BG: A Modular Treatment of BFT Consensus

Towards a Unified Theory of BFT Replication

Xiao Sui[1], Sisi Duan[2], and Haibin Zhang[3]

[1] Shandong University <suixiao@mail.sdu.edu.cn>
[2] Tsinghua University <duansisi@tsinghua.edu.cn>
[3] Beijing Insitute of Technology <haibin@bit.edu.cn>

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

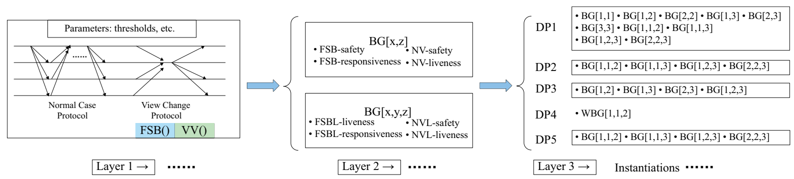

Fig. 1: The three layers of BG framework.

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 FSB() and VV() 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: BG$[x, z]$ and BG$[x, y, z]$. BG$[x, z]$ models protocols *without lock state*, where the view change rules rely purely on the information collected in the view change messages. 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 block *locked* in the y-th phase of the normal case (for some $y < z$).

	protocol	predicates	replicas	message pattern	steps	authenticator complexity		message complexity	
						normal-case	view change	normal-case	view change
1-phase	FaB5[23]	—	$5f+1$	AtoA	2	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	BG[1, 1]	`DP1`	$5f+1$	1toA	3	$\mathcal{O}(n)$	$\mathcal{O}(n)/\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2-phase	PBFT[12]	—	$3f+1$	AtoA	3	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	Fast-Hotstuff[20]	—	$3f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2]	`DP3`	$3f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 2]	`DP1`	$5f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 2]	`DP2`	$4f+1$	1toA	5/7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 2]	`DP5`	$3f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n)/\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	$BG[1,1,2]^*$	`DP5`	$3f+1$	1toA	5/9	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
3-phase	Hotstuff[31]	—	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2, 3]	`DP3`	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

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.

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 fined-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.

2 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 σ_i. A combining algorithm *tcombine* takes as input pk, a message m, and a set of t valid partial signatures, and outputs a signature σ. A signature verification algorithm *tverify* takes as input pk, a message m, and a signature σ, and outputs a bit. We also use a collision-resistant hash function $hash$ mapping a message of arbitrary length to a fixed-length output.

3 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

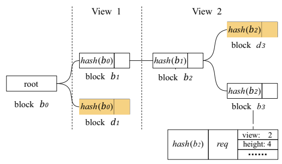

Fig. 2: Example of tree of blocks (nodes).

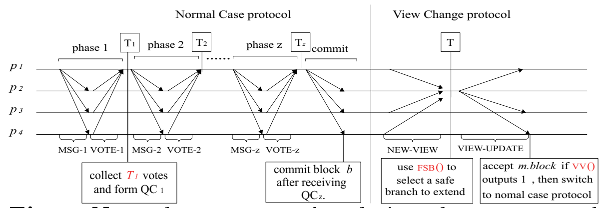

Fig. 3: Normal-case protocol and view change protocol of BG framework (Layer 1).

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 *height* and *view*: *height* for b is the number of blocks on the branch led by b, while *view* for b is the view during which b is added to the tree. The *view* for b is equal to or greater than that of the parent block of b. Two branches are *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 *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 $\langle \text{TYPE}, view, height, block, justify \rangle$ to denote the messages sent among replicas. Some fields can be set as $\perp$. 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. $m.height$ is set as the height of $m.block$. $m.justify$ 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 $m.\text{TYPE}=$ NEW-VIEW, $m.block$ and $m.height$ are set as $\perp$. When $m.\text{TYPE}=$VIEW-UPDATE, $m.block$ is an associated block and $m.height$ is the height of the block. In both cases, $m.justify$ 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 *tsign* algorithm to generate partial signatures for a message. One can run the *tcombine* 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 QCVOTE(m) denote the output of the *tsign* algorithm for m. Otherwise, we implicitly use a signature or a threshold signature for authentication. To hide the implementation detail, let QCCREATE(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 $m.block = b$. $b.QC_j$ 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 QCBLOCK(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, *rank*(b) depends on $b.view$ and $b.height$. 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 *cview*, the highest quorum certificates (received in different phases) $QC_1, \cdots, QC_z$, and *last voted block vb*. In protocols with

Algorithm 1: Normal-case protocol for p_i

localstate: the current view $cview$, the last voted block vb, the locked block lb (in BG$[x, y, z]$), certificates QC_1, QC_2, $\cdots$, and QC_z.
parameters: thresholds for different voting phases T_1, T_2, $\cdots$, and T_z.
As a leader: //$\text{LEADER}(cview) = p_i$
- propose a block b extending $\text{QCBLOCK}(QC_x)$, broadcast $\langle$MSG-1$\rangle$ message m for b, where $m.justify$ is QC_x
• **upon** receiving T_j signed $\langle$VOTE-$j\rangle$ messages for block b: //$j \in [1, z)$
  - propose $b.QC_j$, update $QC_j \leftarrow b.QC_j$
  - broadcast a $\langle$MSG-$(j+1)\rangle$ message m for b, where $m.justify$ is QC_x
• **upon** receiving T_z signed $\langle$VOTE-z$\rangle$ messages for block b':
  - propose $b.QC_z$, update $QC_z \leftarrow b.QC_z$
  - broadcast a COMMIT message m for b, where $m.justify$ is QC_z, propose another block
As a replica:
• **upon** receiving a valid $\langle$MSG-1$\rangle$ message for a block b: //$rank(b) > rank(vb)$
  - update $vb \leftarrow b$, $QC_x \leftarrow m.justify$ //vote for a new block
  - send a $\langle$VOTE-1$\rangle$ message for b to $\text{LEADER}(cview)$
• **upon** receiving a valid $\langle$MSG-$j\rangle$ message m for b: //$j \in (1, z)$
  - update $QC_{j-1} \leftarrow m.justify$, send a $\langle$VOTE-$j\rangle$ message for b to $\text{LEADER}(cview)$
• **upon** receiving a valid $\langle$MSG-$(y+1)\rangle$ for b in BG$[x, y, z]$: update $lb \leftarrow b$
• **upon** receiving a valid COMMIT message m for b:
  - update $QC_z \leftarrow m.justify$, commit b //execute the requests in b in order
  - **wait for** a $\langle$MSG-1$\rangle$ message for another block

lock state, replica need 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$-j$ to all replicas and enters the next phase. Upon receiving a valid MSG-j 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 essage, each replica commits the corresponding block.

Algorithm 2: View change protocol for p_i

criticalstate: contains many variables in *localstate* specified in layer 3.
parameters: thresholds T_1, T_2, $\cdots$, and T_z, and the view change threshold T.
• **upon** *timeout*:
  - update $cview \leftarrow cview + 1$
  - send the *criticalState* in a NEW-VIEW message to $\textsc{Leader}(cview)$
As a new leader: //$\textsc{Leader}(cview) = p_i$
• **upon** receiving a set M of T signed NEW-VIEW messages for view $cview$:
  - $(b'.\pi) \leftarrow$ FSB(M), propose a block b extending b'
  - broadcast a VIEW-UPDATE message m for b, where $m.justify$ is π
  - $(flag = 1)$: : vote for b but does not update *localstate*
    wait until $b.QC_x$ is collected:
    - update $QC_x \leftarrow b.QC_x$, propose a block b_1 extending b
    - broadcast b_1 in a $\langle$MSG-1$\rangle$ message m, where $m.verify$ is $b.QC_x$
  - switch to normal case operation
As a replica:
• **upon** receiving a VIEW-UPDATE essage m from $\textsc{Leader}(cview)$ for a block b:
  - **if** VV$(m,\cdot)=0$ in BG$[x,z]$ or VV$(m,lb)=0$ in BG$[x,y,z]$, discard m
  - broadcast a $\langle$VOTE-1$\rangle$ message for b to $\textsc{Leader}(cview)$
  - $(flag = 1)$: : vote for b but does not update *localstate*
    wait until a $\langle$MSG-1$\rangle$ message m for a block b^* extending b is received:
    - update $vb \leftarrow b$, $QC_x \leftarrow m.justify$, send a $\langle$VOTE-1$\rangle$ message for b^* to $\textsc{Leader}(cview)$
  - switch to normal-case protocol

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 *cview* by one. Then p_i sends *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 FSB(M) to obtain (b', π), where b' is a block and π 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 $flag$ (specified in Sec. 6). If $flag = 0$, the leader directly switches to normal-case protocol. If $flag = 1$, the leader still switches to phase 2 but does not update its *lockState* until $b.QC_x$ is collected.

▷ A replica accepts a VIEW-UPDATE message m in view v from the new leader only if VV(m) outputs 1 in BG$[x,z]$ or VV(m, lb) outputs 1 in BG$[x,y,z]$. Then the replica sends a VOTE-1 message for $m.block$. Similar to the two cases for the leader, according to the $flag$ parameter, the replica may take different actions. If $flag = 0$, the replica switches to normal-case protocol. If $flag = 1$, 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 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—FSB(), and VV(). These properties are *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: BG$[x, z]$ and 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 FSB() and VV() for BG$[x, z]$ and 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.

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

Both BG$[x, z]$ and BG$[x, y, z]$ are z-phase protocols. BG$[x, z]$ models protocols *without lock state*, where the view change rules rely purely on the information provided by replicas during the view change. BG$[x, y, z]$ represents protocols *with lock state*, where the view change rules rely on the information collected during the view change and the information on the block *locked* in the y-th phase (for some $y < z$).

The parameter x applies to both BG$[x, z]$ and BG$[x, y, z]$. Let b be a block and m be a MSG-1 message for b. $m.justify$ is set as $b'.QC_x$, where b' is the parent block of b and QC_x is the QC formed in the x-th phase for b'.

The parameter y in BG$[x, y, z]$ represents the phase where a replica updates its state parameter *lockState* in the normal-case protocol. We say b a *locked block*, if at least one correct replica has updated its *lockState* to b. Intuitively, *lockState* is crucial for a replica in BG$[x, y, z]$ to decide whether to accept a VIEW-UPDATE message.

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

Lemma 1. *Let $B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}$. If $T_j > f$ for $j \in [1..z]$, and $T_1 \geq \lceil \frac{n+f+1}{2} \rceil$, then there exists $b^v \in B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$, we have $rank(b^v) > rank(b')$.*

FSB() and VV() for BG$[x, z]$. We now elaborate the properties of FSB() and VV() functions for 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 M_v of T NEW-VIEW messages (where we call the set M_v a *view change snapshot*), FSB() takes as input

M_v and outputs some (b, π). For $\textsc{fsb}()$, we consider two properties: FSB-safety and FSB-liveness. FSB-safety ensures that a correct leader selects the longest committed branch in the tree to extend. On the other hand, FSB-liveness is relatively straightforward. Here we emphasize that the threshold T cannot be more than $n-f$, as there are f faulty replicas. T could be less than $n-f$ for some protocols.

Given a VIEW-UPDATE message m, let b denote the parent block of $m.block$, and let v denote $m.view$. $\textsc{vv}()$ 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 $\textsc{vv}(m)$. 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 $\textsc{fsb}()$. Essentially, VV-safety ensures that a committed block by any correct replica will remain committed after the view change.

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

Both $\textsc{fsb}()$ properties and $\textsc{vv}()$ 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], $\textsc{vv}()$ 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.

$\textsc{fsb}()$ takes as input M_v and outputs some (b, π). 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 $m.block$ and v denote $m.view$. $\textsc{vv}()$ takes as an input m together with $lockState$ of a replica and outputs a binary value. we define both safety and liveness properties for $\textsc{vv}(m, lockState)$ in VVL-safety and VVL-liveness.

