $\text{FSB}$-liveness holds for $BG[x, z]$.

**Proof.** Given a $M_v$, we need to prove that $\text{FSB}(M_v)$ outputs some $(b, \pi)$. By Lemma 10, $b_2$ is a unique block based on $M_v$. Therefore, after comparing ranks
of blocks whose \( QC_x \) in \( M_e \), \( \text{fsb}(M_v) \) will output a unique block together with a proof \( \pi \).

\textbf{Lemma 27.} If \( BG[x, z] \) satisfies \( \text{DP3} \), \( T_j > f \) for \( j \in [1..z] \), \( T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil \), \( T > f \), and there exists a block committed by at least one correct replica in view \( v - 1 \), then \( \text{FSB-safety} \) holds in view \( v \) for \( BG[x, z] \).

\textit{Proof.} Let \( B^v = \{ b \mid \text{block } b \text{ has been committed before view } v \} \). According to Lemma 1, there exists \( b' \in B^v \) such that for all \( b' \in B^v \) and \( b' \neq b^v \), we have \( \text{rank}(b^v) > \text{rank}(b') \). Since there exists a committed block in view \( v - 1 \), we know \( b^v, \text{view} = v - 1 \). By Lemma 26, \( \text{fsb}(M_v) \) outputs some \( (b_2, \pi) \). We now prove that \( b_2 \) is either \( b^v \) or an extension of \( b^v \). Since \( \text{DP3}(b^v, T, x) \) holds in \( BG[x, z] \), we consider two cases:

1) \( b^v, \text{QC}_x \in M_v, \text{QC}_x \). In this scenario, the output should satisfy that \( b_2, \text{QC}_x \in M_v, \text{QC}_x \) and \( \text{rank}(b_2) \geq \text{rank}(b^v) \). According to Lemma 10, we have that \( b_2 \) equals \( b^v \) or \( b_2 \) is an extension of \( b^v \).

2) \( b^v, \text{QC}_x \notin M_v, \text{QC}_x \). Since \( \text{DP3} \) holds in \( BG[x, z] \), it is clear that at least one correct sender \( p_i \) of a message in \( M_v \) has committed its \( \text{QC}_x \) from \( b^v, \text{QC}_x \) to \( \text{QC}_x \) for some other block \( b' \) before sending the \text{NEW-VIEW} message. According to Algorithm 3, \( p_i \) changes its \( \text{QC}_x \) in view \( v - 1 \) only if it receives \( b', \text{QC}_x \) and \( \text{rank}(b') \geq \text{rank}(b^v) \). By Lemma 10, we know that \( b' \) must be equal to or an extension of \( b^v \), and \( b_2 \) must be an extension of \( b^v \).

Hence, \( \text{FSB-safety} \) holds in view \( v \) in \( BG[x, z] \).

\textbf{Lemma 28.} If \( BG[x, z] \) satisfies \( \text{DP3} \), \( T_j > f \) for \( j \in [1..z] \), \( T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil \), and \( f < T \leq n - f \), then \( \text{FSB-safety} \) holds for \( BG[x, z] \).

\textit{Proof.} For any view \( v \), let \( B^v = \{ b \mid \text{block } b \text{ has been committed before view } v \} \). According to Lemma 1, there exists \( b^v \in B^v \) such that for all \( b' \in B^v \) and \( b' \neq b^v \), we have \( \text{rank}(b^v) \geq \text{rank}(b') \).

We prove that \( \text{FSB-safety} \) holds for \( BG[x, z] \) by proving that \( \text{FSB-safety} \) holds in every view. For a specific view \( v' \), let \( w \) denote \( b^v, \text{view} \). We prove that \( \text{FSB-safety} \) holds in view \( v' \) by iterative method. The proof consists of the following steps. First, we prove that \( \text{FSB-safety} \) holds in view \( w + 1 \). Then we prove that if \( \text{FSB-safety} \) holds in view \( w + 1, \ldots, w + k \) (for any integer \( 1 \leq k \leq v' - w - 1 \)), \( \text{FSB-safety} \) also holds in view \( w + k + 1 \). When \( k = v' - w - 1 \), we know that \( \text{FSB-safety} \) holds in view \( v' \).

