Network Agnostic Consensus: Tight Bounds across Fault Models

Evan Hart[1,3] Vivek Malik[1,3] Aidan Mokalla[1,3] and Erica Blum[1,2]

[1] Reed College, Portland, OR 97202, USA
[2] EricaBlum@reed.edu
[3] Equal contribution

Abstract. Each classical consensus fault model combines one of 3 network assumptions (synchronous, partially synchronous, or asynchronous network) with one of three adversarial assumptions (crash, omission, or Byzantine faults). We establish tight fault tolerance bounds for *network-agnostic* binary agreement across all $\binom{3\times 3}{2} = 36$ possible *pairs* of classical models. By determining the maximum number of faults tolerable while maintaining consistency, validity, and termination for each model pairing, these bounds allow us to compare the inherent costs of network-agnostic consensus with those of single model and mixed-tolerance protocols.

Keywords: Byzantine Agreement · Consensus · Network Agnostic.

1 Introduction

Distributed consensus protocols typically assume a single combination of network model and fault type. However, distributed consensus protocols must function across diverse network conditions and fault scenarios, since real systems face uncertainty about which model accurately describes their execution environment. Classical results establish optimal fault tolerance bounds for individual models combining specific network assumptions (synchronous, partially synchronous, or asynchronous) with particular fault types (crash, omission, or Byzantine). Recent results regarding mixed-tolerance protocols also analyze resilience to multiple combined fault types simultaneously. Although uncertainty is present about which models are most accurate, and we should strive to propose protocols that function in a variety of contexts, real-world consensus systems are unlikely to be accurately described as explicit combinations of multiple models as they are by mixed-tolerance protocols. Network-agnostic protocols address this by functioning correctly under either of two possible models, without advance knowledge of which applies.

Network-agnostic protocols address this uncertainty by tolerating faults under each of multiple possible models individually. Such protocols must succeed regardless of whether the execution follows one model or another, without advance knowledge of which applies. This work establishes tight bounds for all

$\binom{9}{2} = 36$ possible combinations of classical models.

Each represents a protocol that must function correctly under either of two specified single models. For each pair, we determine the maximum number of faults tolerable while maintaining consistency, validity, and termination properties of binary agreement. Our results include both feasibility theorems with explicit protocol constructions and impossibility theorems demonstrating fundamental limits.

2 Definitions and Notation

Definition 1 (Faults). *The following results prove or disprove the existence of protocols Π achieving a certain task in a distributed network. However, we do not assume that executions of Π involve all parties perfectly following the protocols relevant in each section. Whether due to internet outages, software bugs, or malicious hackers, parties may deviate from the protocol in a few ways.*

- ***Correct party****: a party that follows the protocol exactly. All correct parties are honest.*
- ***Honest party****: A party that is either correct, crashed, or experiencing omissions.*
- ***Crash****: a party executes Π correctly up until some instruction, then fails to execute all later instructions (essentially becoming inert).*
- ***Omission****: a party executes all instructions correctly, except that it fails to receive and/or send some messages (if it fails to receive a message it behaves as a correct party would had it not received that message).*
- ***Byzantine****: A party that may behave arbitrarily, including following or not following any instructions of Π, and sending any messages to any set of parties (subject to computational constraints such as the inability to forge signatures). We emphasize that whether a node is Byzantine is not determined experimentally, and a Byzantine node may behave exactly according to protocol.*

Definition 2 (Network). *In a distributed network, two parties can only communicate by secure point-to-point channels. We also consider each party to have such a channel to itself, which delivers messages instantaneously. For distinct parties p_i, p_j, we do not expect instantaneous delivery. There are three commonly used network models governing the delay involved in message sending.*

- ***Synchrony****: all messages are delivered in less than Δ time, where Δ is a known parameter (perhaps a conservative estimate of internet speed).*
- ***Partial synchrony****: there must be some (unknown) time GST such that messages sent at time t must be delivered before time $\max\{GST, t\} + \Delta$. A protocol does not have access to the value of GST and must function no matter how large it is.*
- ***Asynchrony****: a message that is sent is eventually delivered.*

Definition 3 (Models). *A* **model** *is a set of rules governing the executions that a protocol runs in. A protocol Π cannot succeed in all possible executions, so we say that a protocol* **tolerates** *a model $M(t)$ if, when we constrain executions to follow the rules of model $M(t)$, the protocol succeeds with probability 1. In this notation, M is a* model family*, specified by a particular type of faults the adversary is allowed to create and a network type. t is the number of faults the adversary is allowed to create.*

Definition 2 lists 3 network types: ***s****ynchrony,* ***p****artial synchrony, and* ***a****synchrony. Definition 1 lists 3 fault types:* ***c****rash,* ***o****mission, and* ***b****yzantine. Combining these, we name 9 model families according to the initials of the network rule and the fault rule: SC, SO, SB, PC, PO, PB, AC, AO, AB. For example, SB represents the set of synchronous executions in which some number of parties may fail arbitrarily.*

We say $M_1 \subseteq M_2$ when, for a given t, all executions of $M_1(t)$ are also executions of $M_2(t)$. We also may say that M_1 is more restrictive *(for the adversary).*

Definition 4 (Model pairs). *A protocol that should function in two different models $M_1(t), M_2(u)$ is called an* **agnostic protocol***, and works in the* **model pair** *denoted $M_1(t) + M_2(u)$. More generally, we will say the protocol is for the model pair $M_1 + M_2$ without reference to the faults it tolerates in each model.*

We can also compare model pairs with the following rule: $M_1 + M_2 \subseteq M_3 + M_4$ when $M_1 \subseteq M_3$ and $M_2 \subseteq M_4$, or when $M_1 \subseteq M_4$ and $M_2 \subseteq M_3$.

Definition 5 (Binary Agreement). *In the binary agreement problem, each party in a collection of n parties (some of which may be faulty) receives a binary input, 0 or 1. Each party may eventually* decide *an irrevocable binary output. We require the following properties of a (possibly randomized) protocol that solves the binary agreement problem:*

- ***Consistency****: if p_i, p_j are correct parties that decide bits b_i, b_j respectively, then $b_i = b_j$.*
- ***Validity****: if all honest parties receive the same input b, then all correct parties decide b. (Note the distinction between honest and correct parties.)*
- ***Termination****: with probability 1, all correct parties eventually decide.*
- ***Nontriviality****: with probability 1, at least one party decides. We note that this is not a standard property in the consensus literature, but it is weaker than the termination property when there is at least one correct party, so it does not effectively change the problem. We only use it to resolve a few edge cases.*

3 Classical Single-Model Results

We globally assume access to a public-key infrastucture (PKI) and randomness. The former allows us to bypass a well-known result that limits fault tolerance in the SB model to $t < n/3$; with a PKI we can get up to $t < n/2$ as explained

	C	**O**	**B**
S	$t < n$	$t < \frac{n}{2}$	$t < \frac{n}{2}$
P	$t < \frac{n}{2}$	$t < \frac{n}{2}$	$t < \frac{n}{3}$
A	$t < \frac{n}{2}$	$t < \frac{n}{2}$	$t < \frac{n}{3}$

Table 1. Classical results for each of the 9 single models we consider.

below. The latter allows us to bypass the celebrated result [10] that consensus in asynchrony is impossible if the adversary is given access to even one crash fault. Below we reference much better bounds that take advantage of unpredictability.

A protocol for the SC (synchronous crash) model achieving $t < n$ is presented in [12]. $t = n$ is impossible by nontriviality.

A protocol for the SB (synchronous Byzantine) model achieving $t < n/2$ is to repeat n copies of the broadcast protocol presented in [8] and decide on the majority output. $t \geq n/2$ is impossible by validity.

The protocol of the previous point also works in the omission model. $t \geq n/2$ is impossible by [9].

A protocol for the AO model (and thus, the AC, PO, and PC models) achieving $t < n/2$ is given in [3], which was intended for the AC model but also works in the AO model because it does not actually use the assumption that once it faults a party does not continue to message, only that some parties may not send messages. No better fault tolerance is possible by the same experiment above, in which permanent send and receive omissions on one subset of the parties is replaced by a high value of GST (longer than it takes all parties in the experiment to decide) and no delivery between S_0, S_1 before GST.