- **FSBL-liveness**: Let $T \leq n-f$. $\textsc{fsb}(M_v)$ outputs some (b, π).
- **VVL-safety**: Let $P = \{p_i | p_i \in C$ (the set of correct replicas), $\textsc{vv}(m, lockState)$ outputs 1 by p_i in view $v\}$. If b is conflicting with b^v or $b.height$ is lower than $b^v.height$, then $|P| < T_1 - f$.
- **VVL-liveness**: If $(b, m.justify)$ is the output of $\textsc{fsb}(M_v)$ function on some M_v, $\textsc{vv}(m, lockState)$ 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-liveness 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 $T_1 \geq \lceil \frac{n+f+1}{2} \rceil$ and $T_j > f$ for all $j \in [1..z]$.*

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

Theorem 3. *BG[x, z] achieves optimistic responsiveness, if $T_j \leq n - f$ 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 $T_1 \geq \lceil \frac{n+f+1}{2} \rceil$ and $T_j > f$ for all $j \in [1..z]$.*

Theorem 5. *BG[x, y, z] achieves safety-II, if $T_1 \geq \lceil \frac{n+f+1}{2} \rceil$, $T_j > f$ 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 FSBL-liveness and VVL-liveness hold.*

These theorems are find-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 $T_1 = \left\lceil \frac{n+f+1}{2} \right\rceil$ 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 FSB(), and VV() functions satisfying the properties we define in our Layer 2 framework.

We find that directly working on FSB() and 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 FSB() and 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 FSB() and VV() functions. We can show that realizations of FSB() when taking M^b as input—according to these dominant predicates—can satisfy the properties defined in layer 2.

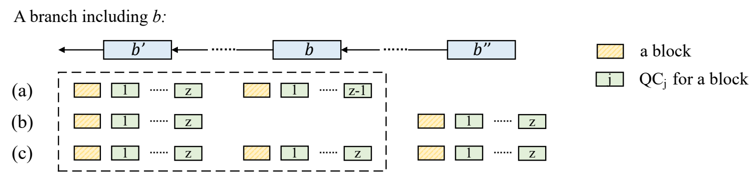

Fig. 4: Examples of virtual view change snapshots.

– We are finally able to prove that using our framework, as long as *all* committed blocks in $\mathrm{BG}[x, z]$ (resp., locked blocks in $\mathrm{BG}[x, y, z]$) 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 $b.view$. 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 $m \in M^b$ 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' and b'', but 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^v}, emphasizing the information for block b^v, focuses on whether the real state of b^v can be obtained. Therefore, M^{b^v} can be viewed as a special case of M_v but includes more information about b^v. Indeed, $criticalState$ in $m \in M^{b^v}$

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 $m \in M_v$ is sent at another moment, m may not directly include any evidence about the real state of b^v. Accordingly, a correct 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 FSB() and VV() accordingly.

6.2 Dominant Predicates and Realizations of FSB() and 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 FSB() and 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., FSB() and 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 $M^b.QC_x$ 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

pred	protocol	FSB()	VV()
DP1	BG[x,z] (vb,QC_x) $flag=1$	**func** FSB(M_v) 01 $b_1 \leftarrow null, b_2 \leftarrow null$ 02 **for** $b \in M_v.vb$ 03 **if** $num(b, M_v.vb) > T/2$ **then** $b_1 \leftarrow b$ 06-1 $\pi_1 \leftarrow M_v$ **return** (b_1, π_1) //BG[x, z] 06-2 $\pi_1 \leftarrow M_v.vb$ **return** (b_1,π_1) //BG[x, y, z] 04 **for** $b : b.QC_x \in M_v.QC_x$ 05 **if** $rank(b) > rank(b_2)$ **then** $b_2 \leftarrow b$ 07-1 $\pi_2 \leftarrow (b_2.QC_x, M_v)$ **return** (b_2, π_2) //BG[x, z] 07-2 $\pi_2 \leftarrow (b_2.QC_x)$ **return** (b_2, π_2) //BG[x, y, z]	**func** VV($\langle$VIEW-UPDATE, v, $b'.height$, b', $\pi\rangle$) 08 $b \leftarrow$ the parent block of b', $v_b \leftarrow b.view$ 09 **if** $M_v \in \pi$ **and** $(b, \pi) =$ FSB(M_v) **and** $v_b < v$ 10 **then return** 1 11 **return** 0
	BG[x,y,z] (vb,QC_x) $flag=1$		**func** VV($\langle$VIEW-UPDATE, v, $b'.height$, b', $\pi\rangle$, lb) 12 $b \leftarrow$ the parent block of b', $v_b \leftarrow b.view$ 13 **if** $M_v.vb \in \pi$ **and** $num(b, M_v.vb) > T/2$ 14 **if** $rank(lb) \leq rank(b)$ **and** $v_b < v$ **then** **return** 1 15 **if** $b.QC_x \in \pi$ **and** $v_b < v$ 16 **if** $rank(lb) \leq rank(b)$ **then return** 1 17 **return** 0
DP2	BG[x,y,z] (vb,QC_x) $flag=1$	**func** FSB(M_v) 18 $b_0, b_1, b_2, b_3 \leftarrow null$ 19 **for** $b \in M_v.vb$ 20 **if** $num(b, M_v.vb) \geq f + 1$ **and** $rank(b) > rank(b_1)$ 21 **then** $b_1 \leftarrow b, b_0 \leftarrow b_1$ 22 **else if** $num(b, M_v.vb) \geq f + 1$ **and** $rank(b) > rank(b_2)$ 23 **then** $b_2 \leftarrow b$ 24 **for** $b : b.QC_x \in M_v.QC_x$ 25 **if** $rank(b) > rank(b_3)$ **then** $b_3 \leftarrow b$ 26 **if** $rank(b_0) > rank(b_3)$ **then** 27 **if** $rank(b_1) = rank(b_2)$ **then** 28 $\pi \leftarrow (b_3.QC_x, M_v.QC_x)$ **return** (b_3,π) 29 **return** (b_0, π) s.t. $\pi = (M_v.vb)$ 30 **return** (b_3, π) s.t. $\pi = (b_3.QC_x)$	**func** VV($\langle$VIEW-UPDATE, v, $b'.height$, b', $\pi\rangle$, lb) 31 $b \leftarrow$ the parent block of b', $v_b \leftarrow b.view$ 32 **if** $b.QC_x \in \pi$ **and** $rank(b) \geq rank(lb)$ 33 **and** $v_b < v$ **then return** 1 34 **if** $M_v.vb \in \pi$ **and** $num(b, M_v.vb) \geq f+1$ 35 **if** $rank(b) > rank(lb)$ **and** $v_b < v$ **then return** 1 36 **if** $M_v.vb \in \pi$ **and** $num(b, M_v.vb) \geq f + 1$ 37 **if** $b = lb$ **then return** 1 38 **if** $M_v.QC_x \in \pi$ **and** $b.QC_x \in \pi$ **and** $v_b < v$ 39 **if** $num(b.QC_x, M_v.QC_x) > 2f + 1$ **then return** 1 40 **return** 0
DP3	BG[x, z] (QC_x) $flag=0$	**func** FSB(M_v) 41 $b_2 \leftarrow null$ 42 **for** $b : b.QC_x \in M_v.QC_x$ 43 **if** $rank(b) > rank(b_2)$ **then** $b_2 \leftarrow b$ 44-1 $\pi \leftarrow (b_2.QC_x, M_v)$ **return** (b_2,π) //BG[x, z] 44-2 $\pi \leftarrow (b_2.QC_x)$ **return** (b_2,π) //BG[x, y, z]	**func** VV($\langle$VIEW-UPDATE, v, $b'.height$, b', $\pi\rangle$) 45 $b \leftarrow$ the parent block of b', $v_b \leftarrow b.view$ 46 **if** $M_v \in \pi$ **and** $(b, \pi) =$ FSB(M_v) **and** $v_b < v$ 47 **then return** 1 48 **return** 0
	BG[x,y,z] (QC_x) $flag=0$		**func** VV($\langle$VIEW-UPDATE, v, $b'.height$, b', $\pi\rangle$, lb) 49 $b \leftarrow$ the parent block of b', $v_b \leftarrow b.view$ 50 **if** $b.QC_x \in \pi$ **and** $v_b < v$ 51 **if** $rank(b) \geq rank(lb)$ **then return** 1 52 **return** 0
DP5	BG[x,y,z] (vb,QC_x) $flag=0$	**func** FSB(M_v) 53 $b_1 \leftarrow null, b_2 \leftarrow null$ 54 **for** $b \in M_v.vb$ 55 **if** $rank(b) > rank(b_1)$ **then** $b_1 \leftarrow b$ 56 **for** $b : b.QC_x \in M_v.QC_x$ 57 **if** $rank(b) > rank(b_2)$ **then** $b_2 \leftarrow b$ 58 **if** $rank(b_1) > rank(b_2)$ **then** 59 $\pi \leftarrow (b_2.QC_x, M_v.QC_x)$ **return** (b_2,π) 60 $\pi \leftarrow (b_2.QC_x)$, **return** (b_2, π)	**func** VV($\langle$VIEW-UPDATE, v, $b'.height$, b', $\pi\rangle$, lb) 61 $b \leftarrow$ the parent block of b', $v_b \leftarrow b.view$ 62 **if** $b.QC_x \in \pi$ **and** $v_b < v$ 63 **if** $rank(b) \geq rank(lb)$ **then return** 1 64 **if** $b.QC_x \in \pi$ **and** $M_v.QC_x \in \pi$ **and** $v_b < v$ 65 **for** $d : d.QC_x \in M_v.QC_x$ 66 **if** $rank(d) > rank(b)$ **then return** 0 67 **return** 1 68 **return** 0

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.

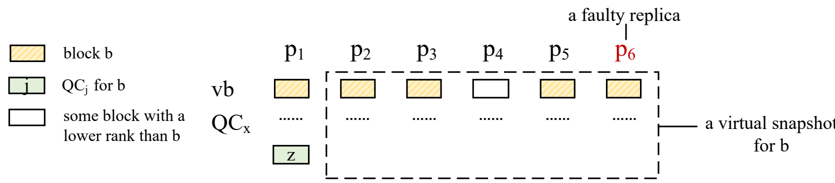

Fig. 5: Examples of `DP1`

correct FSB(M^b) function should output block b if the number of b in $M^b.vb$ is larger than $T/2$, i.e., at least $T/2$ replicas have voted for b.

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

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

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

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

For `DP1`, we obtain the following theorems.

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

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) $2f < T \leq n - f$; 3) $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n - f$; and 4) $f < T_j \leq n - f$ for $j \in [1..z]$.*

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* `DP1`*; 2) $2f < T \leq n - f$; 3) $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n - f$; 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 of these theorems in Appendix F and Appendix G.

Dominant predicate `DP2`. `DP1` has restricted on the number of replicas that vote for certain block, i.e., 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 `DP2`. For BG$[x, y, z]$, it satisfies

DP2 iff $\textsc{Votes}(b, T, y) \geq f + 1$ and $\textsc{Certs}(b, T, x, z) \geq T - (2f + 1)$. 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. FSB() takes as input M_v and outputs (b, π), where b is a block and π 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 FSB().

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

VV() takes as input a VIEW-UPDATE message m and $lockState$ (i.e., lb). Let b' denote the parent block of $m.block$, then VV() outputs 1 if one of the following four conditions is satisfied: 1) $b'.QC_x \in \pi$ and $\mathit{rank}(b') \geq \mathit{rank}(lb)$ (lines 32-33); 2) $M_v.vb \in \pi$ and $\mathit{num}(b', M_v.vb) \geq f + 1$ and $\mathit{rank}(b') > \mathit{rank}(lb)$ (lines 34-35); 3) $M_v.vb \in \pi$ and $\mathit{num}(b', M_v.vb) \geq f + 1$ and $b' = lb$ (lines 36-37); 4) $M_v.QC_x \in \pi$ and $b'.QC_x \in \pi$ and $\mathit{num}(b'.QC_x, M_v.QC_x) > 2f + 1$ (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) $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n - f$; 4) $f < T_j \leq n - f$ for $j \in [1..z]$; and 5) $x \leq y < z$, $n - T_1 + f + 1 \leq T_{y+1}$.*

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$[x, z]$ satisfies DP3 iff $\textsc{Certs}(b, T, x, z) \geq 1$; BG$[x, y, z]$ satisfies DP2 iff $\textsc{Certs}(b, T, x, y) \geq 1$.

