Distributed Computing Column 39
Byzantine Generals: The Next Generation

Idit Keidar
Dept. of Electrical Engineering, Technion
Haifa, 32000, Israel
idish@ee.technion.ac.il

After almost 30 years of research on *Byzantine Agreement (BA)*, the problem continues to be relevant and to re-invent itself in new ways. This column discusses two new research directions that further push the scale of BA. It suggests new domains where BA can, and perhaps should, be deployed.

First, our main contribution, by Valerie King and Jared Saia, argues for running BA in setting with a large number of nodes (or processors). Valerie and Jared survey new BA protocols whose communication complexity is scalable in the number of participating processors. This, they argue, enables their deployment in larger-scale domains for which BA was considered infeasible before. The second contribution, by Marko Vukolić, considers another emerging domain for BA. It calls for wider-scale deployment of BA protocols, not among many processors, but rather over multiple cloud computing providers.

The column ends with a short announcement about Morgan Claypool's new monograph series on Distributed Computing Theory, edited by Nancy Lynch.

Many thanks to Valerie, Jared, and Marko for sharing their insights!

Call for contributions: I welcome suggestions for material to include in this column, including news, reviews, opinions, open problems, tutorials and surveys, either exposing the community to new and interesting topics, or providing new insight on well-studied topics by organizing them in new ways.

Scalable Byzantine Computation

Valerie King
Dept. of Computer Science
University of Victoria
Victoria, B.C., Canada
val@cs.uvic.ca

Jared Saia
Dept. of Computer Science
University of New Mexico
Albuquerque, NM, USA
saia@cs.unm.edu

1 The Byzantine Agreement Problem

Byzantine agreement is arguably one of the most fundamental problems in distributed computing. The basic idea of the problem is this: n players each vote on one of two choices, and the players want to hold an election to decide which choice wins. The problem is made more interesting by the fact that there is no leader, i.e. no single player who everyone knows is trustworthy and can be counted on to tabulate all the votes accurately.

We now give a more precise statement of the problem that was first defined in [53]. Each of n players starts with either a 0 or a 1 as input. A certain fraction of the players (say some fraction less than $1/3$) are bad, and will collude to thwart the remaining good players. Unfortunately, the good players have no idea who the bad players are. The goal is to create an algorithm for the good players that ensures that

- All good players output the same bit in the end
- The bit output by all good players is the same as the input bit of at least one good player

At first, the second condition may seem very easy to satisfy but completely useless. In fact, it is surprisingly hard to satisfy and extremely useful. To show the problem is hard, consider a naive algorithm where all players send out their bits to each other and then each player outputs the bit that it received from the majority of other players. This algorithm fails because the bad guys can send different messages to different people. In particular, when the vote is close, the bad guys can make one good guy commit to the bit 0 and another good guy commit to the bit 1.

We note that the term *consensus* [34] is often used to refer to the above problem in the setting where bad players are not necessarily malicious. In particular, consensus is frequently used in the situation where the bad players are simply unresponsive. In such a situation, the bad players are said to suffer "crash faults", which is in contrast to the more challenging situation, where the bad players suffer "Byzantine faults". In this paper, we will consider only the latter situation, and will use the term Byzantine agreement to make this clear. We emphasize that Byzantine agreement is a significantly harder problem than consensus. In particular, as mentioned above, in Byzantine agreement, the bad players collude to thwart the good players

and may behave in an arbitrary manner: for example, deviating from the protocol only at a critical moments, and in exactly the right way to cause the protocol to fail. In particular, in Byzantine agreement, we assume that the bad players are all controlled by a malicious adversary that knows the protocol run by the good players and can read all messages that are not specifically hidden from the bad players in some way.

Byzantine agreement (BA) is interesting for several reasons. First, the problem is broadly useful in the engineering sense of helping us build tools. If you can solve BA, you can solve the problem of how to build a system that is more robust than any of its components. For example, one can have each component vote on the outcome of a computation and then with BA its easy to determine which outcome is decided by a plurality of the components (e.g. first everyone sends out their votes, then everyone calculates the majority vote, then everyone solves BA to commit to a majority vote.) It is not surprising then that solutions to the BA problem have been used in all kinds of systems ranging from flight control, to data bases, to peer-to-peer; Microsoft uses BA in Farsite [4, 31]; many structured peer-to-peer systems (e.g. Oceanstore [61, 52]) use BA in some way or another. Any distributed system built with any pretension towards attack-resistance that is not using BA probably should be.