A protocol specifically for the PO model is in [13], the well-known Paxos protocol.

A protocol for the AB model (and thus, the PB model, which is more restrictive) achieving $t < n/3$ is presented in [5]. If $t \geq n/3$ parties are Byzantine, consensus is impossible (even in partial synchrony) by [9].

4 Related Work

4.1 Mixed Tolerance

Our results assume that the adversary either creates t_1 of one fault type *or* t_2 of a (possibly different) fault type. A line of works including [6] and [2], in contrast, consider (respectively) asynchronous and synchronous settings in which t_1 of one fault type and t_2 of another occur at the same time in one execution. Our results achieve an approximation of the results in these two papers.

Remark 1. $\Pi_{\text{ABA}}^{t_o+t_b}$ can tolerate t_o omissions and t_b Byzantine faults simultaneously when $2t_o + 3t_b < n$, which is optimal.

Proof. Every party will eventually certify at least $n-(t_o+t_b)$ messages, because there are at least that many correct parties. Also, there are at most $t_b < n - 2(t_o + t_b)$ Byzantine faults when $2t_o + 3t_b < n$. Thus all the lemmas pertaining to $\Pi_{\text{ABA}}^{t_o+t_b}$ continue to apply.

This bound is well-known to be optimal, so no protocol can tolerate t_o omissions and t_b Byzantine faults simultaneously when $2t_o + 3t_b \geq n$.

Remark 2. The protocol of Theorem 2 can tolerate t_c crashes and t_b Byzantine faults simultaneously when $t_c + 2t_b < n$, which is optimal.

Proof. Due to unconditional consistency of the Dolev-Strong protocol in synchrony, consistency is automatic. Termination is also guaranteed because all parties stop executing after $(n+1) \cdot \Delta$ time. If all honest parties receive input v then: by validity of DS, at least $n - t_c - t_b$ of the outputs received by a correct party from the instances of DS broadcast will be for v, and at most t_b will be for $1 - v$, so since $n - t_c - t_b > t_b$ it follows that every correct party will decide v.

This bound, too, is well-known to be optimal, so none does better.

Remark 3. This result is incomparable to [1] because

- Their protocol allows for multivalued agreement, whereas ours relies on the fact that only two inputs and outputs are possible. Their protocol also demands proofs of correctness and external validity, which simple repetition of our protocol to decide on multiple values could not provide.
- However, it does not seem likely that their paper can produce a single instantiation of a protocol that tolerates more Byzantine faults when less parties crash and more crashes when less parties are dishonest.

Granular synchrony, defined by [11], involves a mixed-network model in which certain communication links are synchronous and others are not (either partially synchronous or asynchronous). As with the mixed-fault model, we expect that results in this area are incomparable to our results, because our results assume a consistent network model in each execution, though the network model may differ from execution to execution.

4.2 Threshold Separation

The recent work [14] provides an interesting refinement of network-agnostic consensus, separating the fault threshold for each model into two: one threshold β, below which safety (consistency) is guaranteed; and another threshold γ, below which liveness is guaranteed. Thus, they consider consensus with four thresholds. Their results largely pertain to state machine replication and focus on the SB+AB pair; our bounds reflect binary agreement across all model pairs. Extending the multiple threshold concept to generalized model-agnostic protocols could be an insightful refinement of our results in the future.

From an implementation perspective, compiler-based approaches offer a complementary direction. Deligios & Erbes (2024) describe a black-box compiler

that composes a synchronous BA and an asynchronous BA into a single timing-agnostic protocol. Such compiler approaches incur overhead from switching between protocols and maintaining dual execution modes. Our bounds apply to any single protocol regardless of its internal structure. Such compilers have practical efficiency but necessarily inherit the fault thresholds of their component protocols, whereas our results establish tight feasibility and impossibility bounds that apply to any single protocol irrespective of its internal structure.

Orthogonal refinements to the fault model split omissions into distinct send and receive faults. Loss & Stern (2023) analyze simultaneous Byzantine faults together with s send-faulty and r receive-faulty parties, obtaining a tight bound of $n > 2t + s + r$. This is largely incomparable to our setting, which treats omission faults as encompassing both behaviors.

Network heterogeneity can also be modeled at the link level. Granular synchrony [11] permits some links to be synchronous while others are partially synchronous or asynchronous. Our results assume a single timing model per execution (though it may vary across executions), so granular synchrony is conceptually complementary rather than directly comparable.

4.3 Classical Results

Finally, our discussion relies on classical single-model baselines: synchronous crash tolerance up to $t < n$ [12]; synchronous omission up to $t < n/2$ via majority-based authenticated broadcast with a matching impossibility at $t \geq n/2$; synchronous Byzantine with PKI up to $t < n/2$ via authenticated broadcast [8]; partial synchrony for crash/omission via Paxos [13] and BFT bounds $t < n/3$ [9, **?**]; asynchronous Byzantine agreement up to $t < n/3$ [5]; agreement for asynchronous crash/omission faults [3]; and the impossibility of deterministic asynchronous consensus with a single crash [10].

5 Results

5.1 Network Agnostic Consensus Bounds

This figure shows the feasibility regions for each possible pairing.

This table summarizes the tight bounds for network agnostic consensus across all combinations of network models. Each entry (i, j) shows the conditions under which consensus is possible when the adversary can choose between network model i or network model j. The bounds are expressed as constraints on the number of faults t_i in model i and t_j in model j.

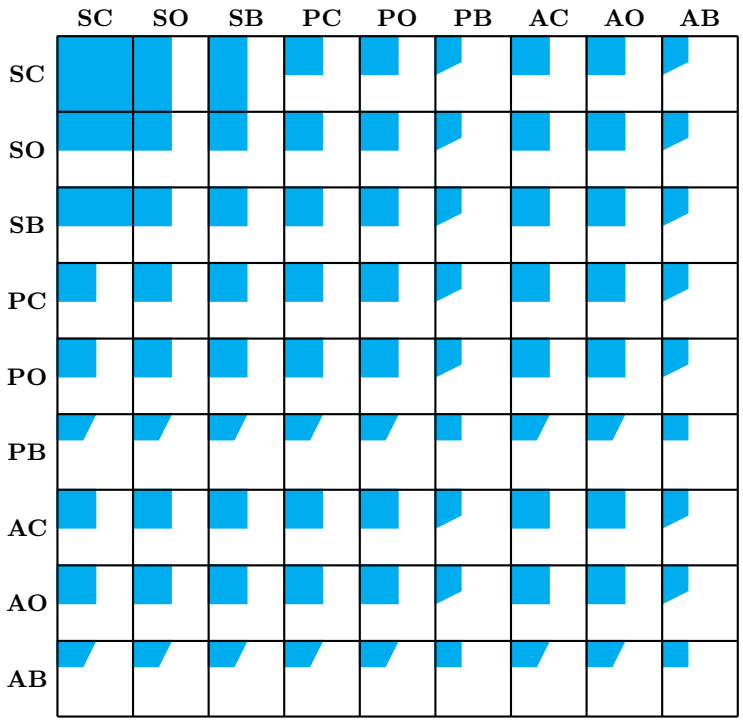

Fig. 1. Each cell is considered to be $n \times n$. A point $(x, y) \in [0, n]^2$ in the cell is colored blue if there is an agreement protocol tolerating t_x faults in the column's model, and t_y faults in the row's model. For example, in the (SC+SB) pair, there is a protocol tolerating t_c crash faults or t_b Byzantine faults when $t_c < n$ and $t_b < n/2$, shown as a rectangle filling half the square. (The figure shows some identical results twice, and some classical results, to highlight patterns in the results.)

Theorem 1 (Impossibility for SC+PC). *There is no consensus protocol tolerating t_s crashes in synchrony or t_p crashes in partial synchrony if $2t_s \geq n$ or if $2t_p \geq n$.*

Proof. We will specifically show that no such protocol has (non-conflicting liveness and consistency). The impossibility of $2t_p \geq n$ is inherited from the impossibility result from the partially synchronous crash model. Now suppose a protocol Π tolerates t_s crashes in synchrony, where $2t_s \geq n$. We show with the following experiment that Π cannot tolerate partial synchrony whatsoever, even with 0 crashes.

