Tight Bounds for Shared Memory Systems Accessed by Byzantine Processes

(Extended Abstract)

Michael Merritt[1], Omer Reingold[1], Gadi Taubenfeld[1,2], and Rebecca N. Wright[3]

[1] AT&T Labs, 180 Park Ave., Florham Park, NJ 07932 USA
[2] The Open University and the Interdisciplinary Center, Israel
[3] Computer Science Dept., Stevens Institute of Technology, Hoboken, NJ 07030 USA

Abstract. We provide efficient constructions and tight bounds for shared memory systems accessed by n processes, up to t of which may exhibit Byzantine faults, in a model previously explored by Malkhi et al. [MMRT00]. We show that sticky bits are universal in the Byzantine failure model for $n \geq 3t+1$, an improvement over the previous result requiring $n \geq (2t+1)(t+1)$. Our result follows from a new strong consensus construction that uses sticky bits and tolerates t Byzantine failures among n processes for any $n \geq 3t+1$, the best possible bound on n for strong consensus. We also present tight bounds on the efficiency of implementations of strong consensus objects from sticky bits and similar primitive objects.

1 Introduction

Although Byzantine fault tolerance in message-passing systems has been extensively investigated, it was only recently that Malkhi et al. initiated the study of Byzantine fault tolerance in asynchronous shared memory systems [MMRT00]. Their work establishes a formal model and shows how the use of access control lists (ACLs) can constrain Byzantine behavior and permit reliable distributed computation. They investigate *universal* objects, which can be used to implement any shared object. Specifically, they show that sticky bits, a simple shared memory primitive long known to be universal in the crash failure model [Plo89], are also universal in the Byzantine failure model, provided that $n \geq (2t+1)(t+1)$, where n is the number of processes and t bounds the number of processes that may fail (exhibiting unconstrained, or Byzantine, behavior). One of the main results of this paper is to strengthen their result, showing that sticky bits are universal in the Byzantine failure model for any $n \geq 3t+1$.

The universality results of [MMRT00] first use constructions from sticky bits to build strong consensus objects, then use strong consensus objects in an explicit universal construction of an arbitrary shared object. (Definitions of sticky bits, weak and strong consensus, and other objects mentioned in this introduction are provided in Section 2.) The bound $n \geq (2t+1)(t+1)$ follows from the

In *Proceedings of 16th International Symposium on Distributed Computing (DISC)*, 2002.

construction in the first step, building strong consensus from sticky bits. In this paper, we present a novel construction of strong consensus from sticky bits for any $n \geq 3t+1$. The consequence for universality is immediate: "Constructions of strong consensus from sticky bits for larger values of t would imply a more resilient universality result." [MMRT00]. Malkhi et al. demonstrate that strong consensus objects can only exist if $n \geq 3t+1$, so our result is the best possible unless a different universal construction is used. Beyond strengthening the universality result for sticky bits, we present tight bounds on the efficiency of implementations of strong consensus objects from sticky bits and similar "powerful" shared objects. (Powerful objects include sticky bits that can be set by more than one process, but exclude registers and single-writer sticky bits. "Powerful" operations are those other than wait-free reads and writes.)

In Section 2, we review the model and definitions. Section 3 presents a general protocol schema that can be used to implement strong consensus from sticky bits. Instantiated separately for $n = 3t+1$ and $n = (t+1)^2$, the schema results in two strong consensus algorithms, which use $\binom{2t+1}{t}$ and $t+1$ powerful sticky bits, correspondingly. The contrast in efficiency of these constructions is striking: exponentially many powerful objects for $n = 3t+1$ and just $t+1$ for $n = (t+1)^2$. We then modify the latter protocol to show that t powerful sticky bits are sufficient, provided $n \geq t^2 + 5t + 1$. Surprisingly, this algorithm works despite the fact that all of the t powerful shared objects could be written by t Byzantine processes.

In Section 4, we demonstrate bounds and tradeoffs among n, t, and the number and type of objects used. We show that the protocols of Section 3 are essentially optimal in the number of powerful shared objects used. In Section 4.1, we present a general lower bound that constrains the types of powerful shared objects and associated access control lists required to implement even weak consensus. Sections 4.2 and 4.3 investigate the tradeoff on the number of powerful shared objects that are necessary and sufficient for constructing strong consensus objects as n increases relative to t. The sufficiency results are constructive, providing explicit instantiations of the protocol schema of Section 3.

To summarize the main contributions of this paper, we present:

- A t-resilient strong consensus protocol (and therefore a universality result) from sticky bits, for $n \geq 3t+1$.
- A t-resilient strong consensus protocol using only t powerful shared objects. (Surprising since t Byzantine processes can access all the powerful shared objects.)
- A proof that at least t shared objects (such as sticky bits) must be used in t-resilient strong consensus protocols.
- A proof that any weak consensus protocol that tolerates t crash failures must use at least one object on which at least $t+1$ processes can invoke powerful operations.
- A tight tradeoff characterizing the number of a kind of powerful shared object that must be used and how big n must be as a function of t. Specifically, we show that the number k of powerful shared objects needed to implement consensus is essentially $k = t \cdot 2^{\Theta(t^2/n)}$.