But that is just the engineering side of things. BA is useful because it allows us to study synchronization in complex systems. How can systems like birds, bees, bacteria, markets come to a decision robustly even when there is no leader. We know they do it, and that they do it robustly, but exactly how do they do it and what is the trade-off they pay between robustness and the time and communication costs of doing it? Studying upper and lower bounds on BA gives insight into these natural systems. The study of how these agreement building processes occur in nature has become quite popular lately, since they occur so pervasively.

BA is also useful because it helps us study fundamental properties of computation. The first major impossibility result on consensus (and hence BA) due to Fischer, Lynch and Patterson, in 1982, was that consensus is impossible for any deterministic algorithm with even one bad player (in the asynchronous communication model) [35]. However, a follow up paper by Ben-Or [12] showed that with a randomized algorithm, it was possible to solve BA even with a constant fraction of bad players, albeit in exponential time. This was a fundamental result in computer science giving some idea of how useful randomization can be for computation under an adversarial model.

In this article, we propose that 1) BA for modern large-scale networks is a fundamental and well-motivated problem; and 2) designing a practical solution to BA for large-scale networks is a problem that recent results suggest may be solvable.

The rest of this article is organized as follows. In Section 2, we make the case for developing a *scalable* algorithm for Byzantine agreement, where by scalable, we mean with latency and communication costs are at most polylogarithmic in the network size. To this end, we discuss several possible applications of scalable Byzantine agreement, including a relatively new connection with a game theoretic problem. In Section 3, we discuss what can and can not be accomplished in various models of Byzantine agreement. Next, in Section 4, we describe mathematical and algorithmic techniques that the authors have found useful in addressing the scalable Byzantine agreement problem. Finally, we conclude and describe areas for future work in Section 5.

2 A Case for Scalable BA

For many reasons, adversarial, or Byzantine, attacks are garnering increasing attention in the research community. For example, the following position paper, from 2008, makes a case for Byzantine Fault Tolerance (BFT): "BFT: The Time is Now" [23]. Unfortunately, system engineers have been less than enamored with

the efficiency of current Byzantine agreement algorithms:

- *"Unfortunately, Byzantine agreement requires a number of messages quadratic in the number of participants, so it is infeasible for use in synchronizing a large number of replicas"* [61]
- *"Eventually batching cannot compensate for the quadratic number of messages [of Practical Byzantine Fault Tolerance (PBFT)]"* [25]
- *"The communication overhead of Byzantine Agreement is inherently large"* [20]

There have been many attempts to circumvent this inefficiency by 1) trying to ensure Byzantine fault tolerance in a particular system without solving Byzantine agreement; and 2) trying to solve Byzantine agreement in special cases for a type of adversary that is unique to that paper. This approach is misguided. Why? To build secure systems, we need secure and trusted components. We should not create robust components from scratch for each new system.

So why the problem with Byzantine agreement? The problem is not the problem. Despite what naysayers say about Byzantine agreement being too paranoid, too hard, or too abstract, BA is the right problem formulation for building secure networks. To see this, consider the many systems papers (e.g. [1, 50, 19, 26, 21, 24, 22, 55]) from the recent years that say that a key component of what they want to do requires BA. The problem is the solution. Or rather the lack of a solution. We still do not have a general algorithm to solve BA that is practical enough to plug in to the many systems that want to solve this problem. So despite all the work in this area, we contend that BA is still a major unsolved problem in distributed computing.

2.1 Applications

We now review applications of Byzantine agreement in potentially large-scale networks. Conspicuous by its absence from our discussion is the state machine approach for implementing fault-tolerant services (see e.g. [62] for a survey of this technique). This method relies critically on BA and is of interest to researchers and Internet companies alike [24, 50, 62]. However, in practice, the goal of this approach seem always to tolerate only a single adversarial fault and so typically networks consist of only 4 machines. In addition, there seems to be no evidence of a trend to increase network sizes for this approach in recent years. Thus, we focus on other applications, where there is a very clear trend of growing networks.