Consider a thought experiment in which the adversary partitions the n parties into a set S_0 of $\lceil n/2 \rceil$ parties and a set S_1 of $\lfloor n/2 \rfloor$ parties. All parties in S_0 receive input 0 and all parties in S_1 receive input 1. No party crashes, and communication within S_0 and within S_1 is bounded by Δ, but all communication between S_0 and S_1 is blocked indefinitely.

	SC	**SO**	**SB**	**PC**	**PO**	**PB**	**AC**	**AO**	**AB**
SC	t_{sc}								
SO	t_{sc} t_{so}	$2t_{so}$							
SB	t_{sc} $2t_{sb}$	$2t_{sb}$ $2t_{so}$	$2t_{sb}$						
PC	$2t_{sc}$ $2t_{pc}$	$2t_{so}$ $2t_{pc}$	$2t_{sb}$ $2t_{pc}$	$2t_{pc}$					
PO	$2t_{sc}$ $2t_{po}$	$2t_{so}$ $2t_{po}$	$2t_{sb}$ $2t_{po}$	$2t_{pc}$ $2t_{po}$	$2t_{po}$				
PB	$3t_{pb}$ $2t_{sc} + t_{pb}$	$3t_{pb}$ $2t_{so} + t_{pb}$	$3t_{pb}$ $2t_{sb} + t_{pb}$	$3t_{pb}$ $2t_{pc} + t_{pb}$	$3t_{pb}$ $2t_{po} + t_{pb}$	$3t_{pb}$			
AC	$2t_{sc}$ $2t_{ac}$	$2t_{so}$ $2t_{ac}$	$2t_{sb}$ $2t_{ac}$	$2t_{pc}$ $2t_{ac}$	$2t_{po}$ $2t_{ac}$	$3t_{pb}$ $2t_{ac} + t_{pb}$	$2t_{ac}$		
AO	$2t_{sc}$ $2t_{ao}$	$2t_{so}$ $2t_{ao}$	$2t_{sb}$ $2t_{ao}$	$2t_{pc}$ $2t_{ao}$	$2t_{po}$ $2t_{ao}$	$3t_{pb}$ $2t_{ao} + t_{pb}$	$2t_{ac}$ $2t_{ao}$	$2t_{ao}$	
AB	$3t_{ab}$ $2t_{sc} + t_{ab}$	$3t_{ab}$ $2t_{so} + t_{ab}$	$3t_{ab}$ $2t_{sb} + t_{ab}$	$3t_{ab}$ $2t_{pc} + t_{ab}$	$3t_{ab}$ $2t_{po} + t_{ab}$	$3t_{ab}$ $3t_{pb}$	$3t_{ab}$ $2t_{ac} + t_{ab}$	$3t_{ab}$ $2t_{ao} + t_{ab}$	$3t_{ab}$

Table 2. Network Agnostic consensus bounds. Rows and columns represent network models: SC (Synchronous Crash), SO (Synchronous Omission), SB (Synchronous Byzantine), PC (Partial Synchrony Crash), PO (Partial Synchrony Omission), PB (Partial Synchrony Byzantine), AC (Asynchronous Crash), AO (Asynchronous Omission), AB (Asynchronous Byzantine). Each cell shows the constraint(s) that must be satisfied (all implicitly $< n$). Diagonal entries show non-agnostic bounds. When two constraints are shown, both must be satisfied.

The views of parties in S_1 are identically distributed to their views in a *synchronous* execution of Π in which all parties receive input 1 but all parties in S_0 crash immediately, which Π must tolerate since $\lceil n/2 \rceil \leq t_s$ if $2t_s \geq n$. Thus, by t_s-validity of Π, parties in S_1 must, if they decide, decide 1. Furthermore, by t_s-termination of Π, there must be a time T such that all parties in S_1 have decided 1 by time T except with negligible probability.

Similarly, the views of parties in S_0 are identically distributed to their views in an execution of Π in which all parties receive input 0 but all parties in S_1 immediately crash, which Π must tolerate since $\lfloor n/2 \rfloor \leq t_s$ if $2t_s \geq n$. As such, by t_s-validity and t_s-termination, there must be some time T' such that all parties in S_0 must decide 0 by some time T' except with negligible probability.

Finally, consider a *partially synchronous* execution of Π in which no parties crash, communication within S_0 and within S_1 is bounded by Δ, but communication between S_0 and S_1 is delayed until time $\max\{T, T'\} + 1$. If the global stabilization time GST occurs after $\max\{T, T'\} + 1$, then the views of all parties in this execution over the time interval $\max\{T, T'\} + 1$ are identically distributed to the views of all parties over the same interval in the original experiment, because all parties terminate by the time any communication between the parties can reveal any difference between the experiment and the execution. This implies that parties in S_0 decide 0 while parties in S_1 decide 1, so consistency is

violated in the partially synchronous execution. (In the edge case where $n = 1$, consistency is not necessarily violated because S_0 could be empty; however, in that case our original assumption gives that t_s must be at least 1, allowing the adversary to break the nontriviality requirement. In the other case, where $n > 1$, both sets are nonempty and so there is a consistency violation.) □

The proof above describes the adversarial strategy that demonstrates the impossibility. We now formalize this adversary:

Algorithm 1: $\mathcal{A}_{\text{partition}}$: Adversarial Strategy for Theorem 1

Initialization.

Partition n parties into sets $S_0 = \{p_1, \ldots, p_{\lceil n/2 \rceil}\}$ and $S_1 = \{p_{\lceil n/2 \rceil + 1}, \ldots, p_n\}$.

Experiment *1: Create first view, violating consistency in synchrony*:

All parties in S_0 receive input 0.
All parties in S_1 receive input 1.
No party crashes.
Within S_0: deliver all messages within time Δ.
Within S_1: deliver all messages within time Δ.
Between S_0 and S_1: block all messages indefinitely.
// Views of S_1 are identical to synchronous execution where S_0 crashes
// Views of S_0 are identical to synchronous execution where S_1 crashes
// By validity and termination: S_0 decides 0 by time T', S_1 decides 1 by time T

Experiment *2: Create second view, violating consistency in partial synchrony*:

Use same partition S_0, S_1 with same inputs.
No parties crash.
Within S_0: deliver all messages within time Δ.
Within S_1: deliver all messages within time Δ.
Between S_0 and S_1: delay all messages until time $\max\{T, T'\} + 1$.
Set GST $> \max\{T, T'\} + 1$.
// Views over interval $[0, \max\{T, T'\} + 1]$ are identical to part (1) of the experiment
// Parties decide before cross-partition communication, violating consistency

Theorem 2 (Tight Impossibility Bound for SC+SB). *There is a protocol tolerating t_c crash faults in synchrony and t_b Byzantine faults in synchrony iff $t_c < n$ and $2t_b < n$.*

Proof. (Impossibility) The claim $t_c < n$ is by nontriviality. The claim that $2t_b < n$ is standard; suppose $n = 2t_b$ and imagine an experiment in which all nodes are correct and half receive 0 and half 1. Then by consistency they must all decide on a bit b. Now the adversary may corrupt all nodes that receive input b but not change their behavior at all; this violates validity.

(Feasibility) Protocol Π, described here, obtains the claimed tolerance. All nodes run a modification of the Dolev-Strong broadcast protocol (DS) as the sender. In our modification, when a node's extracted set is not of size 1, the node outputs $\perp$ from that instance of DS. After $n + 1$ rounds (when every instance of DS completes), each node decides on the majority non-$\perp$ output, breaking a tie by outputting 0.

By DS's consistency even up to $n - 1$ synchronous Byzantine faults, no matter which model Π is run in, all correct nodes end up with the same sequence of outputs from the n instances of DS. Thus, Π attains consistency in all cases.

Additionally, Π obtains validity in the Byzantine case if $t_b < n/2$. Suppose that all correct nodes receive input b. Then more than half of the instances of DS have an honest sender, and by validity of DS, all nodes end up with $> n/2$ copies of b as output. Thus, they all decide b. Finally, Π obtains validity in the crash case for any $t_c < n$. If every node receives input b then every correct node's output sequence contains at least one copy of b (because there is at least one honest sender and thus one successful instance of DS) and no copies of $1 - b$ (because no senders lie) so the majority non-$\perp$ output is always b, so all correct nodes output b.