– A polynomial-time strong consensus protocol (and therefore universality result) for $n = O(t^2/\log t)$, an improvement over the previous $n = \Omega(t^2)$.

2 Model and definitions

The model of computation we consider was introduced by Malkhi et al., and parts of this section are adapted from [MMRT00]. This model consists of an asynchronous collection of n processes, denoted $p_1, \ldots, p_n$, that communicate via shared objects. Wait-free shared memory fault models assume no bound on the number of potentially faulty processes—each operation by a process p on a shared object must terminate, regardless of the concurrent actions of other processes. Following [MMRT00], this model differs in two ways: we make the more pessimistic assumption that process faults are Byzantine, and we make the more optimistic assumption that the number of faults is bounded by t, where t is less than the total number n of processes. In any run any process may be either *correct* or *faulty.* Correct processes are constrained to obey their specifications. A faulty processes can either crash or behave in a Byzantine way. A process that follows its protocol up to a certain point and then stops sending messages or stops accessing shared objects is called a *crashed process* or a *crash fault*. A process that deviates from its protocol either by crashing or by performing incorrect operations is called a *Byzantine process* or a *Byzantine fault*. We generally use t to denote the maximum number of faulty processes. Whenever we discuss faulty processes we identify the type of fault assumed.

2.1 Shared objects with access control lists

Each shared object presents a set of operations. For example, $x.\mathsf{op}$ denotes operation op on object x. For each such operation $x.\mathsf{op}$ on x, there is an associated access control list, denoted $\mathrm{ACL}(x.\mathsf{op})$, which is the set of processes allowed to *invoke* that operation. Each operation execution begins with an invocation by a process in the operation's ACL, and remains pending until a response is received by the invoking process. The ACLs for two different operations on the same object can differ, as can the ACLs for the same operation on two different objects. The ACLs for an object do not change. For any operation $x.\mathsf{op}$, we say that x is k-op if $|\mathrm{ACL}(x.\mathsf{op})| = k$. A process not in the ACL for $x.\mathsf{op}$ cannot invoke $x.\mathsf{op}$, regardless of whether the process is correct or Byzantine (faulty). That is, a (correct or faulty) process cannot access an object in any way except via the operations for which it appears in the associated ACLs. Byzantine faulty processes can, for example, write different values than their specifications suggest, or refuse to invoke an operation they are supposed to invoke, but they remain constrained against invoking operations for which they are not in the ACL.

Many abstract objects support read operations: operations that return information about the state of the object, without constraining its future behavior (see [Her91]). In this paper, we assume the primitive objects (registers and sticky bits) support wait-free read operations by all processes, and focus on the ACLs for non-read operations. Atomic registers (and some other abstract objects) are *historyless* [FHS98] in that they support only read operations and operations such as wait-free write() operations: operations that do not return a value, and that constrain future object behavior independently of the state in which they are invoked. These operations have long been known to be weak synchronization primitives [LA87,Her91]. Accordingly, we define an operation to be *powerful* if it is neither a wait-free read nor a wait-free write() operation, and for an object (or object type) x, we define $\mathrm{ACL}_{pow}(x)$ to be the union of $\mathrm{ACL}(x.\mathsf{op})$ for all powerful operations $x.\mathsf{op}$ of x. Moreover, we call an object (or object type) x *powerful* if $\mathrm{ACL}_{pow}(x) \geq 2$. That is, powerful objects support powerful operations by at least two different processes. Thus, neither registers nor sticky bits (defined below) writable by only one process are powerful, while sticky bits writable by more than one process are powerful.

2.2 Object definitions

Next, we define some of the types of object used in this paper.

Atomic registers: An atomic register x is an object with two operations: $x.\mathsf{read}$ and $x.\mathsf{write}(v)$ where $v \neq \perp$. An $x.\mathsf{read}$ that occurs before the first $x.\mathsf{write}()$ returns $\perp$. An $x.\mathsf{read}$ that occurs after an $x.\mathsf{write}()$ returns the value written in the last preceding $x.\mathsf{write}()$ operation.

Sticky bits: A sticky bit x is an object with two operations: $x.\mathsf{read}$ and $x.\mathsf{set}(v)$ where $v \in \{0, 1\}$. An $x.\mathsf{read}$ that occurs before the first $x.\mathsf{set}()$ returns $\perp$. An $x.\mathsf{read}$ that occurs after an $x.\mathsf{set}()$ returns the value written in the first $x.\mathsf{set}()$ operation. We will be concerned with wait-free sticky bits. (To highlight the specific semantics of the $x.\mathsf{set}()$ operation and to distinguish it from the write() of atomic registers, we depart from previous work [Plo89,MMRT00], which uses write() to denote both operations.)