Our focus will be on two primary areas that can involve networks of large size: 1) peer-to-peer systems and ad hoc networks; and 2) game theoretic applications.

2.2 Peer-to-peer and Ad-hoc Networks

We now describe several applications and open problems related to scalable Byzantine agreement in the area of peer-to-peer and ad-hoc networks.

Distributed Hash Tables: Byzantine agreement is a critical part of peer-to-peer systems such as Farsite [31] and Oceanstore [59]. These two systems are examples of peer-to-peer systems the provide the functionality of a *distributed hash table(DHT* [64]: essentially a DHT enables scalable, distributed storage and retrieval of data items. In general, peer-to-peer networks have no admission control, so it is to be expected that some nodes in the network will be adversarial in the sense that they will, e.g., attempt to prevent accurate storage and retrieval of content in the system. For example, it is well-known that spam attacks plague popular peer-to-peer file-sharing systems [67]. We ask: Can scalable Byzantine agreement algorithms provide for efficient DHTs that are robust to adversarial attack?

We note that there are techniques for tolerating Byzantine faults in distributed hash tables that do not rely on actually performing Byzantine agreement, see for example the work of Awerbuch and Scheideler on this topic [11]. However, we argue that there are several significant benefits to designing DHTs that derive their robustness from Byzantine agreement including the following: 1) BA provides a formal framework for analyzing distributed robustness: it taps into decades of rich theoretical and practical results, and is used in many application areas, thereby offering the possibility of serendipitous connection between problems in p2p networks and problems in databases, control systems, server farms, and so forth; 2) BA is flexible: it can enable not only robust access to data items, but also is a building block for problems of robust *computation*, which can be useful for voting, data mining, auctions, and so forth in a p2p setting; and 3) techniques in Awerbuch and Scheideler [11] require that the network start in a state where it consists of only good processors, BA allows for the possibility of removing this assumption. Finally, we note that there are gossip-based techniques for tolerating Byzantine faults that are applicable to non-DHT peer-to-peer networks [17, 54, 56]; we believe there are fundamental connections yet to be found between these results also, and the problem of scalable Byzantine agreement.

Enforcing Rules: Many papers study the situation where we want to enforce fair sharing of peer-to-peer resources [57, 66], or to enforce service contracts in grid computing [13, 36]. One simple approach to this problem is to rely on a small set of peers to track usage, and enforce quotas or contracts. In [57], this set is called a "manager set" [57], and as stated there: "To be robust against minority collusion, a remote node would insist that a majority of the manager nodes agree that a given request is authorized, requiring the manager set to perform a Byzantine agreement protocol". Unfortunately, as also mentioned in [57], small manager sets are quite vulnerable to bribery attacks. Thus, to avoid bribery attacks, it is better to have a large manager set, and this large set may frequently need to perform scalable Byzantine agreement.

Wireless and Mobile Networks: Most current algorithms for solving BA generally assume that any two players in the system can communicate directly. Unfortunately, this assumption breaks down for ad hoc and mobile networks. Recent scientific results on quorum sensing in biological systems suggest that it is possible to solve BA *in situ*, provided that the underlying network is well-enough connected. The interesting question here is: Can we design *scalable* algorithms for solving BA in "well-connected" networks, for some reasonable definition of well-connected? An affirmative answer to this question would enable the following engineering applications in wireless networks: secure broadcast (see e.g. [14, 15, 16]); integrity of data aggregation [63]; and robust detection and removal of misbehaving nodes. We note that there are several algorithms for performing BA on networks that are not fully connected (see e.g. [27, 65, 30]), however, these algorithms are not scalable. We note also that there are several algorithms that address the problem of Byzantine broadcast in wireless networks [16, 14, 15, 29, 44]. However, these algorithms do not perform Byzantine agreement, instead they enable correct dissemination of a message from one point in the network to the rest of the network.

2.3 Game Theory

We wrap up this section by describing in some detail a new, exciting connection between Byzantine agreement and problems in game theory.