□

The next few pages do not show tight impossibility for any model, but rather show impossibility and feasibility in separate theorems. Then at the end we complete the remaining 35 network pairings using the impossibility and feasibility results below and above.

Algorithm 2: $\Pi_{\text{SC+SB}}$ Feasibility Protocol from party i's perspective with input $v_i \in \{0, 1\}\}$

Initialization.
  outputs $\leftarrow$ array $\perp^n$.

Phase *Dolev-Strong Broadcast*:
  Run Dolev-Strong (DS) broadcast as sender with input v_i for $n+1$ rounds.
  for $j \leftarrow 1$ ***to*** n **do**
    Let S_j be the extracted set from the jth DS instance.
    if $|S_j| = 1$ **then**
      outputs[j] $\leftarrow$ the unique element in S_j.
    else
      outputs[j] $\leftarrow \perp$.

Phase *Decision*:
  Let majority $\leftarrow$ the most frequent non-$\perp$ value in outputs.
  if *tie in majority* **then**
    majority $\leftarrow 0$.
  Output majority and terminate.

Theorem 3 (Impossibility for SC+PB). *There is no consensus protocol tolerating t_{sc} crash faults in synchrony and t_{pb} Byzantine faults in partial synchrony if $3t_{pb} \geq n$ or if $2t_{sc} + t_{pb} \geq n$.*

Proof. $3t_{pb} < n$ is a standard bound for the Byzantine, partially synchronous setting. $2t_{sc} + t_{pb} < n$ is also proven necessary by [4], because their experiment only involves Byzantine nodes immediately crashing in the synchronous case.

Theorem 4 (Feasibility for SB+AB). *There is a consensus protocol tolerating t_{sb} Byzantine faults in synchrony and t_{ab} Byzantine faults in asynchrony if $3t_{ab} < n$ and $2t_{sb} + t_{ab} < n$.*

Proof. This is a result of [4].[4]

Theorem 5 (Feasibility for AO+AB). *There is a consensus protocol tolerating t_o omission faults in asynchrony or t_b Byzantine faults in asynchrony if $3t_b < n$ and $2t_o + t_b < n$.*

Proof. We describe a protocol $\Pi^{t_o}_{\text{AC+AB}}$, an adaptation of the protocol in section 13.2 of *Distributed Consensus: from Aircraft Control to Cryptocurrencies*, itself adapted from "Cachin et al. [CKS05]". $\Pi^{t_o}_{\text{AC+AB}}$ is parameterized by an integer $t_o < n$. It is divided into rounds, each lasting until sufficiently many messages have been received; we do not expect any kind of round synchronization.
$\Pi^{t_o}_{\text{AC+AB}}$ makes use of a coin-flipping primitive that acts as a random oracle. For each positive integer R we require that

[4] We should probably cite the specific theorem numbers.

- Before $n-t_o$ distinct parties call `CoinFlip`(R), the adversary learns nothing about the value of `CoinFlip`(R).
- Once $n-t_o$ distinct parties call `CoinFlip`(R), all correct parties are eventually notified of its value.
- The result of the coin is 50% likely to be 0 and 50% likely to be 1, and is independent of the values of other coins.

Cachin et al. [CKS05] presents a coin-flipping primitive with a threshold of k requests, secure under t arbitrary faults, for any $k \in (t, n-t]$. When $2t_b \leq 2t_o < n$ it holds that $t_b < n - t_o \leq n - t_b$, so this coin is suitable for our purposes.
$\Pi^{t_o}_{\text{AC+AB}}$ involves messages each of which is marked with a party's signature, a round number, a bit, and a title (either *prevote*,*vote*, or *terminate*), and a justification (a set of $n-t_o$ parties). When a party p_i receives a message it has not forwarded before, it forwards (i.e. multicasts) this message to all other parties. Furthermore, p_i *certifies* a message m when it agrees that the message could have been honestly sent by an honest party upon recieving the prior messages. We say a message is *properly justified* (PJ) if there exists an honest party that certifies it. More formally, p_i certifies the message m when one of these conditions holds at some time:

- m is marked as round 0.
- m is marked as a vote (resp. prevote) from round R, and from each party p_j in m's justification, p_i has certified a prevote (resp. vote) from round R (resp. $R-1$), signed by p_j, such that the protocol would send the message m if it received that set of prevotes (resp. votes).
- m is marked as a termination message from round R, and from each party p_j in m's justification, p_i has certified a vote from round R signed by p_j for m's bit.

Additionally a certified termination message for round R counts as a certified round-$(r+1)$ prevote and a PJ round-r vote for all rounds $r \geq R$.
Because of the recursive nature of this definition, every time p_i receives a message m from round R it must first determine whether to certify m; if it does, it must then check all messages it has received from rounds $r \geq R$ to check if it should certify them as a result. Importantly, in the case of equivocation, there may be multiple messages with the same party, round, and title, but different bits; if *at least one* message from each party cited by m supports m, then m is certified.
We say that a set of votes S is *unanimous for* v if every vote in S is for value v; if the value v is clear from context (or not relevant), we simply say S is unanimous.
With the necessary definitions in place, we present $\Pi^{t_o}_{\text{AC+AB}}$ from the perspective of party i with input v_i. $\langle \text{msg} \rangle_i$ represents a message with party i's signature.

Lemma 1. *(Credibility) If a correct party sends a message, all correct parties eventually certify it (or terminate first).*

Proof. The correct party p_i that sends a message does so on the basis of some messages that p_i certifies, which it forwards when it receives them; p_i certifies

these on the basis of some other p_i-certified messages which p_i also receives and forwards, and so on. All of these messages eventually arrive at all other non-terminating correct parties, so they all eventually certify all of them.

Lemma 2. *(AC validity) In an execution of $\Pi^{t_o}_{ABA}$ in which all parties receive input v, if at most t_o parties experience omissions and all others are correct, then all correct parties output v.*

Proof. There are $\geq n - t_o$ votes from correct parties for v sent in round-0, and no votes for $1 - v$ since all parties are honest. All correct parties receive and certify these and so cast a prevote in round 1 for v, and again no party casts a prevote for $1 - v$. All correct parties certify at least $n - t_o$ round-1 prevotes for v, so cast a round-1 vote for v. So all correct parties certify $n - t_o$ votes for v in round 2. Then they all decide v at the start of round 2 and ignore the coin flip. □

Lemma 3. *(AB validity) Fix $t_b \leq t_o$ such that $t_b < n-2t_o$. If at most t_b parties are faulty, and all correct parties receive input v, then all correct parties decide v.*

Proof. Every correct party sends a round-0 vote for v, so all correct parties receive enough round-0 votes to prevote in round 1. Of the $n-t_o > t_b+t_o \geq 2t_b$ round-0 votes signed by distinct parties received in round 1, more than half of them are for v (at most t_b, less than half, are from faulty parties, so are possibly not for v). Thus, all correct parties prevote for v in round 1. Furthermore, no party will certify a round-1 prevote for $1-v$ because a prevote for $1-v$ requires $> t_b$ parties to have prevoted for $1 - v$ in round 0. Thus, the only prevotes that can be certified are for v, and eventually each correct party certifies $n - t_o$ prevotes, and votes for v in round 1. Again, no party can vote for $1 - v$ or $\perp$ because each of those votes requires at least one round-1 prevote for $1 - v$, but no correct party certifies any such vote. As such, at the start of round 2 each correct party will only certify votes for v, and will receive enough of these to decide v at the start of round 2. □

Let F be the first round for which any correct party eventually certifies a termination message marked for round F. The next lemma shows consistency through round F.