Weak consensus objects: [MMRT00] A weak (binary) consensus object x is an object with one operation: $x.\mathsf{propose}(v)$, where $v \in \{0, 1\}$, satisfying: (1) In any run, the $x.\mathsf{propose}()$ operation returns the same value, called the *consensus value*, to every correct process that invokes it. (2) In any finite run in which all participating processes are correct (no Byzantine faults), if the consensus value is v, then some process invoked $x.\mathsf{propose}(v)$.

Strong consensus objects: [MMRT00] A strong (binary) consensus object x strengthens the second condition above to read: (2) If the consensus value is v, then some *correct* process invoked $x.\mathsf{propose}(v)$. [1]

As indicated in [MMRT00], strong consensus objects do not have sequential runs: the additional condition, using redundancy to mask failures, requires at least $t+1$ processes to invoke $x.\mathsf{propose}()$ before any correct process returns from this operation. It is also shown in [MMRT00] that the notion of strong consensus objects that can tolerate t Byzantine faults is ill-defined unless $n \geq 3t + 1$. To

[1] Note that in the binary case, this definition of strong consensus coincides with one that only requires the consensus value to be v when all correct processes have the same input. In the nonbinary case, this is a strictly stronger definition.

accommodate such a need to wait for other processes before progress can be made in specific constructions, the following properties were introduced:

For any operation $x.\mathsf{op}$, we say that $x.\mathsf{op}$ can *tolerate* t faults if $x.\mathsf{op}$, when executed by a correct process, eventually completes in any run ρ in which at least $n - t$ correct processes invoke $x.\mathsf{op}$.

t-resilience: For any operation $x.\mathsf{op}$, we say $x.\mathsf{op}$ is t-resilient if $x.\mathsf{op}$, when executed by a correct process, eventually completes in any run in which each of at least $n - t$ correct processes infinitely often has a pending invocation of $x.\mathsf{op}$.

As in [MMRT00], we use wait-free sticky bits to implement strong consensus objects that tolerate t Byzantine faults. These are in turn used to implement arbitrary t-resilient objects. (Throughout, atomic registers, sticky bits and any other primitive objects are assumed to be wait-free.)

3 Efficient strong consensus protocols

In this section, we present strong consensus protocols based on an idea of Berman and Garay [BG89,Mis89] for performing consensus in the message passing model. Their idea was to run a protocol in phases, where each phase preserves agreement, and if coordinated by a correct process, assures validity. Concatenating $t + 1$ phases guarantees at least one is coordinated by a correct process.

We first show how to use sticky bits to implement a protocol phase with similar properties for the shared memory model. We then show how several strong consensus protocols with different desirable properties can be constructed from these protocol phases. Our protocols are easily modifiable to implement nonbinary consensus.

3.1 A protocol phase

A protocol phase makes use of two types of shared objects. First, a phase uses n *personal* sticky bits, $s_i : p_i \in P$, (where P is the set of all processes, $|P| = n$). Each s_i is writable only by the single process p_i. Second, a phase also uses a single powerful sticky bit, S, which is writable by some set of $t + 1$ processes. We call the $t + 1$ processes in $\mathrm{ACL}(s_i.\mathsf{set}())$ *active* in the phase. Each process p_i enters the phase with a proposed consensus value in_i and leaves with the output value out_i.

Operation of a protocol phase, for each process $p_i \in P$:

1. Perform the wait-free $s_i.\mathsf{set}(in_i)$ operation. (That is, assign in_i to the personal sticky bit s_i.)
2. Perform wait-free reads of the personal sticky bits $s_1 \dots s_n$ until seeing at least $t + 1$ distinct occurrences of some value v other than $\perp$.
3. If p_i has write access to S, then perform the wait-free $S.\mathsf{set}(v)$ operation. (Of course, the first such scheduled process succeeds; the value of S does not change after that.)

4. Perform wait-free reads of S until returning a value val other than $\bot$.
5. Perform wait-free reads of the personal sticky bits $s_1 \ldots, s_n$ until at least $n - t$ return with values other than $\bot$. If val occurs in at least $t + 1$ of the values read, return $out_i = val$ (and say that p_i *supports* val in this phase), else return $out_i = \overline{val}$.

Lemma 1. *A protocol phase has the following properties (given $n \geq 3t + 1$ and the bound t on the number of Byzantine processes):*

1. *If at least $n-t$ correct processes enter a phase, all correct processes eventually exit it.*
2. *The output value of any correct process is the input value of some correct process.*
3. *If all the active processes are correct, all correct processes exit the phase with the same value v.*

Proof. 1. The first and third steps are wait-free. Once $n - t$ correct processes finish the first step, since $n - t \geq 2t + 1$, some value must occur at least $t + 1$ times, and the second and fifth steps must eventually terminate. The fourth step of all processes must terminate once a single correct active process executes the fourth step, and this must eventually happen because there are $t + 1$ active processes.

2. This follows because the output value of any correct process appears in at least $t + 1$ personal sticky bits, so one must have been set by a correct process.