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 $b_2.QC_x$ is included in $M_v.QC_x$. $\textsc{fsb}(M_v)$ outputs $(b_2, (b_2.QC_x, M_v))$ (line 44-1) 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 π denote $m.justify$. Then $\textsc{vv}(m)$ for BG$[x, z]$ returns 1 in view v if $m.justify$ is M_v, $(b, \pi) = \textsc{fsb}(M_v)$, and $b.view$ is lower than v (see lines 45-48). For BG$[x, y, z]$, $\textsc{vv}()$ has an additional input lb. $\textsc{vv}(m, lb)$ outputs 1 if $b.QC_x \in \pi$, $b.view < v$ and $rank(b) \geq rank(lb)$ (see lines 49-52). Otherwise, $\textsc{vv}()$ 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) $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n - f$; 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) $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n - f$; 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 $\textsc{Certs}(b, T, x, z) \geq 1$ and $\textsc{Votes}(b, T, y) \geq 1$. In our concrete constructions of $\textsc{fsb}()$ and $\textsc{vv}()$, 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, ln 53-68. $\textsc{fsb}()$ takes as input M_v and outputs some (b, π). From M_v,

protocols	pred	replicas	thresholds	features
BG[1, 1]	DP1	$5f{+}1$	$T=T_1=4f{+}1$	* an improvement of FaB5 * the most efficient 1-phase protocol
BG[1, 2]	DP3	$3f{+}1$	$T=T_1=2f{+}1$	* almost identical with Fast-Hotstuff * the most efficient 2-phase protocol without lock state
BG[1, 1, 2]	DP1	$5f{+}1$	$T=T_1=T_2$ $=4f{+}1$	* the first 2-phase protocol which achieves $O(n)$ message complexity and $O(n)$ authenticator complexity for both normal-case and view change
BG[1, 1, 2]	DP2	$4f{+}1$	$T=T_1=T_2$ $=3f{+}1$	* the first $4f{+}1$ BFT protocol to the best of our knowledge * $O(n)$ authenticator complexity for normal case and view change
BG[1, 1, 2]	DP5	$3f{+}1$	$T=T_1=T_2$ $=2f{+}1$	* $O(n)$ authenticator complexity in the normal case and view change in the fast path
$BG[1,1,2]^*$	DP5	$3f{+}1$	$T=T_1=T_2$ $=2f{+}1$	* the protocol can be further optimized to cover Marlin [30] * a variant of BG[1, 1, 2] that achieves $O(n)$ authenticator complexity at the cost of two more phases for view change
BG[1, 2, 3]	DP3	$3f{+}1$	$T=T_1=T_2$ $=T_3=2f{+}1$	* almost identical with HotStuff * with minor differences in data structure and information carried in the NEW-VIEW message

Table 3: Representative BG BFT protocols generated using our framework for $z \leq 3$.

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 $M_v.vb$. 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 FSB() outputs $(b_2, b_2.QC_x, M_v.QC_x)$, if $rank(b_1) > rank(b_2)$. Otherwise, FSB() outputs $(b_2, b_2.QC_x)$.

Let m denote a VIEW-UPDATE message for view v, b denote the parent block of $m.block$, and π denote $m.justify$. VV(m, lb) outputs 1 if one of the following conditions is satisfied: 1) $b.QC_x \in \pi$ and $rank(b) \geq rank(lb)$ (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 BG$[x, y, z]$ satisfies DP5.*

Theorem 12. *BG$[x, y, z]$ (in Table 2) achieves safety and responsiveness if the following are satisfied: 1) BG$[x, y, z]$ satisfies DP5; 2) $f < T \leq n - f$; 3) $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n - f$; 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 proof for the above theorems in Appendix K.

A DP5 variant. According to the construction of FSB() function for DP5, a new leader may need to include M_v in its VIEW-UPDATE essage. 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 BG$[x, z]$ or a BG$[x, y, z]$ protocol. For each BG$[x, z]$ or 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.

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.

predicates	candidates	conditions to satisfy predicates*	other conditions (from layer 2): x, y-dependent conditions	other conditions (from layer 2): other general conditions	Thm
DP1 $\textsc{Votes}(b, T, z) > T/2$	BG[1, 1]	$T-(n-T_1+f) > T/2$	—	$2f < T \leq n-f$ $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n-f$ $f < T_j \leq n-f$ for $j \in [1..z]$	7
	BG[1, 1, 2]		$n-T_1+f+1 \leq T_2 \leq n-f$		8
DP2 $\textsc{Votes}(b, T, y) \geq f+1$ and $\textsc{Certs}(b, T, x, z) \geq T-(2f+1)$	BG[1, 1, 2]	$T-(n-T_1+f) \geq f+1$ $T-(n-T_2+f) \geq T-(2f+1)$	$n-T_1+f+1 \leq T_2 \leq n-f$	$f < T \leq n-f$ $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n-f$ $f < T_j \leq n-f$ for $j \in [1..z]$	9
DP3 $\textsc{Certs}(b, T, x, z) > 0$ for BG[x, z] $\textsc{Certs}(b, T, x, y) > 0$ for BG[x, y, z]	BG[1, 2]	$T-(n-T_2+f) > 0$	—	$f < T \leq n-f$ $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n-f$ $f < T_j \leq n-f$ for $j \in [1..z]$	10
	BG[1, 2, 3]	$T-(n-T_2+f) > 0$	$n-T_1+f+1 \leq T_3 \leq n-f$		11
DP5 $\textsc{Certs}(b, T, x, z) > 0$ and $\textsc{Votes}(b, T, y) > 0$	BG[1, 1, 2]	$T-(n-T_1+f) > 0$ $T-(n-T_2+f) > 0$	$n-T_1+f+1 \leq T_2 \leq n-f$	$f < T \leq n-f$ $\lceil \frac{n+f+1}{2} \rceil \leq T_1 \leq n-f$ $f < T_j \leq n-f$ for $j \in [1..z]$	12

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 BG[$1, 2, 3$], a protocol with *lockState* based on `DP3`. To generate 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:

$$\begin{cases} T - (n - T_2 + f) > 0 \\ n - T_1 + f + 1 \leq T_3 \leq n - f \\ f < T < n - f \\ \left\lceil \dfrac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f \\ f < T_1 \leq n - f \\ f < T_2 \leq n - f \\ f < T_3 \leq n - f \end{cases} \tag{1}$$

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 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:

$$\begin{cases} T-(n-T_2+f)>0 \\ f<T<n-f \\ \left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f \\ f<T_1\le n-f \\ f<T_2 \le n-f \end{cases} \tag{2}$$

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

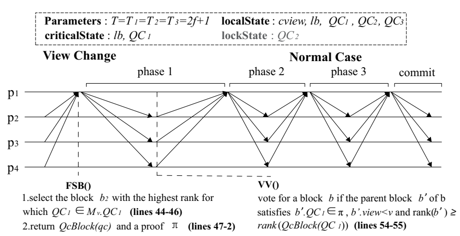

Fig. 6: BG[1,2,3] with DP3.

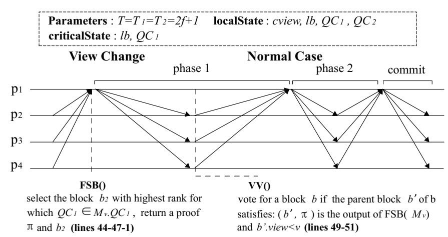

Fig. 7: BG[1,2] with DP3.

BG$[1,1,2]$ 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):

$$\begin{cases} T-(n-T_1+f)>T/2 \\ n-T_1+f+1 \le T_2 \le n-f \\ f<T<n-f \\ \left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f \\ f<T_1\le n-f \\ f<T_2 \le n-f \end{cases} \tag{3}$$

The above system of inequalities has solution only if $n>5f$. We thus set n as $5f+1$, and set T_1, T_2 and T as $4f+1$. According to Algorithm 3, Algorithm 4, and the realization of the FSB() and VV() in Table 2, we obtain 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):

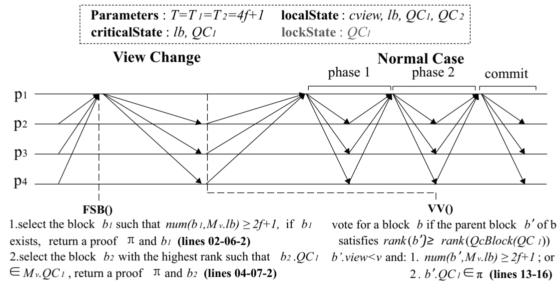

Fig. 8: BG[1,1,2] with DP1

$$
\begin{cases}
T - (n - T_1 + f) \geq f + 1 \\
T - (n - T_2 + f) > 0 \\
n - T_1 + f + 1 \leq T_2 \leq n - f \\
f < T < n - f \\
\left\lceil \dfrac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f \\
f < T_1 \leq n - f \\
f < T_2 \leq n - f
\end{cases}
\tag{4}
$$

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.

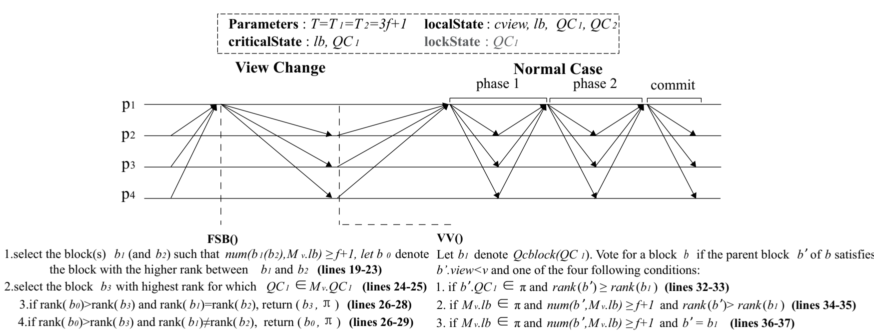

Fig. 9: BG[1,1,2] with DP2

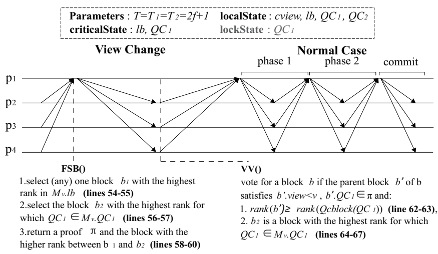

Fig. 10: BG[1,1,2] with DP5

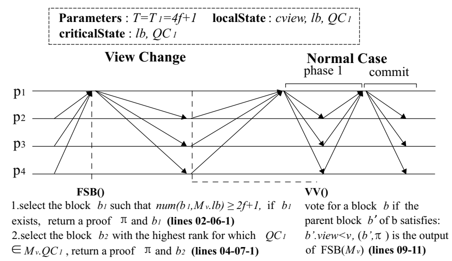

Fig. 11: BG[1,1] with `DP1`

BG$[1,1,2]$ with `DP5`. According to Table 4, we obtain the following system of inequalities (5):

$$\begin{cases} T-(n-T_1+f)>0 \\ T-(n-T_2+f)>0 \\ n-T_1+f+1 \leq T_2 \leq n-f \\ f<T<n-f \\ \left\lceil \dfrac{n+f+1}{2} \right\rceil \leq T_1 \leq n-f \\ f<T_1 \leq n-f \\ f<T_2 \leq n-f \end{cases} \tag{5}$$

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 BG$[1,1,2]$ as shown in Fig. 10.

BG$[1,1]$ with `DP1`. This time, we have the following system of inequalities (6):

$$\begin{cases} T-(n-T_1+f)>T/2 \\ f<T<n-f \\ \left\lceil \dfrac{n+f+1}{2} \right\rceil \leq T_1 \leq n-f \\ f<T_1 \leq n-f \end{cases} \tag{6}$$

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 BG$[1,1]$ is shown in Fig. 11.