Lemma 4. *(Intra-round consistency) In an execution of $\Pi^{t_o}_{AC+AB}$, if up to $t_o < n/2$ parties experience omissions, or up to $t_b < n - 2t_o$ parties are Byzantine, then the following hold for any R with $1 \leq R \leq F$:*

- *(a) Every correct party prevotes and votes exactly once in round R.*
- *(b) There is not both a PJ round-R vote for* `0` *and a PJ round-R vote for* `1`*.*

Proof. We induct on R. In round 1, every correct party receives at least $n - t_o$ round-0 votes, all of which are immediately certified, so every correct party prevotes exactly once. Since at least $n - t_o$ parties are correct, every correct

party receives (and certifies, by 1) enough prevotes to vote, and does so once. Thus, (a) holds in round 1. Next, if there is a PJ round-1 vote for `0` that cites a set of $n-t_o$ parties and a PJ round-1 vote for `1` that cites a set of $n-t_o$ parties, these two sets intersect in at least $n-2t_o > t_b$ parties, so some honest party is in the intersection. However, no honest party prevotes for both 0 and 1, by (a). Thus, (b) holds for $R = 1$.
Now let $R > 1$ and suppose (a) and (b) hold for round $R-1$. Again, each correct party prevotes at least once, since it eventually certifies enough votes from correct parties. Note that this relies on a correct party not terminating, and none does at this round: terminating based on round-F votes only happens after round F, while terminating based on votes from a round $< F$ would lead to a termination message being sent and certified unless all correct parties terminate in which case no round-F message would be PJ. Correct parties do not prevote twice, because that would require PJ round-$R-1$ votes for both 0 and 1, which cannot exist by the inductive hypothesis. This shows point (a). Point (b) holds by the same reasoning as given above for round 1. □

The arguments above do not work in rounds $\geq F+1$. This is because termination messages are only justified by votes from one round, but can be used in later rounds as both votes and prevotes; as such, the overlapping argument above does not hold. The following lemma shows, however, that in rounds after F the behavior of $\Pi^{t_o}_{\text{AC+AB}}$ is even more constrained.

Lemma 5. *(Consistency) In an execution of $\Pi^{t_o}_{AC+AB}$ in which any correct party certifies a termination message m for round F, all correct parties output b by the end of round $F+2$.*

Proof. First, note that round-F votes cannot justify both a termination message for 0 and for 1, by Lemma 4, which holds in round F. Next, a termination message must be justified by at least $n-t_o$ round-F votes, which overlaps with any set of $n-t_o$ round-F votes by at least $n-2t_o > t_b$ votes; thus, every correct and yet-to-terminate party certifies, at the start of round $F+1$, at least one vote from an honest party cited by m. Since honest parties do not equivocate, such a vote must be for b. As such, each correct party casts a prevote (possibly through a termination message) for b in round $F+1$. Also, no correct party justifies any prevote for $1-b$ since justifying one necessitates one of

- A vote for $1-b$. This is impossible by Lemma 4.
- $n-t_o$ PJ $\perp$ votes from distinct parties. This is impossible because there are at least $n-t_o-t_b$ honest parties that vote for b, so at most $n-(n-t_o-t_b) < n-t_o$ votes for $\perp$.

It follows that all correct parties vote for b in round $F+1$ (possibly through a termination message), so all remaining correct parties output b at the start of round $F+2$. □

We now show that parties are unlikely to continue through infinitely many rounds.

Following [Blockchain Book], let a round $R \geq 2$ be *lucky* if there does not and will never exist a PJ round-$(R-1)$ vote for $1-\texttt{CoinFlip}(R)$. In other words, the coin does not conflict with any PJ previous-round vote, so they are all for the coin's value or for $\perp$.

Lemma 6. *If R is lucky then all parties terminate by round $R+2$, even if t_o parties are Byzantine.*

Proof. Let $c = \texttt{CoinFlip}(R)$. In this case, all correct parties cast round-R prevotes for c, either justified by at least one round-$(R-1)$ vote for c, or by $n - t_o$ round-$(R-1)$ votes for $\perp$ (and based on the coin). Furthermore, no party will certify a prevote for $1-c$ because the necessary justification cannot exist. Thus, every correct party eventually certifies $n-t_o$ prevotes for c, and casts a round-R vote for c. In round $R+1$, then, there are $n - t_o$ votes for c (and certainly at most $t_o < n - t_o$ votes for $1 - c$, though in fact there cannot be any), so all correct parties output c at the start of round $R+1$. □

Lemma 7. *Fix a round $R \geq 2$. The probability that at least one of the rounds $R, R+1$ is lucky is at least 50%, no matter the events of earlier rounds.*

Proof. There are two possibilities, both of which lead to one of these rounds being lucky with 50% probability.
Case 1: At the time the round-R coin is flipped, when the adversary learns its value c, some correct party has already certified a vote for a bit $b \in \{0, 1\}$. Then, by the unpredictability of the coin, there is a 50% chance $c = b$, making R lucky.
Case 2: At the time the round-R coin is flipped, all votes that have been received and certified are for $\perp$. Then, if an honest party p_i is one of the $n - t_o$ parties that flipped the coin, its vote set V'_{R-1} is unanimous for $\perp$. Therefore p_i will, once the coin is flipped, not cast a round-R prevote for $1 - c$. This means there are at least $n - t_o - t_b$ correct parties that will not cast one for $1 - c$.
Now, a round-R vote for $1 - c$ would require $n - t_o$ round-R prevotes for $1 - c$, but only $t_o + t_b < n - t_o$ such prevotes from distinct parties can exist. By the unpredictability of the coin and the fact that at this time parties only just called $\texttt{CoinFlip}(R)$ and so none has gotten to $\texttt{CoinFlip}(R+1)$, there is a 50% chance that $\texttt{CoinFlip}(R+1)$ is going to be c. Thus, there is a 50% chance that round $R+1$ will be lucky. □

Now, fix t_o, t_b with $3t_b < n$ and $2t_o + t_b < n$. Then we claim $\Pi_{\text{AC+AB}}^{\max(t_o,t_b)}$ obtains binary agreement if at most t_o parties crash and other parties are correct, or at most t_b parties are Byzantine and other parties are correct. As proof, notice that at most $t_b \leq \max(t_o, t_b)$ parties can be dishonest, and $t_b < n - 2\max\{t_o, t_b\}$ whether $\max\{t_o, t_b\}$ is equal to t_o or t_b. By the lemmas above, these conditions on the faults allow the protocol to reach consistent, valid, terminating agreement.
□

Theorem 6 (Feasibility for SB+AO). *There is a consensus protocol tolerating t_{sb} Byzantine faults in synchrony or t_{ao} omission faults in asynchrony if $2t_{sb} < n$ and $2t_{ao} < n$.*

Feasibility is achieved by this protocol $\Pi_{\text{SB+AO}}$, presented from the perspective of party p_i with initial input v_i:

1. **Phase 1:** Run an instance (referred to as the ith instance) of Dolev-Strong (DS) broadcast as the sender, broadcasting v_i.
2. After $(\lceil n/2 \rceil + 1) \cdot \Delta$ time passes, consider all n instances of DS complete. Take $\perp$ to be the default output to the jth instance if the extracted set for that instance is not of size 1 at this time.
3. There is now a sequence of n outputs from the DS instances. Let v be the majority non-$\perp$ output, breaking a tie with 0.
4. **Phase 2:** Run protocol Π_{VAC} (4) with input v.

Let $h = \lceil n/2 \rceil - 1$ be the largest integer less than $n/2$. Π_{VAC} ("valid asynchronous consensus") is the same as $\Pi^h_{\text{AC+AB}}$ above except for these modifications.

- Instead of just waiting for $n - h$ PJ prevotes/votes before moving on, party p_i waits for $n - h$ PJ prevotes/votes *and* Δ time (on its local clock) since it last prevoted/voted before moving on (See lines 4, 7, 15, 21 of refVACProt).
- In round $r \geq 1$, party p_i instead votes for a bit b if it certifies $n - h$ round-r prevotes for b, rather than unanimity (See lines 8, 22 of 4).

We now go through "the long way".

Lemma 8. *(Credibility) In an execution of Π_{VAC}, if a correct party sends a message, all correct parties eventually certify it (or terminate first).*

Proof. The correct party p_i that sends a message does so on the basis of some messages that p_i certifies, which it forwards when it receives them; p_i certifies these on the basis of some other p_i-certified messages which p_i also receives and forwards, and so on. All of these messages eventually arrive at all other non-terminating correct parties, so they all eventually certify all of them.