According to Lemma 27, we know that \( \text{FSB-safety} \) holds in view \( w + 1 \). Now we assume that \( \text{FSB-safety} \) holds in view \( w + 1, \ldots, w + k \) (for any integer \( 1 \leq k \leq v' - w - 1 \)). We need to show that \( \text{FSB-safety} \) holds in view \( w + k + 1 \). Let \( (b, \pi) \) denote the output of \( \text{fsb}(M_{w+k+1}) \) based on a snapshot \( M_{w+k+1} \). There are two situations to consider:

1) \( b^v, \text{QC}_x \in M_{w+k+1}, \text{QC}_x \). In this scenario, the output should satisfy that \( b, \text{QC}_x \in M_{w+k+1}, \text{QC}_x \) and \( \text{rank}(b) \geq \text{rank}(b^v) \). According to Lemma 10, Lemma 11 and the inductive hypothesis, we have that \( b \) equals \( b^v \) or \( b \) is an extension of \( b^v \).
Lemma 5. If \( x, y, z \) and \( r \) are such that \( x < y \), \( y < z \), and \( x < z \), it is clear that at least one correct sender \( p_i \) of a message in \( M_{w+k+1} \) has changed its \( QC_x \) from \( b' \) to \( QC_x \) for some other block \( b' \) before sending a new-view message. According to Algorithm 3, \( p_i \) changes its \( QC_x \) only if it receives \( b', QC_x \) and \( rank(b') \geq rank(b') \). Therefore, we have that \( rank(b) \geq rank(b') \), and \( QC_x \) for blocks \( b, b', b'' \) are formed in view \( \{ w, \cdots, w+k \} \) and \( b \neq b'' \). By Lemma 10, Lemma 11 and the inductive hypothesis, we know that \( b' \) must be equal to or an extension of \( b'' \), and \( b \) must be an extension of \( b'' \).

In both cases \( b \) is either \( b'' \) or an extension of \( b'' \), then FSB-safety holds in view \( w+k+1 \). When \( k = v' - w - 1 \), we know that FSB-safety holds in view \( v' \). We conclude that FSB-safety holds for any \( v' \). Hence, FSB-safety holds in \( BG[x, z] \).

Lemma 29. If \( T_1 > f \), VV-safety holds in \( BG[x, z] \).

Proof. The proof is the same with that of Lemma 16.

Lemma 30. If \( T_j \leq n - f \) for \( 1 \leq j \leq z \), VV-liveness holds in \( BG[x, z] \).

Proof. The proof is the same with that of Lemma 17.

Theorem 10. \( BG[x, z] \) (in Table 2) achieves safety and optimistic responsiveness if the following are satisfied: 1) \( BG[x, z] \) satisfies DP3; 2) \( f < T \leq n - f \); 3) \( \left\lfloor \frac{n-f+1}{2} \right\rfloor \leq T_1 \leq n - f ; \) and 4) \( f < T_j \leq n - f \) for \( j \in [1..z] \).

Proof. Correctness follows from Theorem 1, Theorem 2, Theorem 3, Lemma 26, Lemma 28, Lemma 29 and Lemma 30.

J Proofs of Theorems for \( BG[x, y, z] \) with DP3

Lemma 5. If \( x < y \) and \( T - (n - T_{x+1} + f) > 0 \) or if \( x = y \) and \( T - (n-1) > 0 \), then \( BG[x, y, z] \) satisfies DP3.

Proof. We consider two cases in the lemma separately: \( x < z \) and \( x = z \).