References

1. Abraham, I., Gueta, G., Malkhi, D., Alvisi, L., Kotla, R., Martin, J.P.: Revisiting fast practical Byzantine fault tolerance. arXiv preprint arXiv:1712.01367 (2017)
2. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Reaching consensus for asynchronous distributed key generation. In: PODC (2021)
3. Abraham, I., Malkhi, D., Nayak, K., Ren, L., Yin, M.: Sync HotStuff: Simple and practical synchronous state machine replication. In: IEEE Symposium on Security and Privacy (S&P) (2020)
4. Abraham, I., Malkhi, D., Spiegelman, A.: Asymptotically optimal validated asynchronous byzantine agreement. In: PODC. pp. 337–346. ACM (2019)
5. Abspoel, M., Attema, T., Rambaud, M.: Malicious security comes for free in consensus with leaders. Cryptology ePrint Archive (2020)

6. Alhaddad, N., Varia, M., Zhang, H.: High-threshold AVSS with optimal communication complexity. In: Financial Cryptography. Lecture Notes in Computer Science, Springer (2021)
7. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Public Key Cryptography — PKC 2003 (2002)
8. Buchman, E.: Tendermint: Byzantine fault tolerance in the age of blockchains (2017)
9. Buterin, V., Griffith, V.: Casper the friendly finality gadget. arXiv: Cryptography and Security (2017)
10. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. Journal of Cryptology **18**(3), 219–246 (2005)
11. Cachin, C., Vukolic, M.: Blockchain consensus protocols in the wild. In: DISC (2017)
12. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. TOCS **20**(4), 398–461 (2002)
13. Das, S., Xiang, Z., Ren, L.: Asynchronous data dissemination and its applications. CCS (2021)
14. Duan, S., Zhang, H.: Pace: Fully parallelizable bft from reproposable byzantine agreement. CCS (2022)
15. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of ACM **32**(2), 288–323 (1988)
16. Gelashvili, R., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A., Xiang, Z.: Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. arXiv preprint arXiv:2106.10362 (2021)
17. Giridharan, N., Howard, H., Abraham, I., Crooks, N., Tomescu, A.: No-commit proofs: Defeating livelock in bft. Cryptology ePrint Archive (2021)
18. Guerraoui, R., Raynal, M.: The alpha of indulgent consensus. The Computer Journal **50**(1), 53–67 (2007)
19. Guerraoui, R., Knežević, N., Quéma, V., Vukolić, M.: The next 700 bft protocols. ACM Transactions on Computer Systems **32**(4), 12:1–12:45 (2015)
20. Jalalzai, M.M., Niu, J., Feng, C., Gai, F.: Fast-hotstuff: A fast and resilient hotstuff protocol. arXiv preprint arXiv:2010.11454 (2021)
21. Kokoris Kogias, E., Malkhi, D., Spiegelman, A.: Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In: CCS (2020)
22. Maric, O., Sprenger, C., Basin, D.: Consensus refined. In: DSN (2015)
23. Martin, J.P., Alvisi, L.: Fast byzantine consensus. IEEE Transactions on Dependable and Secure Computing **3**(3), 202–215 (2006)
24. Milosevic, Z., Hutle, M., Schiper, A.: Unifying byzantine consensus algorithms with weak interactive consistency. In: Principles of Distributed Systems. pp. 300–314. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
25. Mostefaoui, A., Rajsbaum, S., Raynal, M.: A versatile and modular consensus protocol. In: DSN (2002)
26. Rütti, O., Milosevic, Z., Schiper, A.: Generic construction of consensus algorithms for benign and byzantine faults. In: DSN. pp. 343–352 (2010)
27. Shoup, V.: Practical threshold signatures. In: Advances in Cryptology — EUROCRYPT 2000 (2000)
28. Song, Y.J., Van Renesse, R., Schneider, F.B., Dolev, D.: The building blocks of consensus. In: ICDCN. pp. 54–72. Springer (2008)

29. Stern, G., Abraham, I.: Information theoretic hotstuff. In: OPODIS (2020)
30. Sui, X., Duan, S., Zhang, H.: Marlin: Two-phase bft with linearity. DSN (2022)
31. Yin, M., Malkhi, D., Reiter, M.K., Gueta, G.G., Abraham, I.: Hotstuff: Bft consensus with linearity and responsiveness. In: PODC (2019)

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 $m = \langle \textsc{msg}\text{-}1, cview, b.height, b, QC_x \rangle$ to all replicas. Upon receiving a MSG-1 message m, each replica verifies $m.block$. The replica then sends the leader $\textsc{qcVote}(\langle \textsc{vote}\text{-}1, b.height, curView, b, QC_x \rangle)$.

▷ 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 $\textsc{qcCreate}()$ to obtain a QC_{j-1} for block b, broadcasts $\langle \textsc{msg}\text{-}j, cview, b.height, b, b.QC_{j-1} \rangle$ to all replicas and enters the next phase. Upon receiving a valid MSG-j message, a replica sends the leader QCVOTE(m) using a VOTE-j message. In 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.

▷ In lines 25-32 (the commit step), the leader broadcasts QC_z to all replicas in a COMMIT message. Upon receiving a valid QC_z, each replica commits the corresponding block.

Algorithm 3: Normal-case protocol for BG[x, z] and BG[x, y, z]

```
1  Initialization:
2   localState: cview ← 1, vb ← ⊥. QC_1, QC_2, ···, and QC_z are initialized to ⊥.
3   criticalState: set by layer 3 of the framework; contain variables in localState.
4   lockState: QC_y (in BG[x, y, z]) or ⊥ (in BG[x, z])
5   flag: a system parameter set by layer 3 of the framework.
6  ▷ Phase 1:
7  as a leader
8   b' ← qcBlock(QC_x), b ← ⟨cview, b'.height + 1, req, hash(b')⟩
9   broadcast ⟨MSG-1, cview, b.height, b, b'.QC_x⟩
10 as a replica
11  wait for message ⟨MSG-1, cview, b.height, b, b'.QC_x⟩ from leader(cview)
12  if b'.view = b.view = cview and b.height = b'.height+1 and b.pl = hash(b') and rank(b') ≥ rank(vb)
13    QC_x ← b'.QC_x, vb ← b, m ← ⟨VOTE-1, cview, b.height, b, ⊥⟩
14    send qcVote(m) to Leader(cview)
15 ▷ Phase 2 to Phase z (for 2 ≤ j ≤ z):
16 as a leader
17  wait for T_{j-1} matching votes: M ← {σ|σ is a signature for ⟨VOTE-(j-1), cview, b.height, b, ⊥⟩}:
18      broadcast ⟨MSG-(j), cview, b.height, b, qcCreate(M)⟩
19 as a replica
20  wait for message ⟨MSG-(j), cview, b.height, b, b.QC_{j-1}⟩ from Leader(cview):
21  if b.view = cview and rank(b) > rank(qcBlock(QC_{j-1}))
22    QC_{j-1} ← b.QC_{j-1}, m ← ⟨VOTE-j, cview, b.height, b, ⊥⟩
23    if j = y + 1 then lb ← b
24    send qcVote(m) to leader(cview)
25 ▷ Commit
26 as a leader
27  wait for T_z matching votes: M ← {σ|σ is a signature for ⟨VOTE-(z), cview, b.height, b, ⊥⟩}
28      broadcast ⟨COMMIT, cview, b.height, b, qcCreate(M)⟩
29 as a replica
30  wait for message ⟨COMMIT, cview, b.height, b, b.QC_z⟩ from leader(cview)
31      if b.view = cview and rank(b) > rank(qcBlock(QC_z))
32        QC_z ← b.QC_z, execute the requests in b in order
33 ▷ Finally
34  switch to New-view phase of view change protocol if timeout occurs in any phase
```

B.2 View Change Protocol

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

▷ In lines 1-3, a replica triggers view change by incrementing *cview* by one. Then p_i sends *criticalState* in a NEW-VIEW message to the next leader.

▷ In lines 4-14, the new leader collects a set of T NEW-VIEW messages, denoted as M_v. It then executes FSB(M_v) to obtain (b', π), where b' is a block and π 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 $\langle$NEW-VIEW$, cview, \perp, \perp, criticalState\rangle$ to LEADER$(cview)$
4 ▷ **View-update**
5 **as** a new leader
6 $(b',\ \pi) \leftarrow$ FSB(M_v) //M_v is a set of T NEW-VIEW messages collected in v
7 $b \leftarrow \langle b'.height + 1, cview, req, hash(b')\rangle$
8 broadcast $m = \langle$VIEW-UPDATE$, cview, b.height, b, \pi\rangle$
9 **if** $flag = 0$ **then** switch to Phase 2 of normal-case operation
10 **if** $flag = 1$ **then**
11 **wait for** T_1 matching votes:
12 $M \leftarrow \{m \mid m = \langle$VOTE-1$, cview, b.height, b, \perp\rangle\}$: $QC_1 \leftarrow$ QCCREATE(M)
13 execute Phase 2 to Phase (x+1) without updating $lockState$ until $b.QC_x$ is generated
14 then update QC_x with $b.QC_x$ and switch to phase 1 of normal-case operation
15 **as** a replica
16 **wait for** $\langle$VIEW-UPDATE$, cview, b.height, b, \pi\rangle$ from LEADER$(cview)$
17 **if** (VV$(m)=0$ in BG$[x, z]$ **or** VV$(m, vb)=0$ in BG$[x, y, z]$) **then** discard m
18 **if** $flag=0$ **then** switch to Line 13 of normal-case without updating QC_x
19 **if** $flag = 1$ **then**
20 execute Phase 1 to Phase x of normal-case operation without updating $lockState$
21 **if** a $\langle$MSG-1$\rangle$ message for a block b^* extending b is received
22 switch to Phase 1 of normal-case operation
23 ▷ **Finally**
24 switch to New-view phase of view change protocol if $timeout$ 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 VIEW-UPDATE message m. Depending on the parameter $flag$, there are two cases. If $flag = 0$, the leader directly switches to phase 2 of normal-case operation. If $flag = 1$, the leader still switches to phase 2 but does not update its $lockState$ until $b.QC_x$ is generated.

▷ In lines 15-22, a replica accepts a VIEW-UPDATE message m in view v from the new leader only if VV(m) outputs 1 in BG$[x, y, z]$ or VV(m, lb) outputs 1 in BG$[x, y, z]$. Similar to the two cases for the leader, according to the $flag$ parameter, the replica may take different actions upon receiving m. If $flag = 0$, the replica switches to Line 13 of normal-case without updating QC_x. If $flag = 1$, the replica still votes for m but does not update $lockState$ or commit any block until a 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).

predicates	candidates	conditions to satisfy predicates	other conditions (from layer 2)		solvable?
			x, y-dependent conditions	other general conditions	
DP1 $\textsc{Votes}(b, T, z) > T/2$	BG[1, 1]	$T-(n-T_1+f) > T/2$	—	$f < T \le n-f$ $\left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f$ $f < T_j \le n-f$ for $j \in [1..z]$	✓
	BG[1, 2]		—		✓
	BG[2, 2]		—		✓
	BG[1, 3]		—		✓
	BG[2, 3]		—		✓
	BG[3, 3]		—		✓
	BG[1, 1, 2]		$n-T_1+f+1 \le T_2 \le n-f$	$f < T \le n-f$ $\left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f$ $f < T_j \le n-f$ for $j \in [1..z]$	✓
	BG[1, 1, 3]		$n-T_1+f+1 \le T_2 \le n-f$		✓
	BG[1, 2, 3]		$n-T_1+f+1 \le T_3 \le n-f$		✓
	BG[2, 2, 3]		$n-T_1+f+1 \le T_3 \le n-f$		✓
DP2 $\textsc{Votes}(b, T, y) \ge f+1$ and $\textsc{Certs}(b, T, x, z) \ge T-(2f+1)$ (for BG[x, y, z])	BG[1, 1, 2]	$T-(n-T_1+f) \ge f+1$ $T-(n-T_2+f) \ge T-(2f+1)$	$n-T_1+f+1 \le T_2 \le n-f$	$f < T \le n-f$ $\left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f$ $f < T_j \le n-f$ for $j \in [1..z]$	✓
	BG[1, 1, 3]	$T-(n-T_1+f) \ge f+1$ $T-(n-T_2+f) \ge T-(2f+1)$	$n-T_1+f+1 \le T_2 \le n-f$		✓
	BG[1, 2, 3]	$T-(n-T_1+f) \ge f+1$ $T-(n-T_2+f) \ge T-(2f+1)$	$n-T_1+f+1 \le T_3 \le n-f$		✓
	BG[2, 2, 3]	$T-(n-T_1+f) \ge f+1$ $T-(n-T_3+f) \ge T-(2f+1)$	$n-T_1+f+1 \le T_3 \le n-f$		✓
DP3 $\textsc{Certs}(b, T, x, z) > 0$ (for BG[x, z])	BG[1, 1]	$T-(n-1) > 0$	—	$f < T \le n-f$ $\left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f$ $f < T_j \le n-f$ for $j \in [1..z]$	×
	BG[1, 2]	$T-(n-T_2+f) > 0$	—		✓
	BG[2, 2]	$T-(n-1) > 0$	—		×
	BG[1, 3]	$T-(n-T_2+f) > 0$	—		✓
	BG[2, 3]	$T-(n-T_3+f) > 0$	—		✓
	BG[3, 3]	$T-(n-1) > 0$	—		×
DP3 $\textsc{Certs}(b, T, x, y) > 0$ (for BG[x, y, z])	BG[1, 1, 2]	$T-(n-1) > 0$	$n-T_1+f+1 \le T_2 \le n-f$	$f < T \le n-f$ $\left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f$ $f < T_j \le n-f$ for $j \in [1..z]$	×
	BG[1, 1, 3]	$T-(n-1) > 0$	$n-T_1+f+1 \le T_2 \le n-f$		×
	BG[1, 2, 3]	$T-(n-T_2+f) > 0$	$n-T_1+f+1 \le T_3 \le n-f$		✓
	BG[2, 2, 3]	$T-(n-1) > 0$	$n-T_1+f+1 \le T_3 \le n-f$		×
DP5 $\textsc{Certs}(b, T, x, z) > 0$ and $\textsc{Votes}(b, T, y) > 0$ (for BG[x, y, z])	BG[1, 1, 2]	$T-(n-T_1+f) > 0$ $T-(n-T_2+f) > 0$	$n-T_1+f+1 \le T_2 \le n-f$	$f < T \le n-f$ $\left\lceil \frac{n+f+1}{2} \right\rceil \le T_1 \le n-f$ $f < T_j \le n-f$ for $j \in [1..z]$	✓
	BG[1, 1, 3]	$T-(n-T_1+f) > 0$ $T-(n-T_2+f) > 0$	$n-T_1+f+1 \le T_2 \le n-f$		✓
	BG[1, 2, 3]	$T-(n-T_1+f) > 0$ $T-(n-T_2+f) > 0$	$n-T_1+f+1 \le T_3 \le n-f$		✓
	BG[2, 2, 3]	$T-(n-T_1+f) > 0$ $T-(n-T_3+f) > 0$	$n-T_1+f+1 \le T_3 \le n-f$		✓