Lemma 9. *(AC validity) In an execution of Π_{VAC} in which all parties receive input v, if at most h parties experience omissions and all others are correct, then all correct parties output v.*

Proof. There are $\geq n - h$ votes from correct parties for v sent in round-0, and no votes for $1 - v$ since all parties are honest. Therefore, all correct parties receive and certify $\geq n - h$ round-0 votes for v. Once a correct party p_i has waited Δ time, a waiting period that eventually ends, p_i casts a prevote in round 1 for v. Again, no party casts a prevote for $1 - v$. All correct parties certify at least $n - h$ round-1 prevotes for v, so (after another finite-length wait) cast a round-1 vote for v. So all correct parties certify $n - h$ votes for v in round 2. Then they all decide v at the start of round 2 and ignore the coin flip. □

Let F be the first round for which any correct party eventually certifies a termination message marked for round F. The next lemma shows consistency through round F.

Lemma 10. *(Intra-round consistency) In an execution of Π_{VAC}, if up to h parties experience omissions and other parties are correct, then the following hold for any R with $1 \leq R \leq F$:*

- *(a) Every correct party prevotes and votes exactly once in round R.*
- *(b) There is not both a PJ round-R vote for* `0` *and a PJ round-R vote for* `1`*.*

Proof. We induct on R. In round 1, every correct party receives at least $n - h$ round-0 votes, all of which are immediately certified, so every correct party prevotes exactly once. Since at least $n - h$ parties are correct, every correct party receives (and certifies, by 8) enough prevotes to vote, and does so once. Thus, (a) holds in round 1. Next, if there is a PJ round-1 vote for `0` that cites a set of $n - h$ parties and a PJ round-1 vote for `1` that cites a set of $n - h$ parties, these two sets intersect in at least $n - 2h \geq 1$ parties, so some party is in the intersection. However, every party is honest, and no honest party prevotes for both 0 and 1, by (a). Thus, (b) holds for $R = 1$.
Now let $R > 1$ and suppose (a) and (b) hold for round $R-1$. Again, each correct party prevotes at least once, since it eventually certifies enough votes from correct parties even if the incorrect parties fail to send messages. Note that this relies on a correct party not terminating, and none does at this round: terminating based on round-F votes only happens after round F, while terminating based on votes from a round $< F$ would lead to a termination message being sent and certified unless all correct parties terminate in which case no round-F message would be PJ. Correct parties do not prevote twice, because that would require PJ round-$R - 1$ votes for both 0 and 1, which cannot exist by the inductive hypothesis. This shows point (a). Point (b) holds by the same reasoning as given above for round 1. □

The arguments above do not work in rounds $\geq F+1$. This is because termination messages are only justified by votes from one round, but can be used in later rounds as both votes and prevotes; as such, the overlapping argument above does not hold. The following lemma shows, however, that in rounds after F the behavior of Π_{VAC} is even more constrained.

Lemma 11. *(Consistency) In an execution of Π_{VAC} in which any correct party certifies a termination message m for round F, all correct parties output b by the end of round $F + 2$.*

Proof. First, note that round-F votes cannot justify both a termination message for 0 and for 1, by Lemma 10, which holds in round F. Next, a termination message must be justified by at least $n - h$ round-F votes, which overlaps with any set of $n - h$ round-F votes by at least $n - 2h \geq 1$ votes; thus, every correct and yet-to-terminate party certifies, at the start of round $F+1$, at least one vote cited by m. Since all parties are honest, and honest parties do not equivocate,

such a vote must be for b. As such, each correct party casts a prevote (possibly through a termination message) for b in round $F+1$. Also, no correct party justifies any prevote for $1-b$ since justifying one necessitates one of

- A vote for $1-b$. This is impossible by Lemma 10.
- $n-h$ PJ $\perp$ votes from distinct parties. This is impossible because there are at least $n-h$ honest parties that vote for b, so at most $h < n-h$ votes for $\perp$.

It follows that all correct parties vote for b in round $F+1$ (possibly through a termination message), so all remaining correct parties output b at the start of round $F+2$. □

We now show that parties are unlikely to continue through infinitely many rounds.
Following [Blockchain Book], let a round $R \geq 2$ be *lucky* if there does not and will never exist a PJ round-$(R-1)$ vote for $1-\texttt{CoinFlip}(R)$. In other words, the coin does not conflict with any PJ previous-round vote, so they are all for the coin's value or for $\perp$.

Lemma 12. *In an execution of Π_{VAC} in which at most h parties experience omissions and all others are correct, if R is lucky then all parties terminate by round $R+2$.*

Proof. Let $c = \texttt{CoinFlip}(R)$. In this case, all correct parties cast round-R prevotes for c, either justified by at least one round-$(R-1)$ vote for c, or by $n-h$ round-$(R-1)$ votes for $\perp$ (and based on the coin). Furthermore, no party will certify a prevote for $1-c$ because the necessary justification cannot exist. Thus, every correct party eventually certifies $n-h$ prevotes for c, and casts a round-R vote for c. In round $R+1$, then, there are $n-h$ votes for c (and certainly at most $h < n-h$ votes for $1-c$, though in fact there cannot be any), so all correct parties output c at the start of round $R+1$. □

Lemma 13. *Fix a round $R \geq 2$. The probability that at least one of the rounds $R, R+1$ is lucky is at least 50%, no matter the events of earlier rounds.*

Proof. There are two possibilities, both of which lead to one of these rounds being lucky with 50% probability.
Case 1: At the time the round-R coin is flipped, when the adversary learns its value c, some correct party has already certified a vote for a bit $b \in \{0,1\}$. Then, by the unpredictability of the coin, there is a 50% chance $c = b$, making R lucky.
Case 2: At the time the round-R coin is flipped, all votes that have been received and certified are for $\perp$. Then, if an honest party p_i is one of the $n-h$ parties that flipped the coin, its vote set V'_{R-1} is unanimous for $\perp$. Therefore p_i will, once the coin is flipped, not cast a round-R prevote for $1-c$. Thus, there are at least $n-h$ correct parties that will not cast a prevote for $1-c$.
Now, a round-R vote for $1-c$ would require $n-h$ round-R prevotes for $1-c$, but only $h < n-h$ such prevotes from distinct parties can exist. By the

unpredictability of the coin and the fact that at this time parties only just called $\texttt{CoinFlip}(R)$ and so none has gotten to $\texttt{CoinFlip}(R+1)$, there is a 50% chance that $\texttt{CoinFlip}(R+1)$ is going to be c. Thus, there is a 50% chance that round $R+1$ will be lucky. □

Thus ends "the long way".

Lemma 14. *(Validity) In an synchronous execution of Π_{VAC} in which less than $n/2$ parties are faulty, and all honest parties receive input v, all correct parties output v by the end of round 2.*

Proof. All correct parties prevote v in round 0. At the start of round 1, all correct parties receive all others' prevotes, because they wait an additional Δ time. Since $< n/2$ parties are Byzantine, all correct parties prevote v in round 1. Again, all correct parties receive all of these prevotes within Δ waiting. Also, all of these prevotes will be justified by a set containing every correct party, so each correct party will certify all of these prevotes, since it will have received messages from those $\geq n - h$ correct parties implying a majority for v. Therefore, in the second step of round 1 all correct parties have sufficiently many PJ prevotes for v to vote for v. Furthermore, casting a PJ vote for $1-v$ requires $n-h > n/2$ distinct parties to cast a round-1 prevote for $1-v$, but no honest party casts one, so this is impossible; as such, no Byzantine party can cast a PJ vote for $1-v$. Thus, at the start of round 2 all correct parties receive (due to synchrony) and certify (due to honest-majority prevote unanimity for v) more than $n/2$ votes for v and no votes for $1-v$. At this point all correct parties decide v and terminate. □

Theorem 7. *Π_{SB+AO} achieves termination with probability 1, consistency, and validity in a synchronous network with $< n/2$ Byzantine faults, and the same in an asynchronous network with $< n/2$ omission faults.*

Proof. Suppose the network is synchronous. The unconditional consistency of the Dolev-Strong protocol in synchrony shows that all correct parties end the first phase with the same sequence of n bits. Thus, all correct parties begin the second phase with the same bit as input. Since Π_{VAC} is valid under $< n/2$ Byzantine faults in synchrony, consistency is obtained, in addition to termination by 14. Also, if all correct parties begin with the same bit v, then by validity of DS, every correct party ends the first phase with $> n/2$ of the elements of its output sequence equal to v. Therefore, all correct parties enter the second phase with input v, and all decide v by validity of Π_{VAC}.