In a \( BG[x, y, z] \) protocol such that \( x < z \), for any block \( b \), we know that if \( b \) has been locked by any correct replica \( p_i \) in view \( v \), then \( b.view = v \) and \( p_i \) has also set its \( QC_x \) to \( b.QC_y \). According to \( x < y \) and Lemma 7, \( b.QC_{x+1} \) is also formed by the leader in view \( v \). Accordingly, at least \( T_{x+1} - f \) correct replicas have sent \( vote = (x+1) \) messages for \( b \) such that \( b.QC_{x+1} \) is formed. As \( p_i \) set its \( QC_y \) to \( b.QC_y \) in view \( v \), \( b \) is block proposed in normal case and the \( T_{x+1} - f \) replicas set their \( vb \) to \( b \) in view \( v \). The \( T_{x+1} - f \) correct replicas set their \( QC_x \)’s to \( b.QC_x \) in view \( v \). Thus, fewer than \( n - T_{x+1} \) correct replicas have not yet set their \( QC_x \)’s to \( b.QC_x \) in view \( v \). Therefore, for any \( M^b \), at most \( n - T_{x+1} + f \) messages are sent by replicas who have not set their \( QC_x \)’s to \( b.QC_x \), i.e., there at least \( T - (n - T_{x+1} + f) \) \( b.QC_x \) in \( M^b.QC_x \). Since \( T - (n - T_{x+1} + f) > 0 \), we have that \( Cerfs(b, T, x, y) > 0 \).
In a BG\([x, y, z]\) protocol such that \(x = z\), for any block \(b\), we know that if \(b\) has been locked by any correct replica \(p_i\) in view \(v\), then \(b . view = v\) and \(p_i\) has also set its \(QC_y\) to \(b . QC_y\). In this case, \(b . QC_y\) is received by at least only one replica. If \(T - (n - 1) > 0\), a new leader need to collect new-view messages from all the replicas in the system. Then \(b . QC_y\) is included in \(M^b . QC_x\) and we have that \(\text{Certs}(b, T, x, y) > 0\).

That completes the proof.

**Lemma 31.** If \(BG[x, y, z]\) satisfies DP3, \(T_j > f\) for \(j \in [1..z]\), \(T_1 \geq \left\lceil\frac{n+f+1}{2}\right\rceil\), and \(f < T \leq n - f\), then FSBL-liveness holds for \(BG[x, y, z]\).

**Proof.** The proof resembles the proof of Lemma 26. In any view \(v\), the leader can obtain block \(b_2\) based on any \(M_v\) (a valid view change snapshot) and output \((b_2, b_2 . QC_x)\) according to Lemma 10.

**Lemma 32.** If \(BG[x, y, z]\) satisfies DP3, \(T_j > f\) for \(j \in [1..z]\), \(T_1 \geq \left\lceil\frac{n+f+1}{2}\right\rceil\), \(T > 2f\), \(T_{y+1} \geq n - T_1 + f + 1\), and there exists a block committed by at least one correct replica in view \(v - 1\), then VVL-safety holds in view \(v\) for \(BG[x, y, z]\).

**Proof.** Let \(B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}\). According to Lemma 1, we can find \(b^v \in B^v\) such that for all \(b^v \in B^v\) and \(b^v \neq b^v\), we have \(\text{rank}(b^v) > \text{rank}(b^v)\). Since there exists a committed block in view \(v - 1\), we know \(b^v . view = v - 1\).

As there exists \(QC_z\) formed for \(b^v\) in view \(v - 1\) and \(z < y\), at least \(T_{y+1} - f \geq n - T_1 + f + 1\) correct replicas have locked \(b^v\) and sent \(\text{VOTE}(y+1)\) for \(b^v\) to form \(b^v . QC_y\) in view \(v - 1\). Let \(m\) denote a \text{VIEW-UPDATE} message such that \(m . view = v\) and the parent block \(b'\) of \(m . block\) is conflicting with \(b^v\) or \(\text{rank}(b') > \text{rank}(b^v)\). Let \(P = \{p_i \mid p_i \in C\} \) (the set of correct replicas), \(\text{VV}(m, \text{lockState})\) returns true in view \(v\) by \(p_i\).