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 $flag$ 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 $M_v.vb$ is larger than that of the highest QC_x qc in $M_v.QC_x$, then p_i broadcasts qc to ask replicas whether qc is indeed the highest QC_x they receive.

If T replicas respond "yes", then according to DP5, p_i knows that a block with a higher rank than that of qc cannot be committed. These responses form a certificate, and p_i can use this certificate to validate its new block that extends qc.

	protocol	predicates	replicas	message pattern	steps	authenticator complexity		message complexity	
						normal-case	view change	normal-case	view change
1-phase	FaB5[23]	—	$5f+1$	AtoA	2	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	BG[1, 1]	DP1	$5f+1$	1toA	3	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
2-phase	PBFT[12]	—	$3f+1$	AtoA	3	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	Tendermint[8]	—	$3f+1$	AtoA	3	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	Casper[9]	—	$3f+1$	AtoA	3	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
	Fast-Hotstuff[20]	—	$3f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2]	DP1	$5f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[2, 2]	DP1	$5f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2]	DP3	$3f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 2]	DP1	$5f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 2]	DP2	$4f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 2]	DP5	$3f+1$	1toA	5	$\mathcal{O}(n)$	$\mathcal{O}(n^2)/\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
3-phase	Hotstuff[31]	—	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 3]	DP1	$5f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[2, 3]	DP1	$5f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[3, 3]	DP1	$5f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 3]	DP3	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[2, 3]	DP3	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 3]	DP1	$5f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2, 3]	DP1	$5f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[2, 2, 3]	DP1	$5f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 3]	DP2	$4f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2, 3]	DP2	$4f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[2, 2, 3]	DP2	$4f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2, 3]	DP3	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 1, 3]	DP5	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)/\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[1, 2, 3]	DP5	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)/\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
	BG[2, 2, 3]	DP5	$3f+1$	1toA	7	$\mathcal{O}(n)$	$\mathcal{O}(n)/\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table 6: BFT protocols generated using BG$[x, z]$ and BG$[x, y, z]$ for $z \leq 3$. For instance, BG[1, 1] with a predicate DP1 is a 1-phase protocol. One can have many instantiations for the same parameters (e.g., 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.

protocol	pred	T	T_1
BG[1, 1]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$

protocol	pred	T	T_1	T_2
BG[1, 2]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$
BG[2, 2]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$
BG[1, 2]	`DP3`	$(f, n-f]$	$[\lceil \frac{n+f+1}{2} \rceil, n-f]$	$[n+f-T+1, n-f]$
BG[1, 1, 2]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[n+f-T_1+1, n-f]$
BG[1, 1, 2]	`DP2`	$(f, n-f]$	$[\max(n-T+2f+1, \lceil \frac{n+f+1}{2} \rceil), n-f]$	$[\max(n-f+1, n-T_1+f+1), n-f]$
BG[1, 1, 2]	`DP5`	$(f, n-f]$	$[\max(n-T+f+1, \lceil \frac{n+f+1}{2} \rceil), n-f]$	$[\max(n-T+f+1, n-T_1+f+1), n-f]$

protocol	pred	T	T_1	T_2	T_3
BG[1, 3]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$	$[f+1, n-f]$
BG[2, 3]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$	$[f+1, n-f]$
BG[3, 3]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$	$[f+1, n-f]$
BG[1, 3]	`DP3`	$(f, n-f]$	$[\lceil \frac{n+f+1}{2} \rceil, n-f]$	$[n+f-T+1, n-f]$	$[f+1, n-f]$
BG[2, 3]	`DP3`	$(f, n-f]$	$[\lceil \frac{n+f+1}{2} \rceil, n-f]$	$[f+1, n-f]$	$[n+f-T+1, n-f]$
BG[1, 1, 3]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[n+f-T_1+1, n-f]$	$[f+1, n-f]$
BG[1, 2, 3]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$	$[n+f-T_1+1, n-f]$
BG[2, 2, 3]	`DP1`	$(f, n-f]$	$(\max\{\lceil \frac{n+f+1}{2} \rceil, n+f-\frac{T}{2}\}, n-f]$	$[f+1, n-f]$	$[n+f-T_1+1, n-f]$
BG[1, 1, 3]	`DP2`	$(f, n-f]$	$[\max\{n-T+2f+1, \lceil \frac{n+f+1}{2} \rceil\}, n-f]$	$[n+f+1-\min\{2f, T_1\}, n-f]$	$[f+1, n-f]$
BG[1, 2, 3]	`DP2`	$(f, n-f]$	$[\max\{n-T+2f+1, \lceil \frac{n+f+1}{2} \rceil\}, n-f]$	$[n-f+1, n-f]$	$[n-T_1+f+1, n-f]$
BG[2, 2, 3]	`DP2`	$(f, n-f]$	$[\max\{n-T+2f+1, \lceil \frac{n+f+1}{2} \rceil\}, n-f]$	$[f+1, n-f]$	$[n+f+1-\min\{2f, T_1\}, n-f]$
BG[1, 2, 3]	`DP3`	$(f, n-f]$	$[\lceil \frac{n+f+1}{2} \rceil, n-f]$	$[n+f-T+1, n-f]$	$[n+f-T_1+1, n-f]$
BG[1, 1, 3]	`DP5`	$(f, n-f]$	$[\max\{n-T+f+1, \lceil \frac{n+f+1}{2} \rceil\}, n-f]$	$[n+f+1-\min\{T, T_1\}, n-f]$	$[f+1, n-f]$
BG[1, 2, 3]	`DP5`	$(f, n-f]$	$[\max\{n-T+f+1, \lceil \frac{n+f+1}{2} \rceil\}, n-f]$	$[n-T+f+1, n-f]$	$[n-T_1+f+1, n-f]$
BG[2, 2, 3]	`DP5`	$(f, n-f]$	$[\max\{n-T+f+1, \lceil \frac{n+f+1}{2} \rceil\}, n-f]$	$[f+1, n-f]$	$[n+f+1-\min\{T, T_1\}, n-f]$

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.

When a replica responds *no*, the replica is also required to send its QC_x qc' in this response, where $rank(qc') > rank(qc)$. 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 $b'.view = b.view$. 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.

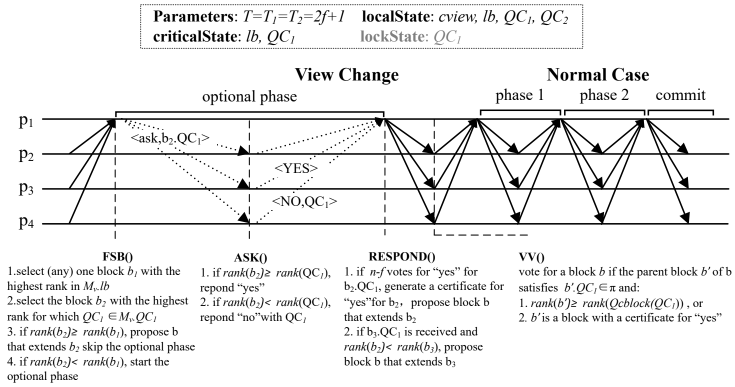

Fig. 12: BG[1,1,2]* with DP5

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 $\perp$ 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 $\perp$. 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 $\lceil (T+1)/2 \rceil$, 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 been included in the NEW-VIEW message if another view change occurs.

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

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

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

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

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

Realization of FSB() and VV() for DP1. We present a modified realization of the FSB() and VV() functions for BG$[x, z]$ and BG$[x, y, z]$ in Table 8. FSB() takes as input M_v and outputs (b, π). 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 $num(b, M_v.vb) > T/2$, then these votes for b_1 can form a VC vc and b_1 is set as b (lines 02-04). Then FSB() outputs $(b_1, (M_v, vc))$ in BG$[x, z]$ and $(b_1, (\perp, vc))$ in BG$[x, y, z]$ (lines 05-1 and 05-2). Otherwise, we have $b_1 = null$. 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 VC vc for b_3 is included in $M_v.QC_{vc}$ and vc is the view-change certificate with the highest view contained in $M_v.QC_{vc}$. If the view of vc is larger than b_2, then FSB() returns $(b_2, (M_v, vc))$ in BG$[x, z]$ and $(b_2, (\perp, vc))$ in BG$[x, y, z]$ (lines 12-1 and 12-2) Otherwise, FSB() returns $(b_2, (M_v, b_2.QC_x))$ in BG$[x, z]$ and $(b_2, (\perp, b_2.QC_x))$ in BG$[x, y, z]$ (lines 13-1 and 13-2).

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

Algorithm 5: View change protocol

▷ **New-view:**
as a replica
$cview \leftarrow cview + 1$, $m \leftarrow \perp$ $m_1 \leftarrow \langle \text{NEW-VIEW}, cview, \perp, \perp, criticalState \rangle$
if $rank(vb) > rank(\text{QCBLOCK}(QC_x))$
  $m \leftarrow \langle \text{NEW-VIEW}, cview, vb, vb.height, \perp \rangle$
send m_1 and $\text{QCVOTE}(m)$ to $\text{LEADER}(cview)$
▷ **View-update**
as a new leader
$(b', \pi) \leftarrow \text{FSB}(M_v)$ // M_v is a set of T NEW-VIEW messages collected in v
$b \leftarrow \langle b'.height + 1, cview, req, hash(b') \rangle$
broadcast $m = \langle \text{VIEW-UPDATE}, cview, b.height, b, \pi \rangle$
wait for T_1 matching votes:
$M \leftarrow \{m \mid m = \langle \text{VOTE-1}, cview, b.height, b, \perp \rangle\}$: $QC_1 \leftarrow \text{QCCREATE}(M)$
//switch to Phase 2 of normal-case operation
as a replica
wait for $\langle \text{VIEW-UPDATE}, cview, b.height, b, \pi \rangle$ from $\text{LEADER}(cview)$
if $\text{VV}(m, \cdot) = 1$ **and** $\pi = (\pi_1, \pi_2)$
  if π_2 is a view change certificate **then** $QC_{vc} \leftarrow \pi_2$
  if π_2 is a QC_x **and** $rank(\pi_2) > rank(QC_x)$ **then** $QC_x \leftarrow \pi_2$
  //switch to line 13 of normal-case without updating QC_x
▷ **Finally**
switch to New-view phase of view change protocol if *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 $\text{FSB}()$ function:

- **FSBL-wliveness**: If $C \subseteq S(M_v)$, then $\text{FSB}(M_v)$ outputs (b, π).