3. If all the active processes are correct, then the value that S is set to will be supported by every correct process. (Note that this property implies that if all correct processes enter the phase with the same input value, then all correct processes exit the phase with the same output value.) ■

Now consider a consensus protocol, *Schema*, constructed by chaining finitely many separate protocol phases (with different sticky bits) together in a fixed sequence. Each process enters the first phase with its proposed value, uses the value returned from each phase as the input to the next phase, and returns as the protocol output the value returned from the last phase.

Lemma 2. *Given $n \geq 3t + 1$ and the bound t on the number of Byzantine processes, if any phase has only correct active processes, Schema implements strong consensus.*

Proof. Part 1 of Lemma 1 guarantees the correct processes eventually exit each phase and so *Schema*. Part 2 guarantees the correct processes enter each phase with a valid input. The assumption and part 3 guarantee the correct processes eventually agree in a phase, and again part 2 guarantees the correct processes do not change their value thereafter. ■

3.2 Strong consensus protocols

We present three protocols that use the protocol phases of Section 3.1 to implement strong consensus. The first protocol (Theorem 1) works for any $n \geq 3t+1$, but requires exponentially many powerful objects. Since Malkhi et al. [MMRT00] show that $n \geq 3t+1$ is necessary for strong consensus in this model, our protocol is optimal in terms of the ratio between t and n. The second protocol (Theorem 2) uses only $t+1$ powerful objects, but requires $n \geq (t+1)^2$. The third protocol (Theorem 3) is surprising because it works even if the faulty processes have access to all the powerful objects. It modifies *Schema* by replacing the requirement that some phase contains only correct active processes by a voting step at the end. It uses only t powerful objects and requires $n \geq t^2 + 5t + 1$.

Theorem 1. *A strong consensus object tolerating t Byzantine faults can be implemented using $(t+1)$-$\mathsf{set}()$, n-read sticky bits and 1-$\mathsf{set}()$, n-read sticky bits, provided that $n \geq 3t+1$.*

Proof. Let P' be a subset of P with $2t+1$ processes. The protocol consists of $\binom{2t+1}{t}$ phases, each following the other, where the active processes in each phase consist of a distinct subset of $t+1$ processes from P'. Since only t processes are faulty, one such phase contains only correct active processes, and the theorem follows from Lemma 2. ■

The implication for universality is immediate from Malkhi et al. [MMRT00]:

Corollary 1. *Any t-resilient object can be implemented using $(t+1)$-$\mathsf{set}()$, n-read sticky bits and 1-$\mathsf{set}()$, n-read sticky bits, provided that $n \geq 3t+1$.*

Though optimal in n and t, the protocol of Theorem 1 is not efficient in time or the number of powerful objects, as it uses a number of rounds and of powerful objects exponential in t. We will show in Section 4 that the space bound is inherent: an exponential number of powerful objects is *required* when $n = 3t+1$.

The following instantiation of *Schema* uses only $t+1$ rounds and $t+1$ powerful objects, but requires $n \geq (t+1)^2$. In this instantiation, a completely new set of $t+1$ active processes is used for each protocol phase:

Theorem 2. *A strong consensus object that can tolerate t Byzantine faults can be implemented using $t+1$ $(t+1)$-$\mathsf{set}()$, n-read sticky bits, together with 1-$\mathsf{set}()$, n-read sticky bits, provided $n \geq (t+1)^2$.*

Proof. There are $t+1$ phases, each with a distinct set of active processes, so for at least one phase all active processes are correct. The result follows by Lemma 2. ■

Theorems 1 and 2 are two extreme points in a tradeoff between the number of powerful objects and the ratio of t to n. We examine this tradeoff more closely in Section 4.2. First, we present a final protocol that is surprising because there are only t powerful objects, and therefore it works even if t Byzantine processes can access *all* the powerful objects.

Theorem 3. *A strong consensus object tolerating t Byzantine faults can be implemented using t powerful* $(t+1)$-$\mathsf{set}()$, n-read *sticky bits, together with* 1-$\mathsf{set}()$, n-read *sticky bits, provided that* $n \geq t^2 + 5t + 1$.

Proof. We modify the protocol of Theorem 2 by omitting the last phase. That is, there are t phases, each involving a distinct set of active processes accessing the $(t+1)$-$\mathsf{set}()$, n-read sticky bit for that phase. We then designate exactly $4t+1$ additional processes (there are at least that many) not active in any phase as *voters*. Note that either some phase contains only correct active processes, or all the voters are correct. Each of the voting processes takes their output from the last phase and writes their resulting value in a personal sticky bit. After the last phase, all processes read the voters' personal sticky bits, and decide on the first value they see occurring $2t+1$ times.