Otherwise, the network is asynchronous. Each party eventually exits the first phase, and by 6,7 the second phase has termination with probability 1 under $< n/2$ honest faults in asynchrony. Thus, Π_{VAC} achieves termination with probability 1.
Because Π_{VAC} is consistent in the presence of $< n/2$ omission faults even in asynchrony, consistency of $\Pi_{\text{SB+AO}}$ is guaranteed. Also, if every correct party begins with input v, each must output $\perp$ or v from every instance of DS, as no message will ever be for bit $1-v$ when all parties are honest and start with input v. Furthermore, each party must output v from its own instance of DS (the adversary cannot block messages from a party to itself). Therefore, the majority non-$\perp$ output will be v. So the validity of Π_{VAC} in asynchrony implies $\Pi_{\text{SB+AO}}$ also achieves validity in asynchrony. □

Figure 2 shows the above results graphically. Figure 3 shows many of the following theorems as model pairs sandwiched between an more restrictive model pair with an impossibility result, and a less restrictive model pair with a matching feasibility result.

Theorem 8 (Trapezoid Pairings). *There is a protocol tolerating each of these model pairs:*

1. $SC(t_1) + PB(t_2)$
2. $SC(t_1) + AB(t_2)$
3. $SO(t_1) + PB(t_2)$
4. $SO(t_1) + AB(t_2)$
5. $SB(t_1) + PB(t_2)$
6. $SB(t_1) + AB(t_2)$

if and only if $3t_2 < n$ *and* $2t_1 + t_2 < n$.

Proof. Impossibility holds by 3 and the fact that all of these models are at most as restrictive as $\mathrm{SC}(t_1) + \mathrm{PB}(t_2)$. Feasibility holds by 4 and the fact that all of these models are at least as restrictive as $\mathrm{SB}(t_1) + \mathrm{AB}(t_2)$. □

Theorem 9 (More Trapezoid Pairings). *There is a protocol tolerating one of these model pairs:*

1. $PC(t_1) + PB(t_2)$
2. $PC(t_1) + AB(t_2)$
3. $PO(t_1) + PB(t_2)$
4. $PO(t_1) + AB(t_2)$
5. $AC(t_1) + PB(t_2)$
6. $AC(t_1) + AB(t_2)$
7. $AO(t_1) + PB(t_2)$
8. $AO(t_1) + AB(t_2)$

if and only if $3t_2 < n$ *and* $2t_1 + t_2 < n$.

Proof. Impossibility holds by 3 and the fact that all of these models are at most as restrictive as $\mathrm{SC}(t_1) + \mathrm{PB}(t_2)$. Feasibility holds by 6 and the fact that all of these models are at least as restrictive as $\mathrm{AO}(t_1) + \mathrm{AB}(t_2)$. □

Theorem 10 ($n/2$ Square Pairings). *There is a protocol tolerating one of these model pairs:*

1. $SC(t_1) + PC(t_2)$
2. $SO(t_1) + PC(t_2)$
3. $SB(t_1) + PC(t_2)$
4. $SC(t_1) + PO(t_2)$
5. $SO(t_1) + PO(t_2)$
6. $SB(t_1) + PO(t_2)$
7. $SC(t_1) + AC(t_2)$

8. $SO(t_1) + AC(t_2)$
9. $SB(t_1) + AC(t_2)$
10. $SC(t_1) + AO(t_2)$
11. $SO(t_1) + AO(t_2)$
12. $SB(t_1) + AO(t_2)$

if and only if $2t_1 < n$ *and* $2t_2 < n$.

Proof. Impossibility holds by 1 and the fact that all of these models are at most as restrictive as $\mathrm{SC}(t_1) + \mathrm{PC}(t_2)$. Feasibility holds by 6 and the fact that all of these models are at least as restrictive as $\mathrm{AO}(t_1) + \mathrm{AB}(t_2)$. □

Theorem 11 (Trivial $n/2$ Square Pairings). *There is a protocol tolerating one of these model pairs:*

1. $PC(t_1) + PO(t_2)$
2. $PC(t_1) + AO(t_2)$
3. $AC(t_1) + PO(t_2)$
4. $AC(t_1) + AO(t_2)$
5. $PC(t_1) + AC(t_2)$
6. $PO(t_1) + AO(t_2)$
7. $SO(t_1) + SB(t_2)$

if and only if $2t_1 < n$ *and* $2t_2 < n$.

Proof. Feasibility for items 1 through 6 can be achieved by running the protocol of [3] that tolerates $< n/2$ omission faults in asynchrony, as explained above in the section on classical results. This is because each model family in each pair is at least as restrictive as AO. Meanwhile, feasibility for item 7 is achieved by running a protocol that tolerates $< n/2$ Byzantine faults in synchrony, e.g. having each party run an instance of DS broadcast as the sender and take the majority output.

Impossibility for all items is inherited from impossibility results for each network in the pair. □

Theorem 12 (Worst case). *There is a protocol tolerating* $PB(t_1) + AB(t_2)$ *if and only if* $3t_1 < n$ *and* $3t_2 < n$.

Proof. Feasibility is achieved by running any protocol that tolerates $< n/3$ Byzantine faults in asynchrony. Impossibility is inherited from impossibility results for each network in the pair. □

Theorem 13 (SC+SO). *There is a protocol tolerating* $SC(t_1) + SO(t_2)$ *if and only if* $t_1 < n$ *and* $t_2 < n$.

Proof. Feasibility holds by 2 and the fact that $SC(t_1)+SO(t_1)$ is more restrictive than $SC(t_1) + SB(t_1)$. Impossibility is inherited from impossibility results for each network in the pair. □

6 Conclusion

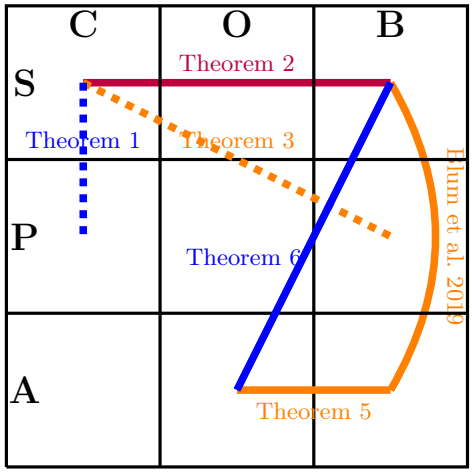

Fig. 2. Impossibility theorems are shown as dashed lines, while feasibility theorems are shown as solid lines. Lines of the same color bind faults in the same way.

7 Future Work

The most natural extension is to k-agnostic protocols that must function correctly under any of $k > 2$ possible models without advance knowledge of which applies. For example, one could imagine deriving bounds similarly for 3-agnostic models, where protocols must succeed under any of three specified network-fault combinations. This would further show how fault tolerance degrades as uncertainty increases across multiple execution environments.

Our techniques for establishing tight bounds should extend to multivalued consensus, reliable broadcast, and state machine replication. The fundamental trade-offs between uncertainty and fault tolerance we identify likely apply broadly across distributed agreement problems, though the specific bounds will differ.

Following Momose and Ren's work [14], separating safety and liveness thresholds in the network-agnostic setting could give more nuanced protocols still. Rather than requiring both properties under the same fault bound, protocols might guarantee consistency under threshold β and termination under threshold γ for each model.

Combining our approach with granular synchrony [11], where individual communication links have different timing guarantees, could capture more realistic network uncertainty. Protocols might need to handle both overall network-model

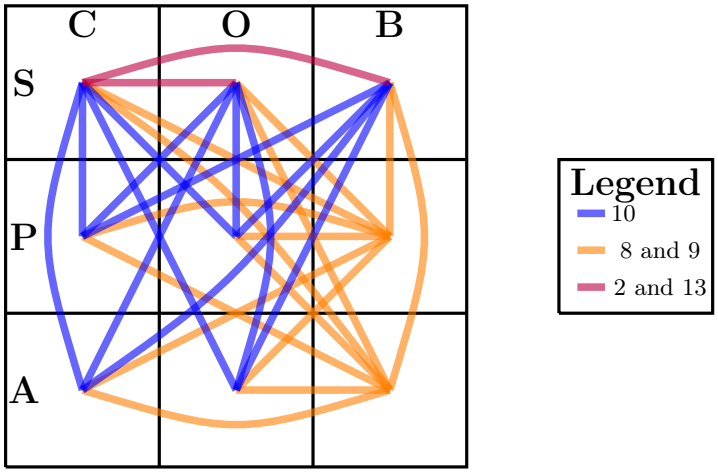

Fig. 3. Results of Theorems 2, 8, 9, 10, 13, essentially obtained by interpolating between comparable models in Figure 2.

uncertainty and link-level timing heterogeneity simultaneously.