<table>
<tr><th>pred</th><th>protocol</th><th>FSB()</th><th>VV()</th></tr>
<tr><td rowspan="2">DP1</td><td>BG$[x,z]$
(vb, QC_x)
$flag = 1$</td><td rowspan="2">func FSB(M_v)
$b_1 \leftarrow null, b_2 \leftarrow null, b_3 \leftarrow null$
for $b \in M_v.vb$
if $num(b, M_v.vb) > T/2$ then $b_1 \leftarrow b$
$\pi_2 \leftarrow$ a VC for b
return $(b_1, (M_v, \pi_2))$ BG$[x,z]$
return $(b_1, (\perp, \pi_2))$ BG$[x,y,z]$
for $b : b.QC_x \in M_v.QC_x$
if $rank(b) > rank(b_2)$ then $b_2 \leftarrow b$
for b : a ceritificate qc for $b \in M_v.QC_{vc}$
if $qc.view > b_3.view$
$b_3 \leftarrow b$, $vc \leftarrow qc$
if $vc.view > b_3.view$
return $(b_3, (M_v, vc))$ BG$[x,z]$
return $(b_3, (\perp, vc))$ BG$[x,y,z]$
return $(b_2, (M_v, b_2.QC_x))$ BG$[x,z]$
return $(b_2, (\perp, b_2.QC_x))$ BG$[x,y,z]$</td><td>func VV$(\langle$VIEW-UPDATE$, cview, b'.height, b', \pi\rangle)$
$b \leftarrow$ the parent block of b'
if $M_v \in \pi$ and $(b, \pi) =$ FSB(M_v) and $b.view < v$
then return 1
return 0</td></tr>
<tr><td>BG$[x,y,z]$
(vb, QC_x)
$flag = 1$</td><td>func VV$(\langle$VIEW-UPDATE$, v, b'.height, b', \pi\rangle, lb)$
$b \leftarrow$ the parent block of b'
if a VC vc for $b \in \pi$ and $b.view < cview$
if $rank(lb) \leq rank(b)$ then return 1
if $b.QC_x \in \pi$ and $b.view < v$
if $rank(lb) \leq rank(b)$ then return 1
return 0</td></tr>
</table>

Table 8: The realization of FSB() and VV() for the optimized protocols according to DP1.

The other properties are exactly the same as those for BG$[x, y, z]$. We have the following liveness theorem:

Theorem 13. *BG$[x, y, z]$ achieves weak liveness, if $T_j \leq n - f$ for all $j \in [1..z]$, and FSBL-wliveness and VVL-liveness hold.*

D.2 Weak Liveness: Layer 3 and Instantiations

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 QC_x 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 $M_v(C)$. We also define $M^b(C)$ such that $M^b(C)$ is identical to M^b, except that $M^b(C)$ 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., Tendermint and Casper): $flag$ is set to 0, BG$[x, y, z]$ satisfies DP4 iff $b.QC_x \in M^b(C)$ 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) $\left\lceil \frac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f$; 3) $f < T_j \leq n - f$ for $1 \leq j \leq z$; and 4) $x \leq y < z$, $n - T_1 + f + 1 \leq T_{y+1}$.*

pred	*critical state*	FSB()	VV()
DP4	BG$[x, y, z]$ (QC_x) $flag = 0$	**func** FSB(M_v) $b_2 \leftarrow null$ **for** $b : b.QC_x \in M_v.QC_x$ **if** $rank(b) > rank(b_2)$ **then** $b_2 \leftarrow b$ **return** (b_2, π) s.t. $\pi = (b_2.QC_x)$	**func** VV($\langle$VIEW-UPDATE$, cview, b'.height, b', \pi\rangle, lb$) $b \leftarrow$ the parent block of b' **if** $b.QC_x \in \pi$ **and** $rank(b) \geq rank(lb)$ **then** **if** $b.view < cview$ **then return** 1 **return** 0

Table 9: Realizing FSB() and VV() according to DP4.

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 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 Tendermint 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 $b.view$ for some $k \in [1..z]$, then $b.QC_1, \cdots, b.QC_k$ 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 VOTE-k messages for b. As $T_k > f$, at least one correct replica p_i has sent VOTE-k messages for b. Hence, p_i must have received $b.QC_{k-1}$ contained in a 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 VOTE-1 message for b_v to form the $b_v.QC_1$ in view v. If $b_v{}'.view = v$, then p_i must have received a QC_x for $b_v{}'$ and $rank(b_v{}') < rank(b_v)$. 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, $b_v{}'.view < v$. 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 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 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 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−1 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−1 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_1 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−1 message for b or d are sent during normal-case protocol. Note that each correct replica sends a VOTE−1 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 $d.height > b_v$, 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 $B^v = \{b \mid \text{block } b \text{ has been committed before view } v\}$. 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 $rank(b^v) > rank(b')$.*

Proof. For any block $b \in B^v$, according to Lemma 7, a QC_x for b is formed in $b.view$. 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 $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$ 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_z 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 $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$ 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 $v+1, \cdots, v+k$ $(k \geq 1)$, then for any QC_x qc formed in view v' $(v+1 \leq v' \leq k)$,* 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' 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 $b^{v'}.height \geq b.height$ 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 has been formed in view v'. Since $T > f+1$, at least one correct replica has sent VOTE-1 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 $v+1, \cdots, v+k_0$, 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_z 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 VOTE-1 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 $v, \cdots, v+k$ ($k \geq 1$), then for any QC_x qc formed in view $v, \cdots, v+k$, 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 $v+1, \cdots, v+k_0$ 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 $b^{v'} = b$ 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, 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, 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'. Note that FSB-safety and VV-safety hold in view $v, \cdots, v'$. 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, z] satisfies safety II.

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.*

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, π). Then p_i sends a view-update message m such that $m.block = b_v$. We distinguish two case:

1) $flag = 1$. According to VV-liveness, b_v can be voted by enough replicas to form $b_v.QC_1$. Then b_v will be voted by all the correct replicas to form $b_v.QC_2, \cdots, b_v.QC_x$. As $T_j \leq n - f$ for all $j \in [1..x]$, $b_v.QC_x$ 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 $b'.QC_1, \cdots, b'.QC_z$ 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 $b_v.QC_1, \cdots, b_v.QC_z$ 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 FSBL-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 FSBL-liveness, p_i can run fsb() and obtain some (b, π). 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 DP1

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

Proof. In a BG$[x,z]$ or BG$[x,y,z]$ protocol, 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 Lemma 7, $b.QC_1$ is also formed by the leader in view v. Accordingly, at least $T_1 - f$ correct replicas have sent VOTE-1 messages for b such that $b.QC_1$ is formed. As p_i set its QC_z to $b.QC_z$ 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 VOTES$(b, T, z) > T/2$ and the BG$[x,z]$ or BG$[x,y,z]$ satisfies 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 FSB(M_v) outputs some (b, π). Based on M_v, we can obtain two intermediate variables, block b_1 and block b_2. Since $num(b_1, M_v.vb) > T/2$, b_1 is a unique block or *null*. By Lemma 10, b_2 is also a unique block. Therefore, after comparing ranks of b_1 and b_2, FSB(M_v) will output a block together with a proof π.

Lemma 14. *If BG[x, z] satisfies DP1, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $T > 2f$, 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 $rank(b^v) > rank(b')$. 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, FSB(M_v) will output some (b, π). 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 DP1 holds in BG$[x,z]$, we consider two cases:

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

2) $num(b^v, M^v.vb) \leq T/2$. Since $num(b^v, M^b.vb) > T/2$ 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 $rank(b') > rank(b^v)$ 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 NEW-VIEW message must be an extension of b^v. So b_1 must be an extension of b^v or *null*. In addition, p_i will send its QC_x qc' in a NEW-VIEW message, QCVIEW$(qc') = v - 1$, and QCHEIGHT$(qc') \geq b^v.height$. Then $rank(b_2) \geq rank(qc')$. According to Lemma 10, we now know b_2 is b^v or an

extension of b^v. Hence, b must be b^v or an extension of b^v 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 $rank(b^v) > rank(b')$.

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 $b^{v'}.view$. 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 $w+1, \cdots, w+k$ (for any constant 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 $w+1, \cdots, w+k$ (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, $\textsc{fsb}(M_{w+k+1})$ will output some (b, π). 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 $b^{v'}$ or an extension of $b^{v'}$. Since `DP1` holds in BG$[x,z]$, we consider two cases:

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

2) $num(b^{v'}, M_{w+k+1}.vb) \leq T/2$. Since $num(b^v, M^b.vb) > T/2$ 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' during view $w, w+1, \cdots, w+k$. Since $flag = 1$, p_i changed its vb only if $rank(b') > rank(b^{v'})$. According to Algorithm 1, p_i has received a QC_x qc for the parent block b'' of b' and $rank(b'') \geq rank(b^{v'})$. By Lemma 10, Lemma 11 and the hypothesis, we know that 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 NEW-VIEW message must be an extension of b^v. So b_1 must be an extension of $b^{v'}$ 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 $rank(\textsc{qcBlock}(qc')) \geq rank(\textsc{qcBlock}(qc))$. Therefore, $rank(b_2) \geq rank(b^{v'})$ and $b_2.QC_x$ is included in $M_{w+k+1}.QC_x$. If $b_2.view = w$, then according to Lemma 10, b_2 is either $b^{v'}$ or an extension of $b^{v'}$. If $b_2.view > w$, then according to Lemma 11 and the inductive hypothesis, b_2 is either $b^{v'}$ or an extension of $b^{v'}$. Hence, b is either $b^{v'}$ or an extension of $b^{v'}$ no matter b equals b_1 or b equals b_2.

In both cases b is either $b^{v'}$ or an extension of $b^{v'}$, 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 $BG[x,z]$.*

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).

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 $m.block$. If $(b, m.justify)$ is an output of FSB(M_v), then $M_v \in m.justify$ and VV(m) outputs 1. This completes the proof.

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) $\left\lceil \frac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f$; 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 $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, and $f < T \leq n - f$, then FSBL-liveness holds for BG[x, y, z].*

Proof. The proof resembles the proof of Lemma 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, π) after comparing ranks of b_1 and b_2.

Lemma 19. *If BG[x, y, z] satisfies DP1, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $T > 2f$, $x \leq y < z$, $T_{y+1} \geq n - T_1 + f + 1$, 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 $rank(b^v) > rank(b')$. 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 $y < z$, 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 $rank(b_v) < rank(b^v)$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), VV($m, lockState$) outputs in view v 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_l denote QCBLOCK(qc). Since a correct replica only change its QC_x to a QC_x with the same or a higher rank, $rank(qc) \geq rank(b^v)$. VV($m, lockState$) returns true by p_i if one of the following two conditions is satisfied:

1) $m.justify$ contains $b_v.QC_x$, $b_v.view < v$ and $rank(b_v) \geq rank(b_l)$ (lines 15-16 in Table 2).
2) $m.justify$ contains $M_v.vb$, $num(b_v, M_v.vb) > T/2$, $b_v.view < v$ and $rank(b_v) \geq rank(b_l)$ (lines 13-14 in Table 2).

Suppose that $rank(b_v) < rank(b^v)$. In this case, VV$(m, lockState)$ outputs 0 since none of the above conditions is satisfied. Suppose that $rank(b_v) \geq rank(b^v)$. If case 1) is satisfied, then according to Lemma 10, b_v must be equal to b^v or an extension of b^v. If case 2) is satisfied, then at least one correct sender of a message in M_v has changed its vb from b^v to b_v in view $v-1$ since BG$[x, y, z]$ satisfies DP1. According to Algorithm 3, $rank(b_v) > rank(b^v)$ and the 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^v. According to the assumption that either b^v is conflicting with b^v or $rank(b_v) < rank(b^v)$, for all the correct replicas who have locked b^v, 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 20. *If BG[x, y, z] satisfies 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 BG[x, y, z].*

Proof. For any view v, let $B^v = \{b \mid$ block b 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 $rank(b^v) > rank(b')$.