For any one correct replica \(p_i\) who has locked \(b^v\), let \(b_i\) denote the locked block of \(p_i\) when \(p_i\) received \(m\). Since a correct replica only changes its \(QC_x\) to a \(QC_x\) with the same or a higher rank, \(\text{rank}(b_i) \geq \text{rank}(b^v)\). \(\text{VV}(m, \text{lockState})\) returns true by \(p_i\) if \(m . justify\) contains \(b^v . QC_x\), \(b_i . view < v\) and \(\text{rank}(b') \geq \text{rank}(b_i)\) (lines 50-51 in Table 2).

If the condition is satisfied, then we know that \(\text{rank}(b') \geq \text{rank}(b_i) \geq \text{rank}(b^v)\) and \(b^v . QC_x\) is formed in view \(v - 1\). According to Lemma 10, \(b'\) must be equal to \(b^v\) or an extension of \(b^v\), contradicting to the assumption.

Therefore, for all the correct replicas who have locked \(b^v\), \(\text{VV}(m, \text{lockState})\) return false. Since \(T_{y+1} \geq n - T_1 + f + 1\) and at least \(T_{y+1} - f \geq n - T_1 + 1\) correct replicas have locked \(b^v\), we know that \(|P| < T_1 - f\) and VVL-safety holds in view \(v\).

**Lemma 33.** If \(BG[x, y, z]\) satisfies DP3, \(T_j > f\) for \(j \in [1..z]\), \(T_1 \geq \left\lceil\frac{n+f+1}{2}\right\rceil\), and \(T_{y+1} \geq n - T_1 + f + 1\), then VVL-safety holds in \(BG[x, y, z]\).
Proof. For any view \( v \), let \( B^v = \{ b \mid \text{block } b \text{ has been committed before view } v \} \). According to Lemma 1, we can find \( b^v \in B^v \) such that for all \( b' \in B^v \) and \( b' \neq b^v \), we have \( \text{rank}(b^v) > \text{rank}(b') \).

We prove that VVL-safety holds for \( BG[x, y, z] \) by proving that VVL-safety holds in every view. For a specific view \( v' \), let \( w \) denote \( b^{v'}\cdot \text{view} \). We need to prove that VVL-safety holds in view \( v' \). Our proof consists of the following steps. First, we prove that VVL-safety holds in view \( w + 1 \). Then we prove that if VVL-safety holds in view \( w + 1, \cdots, w + k \) (for any integer \( 1 \leq k \leq v' - w - 1 \)), VVL-safety also holds in view \( w + k + 1 \). Then for \( k = v' - w - 1 \), we know that VVL-safety holds in view \( v' \).

According to Lemma 32, we know that VVL-safety holds in view \( w + 1 \). Then, assume that VV-safety holds in view \( w + 1, \cdots, w + k \) (for any integer \( 1 \leq k \leq v' - w - 1 \)). We need to show that VV-safety also holds in view \( w + k + 1 \). As there exists \( QC_x \) for \( b^{v'} \) in view \( w \) and \( z > y \), at least \( T_{y+1} - f \geq n - T_1 + f + 1 \) correct replicas have locked \( b^{v'} \) and sent a \( \text{vote}-(y+1) \) message for \( b^{v'} \) to form \( b^{v'}\cdot QC_{y+1} \) in view \( w \). Let \( m \) denote a \( \text{view-update} \) message such that \( m.\text{view} = w + k + 1 \) and the parent block \( b' \) of \( m.\text{block} \) is conflicting with \( b^{v'} \) or \( \text{rank}(b') < \text{rank}(b^{v'}) \). Let \( P = \{ p_i \mid p_i \in C \text{ (the set of correct replicas)} \}, \text{VV}(m, lockState) \) outputs 1 in view \( w + k + 1 \) by \( p_i \).

For any one correct replica \( p_i \) who has locked \( b^{v'} \), let \( b_i \) denote the locked block of \( p_i \) when \( p_i \) received \( m \). Since a correct replica only change its \( QC_x \) to a \( QC_x \) with the same or a higher rank, \( \text{rank}(b_i) \geq \text{rank}(b^{v'}) \). \text{VV}(m, lockState) outputs 1 by \( p_i \) if \( m.\text{justify} \) contains \( b', QC_x, b'.\text{view} < w + k + 1 \) and \( \text{rank}(b') \geq \text{rank}(b_i) \) (lines 50–51 in Table 2).