To see that this works, first note that at most one value can occur $2t+1$ or more times among the $4t+1$ voters. We now show one value must occur at least that often, and that the value is valid. By parts (2) and (3) of Lemma 1, if there is a phase in which all the active process are correct, then all correct voters will write the same valid value to their own personal sticky bit. Since in this case at least $3t+1$ voters are correct, eventually at least $3t+1 \geq 2t+1$ votes will be written and will agree on some valid value v. If there is no phase in which all the active processes are correct, then as argued above, no voter is faulty. In this case, all voters will write a valid value, so one value will be written at least $\lceil (4t+1)/2 \rceil = 2t+1$ times. ∎

Since there are only t powerful objects, this algorithm works even in the case that no single powerful object is accessible exclusively by correct processes. In this case, the size, $t+1$, of the ACLs is needed to ensure that each powerful object will eventually be written (and so other processes can wait for it to be set), rather than ensuring that some powerful object will be written by a correct process. Other protocols derived from the protocol *Schema*, such as the algorithm in the proof of Theorem 1, can be similarly modified, removing one protocol phase and replacing it with a set of $4t+1$ voters.

One might expect that such tricks could be used to reduce the number of powerful objects even further. For example, can strong consensus be implemented with even fewer than t powerful sticky bits? Do t sticky bits suffice for $n = 3t+1$? (The t bit algorithm of Theorem 3 requires $n \geq t^2 + 5t + 1$.) We explore these questions in the next section.

4 Lower Bounds and Tradeoffs

We begin this section by showing in Theorem 4 that t is in fact a lower bound, regardless of n, on the number of powerful sticky bits needed for strong consensus. In the ensuing subsections, we examine the general tradeoff of number of sticky bits and n.

Theorem 4. *No t-resilient strong consensus object can be implemented from fewer than t powerful sticky bits (using no other powerful objects).*

Proof. Suppose such a protocol exists. Since it uses at most $t-1$ powerful sticky bits, the protocol must remain 1-resilient even in the case that $t-1$ Byzantine processes first set these bits to 0 before any correct process takes a step, and subsequently take no action. These sticky bits are clearly useless: omitting them and the $t-1$ Byzantine processes results in a 1-resilient protocol with *no* powerful objects that uses registers and single writer sticky bits. Obviously, this protocol is also correct in the crash-fault model (against the failure of a single process). But in the crash model, single writer sticky bits can be implemented by registers, and the resulting protocol contradicts the well-known results of Fischer, Lynch and Paterson [FLP85] and Loui and Abu-Amara [LA87]. ■

4.1 The ACL Theorem

We next prove a general theorem establishing a necessary condition for implementing *weak* consensus even in the presence of only *crash* faults. Inspired by proofs of [FLP85,LA87], we show that when at most t out of n processors may crash, the weak consensus problem is only solvable in shared memory systems containing a powerful object o such that $\mathrm{ACL}_{pow}(o) \geq t+1$. Trivially, a weak consensus object o defined with $|\mathrm{ACL}(o.\mathsf{propose}())| = t+1$ shows this condition is sufficient. Similarly, a sticky bit o with $|\mathrm{ACL}(o.\mathsf{set}())| = t+1$ easily implements a weak consensus object that tolerates up to t Byzantine faults.

Theorem 5. *Any weak consensus protocol that tolerates t crash failures must use at least one object, o, such that* $|ACL_{pow}(o)| \geq t+1$.

Proof (sketch). For the purposes of this proof, we denote a run of a protocol by the sequence in which processes invoke operations. A finite run x is *v-valent* if in all extensions of x where a decision is made, the decision value is v ($v \in \{0,1\}$). A run is *univalent* if it is either 0-valent or 1-valent, otherwise it is *bivalent.* In the following, P denotes a set of processes, x and x' denote runs and $x'p$ is an extension of the run x' by one step of process p.

Assume π is a t-resilient consensus protocol. By familiar arguments, π has an empty bivalent run x_0. We begin with x_0 and pursue the following round-robin *bivalence-preserving scheduling* discipline:

$x := x_0$; $P := \phi$; $i := 0$;
`repeat`
  $j := i + 1$
  `if` x has a bivalent extension $x'p_j$
  `then` $x := x'$
  `else` $P := P + p_j$
  $i := i + 1 (\mathrm{mod}\ n)$
`until` $|P| = t + 1$.

If this procedure does not terminate, then there is an $(n-t)$-fair run with only bivalent finite prefixes. However, the existence of such a run contradicts the definition of t-resilient consensus protocols. Hence, the procedure will terminate with some bivalent finite run x, and a set P of $t+1$ processes such that any extension $x'p$ of x, for any process p in P, is univalent.

Pick any $p \in P$, and let v be such that the run xp is v-valent. Since x is bivalent, there is a (shortest) extension z of x which is $\overline{v}$-valent.

Let z' be the longest prefix of z that does not contain a step of p, and note that either $z = z'$ or $z = z'p$. From the assumption about z', it follows that $z'p$ is $\overline{v}$-valent, and $z' \neq x$.

Consider the extensions of x that are also prefixes of z'. Since xp and $z'p$ have opposite valencies, there must exist a P-free extension y of x and a process $q \neq p$, $x \leq y < yq \leq z'$, such that yp and yqp are univalent but with opposite valencies.