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'. 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 $b^{v'}$ has been committed in view w and $y < z$, at least $T_{y+1} - f \geq n - T_1 + 1$ correct replicas have locked $b^{v'}$ and sent VOTE-(y+1) for $b^{v'}$ to form $b^{v'}.QC_{y+1}$ in view w. Let m denote a VIEW-UPDATE message such that $m.view = w+k+1$ and the parent block b' of $m.block$ is conflicting with $b^{v'}$ or $rank(b') < rank(b^{v'})$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), VV$(m, lockState)$ returns true in view $w+k+1$ 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_l denote QCBLOCK(qc). Since a correct replica only change its QC_x to a QC_x with the same or a higher rank, $rank(qc) \geq rank(b^{v'})$. VV$(m, lockState)$ 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 $rank(b') \geq rank(b_l)$ (lines 15-16 in Table 2).
2) $m.justify$ contains $M_{w+k+1}.vb$, $b'.view < w+k+1$ and $num(b', M_{w+k+1}.vb) > T/2$ and $rank(b') \geq rank(b_l)$ (lines 13-14 in Table 2).

Suppose that $rank(b'_{w+1}) < rank(b^{v'})$. In this case, $\textsc{vv}(m, lockState)$ outputs 0 since both conditions are not satisfied. Suppose that $rank(b') \geq rank(b^{v'})$. If case 1) is satisfied, then according to Lemma 7, Lemma 10, Lemma 12 and the inductive hypothesis, b' must be equal to $b^{v'}$ or an extension of $b^{v'}$. If case 2) is satisfied, then at least one correct sender of a message in M_{w+k+1} has changed its vb from $b^{v'}$ to b' in view $w, \cdots, w+k$ since BG$[x,y,z]$ satisfies `DP1` and $T > 2f$. According to Algorithm 3, $rank(b') > rank(b^{v'})$, the QC_x for the parent block b'' of b' is received by the replica, $b''.view = b'.view$ and $rank(b'') \geq rank(b^{v'})$. By Lemma 10, Lemma 12 and the inductive hypothesis, b' must be an extension of $b^{v'}$. According to the assumption that either b' is conflicting with $b^{v'}$ or $rank(b') < rank(b^{v'})$, for all the correct replicas who have locked $b^{v'}$, $\textsc{vv}(m, lockState)$ 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 $b^{v'}$, 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 21. *If BG[x,y,z] satisfies* `DP1`*, $f < T_j \leq n-f$ 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 $B_l^v = \{b \mid \text{block } b \text{ has been locked before view } v\}$. According to Lemma 7 and Lemma 10, we can fine $b_l^v \in B_l^v$ such that for all $b' \in B_l^v$ and $b' \neq b_l^v$, $rank(b_l^v) > rank(b')$. Given a VIEW-UPDATE message m in view v, let b denote the parent block of $m.block$. If $(b, m.justify)$ is the output of $\textsc{fsb}(M_v)$ on some M_v, there are two cases to consider:

1) $b_l^v.QC_x \in M_v.QC_x$. In this case, b is a block such that $rank(b) \geq b_l^v$. 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 $b_l^v.QC_x \in M_v.QC_x$, we have that $rank(b_2) \geq b_l^v$. If b_1 exists, we can also have that $rank(b_1) \geq b_l^v$ since BG$[x,y,z]$ satisfies `DP1`. Therefore, we know that $\textsc{vv}(m, lockState)$ 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) $b_l^v \notin M_v.QC_x$. We distinguish two cases. If there exists a block b_v such that $b_v.QC_x \in M_v.QC_x$ and $rank(b_v) \geq rank(b_l^v)$, then block b should satisfies that $rank(b) \geq rank(b_l^v)$. Therefore, we know that $\textsc{vv}(m, lockState)$ will output 1 by all correct replicas. If any $qc \in M_v.QC_x$ satisfies that $rank(\textsc{qcBlock}(qc)) < rank(b_l^v)$, we prove that the output of $\textsc{fsb}(M_v)$ is $(b_l^v, M_v.vb)$. Since BG$[x,y,z]$ satisfies `DP1`, we have $num(b_l^v, M^{b_l^v}.vb) > T/2$. Any correct replica changes its vb and QC_x only in the case that it has received a QC_x qc such that $rank(\textsc{qcBlock}(qc)) \geq rank(vb)$. Therefore, no correct replica of any message in M_v has changed its vb from b_l^v to some other block and $num(b_l^v, M^{b_l^v}.vb) > T/2$. Then the output of $\textsc{fsb}(M_v)$ is $(b_l^v, M_v.vb)$ and $\textsc{vv}(m, lockState)$ 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* DP1*; 2) $2f < T \leq n-f$; 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) $x \leq y < z$, $n - T_1 + f + 1 \leq T_{y+1}$.*

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 DP2

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.

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 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 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 $x \leq y < z$, $d.QC_x$ is also formed in view v. Accordingly, at least $T_{x+1} - f$ correct replicas have received $d.QC_x$ and sent 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_x's to $d.QC_x$ in view v. Thus, fewer than $n - T_{x+1}$ correct replicas have not yet set their QC_x'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_x'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 CERTS$(d, T, x, z) \geq T-(2f+1)$.

Therefore, BG[x, y, z] satisfies DP2.

Lemma 22. *If BG[x, y, z] satisfies* DP2*, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, 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 FSB(M_v) outputs some (b, π). Based on M_v, we can obtain four intermediate variables, block b_0, b_1, b_2 and b_3. Since $num(b_1, M_v.vb) > f+1$ and $num(b_2, M_v.vb) > f+1$, b_1, b_2 and b_0 are unique blocks or *null*. By Lemma 10, b_3 is also a unique block. Therefore, after comparing ranks of b_0, b_1, b_2 and b_3, FSB(M_v) will output a block together with a proof π.

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 $rank(b^v) > rank(b')$. 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 $y < z$, at least $T_{y+1} - f \geq n - T_1 + f + 1$ correct replicas have locked b^v and sent VOTE-(y+1) for b^v to form $b^v.QC_{y+1}$ 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 $rank(b_v) < rank(b^v)$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), VV($m, lockState$) outputs 1 in view v by $p_i\}$.

For any correct replica p_i who has locked b^v, let b_l be its lb when receiving m. Since a correct replica only change its lb to a block with the same or a higher rank, $rank(b_l) \geq rank(b^v)$. Note that $x \leq y$. According to Lemma 7 and Lemma 10, $b_l.QC_x$ is formed in view $v - 1$ and b_l is equal to b^v or an extension of b^v. VV($m, lockState$) outputs 1 by p_i if one of the following four conditions is satisfied:

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

Suppose that $rank(b_v) < rank(b^v)$. In this case, VV($m, lockState$) 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 $num(b^v.QC_x, M^{b^v}.QC_x) \geq T-(2f+1)$. As $rank(b_v) < rank(b^v)$ and a correct replicas only updates its QC_x to a QC_x for a block with a higher rank, $num(b_v.QC_x, M_v.QC_x) \leq 2f + 1$, a contradiction.

Suppose that $rank(b_v) \geq rank(b^v)$. 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 $b'.view = b_v.view$. Since $rank(b_v) > rank(b_l) \geq rank(b^v)$, we have that $rank(b') \geq rank(b^v)$. 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 $b_v = b_l$. 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 $rank(b_v) < rank(b^v)$, for all the correct replicas who have locked b^v, $\textsc{vv}(m, lockState)$ 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 24. *If BG[x, y, z] satisfies* `DP2`*, $T_j > f$ for $j \in [1..z]$, $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, 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 $rank(b^v) > rank(b')$.

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'. 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 VOTE-(y+1) message for $b^{v'}$ to form $b^{v'}.QC_{y+1}$ in view w. Let m denote a VIEW-UPDATE message such that $m.view = w + k + 1$ and the parent block b' of $m.block$ is conflicting with $b^{v'}$ or $rank(b') < rank(b^{v'})$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), $\textsc{vv}(m, lockState)$ outputs 1 in view $w+k+1$ 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 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 $lockState$ when receiving m. Let b_l denote QCBLOCK(qc). Since a correct replica only change its QC_x to a QC_x with the same or a higher rank, $rank(qc) \geq rank(b^{v'})$. 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 $b^{v'}$. $\textsc{vv}(m, lockState)$ 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 $rank(b') \geq rank(b_l)$ and $b_v.view < w+k+1$ (lines 32-33 in Table 2).
2) $m.justify$ contains $M_{w+k+1}.vb$, $num(b', M_{w+k+1}.vb) \geq f+1$ and $rank(b') > rank(b_l)$ and $b'.view < w+k+1$ (lines 34-35 in Table 2).
3) $m.justify$ contains $M_{w+k+1}.vb$, $num(b', M_v.vb) \geq f+1$ and $b_v = b_l$ and $b'.view < w+k+1$ (lines 36-37 in Table 2).
4) $m.justify$ contains $M_{w+k+1}.QC_x$, and $num(b'.QC_x, M_v.QC_x) > 2f+1$ and $b'.view < w+k+1$ (lines 38-39 in Table 2).

Suppose that $rank(b') < rank(b^{v'})$. In this case, $\textsc{vv}(m, lockState)$ returns true only when condition 4) is satisfied. Since BG$[x, y, z]$ satisfies DP2 and $b^{v'}$ is committed in view w, we have $num(b^{v'}.QC_x, M^{b^{v'}}.QC_x) \geq T - (2f+1)$. As $rank(b') < rank(b^{v'})$ and correct replicas only change its QC_x to a QC_x the same or a higher rank, $num(b'.QC_x, M_{w+k+1}.QC_x) \leq 2f+1$, a contradiction.

Suppose that $rank(b') \geq rank(b^{v'})$. 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 $b^{v'}$ 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' such that $b''.view = b'.view$. Since $rank(b') > rank(b_l) \geq rank(b^{v'})$, we have that $rank(b'') \geq rank(b^{v'})$. According to $x \leq y$, Lemma 10, Lemma 7, Lemma 12 and the inductive hypothesis, b'' must be equal to $b^{v'}$ or is an extension of $b^{v'}$. Therefore, b' must be an extension of $b^{v'}$, If case 3) is satisfied, then $b' = b_l$. Therefore, b' is equal to $b^{v'}$ or an extension of $b^{v'}$.

According to the assumption that b' is conflicting with $b^{v'}$, for all the correct replicas who have locked $b^{v'}$, $\textsc{vv}(m, lockState)$ will output 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 $b^{v'}$, 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 25. *If BG$[x, y, z]$ satisfies DP2, $f < T_j \leq n - f$ for $j \in [1..z]$, and $T_1 \geq \left\lceil \frac{n+f+1}{2} \right\rceil$, $x < y \leq z$, then VVL-liveness holds in BG$[x, y, z]$.*

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

1) $b_l^v.QC_x \in M_v.QC_x$. In this case, b is a block such that $rank(b) \geq b_l^v$. Therefore, $\textsc{vv}(m, lockState)$ 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) $b_l^v \notin M_v.QC_x$. Let b_3 denote the block with the highest rank such that $b_3.QC_x \in M_v.QC_x$. We distinguish two sub-cases. If $rank(b_3) \geq rank(b_l^v)$, then we know that the block b should satisfies that $rank(b) \geq rank(b_l^v)$. Therefore, we know that $\textsc{vv}(m, lockState)$ outputs 1 by all correct replicas.