Our theoretical bounds may benefit from validation through physical implementations. Key related questions also include the computational overhead of network-agnostic protocols and their performance in real distributed systems.

8 Acknowledgments

References

1. Abraham, I., Ben-David, N., Yandamuri, S.: Efficient and adaptively secure asynchronous binary agreement via binding crusader agreement. In: Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing. pp. 381–391 (2022)
2. Abraham, I., Dolev, D., Eyal, I., Halpern, J.Y.: Distributed computing meets game theory: Robust mechanisms for rational secret sharing and multiparty computation. In: Proceedings of the 40th ACM Symposium on Principles of Distributed Computing. pp. 81–91 (2021)
3. Ben-Or, M.: Another advantage of free choice: Completely asynchronous agreement protocols. In: Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing. pp. 27–30 (1983)
4. Blum, E., Katz, J., Loss, J.: Synchronous consensus with optimal asynchronous fallback guarantees. In: Theory of Cryptography Conference. pp. 131–150 (2019)
5. Cachin, C., Kursawe, K., Shvartsman, V.: Optimistic asynchronous byzantine agreement. In: Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems. pp. 204–213 (2005)
6. Clement, A., Kapritsos, M., Lee, S., Wang, Y., Alvisi, L., Dahlin, M., Riche, T.: Upright cluster services. In: Proceedings of the 22nd ACM Symposium on Operating Systems Principles. pp. 277–290 (2009)
7. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. SIAM Journal on Computing **12**(4), 656–666 (1983). https://doi.org/10.1137/0212045, https://doi.org/10.1137/0212045
8. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. SIAM Journal on Computing **12**(4), 656–666 (1983)
9. Dwork, C., Lynch, N.A., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM **35**(2), 288–323 (1988)
10. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM **32**(2), 374–382 (1985)
11. Giridharan, N., Abraham, I., Crooks, N., Howard, H.: Granular synchrony and its application to flexible bft consensus. In: Proceedings of the 2024 USENIX Annual Technical Conference. pp. 123–138 (2024)
12. Lamport, L.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems **4**(3), 382–401 (1982)
13. Lamport, L.: The part-time parliament. ACM Transactions on Computer Systems **16**(2), 133–169 (1998)
14. Momose, A., Ren, L.: Multi-threshold byzantine fault tolerance. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1686–1699 (2021)

Algorithm 3: $\Pi^{t_o}_{\text{AC+AB}}$, from the perspective of party i with input $v_i \in \{0,1\}$

Round 0**.**
- Multicast $\langle \text{vote}, r = 0, v_i \rangle$.

Round 1**.**
- **Upon** *certifying a set* V'_0 *of* $n - t_o$ *round-*0 *votes signed by* $V_0 = \{n - t_o \text{ parties}\}$**:**
  - $b \leftarrow$ the majority bit in V'_0 (breaking a tie with 0).
  - Multicast $\langle \text{prevote}, r = 1, b, V_0 \rangle_i$.
- **Upon** *certifying a set* U'_1 *of* $n - t_o$ *round-*1 *prevotes signed by* $U_1 = \{n - t_o \text{ parties}\}$**:**
  - **if** *all prevotes in* U'_1 *are for some* b' **then**
    - $v \leftarrow b'$.
  - **else**
    - $v \leftarrow \perp$.
  - Multicast $\langle \text{vote}, r = 1, v, U_1 \rangle_i$.

Round R*, with* $2 \leq R$**.**
- Complete all earlier rounds first.
- **Upon** *certifying a set* V'_{R-1} *of* $n - t_o$ *round-*$R-1$ *votes signed by* $V_{R-1} = \{n - t_o \text{ parties}\}$**:**
  - $c \leftarrow$ `CoinFlip(`R`)`.
  - **if** V'_{R-1} *contains a vote for a bit* $b \in \{0,1\}$ **then**
    - Multicast $\langle \text{prevote}, r = R, b, V_{R-1} \rangle_i$.
  - **else** (if V'_{R-1} is unanimous for $\perp$)
    - Multicast $\langle \text{prevote}, r = R, c, V_{R-1} \rangle_i$.
- **Upon** *certifying a set* U'_R *of* $n - t_o$ *round-*R *prevotes signed by* $U_R = \{n - t_o \text{ parties}\}$**:**
  - **if** *all prevotes in* U'_R *are for some* b' **then**
    - $v \leftarrow b'$.
  - **else**
    - $v \leftarrow \perp$.
  - Multicast $\langle \text{vote}, r = R, v, U_R \rangle_i$.

Upon *there existing a round* $R \geq 1$ *and a bit* $b \in \{0,1\}$ *such that there are at least* $n - t_o$ *certified votes from round* R *on bit* b*, signed by a set* T_R *of* $n - t_o$ *distinct parties***:**
- Complete execution as above for all rounds up to and including R.
- Multicast $\langle \text{terminate}, r = R, b, T_R \rangle_i$.
- Output b and terminate.

Algorithm 4: Π_{VAC}, from the perspective of party i with input $v_i \in \{0,1\}$

Round 0**.**
- Multicast $\langle \text{vote}, r = 0, v_i \rangle$.

Round 1**.**
- **Upon** *certifying a set* V_0' *of* $n-h$ *round-*0 *votes signed by* $V_0 = \{n-h \text{ parties}\}$ **and** Δ **time has passed since last vote/prevote:**
  - $b \leftarrow$ the majority bit in V_0' (breaking a tie with 0).
  - Multicast $\langle \text{prevote}, r = 1, b, V_0 \rangle_i$.
- **Upon** *certifying a set* U_1' *of* $n-h$ *round-*1 *prevotes signed by* $U_1 = \{n-h \text{ parties}\}$ **and** Δ **time has passed since last vote/prevote:**
  - **if** $n-h$ *prevotes in* U_1' *are for some* b' **then**
    - $v \leftarrow b'$.
  - **else**
    - $v \leftarrow \perp$.
  - Multicast $\langle \text{vote}, r = 1, v, U_1 \rangle_i$.

Round R, *with* $2 \leq R$**.**
- Complete all earlier rounds first.
- **Upon** *certifying a set* V_{R-1}' *of* $n-h$ *round-*$R-1$ *votes signed by* $V_{R-1} = \{n-h \text{ parties}\}$ **and** Δ **time has passed since last vote/prevote:**
  - $c \leftarrow$ `CoinFlip(`R`)`.
  - **if** V_{R-1}' *contains a vote for a bit* $b \in \{0,1\}$ **then**
    - Multicast $\langle \text{prevote}, r = R, b, V_{R-1} \rangle_i$.
  - **else** (if V_{R-1}' is unanimous for $\perp$)
    - Multicast $\langle \text{prevote}, r = R, c, V_{R-1} \rangle_i$.
- **Upon** *certifying a set* U_R' *of* $n-h$ *round-*R *prevotes signed by* $U_R = \{n-h \text{ parties}\}$ **and** Δ **time has passed since last vote/prevote:**
  - **if** $n-h$ *prevotes in* U_R' *are for some* b' **then**
    - $v \leftarrow b'$.
  - **else**
    - $v \leftarrow \perp$.
  - Multicast $\langle \text{vote}, r = R, v, U_R \rangle_i$.

Upon *there existing a round* $R \geq 1$ *and a bit* $b \in \{0,1\}$ *such that there are at least* $n-h$ *certified votes from round* R *on bit* b*, signed by a set* T_R *of* $n-h$ *distinct parties***:**
- Complete execution as above for all rounds up to and including R.
- Multicast $\langle \text{terminate}, r = R, b, T_R \rangle_i$.
- Output b and terminate.