Now familiar case analyses [FLP85,LA87] preclude p from invoking wait-free read or write() operations: If p is a read, then ypq and yqp are indistinguishable to processes other than p, yet their p-free extensions must have opposite valencies, a contradiction. If p is a write(), then yp and yqp are indistinguishable to processes other than q, yet their q-free extensions must have opposite valencies, a contradiction. Since we chose p arbitrarily from P, we can conclude that no member of P invokes wait-free read or write() operations as their next step.

Finally, let o be the object accessed by p in the last step of yp. If q does not access o in yq, then ypq and yqp are indistinguishable, another contradiction. Suppose some $p' \in P$ accesses an object other than o. Either yp' is v-valent, and the indistinguishability of $yp'qp$ and $yqpp'$ leads to a contradiction, or yp' is $\overline{v}$-valent, and similarly the indistinguishability of ypp' and $yp'p$ leads to a contradiction. It follows that all $t+1$ processes in P invoke powerful operations on o. We conclude that $|\mathrm{ACL}_{pow}(o)| \geq t+1$. ∎

In the next subsection, Theorem 5, in combination with the protocols of the previous section, is a powerful tool in establishing an asymptotic tradeoff on the number of sticky bits (or other powerful objects) necessary to implement strong consensus, as n varies relative to t.

4.2 On the number of powerful objects

Theorem 5 states that any weak consensus protocol that tolerates t crash failures must use at least one object on which at least $t+1$ processes can invoke powerful operations. Section 3, and in particular Theorem 1, shows that sticky bits are examples of such objects that allow consensus protocols tolerating t Byzantine failures, even for $n = 3t+1$. The combination of these results shows that a primitive shared object type, o, with $|\mathrm{ACL}_{pow}(o)| = t+1$, is necessary and sufficient to implement strong consensus.

However, these results ignore the natural efficiency consideration of the *number* of powerful objects that are needed. In particular, the protocol for $n = 3t+1$ uses an exponential number (in t) of powerful sticky bits. Can we do with much fewer (e.g., polynomially many) powerful objects? Since a strong consensus object *sc* would of course trivially implement itself (with $\mathrm{ACL}_{pow}(sc.\mathsf{propose}()) = n$), some care is needed to generalize the question from the specific case of how many sticky bits are necessary. Accordingly, we define an object o to be *subvertible* if it is powerful and any Byzantine process in $\mathrm{ACL}(o)$ can cause operations by correct processes to be useless—that is, the return values of correct processes are dependent only on the operations of the Byzantine process. Sticky bits are subvertible, since a Byzantine process can invoke $\mathsf{set}(0)$ before any correct process takes a step. Obviously, if the object supports $\mathsf{write}()$ operations it is subvertible, and of course, strong consensus itself is not subvertible.

This section uses a combinatorial analysis to provide a tight asymptotic tradeoff between the number, k, of subvertible objects that must be used and the number of processes, n (as a function of the number of possible failures t), in the case that only subvertible objects are used. This tradeoff is essentially given by $k = t \cdot 2^{\Theta(t^2/n)}$. In particular, when $n = 3t + 1$, an exponential number of subvertible objects is indeed necessary. Interestingly, we obtain a protocol for $n = O(t^2/\log t)$ that uses a polynomial number of sticky bits (such a protocol was only previously known for $n = \Omega(t^2)$ [MMRT00]).

More formally, this section gives a proof of the following complementary theorems.

Theorem 6. *For any $n \geq 3t + 1$, there exists a strong consensus protocol that tolerates t Byzantine failures and the only powerful objects it uses are $\max\{t + 1, t \cdot 2^{O(t^2/n)}\}$ sticky bits with access lists of size $t + 1$.*

Theorem 7. *For any $n \geq 3t + 1$ and any constant $0 < \alpha < 1$, any strong consensus protocol that tolerates t Byzantine failures using only subvertible objects must use at least $t \cdot 2^{\Omega(t^2/n)}$ subvertible objects with $|ACL_{pow}(o)| \geq (1 - \alpha)t$ for each such object o.*

As a corollary of Theorem 6 we get the promised protocol for $n = O(t^2/\log t)$ that only uses a polynomial number of sticky bits

Corollary 2. *For any $n \geq 3t + 1$ such that $n = O(t^2/\log t)$, there exists a strong consensus protocol that tolerates t Byzantine failures and the only powerful objects it uses are a polynomial number (in t) of sticky bits with access lists of size $t + 1$.*

Correctness of the first two protocols in the previous subsection depend on their being enough such objects that at least one is accessed by *only* correct processes. The third allows all the powerful objects to be accessed by Byzantine processes, but in this case the protocol uses the guarantee that there are many additional processes, none of whom is even crash faulty. The former property is a combinatorial property of the sticky bit access control lists, which we call being "t-immune": no set of t (faulty) processes can subvert all the powerful sticky bits. This is formalized in the following definition: A collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of the domain $[n] = \{1, 2, \ldots, n\}$ is *m-immune* if for every set $T \subseteq [n]$ of m elements there exists $S_j \in \mathcal{S}$ such that $T \cap S_j = \emptyset$. We note that related objects have been studied extensively in the past [EFF85,GGL95].