If $rank(b_3) < rank(b_l^v)$, we prove that $num(b_l^v, M_v.vb) \geq f+1$. Since BG$[x, y, z]$ satisfies DP2, we have $num(b_l^v, M^{b_l^v}.vb) > f + 1$. Any correct replica changes its vb and QC_x only in the case that it has received a QC_x qc such that $rank(\textsc{qcBlock}(qc)) \geq rank(vb)$. Thus, for any correct sender of a message in M_v, if a replica has sent a VOTE-1 message for b_l^v, it will not change its vb before sending NEW-VIEW message. We have that $num(b_l^v, M_v.vb) \geq f+1$. If there exists another block d such that $num(d, M_v.vb) \geq f+1$ and $rank(d) = rank(b_l^v)$, Then the output of $\textsc{fsb}(M_v)$ is $(b_3, b_3.QC_x, M_v.QC_x)$. As $num(d, M_v.vb) \geq f+1$ and $num(b_l^v, M_v.vb) \geq f+1$, QC_x for the parent block of b_l^v and d are both formed in view $b_l^v.view$. According to Lemma 10, b_l^v and d have the same parent block b_3. Then $num(b_3.QC_x, M_v.QC_x) \geq 2f+2$. Therefore, we know that $\text{VV}(m, lockState)$ 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 $(b_l^v, M_v.vb)$. Therefore, according to Lemma 10, we know that $\text{VV}(m, lockState)$ 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 $j \in [1..z]$; and 5) $n - T_1 + f + 1 \leq T_{y+1}$.*

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, $b.QC_{x+1}$ is also formed by the leader in view v. Accordingly, at least $T_{x+1} - f$ correct replicas have sent $\text{VOTE}-(x+1)$ 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 $T_{x+1} - f$ correct replicas set their QC_x's to $b.QC_{x+1}$ 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 $d.QC_x$, i.e., there at least $T - (n - T_{x+1} + f)$ b in $M^b.vb$. Since $T - (n - T_{x+1} + f) > 0$, 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, $b.QC_x$ 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 $b.QC_x$ is included in $M^b.QC_x$ 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 FSB-liveness holds for BG[x, z].*

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

Lemma 27. *If BG[x, z] satisfies 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 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, there exists $b^v \in B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$, we have $rank(b^v) > rank(b')$. Since there exists a committed block in view $v-1$, we know $b^v.view = v-1$. By Lemma 26, $\textsc{fsb}(M_v)$ outputs some (b_2, π). We now prove that b_2 is either b^v or an extension of b^v. Since $\texttt{DP3}(b^v, T, x)$ holds in BG[x, z], we consider two cases:

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

2) $b^v.QC_x \notin M_v.QC_x$. Since DP3 holds in BG[x, z], it is clear that at least one correct sender p_i of a message in M_v has changed its QC_x from $b^v.QC_x$ to QC_x for some other block b' before sending the NEW-VIEW message. According to Algorithm 3, p_i changes its QC_x in view $v-1$ only if it receives $b'.QC_x$ and $rank(b') \geq rank(b^v)$. 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, FSB-safety holds in view v in BG[x, z].

Lemma 28. *If BG[x, 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 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, there exists $b^v \in B^v$ such that for all $b' \in B^v$ and $b' \neq b^v$, we have $rank(b^v) \geq rank(b')$

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 $b^{v'}.view$. We prove that FSB-safety holds in view v' by iterative method. The proof consists of the following steps. First, we prove that FSB-safety holds in view $w+1$. Then we prove that if FSB-safety holds in view $w+1, \cdots, w+k$ (for any integer $1 \leq k \leq v'-w-1$), 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 27, we know that FSB-safety holds in view $w+1$.

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

1) $b^{v'}.QC_x \in M_{w+k+1}.QC_x$. In this scenario, the output should satisfy that $b.QC_x \in M_{w+k+1}.QC_x$ and $rank(b) \geq rank(b^{v'})$. According to Lemma 10, Lemma 11 and the inductive hypothesis, we have that b equals $b^{v'}$ or b is an extension of $b^{v'}$.

2) $b^{v'}.QC_x \notin M_{w+k+1}.QC_x$. In this scenario, b isn't equal to $b^{v'}$, we need to prove that b must be an extension of $b^{v'}$. Since DP3 holds in BG$[x, z]$, 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 $b^{v'}.QC_x$ 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 $rank(b') \geq rank(b^{v'})$. Therefore, we have that $rank(b) \geq rank(b') \geq rank(b^{v'})$, QC_x for blocks $b, b', b^{v'}$ are formed in view $\{w, \cdots, w+k\}$ and $b \neq b^{v'}$. By Lemma 10, Lemma 11 and the inductive hypothesis, we know that b' must be equal to or an extension of $b^{v'}$, and b must be an extension of $b^{v'}$.

In both cases b is either $b^{v'}$ or an extension of $b^{v'}$, 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) $\left\lceil \frac{n+f+1}{2} \right\rceil \leq T_1 \leq n - f$; and 4) $f < T_j \leq n - f$ for $j \in [1..z]$.*

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_y 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 $T_1 - f$ 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 CERTS$(b, T, x, y) > 0$.

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 $b' \in B^v$ and $b' \neq b^v$, we have $rank(b^v) > rank(b')$. 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 VOTE-(y+1) for b^v to form $b^v.QC_{y+1}$ in view $v-1$. Let m denote a VIEW-UPDATE message such that $m.view = v$ and the parent block b' of $m.block$ is conflicting with b^v or $rank(b') < rank(b^v)$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), $\text{VV}(m, lockState)$ returns true in view v by $p_i\}$.

For any one correct replica p_i who has locked b^v, let b_l 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, $rank(b_l) \geq rank(b^v)$. $\text{VV}(m, lockState)$ returns true by p_i if $m.justify$ contains $b'.QC_x$, $b'.view < v$ and $rank(b') \geq rank(b_l)$ (lines 50-51 in Table 2).

If the condition is satisfied, then we know that $rank(b') \geq rank(b_l) \geq rank(b^v)$ and $b'.QC_x$ 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, $\text{VV}(m, lockState)$ 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 $rank(b^v) > rank(b')$.

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'. Out 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_z formed 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 VOTE-$(y+1)$ message for $b^{v'}$ to form $b^{v'}.QC_{y+1}$ in view w. Let m denote a VIEW-UPDATE message such that $m.view = w+k+1$ and the parent block b' of $m.block$ is conflicting with $b^{v'}$ or $rank(b') < rank(b^{v'})$. Let $P = \{p_i \mid p_i \in C$ (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_l 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, $rank(b_l) \geq rank(b^{v'})$. $\text{VV}(m, lockState)$ outputs 1 by p_i if $m.justify$ contains $b'.QC_x$, $b'.view < w+k+1$ and $rank(b')$ $\geq rank(b_l)$ (lines 50-51 in Table 2).

If the condition is satisfied, then we know that $rank(b') \geq rank(b_l) \geq rank(b^v)$. 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 $rank(b') < rank(b^{v'})$. 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 $B_l^v = \{b \mid \text{block } b \text{ has been locked before view } v\}$. According to $x \leq y$ and Lemma 10, we can find a block b_l^v in B_l^v such that for all $b' \in B_l^v$ and $b' \neq b_l^v$, we have $rank(b_l^v) > rank(b')$. Given a VIEW-UPDATE message m in view v, let b denote the parent block of $m.block$. If $(b, m.justify)$ is the output of $\text{FSB}(M_v)$ on a snapshot M_v, there are two cases to consider:

1) $b_l^v.QC_x \in M_v.QC_x$. In this case, b is a block such that $rank(b) \geq rank(b_l^v)$. 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) $b_l^v.QC_x \notin M_v.QC_x$. Note that any correct replica changes its vb and QC_x only in the case that it has received a QC_x qc such that $rank(\text{QCBLOCK}(qc)) \geq$

$rank(vb)$. Since BG$[x, y, z]$ satisfies DP3, at least one correct sender of a message in M_v has change its QC_x from $b_l^v.QC_x$ to QC_x for another block b'. Then $rank(b') \geq rank(b_l^v)$. In this case, b is a block such that $rank(b) \geq rank(b_l^v)$. Therefore, we know that $\text{VV}(m, lockState)$ outputs 1 by all correct replicas since conditions (lines 54-55 in Table 2) are satisfied for all of them. This complete the proof.

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 DP1; 2) $2f < T \leq n-f$; 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) $n-T_1+f+1 \leq T_{y+1}$.*

Proof. Correctness follows from Theorem 4, Theorem 5, Theorem 6, Lemma 31, Lemma 33 and Lemma 34.

K Proofs of Theorems for BG$[x, y, z]$ with DP5

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

Proof. In a 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 $b.view = v$ 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 $M^b.vb$. 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 $d.view = v$. 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_x's to $d.QC_x$ in view v. Thus, fewer than $n - T_{x+1}$ correct replicas have not yet set their QC_x'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_x's to $d.QC_x$, 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 $\text{CERTS}(d, T, x, z) \geq 1$.

Therefore, BG$[x, y, z]$ satisfies DP1.

Lemma 35. *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 $f < T \leq n-f$, then FSBL-liveness holds for 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_v and output (b, π) 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 $rank(b^v) > rank(b')$. 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 $y < z$, 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 to form $b^v.QC_{y+1}$ in view $v-1$. Let m denote a VIEW-UPDATE message such that $m.view = v$ and the parent block b' of $m.block$ is conflicting with b^v or $rank(b') < rank(b^v)$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), 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_l 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, $rank(b_l) \geq rank(b^v)$. VV($m, lockState$) outputs 1 by p_i if one of the following two conditions is satisfied:

1) $m.justify$ contains $b'.QC_x$, $b'.view < v$ and $rank(b') \geq rank(b_l)$ (lines 62-63 in Table 2).
2) $m.justify$ contains $M_v.QC_x$, $b'.view < v$ and $b'.QC_x$ 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 $rank(b') < rank(b^v)$. If the condition 2) is satisfied, then we know that b' is the block with highest rank for which a QC_x is included in $M_v.QC_x$. Since BG[x, y, z] satisfies DP5, we have that $num(b^v.QC_x, M^{b^v}.QC_x) \geq 1$. Note that any correct replica changes its QC_x with a QC_x with the same or higher rank. We have $rank(b') \geq rank(b^v)$. 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 $rank(b') < rank(b^v)$.

Therefore, for all the correct replicas who have locked b^v, 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 $rank(b^v) > rank(b')$.

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 $w+1, \cdots, w+k$ (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 36, 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_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 VOTE-(y+1) for $b^{v'}$ to form $b^{v'}.QC_{y+1}$ in view w. Let m denote a VIEW-UPDATE message such that $m.view = w+k+1$ and the parent block b' of $m.block$ is conflicting with $b^{v'}$ or $rank(b') < rank(b^{v'})$. Let $P = \{p_i \mid p_i \in C$ (the set of correct replicas), 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_l 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, $rank(b_l) \geq rank(b^{v'})$. 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 $rank(b') \geq rank(b_l)$ (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 $b^{v'}$ according to Lemma 7, Lemma 10, Lemma 12 and the inductive hypothesis. This is a contradiction with the assumption that either b' is conflicting with b^v or $rank(b') < rank(b^v)$. If the condition 2) is satisfied, then we know that b' is the block with the highest rank for which a QC_x is included in $M_{w+k+1}.QC_x$. Since BG$[x, y, z]$ satisfies DP5, we have that $num(b^{v'}.QC_x, M^{b^{v'}}.QC_x) \geq 1$. Note that any correct replica changes its QC_x only with QC_x with the same or a higher rank, $rank(b') \geq rank(b^{v'})$. 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 the assumption that either b' is conflicting with b^v or $rank(b') < rank(b^v)$. .

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

1) $b_l^v.QC_x \in M_v.QC_x$. In this case, the output of FSB(M_v) is $(b, b.QC_x)$ or $(b, M_v.QC_x)$. For both situations, b is a block such that $rank(b) \geq rank(b_l^v)$. Hence, VV$(m, lockState)$ returns true by all correct replicas since condition 1) (lines 62-63 in Table 2) is satisfied.

2) $b_l^v \notin M_v.QC_x$. We need to consider two sub-cases. If $rank(b) \geq rank(b_l^v)$, then we know that $\textsc{vv}(m, lockState)$ returns true by all correct replicas since condition 1) (lines 77-78 in Table 2) is satisfied. If $rank(b) < rank(b_l^v)$, we need to prove that the output of $\textsc{fsb}(M_v)$ is of the form $(b, b.QC_x, M_v.QC_x)$. Since $\mathrm{BG}[x, y, z]$ satisfies DP5, then we have $b_l^v \in M^{b_l^v}.vb$. Recall again any correct replica changes its vb and QC_x only in the case that it has received a QC_x qc such that $rank(\textsc{qcBlock}(qc)) \geq rank(vb)$. 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_l^v and we know $b_l^v \in M^v.vb$. Then the output of $\textsc{fsb}(M_v)$ is $(b, M_v.QC_x)$ and $\textsc{vv}(m, lockState)$ 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 *BG[x, y, z] (in Table 2) achieves safety and responsiveness if the following are satisfied: 1) BG[x, y, z] satisfies DP5; 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) $x \leq y < z$, $n - T_1 + f + 1 \leq T_{y+1}$.*

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