If the condition is satisfied, then we know that \( \text{rank}(b') \geq \text{rank}(b_i) \geq \text{rank}(b^{v'}) \). According to Lemma 7, Lemma 10, Lemma 12 and the inductive hypothesis, \( b' \) must be equal to \( b^{v'} \) or an extension of \( b^{v'} \), contradicting to the assumption that either \( b' \) is conflicting with \( b^{v'} \) or \( \text{rank}(b') < \text{rank}(b^{v'}) \). Therefore, for all the correct replicas who have locked \( b^{v'} \), \( \text{VV}(m, lockState) \) return false. Since \( T_{y+1} \geq n - T_1 + 1 \) and at least \( T_{y+1} - f \geq n - T_1 + 1 \) correct replicas have locked \( b^{v'} \), we know that \( |P| < T_1 - f \) and VVL-safety holds in view \( w + k + 1 \).

Then for \( k = v' - w - 1 \), we know that VVL-safety holds in view \( v' \).

**Lemma 34.** If \( BG[x, y, z] \) satisfies DP3, \( f < T_j \leq n - f \) for \( j \in [1..z] \), and \( T_1 \geq \left\lceil \frac{n + f + 1}{2} \right\rceil \), then VVL-liveness holds in \( BG[x, y, z] \).

**Proof.** For any view \( v \), let \( B^v_{\text{qc}} = \{ b \mid \text{block } b \text{ has been locked before view } v \} \). According to Lemma 11, we can find one block \( b^v \in B^v_{\text{qc}} \) such that for all \( b' \in B^v_{\text{qc}} \) and \( b' \neq b^v \), we have \( \text{rank}(b^v) > \text{rank}(b') \). Given a \( \text{view-update} \) message \( m \) in view \( v \), let \( b \) denote the parent block of \( m.\text{block} \). If \( (b, m.\text{justify}) \) is the output of \( \text{fsb}(M_v) \) on a snapshot \( M_v \), there are two cases to consider:

1) \( b^v, QC_x \in M_v, QC_x \). In this case, \( b \) is a block such that \( \text{rank}(b) \geq \text{rank}(b^v) \). Therefore, we know that \( \text{VV}(m, lockState) \) outputs 1 by all correct replicas since conditions (lines 50–51 in Table 2) are satisfied for all of them.

2) \( b^v, QC_x \notin M_v, QC_x \). Note that any correct replica changes its \( vb \) and \( QC_x \) only in the case that it has received a \( QC_x \) such that \( \text{rank}(\text{qcBlock}(qc)) \geq \)
Lemma 6. If \( T - (n - T_1 + f) > 0 \) and \( T - (n - T_{x+1} + f) > 0 \), then \( \text{BG}[x, y, z] \) satisfies DP5.

Proof. In a \( \text{BG}[x, y, z] \), for any block \( b \), we know that if \( b \) has been locked by any correct replica \( p_i \) in view \( v \), then \( b.\text{view} = v \) and \( p_i \) has also set its \( QC_y \) to \( b.QC_y \). According to \( x \leq y \) and Lemma 7, \( b.QC_1 \) is also formed by the leader in view \( v \). Accordingly, at least \( T_1 - f \) correct replicas have sent \( \text{VOTE}-1 \) messages for \( b \) such that \( b.QC_1 \) is formed. The \( T_1 - f \) replicas set their \( vb \) to \( b \) in view \( v \). Thus, fewer than \( n - T_1 \) correct replicas have not yet set their \( vb \) to \( b \) in view \( v \). Therefore, for any \( M^b \), at most \( n - T_1 + f \) messages are sent by replicas who have not set their \( vb \) to \( b \), i.e., there are at least \( T - (n - T_1 + f) \) \( b \) in \( M^b \). Hence, since \( T - (n - T_1 + f) > 0 \), we have that \( \text{VOTES}(b, T, y) \geq 1 \).