Theorem 3 shows that strong consensus protocols can exist even when the access lists of the powerful objects are not t-immune. However, we now argue that

m-immunity (for some m that depends on t) is a necessary condition. As a simple corollary of Theorem 5 (and generalizing Theorem 4) we have the following:

Corollary 3. *Let π be any strong consensus protocol that tolerates $t_1 \geq 1$ crash failures and $t_2 \geq 1$ Byzantine failures and uses only subvertible objects. Let $\mathcal{S}$ be the collection of $ACL_{pow}(o)$ for all powerful objects o used by π that are of size at least $t_1 + 1$. Then $\mathcal{S}$ is t_2-immune.*

Proof (sketch). Assume to the contrary that $T_2 = \{i_1, \ldots, i_{t_2}\}$ is a set of t_2 processes that cover $\mathcal{S}$ (i.e. for all $S_j \in \mathcal{S}$, $T_2 \cap S_j \neq \emptyset$). Then π contains runs in which all the processes in T_2 are Byzantine. These processes can subvert all the objects with large (size $t_1 + 1$) access lists, making these objects useless in fighting the remaining t_1 crash failures. More formally, it is easy to modify π so that it will still tolerate t_1 crash failures but will use no object o with $\text{ACL}_{pow}(o)$ bigger than t_1. Since this is a contradiction to Theorem 5, the corollary follows. ∎

The next corollary follows trivially from Corollary 3.

Corollary 4. *Let π be any strong consensus protocol that tolerates t Byzantine failures and uses only subvertible objects. Let $0 < \alpha < 1$ be some constant such that $(1-\alpha)t > 1$, and let $\mathcal{S}$ be the collection of $ACL_{pow}(o)$ for all powerful objects o used by π that are of size at least $\lfloor (1-\alpha)t \rfloor + 1$. Then $\mathcal{S}$ is $\lceil \alpha t \rceil$-immune.*

4.3 Bounds on the size of m-immune collections

Combining Theorem 1 and Corollary 3, both upper and lower bounds on the number of subvertible objects needed by Byzantine consensus protocols (Theorems 6 and 7) can be derived from corresponding bounds on the number of sets in m-immune collections. Such bounds will be given in this section. The techniques we use to derive these bounds are quite common but we were not able to deduce our results directly from previous work. We note that the results given here contain improvements and significant simplifications due to Noga Alon [Alo02].

Theorem 8. *Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be an m-immune collection of subsets of the domain $[n]$. If all the sets $S_j \in \mathcal{S}$ contain at least $t + 1$ elements then*

$$k \geq \max\{m + 1, m \cdot 2^{\Omega(t \cdot m/n)}\}.$$

Proof. That $k \geq m + 1$ is trivial (any collection of less than $m + 1$ subsets can be covered by m elements). To obtain the rest of the bound we apply the probabilistic method (see [ASE92]). Consider a random subset $T' \subset [n]$ of size $m/2$ (we assume without loss of generality that m is even). For any $j \in [k]$,

$$\Pr[T' \cap S_j = \emptyset] < \left(1 - \frac{t+1}{n}\right)^{m/2}.$$

Therefore, the expected number of $j \in [k]$ such that $T' \cap S_j = \emptyset$ is smaller than $k\left(1 - \frac{t+1}{n}\right)^{m/2}$. If $k\left(1 - \frac{t+1}{n}\right)^{m/2} \leq m/2$ than there exists an assignment to the set T' that covers all but $m/2$ of the subsets S_j. The remaining subsets can be covered by additional $m/2$ elements in $[n]$, which contradicts the assumption that $\mathcal{S}$ is m-immune. We can therefore conclude that $k > m/2 \cdot \left(1 - \frac{t+1}{n}\right)^{m/2}$, completing the proof. ■

An upper bound on the number of sets in m-immune collections can also be obtained by an application of the probabilistic method. Nevertheless, we are interested in an *explicit construction* of the m-immune collection (so that the resulting consensus protocol will be explicit as well). The proof of the following theorem provides such a construction.

Theorem 9. *For any n, t and m with $n > t + m + 1$, there exists an m-immune collection $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ of subsets of the domain $[n]$, such that (1) all the sets $S_j \in \mathcal{S}$ contain at least $t + 1$ elements and (2)*

$$k \leq \max\{m + 1, m \cdot 2^{O(t \cdot m/n)}\}.$$

Proof. We can assume without loss of generality that $n < (m + 1)(t + 1)$ and that $n, t + 1$ and m are all (positive) powers of 2. The first assumption is easy to justify: if $n \geq (m + 1)(t + 1)$ we can simply define $\mathcal{S}$ to be a collection of $m + 1$ disjoint subsets of size $t + 1$. To justify the second assumption, we note that it is sufficient to construct our collection for $n' \leq n$, $t' \geq t$, and $m' \geq m$ that are the closest values such that $n', t' + 1$ and m' are all powers of 2. For this argument to work we must have that $n' > t' + m' + 1$ which is true as long as $n > 4(t + m) + 1$. (Otherwise, if $n \leq 4(t+m)+1$, we have a simple direct construction of the desired collection: Just take all the subsets of size $t + 1$ of the first $t + m + 1$ elements. Since $n \leq 4(t + m) + 1$ we have that $k = \binom{t+m+1}{m} < 2^{t+m+1} = 2^{O(t \cdot m/n)}$ as claimed.)