Besides, for any block \( d \), if \( d.QC_z \) is received by a correct replica \( p_i \) and \( p_i \) set its \( QC_z \) to \( d.QC_z \) in view \( v \), then \( d.\text{view} = v \). According to Lemma 7 and \( x \leq y < z \), \( d.QC_{x+1} \) is also formed by the leader in view \( v \). Accordingly, at least \( T_{x+1} - f \) correct replicas have received \( d.QC_x \) and sent \( \text{VOTE}-x+1 \) messages for \( d \) such that \( d.QC_{x+1} \) is formed. The \( T_{x+1} - f \) correct replicas set their \( QC_y \)'s to \( d.QC_z \) in view \( v \). Thus, fewer than \( n - T_{x+1} \) correct replicas have not yet set their \( QC_y \)'s to \( d.QC_x \) in view \( v \). Therefore, for any \( M^b \), at most \( n - T_{x+1} + f \) messages are sent by replicas who have not set their \( QC_y \)'s to \( d.QC_x \), i.e., there at least \( T - (n - T_{x+1} + f) \) \( b.QC_x \) in \( M^b \). Since \( T - (n - T_{x+1} + f) > 0 \), we have that \( \text{CERTS}(d, T, x, z) \geq 1 \).

Therefore, \( \text{BG}[x, y, z] \) satisfies DP1.

Lemma 35. If \( \text{BG}[x, y, z] \) satisfies DP5, \( T_j > f \) for \( j \in [1..z] \), \( T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil \), and \( f < T \leq n - f \), then FSBL-liveness holds for \( \text{BG}[x, y, z] \).

Proof. The proof resembles the proof of Lemma 18. In any view \( v \), the leader can obtain two intermediate variables, block \( b_1 \) and block \( b_2 \) based on a \( M \) and output \((b, \pi)\) after comparing ranks of \( b_1 \) and \( b_2 \).
Lemma 36. If $BG[x, y, z]$ satisfies $DP5$, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $T_{y+1} \geq n - T_1 + f + 1$, and at least one block has been committed by correct replica in view $v-1$, then VVL-safety holds in view $v$ for $BG[x, y, z]$.

Proof. Let $B^v = \{ b \mid \text{block } b \text{ has been committed before view } v \}$. According to Lemma 1, we let $b^v$ denote a block in $B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$, we have $\text{rank}(b^v) > \text{rank}(b')$. Since there exists a committed block in view $v-1$, we know $b^{v-1}.view = v-1$.

As there exists $QC_z$ formed for $b^v$ in view $v-1$ and $y < z$, at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^v$ and sent $\text{vote-}(y+1)$ for $b^v$ to form $b^v.QC_{y+1}$ in view $v-1$. Let $m$ denote a $\text{view-update}$ message such that $m.view = v$ and the parent block $b'$ of $m.block$ is conflicting with $b^v$ or $\text{rank}(b') < \text{rank}(b^v)$. Let $P = \{ p_i \mid p_i \in C \text{ (the set of correct replicas)} \}$, $\text{VV}(m, lockState)$ outputs 1 in view $v$ by $p_i$.

For any one correct replica $p_i$ who has locked $b^v$, let $b_i$ denote the locked block of $p_i$ when $p_i$ received $m$. Since a correct replica only changes its $QC_x$ to a $QC_z$ with the same or a higher rank, $\text{rank}(b_i) \geq \text{rank}(b^v)$. $\text{VV}(m, lockState)$ outputs 1 by $p_i$ if one of the following two conditions is satisfied:

1) $m.justify$ contains $b', QC_z, b'.view < v$ and $\text{rank}(b') > \text{rank}(b_i)$ (lines 62-63 in Table 2).

2) $m.justify$ contains $M_v, QC_z, b'.view < v$ and $b'.QC_z$ is the $QC_x$ with the highest rank in $M_v.QC_x$ (lines 64-67 in Table 2).

If condition 1) is satisfied, then $b'$ must be equal to or an extension of $b^v$ according to Lemma 10, contradicting the assumption that either $b'$ is conflicting with $b^v$ or $\text{rank}(b') < \text{rank}(b^v)$. If the condition 2) is satisfied, then we know that $b'$ is the block with the highest rank for which a $QC_z$ is included in $M_v.QC_x$. Since $BG[x, y, z]$ satisfies $DP5$, we have that $\text{num}(b^v, QC_z, M^{b^v}.QC_z) \geq 1$. Note that any correct replica changes its $QC_x$ with a $QC_z$ with the same or higher rank. We have $\text{rank}(b') \geq \text{rank}(b^v)$. According to Lemma 10, $b'$ must be equal to $b^v$ or an extension of $b^v$, contradicting the assumption that either $b'$ is conflicting with $b^v$ or $\text{rank}(b') < \text{rank}(b^v)$.

Therefore, for all the correct replicas who have locked $b^v$, $\text{VV}(m, lockState)$ return false. Since $T_{y+1} \geq n - T_1 + f + 1$ and at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^v$, we know that $|P| < T_1 - f$ and VVL-safety holds in view $v$.

Lemma 37. If $BG[x, y, z]$ satisfies $DP5$, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, and $T_{y+1} \geq n - T_1 + f + 1$, then VVL-safety holds in $BG[x, y, z]$.

Proof. For any view $v$, let $B^v = \{ b \mid \text{block } b \text{ has been committed before view } v \}$. According to Lemma 1, we can find $b^{v'} \in B^v$ such that for all $b' \in B^v$ and $b' \neq b^{v'}$, we have $\text{rank}(b^{v'}) > \text{rank}(b')$.

We prove that VVL-safety holds for $BG[x, y, z]$ by proving that VVL-safety holds in every view. For a specific view $v'$, let $w$ denote $b^{v'}.view$. We need to prove that VVL-safety holds in view $v'$. The proof consists of the following steps. First, we prove that VVL-safety holds in view $w+1$. Then we prove that if VVL-safety
holds in view \( w+1, \ldots, w+k \) (for any integer \( 1 \leq k \leq v' - w - 1 \)), VVL-safety also holds in view \( w+k+1 \). Then for \( k = v' - w - 1 \), we know that VVL-safety holds in view \( v' \).

According to Lemma 36, we know that VVL-safety holds in view \( w+1 \). Then, assume that VV-safety holds in view \( w+1, \ldots, w+k \) (for any integer \( 1 \leq k \leq v' - w - 1 \)). We need to show that VV-safety also holds in view \( w+k+1 \). As there exists \( QC_z \) formed for \( b'' \) in view \( w \) and \( y < z \), at least \( T_{y+1} - f \geq n - T_1 + 1 \) correct replicas have locked \( b'' \) and sent \( vote-(y+1) \) for \( b'' \) to form \( b'''.QC_{y+1} \) in view \( w \). Let \( m \) denote a \texttt{VIEW-UPDATE} message such that \( m.view = w+k+1 \) and the parent block \( b' \) of \texttt{m.block} is conflicting with \( b'' \) or \( rank(b') < rank(b'') \). Let \( P = \{ p_i \mid p_i \in C \) (the set of correct replicas), \( \texttt{vv}(m, lockState) \) outputs 1 by \( p_i \) if one of the following four conditions is satisfied:

1) \( m.justify \) contains \( b'.QC_x, b'.view < w+k+1 \) and \( rank(b') \geq rank(b_i) \) (lines 62-63 in Table 2).
2) \( m.justify \) contains \( M_{w+k+1}.QC_x, b'.view < w+k+1 \) and \( b'.QC_x \) is the \( QC_x \) with the highest rank in \( M_{w+k+1}.QC_x \) (lines 64-67 in Table 2).