We can now define the desired m-immune collection. Set $\ell = 2(t + 1)m/n$. By our assumptions ℓ is an integer and furthermore ℓ divides $t + 1$ (as both are powers of 2 and $\ell \leq t + 1$). Set $r = n/2(t + 1)$. By our assumptions this also is an integer and furthermore $r < m$. Now, let us divide the set $[n]$ into $2\ell r$ disjoint subsets of equal size: $A_{1,1}, \ldots A_{1,2\ell}, A_{2,1}, \ldots, A_{2,2\ell} \ldots, A_{r,2\ell}$. We can now define the collection $\mathcal{S}$. For any $i \in [r]$ and any ℓ distinct indices $j_1, j_2, \ldots, j_\ell$ in $[2\ell]$, the collection will contain the set

$$S_{i,j_1,j_2,\ldots,j_\ell} = A_{i,j_1} \cup A_{i,j_2} \cup \ldots \cup A_{i,j_\ell}.$$

By definition, the number of sets in the collection is $r \cdot \binom{2\ell}{\ell} < r \cdot 2^{2\ell} = m \cdot 2^{O(t \cdot m/n)}$. Furthermore, each set contains $n/2r = t + 1$ elements. It remains to show that $\mathcal{S}$ is m-immune. Let T be any subset of $[n]$ of size m. It is easy to see that there exists at least one $i \in [r]$ such that T contains at most $m/r = \ell$ elements in $\cup_{j=1}^{2\ell} A_{i,j}$. Therefore, T contains elements in at most ℓ sets $\{A_{i,j}\}_{j=1}^{2\ell}$. Therefore, there exist distinct indices $j_1, j_2, \ldots, j_\ell$ in $[2\ell]$, such that $T \cap S_{i,j_1,j_2,\ldots,j_\ell} = \emptyset$. ■

5 Conclusions

We presented a t-resilient strong consensus protocol (and therefore universality result) from sticky bits for $n \geq 3t+1$. We demonstrated a tight tradeoff between the fault tolerance and the efficiency of any strong consensus protocol using subvertible objects such as sticky bits, in particular showing that any strong consensus protocol using only subvertible objects must use a super-polynomial number of objects. The tradeoff also implies a polynomial time t-resilient strong consensus protocol for $n = O(t^2/\log t)$.

It remains open whether sticky bits are universal for $n \leq 3t$. Since strong consensus is not possible for $n \leq 3t$, a different universality construction not going through strong consensus would be needed.

Acknowledgements

We are grateful to Noga Alon for his simplifications and improvements that appear in Section 4.3. We would also like to thank Yuval Ishai and Juan Garay for useful discussions.

References

[Alo02] N. Alon. Private communication.

[ASE92] N. Alon, J.H. Spencer, and P. Erdős. *The probabilistic method.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., 1992.

[BG89] P. Berman and J.A. Garay. Asymptotical optimal distributed consensus. *Proceedings of the 16th International Colloquium on Automata, Languages and Programming (ICALP 89)*, LNCS 372, pp. 80–94, 1989.

[EFF85] P. Erdös, P. Frankl and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics* 51:75-89, 1985.

[FHS98] F. Fich, M. Herlihy, and N. Shavit, On the Space Complexity of Randomized Synchronization. *Journal of the ACM*, 45(5):843–862, September 1998.

[FLP85] M.J. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[GGL95] M. Grtschel, R. L. Graham, and L. Lovsz, *Handbook of Combinatorics.* Vol. 1, 2. MIT Press. 1995.

[HW90] M.P. Herlihy and J.M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems* 12(3):463–492, July 1990.

[Her91] M.P. Herlihy. Wait-free synchronization, *ACM Transactions on Programming Languages and Systems* 13(1):124–149, January 1991.

[LA87] M.C. Loui and H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163–183, 1987.

[Mis89] J. Misra. A simple proof of a simple consensus algorithm. *Information Processing Letters*, 33(1):21–24, 1989.

[MMRT00] D. Malkhi, M. Merritt, M. Reiter, and G. Taubenfeld. Objects shared by Byzantine processes. *Proceedings of the 14th International Symposium on Distributed Computing (DISC 2000)*, LNCS 1914, pp. 345–359, 2000.

[Plo89] S.A. Plotkin. Sticky bits and universality of consensus. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, pages 159–175, August 1989.