If condition 1) is satisfied, then \( b' \) must be equal to or is an extension of \( b'' \) according to Lemma 7, Lemma 10 and the inductive hypothesis. This is a contradiction with the assumption that either \( b' \) is conflicting with \( b'' \) or \( rank(b') < rank(b'') \). If the condition 2) is satisfied, then we know that \( b' \) is the block with the highest rank for which a \( QC_x \) is included in \( M_{w+k+1}.QC_x \). Since \( BG[x, y, z] \) satisfies \texttt{DP5}, we have that \( num(b'''.QC_x, M^{b''}.QC_x) \geq 1 \). Note that any correct replica changes its \( QC_x \) only with \( QC_x \) with the same or a higher rank, \( rank(b') \geq rank(b'') \). According to Lemma 7, Lemma 10 and the inductive hypothesis, \( b' \) must be equal to \( b'' \) or an extension of \( b'' \), contradicting the assumption that either \( b' \) is conflicting with \( b'' \) or \( rank(b') < rank(b'') \).

Then for \( k = v' - w - 1 \), we know that VVL-safety holds in view \( v' \). That completes the proof.

**Lemma 38.** If \( BG[x, y, z] \) satisfies \texttt{DP5}, \( T_j > f \) for \( j \in [1..z] \), \( T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil \), \( x \leq y < z \) and \( f < T \leq n - f \), then VVL-liveness holds in \( BG[x, y, z] \).

**Proof.** For any view \( v \), let \( B_i^v = \{ b \mid \text{block } b \text{ has been locked before view } v \} \). According to \( x \leq y \) and Lemma 10, we can find \( b_i^v \in B_i^v \) such that for all \( b' \in B_i^v \) and \( b' \neq b_i^v \), we have \( rank(b_i^v) \geq rank(b') \). Given a \texttt{VIEW-UPDATE} message \( m \) in view \( v \), let \( b \) denote the parent block of \texttt{m.block}. If \( (b, m.justify) \) is the output of \texttt{FSB}(\( M_\nu \)) on some \( M_\nu \), there are two cases to consider:

1) \( b_i^v, QC_x \in M_\nu, QC_x \). In this case, the output of \texttt{FSB}(\( M_\nu \)) is \((b, b, QC_x)\) or \((b, M_\nu, QC_x)\). For both situations, \( b \) is a block such that \( rank(b) \geq rank(b_i^v) \).

Hence, \texttt{vv}(\( m, lockState \)) returns true by all correct replicas since condition 1) (lines 62-63 in Table 2) is satisfied.
2) $b^v_l \notin M_v.QC_x$. We need to consider two sub-cases. If $\text{rank}(b) \geq \text{rank}(b^v_l)$, then we know that $\text{vv}(m, lockState)$ returns true by all correct replicas since condition 1) (lines 77-78 in Table 2) is satisfied. If $\text{rank}(b) < \text{rank}(b^v_l)$, we need to prove that the output of $\text{fsb}(M_v)$ is of the form $(b, b.QC_x, M_v.QC_x)$. Since $\text{BG}[x, y, z]$ satisfies $\text{DP5}$, then we have $b^v_l \in M^v_l.vb$. Recall again any correct replica changes its $vb$ and $QC_x$ only in the case that it has received a $QC_x$ such that $\text{rank}(\text{qcBlock}(qc)) \geq \text{rank}(vb)$. Therefore, for any replica that sends a message $m$ such that $m \in M_v$, it will not change its $vb$ or updates $QC_x$, as the replica previously sets $vb$ to $b^v_l$ and we know $b^v_l \in M^v_l.vb$. Then the output of $\text{fsb}(M_v)$ is $(b, M_v.QC_x)$ and $\text{vv}(m, lockState)$ returns true by all correct replicas since condition 2) (lines 64-67 in Table 2) is satisfied for them.

That completes the proof.

**Theorem 12** $\text{BG}[x, y, z]$ (in Table 2) achieves safety and responsiveness if the following are satisfied: 1) $\text{BG}[x, y, z]$ satisfies $\text{DP5}$; 2) $f < T \leq n - f$; 3) $\left\lfloor \frac{n+f+1}{2} \right\rfloor \leq T_1 \leq n - f$; 4) $f < T_j \leq n - f$ for $j \in [1..z]$; and 5) $x \leq y < z$, $n - T_1 + f + 1 \leq T_{y+1}$.

**Proof.** Correctness follows from Theorem 4, Theorem 5, Theorem 6, Lemma 35, Lemma 37 and Lemma 38.