2.3.1 Reduced Social Cost through Mediation

Game theory studies the emergent behavior in a group of agents, when each agent acts individually to maximize their own utility. A well-known phenomenon in most games is the "tragedy of the commons"

effect. Informally, the tragedy of the commons occurs whenever selfish behavior by each player in the game leads to a situation that is disadvantageous to all the players. The focus of much game theoretic work in the computer science community is defining games that occur in computer networks; identifying those games that exhibit a significant tragedy of the commons effect; and suggesting ways to restructure games (via adding incentives or changing global policies) to reduce this effect (see, for example, [7, 9, 5, 6, 2, 68, 42, 37, 58]). Unfortunately, many of these approaches encourage collaboration by changing the game itself, which may be undesirable or impracticable.

In 1999, in a highly influential paper, Papadimitriou and Koutsoupias [51] quantified the tragedy of the commons effect by defining the *price of anarchy*. The price of anarchy is the ratio between the social cost for the worst case Nash equilibrium and the social cost of the social optimum.[1] For example, consider the following "pollution game" for n players. Each player can decide to pollute or to not pollute. Each player incurs a cost of 2 if it does not pollute. Moreover, each player incurs an additional cost equal to total number of players that pollute. If $n > 2$, the only Nash equilibria for this game is for everyone to pollute and the social cost of this Nash equilibria is n^2. On the other hand, in the social optimum, no player pollutes and the social cost is just $2n$. Thus, the price of anarchy for this game is $n/2$.

One approach to reducing the price of anarchy of a game is to use a *mediator*. Intuitively, a mediator can be viewed as a trusted party that gives each player private advice on which action to pursue. The players retain free will and remain selfish, and will thus ignore the advice if it is in their best interest to do so.

To illustrate the ability of a mediator to reduce the social cost of a game, consider an infinite round pollution game, and a mediator which advises each player not to pollute until some player disregards this advice, after which the mediator will advise every player to pollute in every round. It is in the best selfish interest of each player to follow the advise of such a mediator, and when each player does follow this advice, the social cost per round is optimal. This simple mediator requires a game with multiple rounds in order to react to the actions that the players take. However, surprisingly, there are many single-round games for which mediators can reduce social cost, see for example, [40]. Mediators for such games rely critically on randomization.

2.3.2 Implementing a Mediator

In the mediator approach, one may imagine that the advice each player receives comes from an actual trusted external party. However, recent papers by Abraham, Dolev, Gonen, Halpern and Teague [2, 39] describe algorithms that implement a mediator in a distributed fashion, just by having the players talk amongst themselves (i.e. what economists refer to as "cheap talk"). In other words, they show that there exists a distributed algorithm for talking among the players that enables the simulation of a mediator. Moreover, in [2] it is shown that it is possible to achieve this in a robust manner [2]. More specifically, even if t players are adversarial and coalitions of up to k players may exist, it is possible to simulate a mediator provided that $n > 3k + 3t$. In later, work, Abraham, Dolev and Halpern [3] show that the bounds on t, k and n are essentially tight. Dispensing with the need for an actual trusted external party is frequently desirable since a trusted and impartial external party frequently may not exist.

Unfortunately, because these results rely on algorithms for secure multiparty computation, they are not scalable in that they require each player to send and receive at least $O(n)$ bits. This limits the applicability of the results, for example, on games over large networks like the Internet. Byzantine agreement is a frequently used subroutine in protocols for secure multiparty computation, and hence it seems unlikely that scalability for secure multiparty computation can be achieved without a scalable solution to Byzantine agreement.

[1] The social cost of the outcome of a game is the sum of the costs of each player.

Followup work by Kol and Naor [49] improves on [2, 39], and also makes the reliance on Byzantine agreement more explicit. In [49], they describe distributed algorithms for implementing a mediator that do not rely on any cryptographic assumptions. They point out that some past results in [2, 39] make such assumptions and are thus susceptible to the problem of backwards induction. Essentially, backwards induction occurs when there is some round of a protocol, even a round exponentially far in the future, when the players would be able to decrypt secret information. If such a round of the protocol is ever reached, call it r, there is no longer any incentive for players to cooperate with each other during and after round r. Then, there is no incentive for the players to cooperate with each other in round $r-1$ since in the next round there will be cooperation no matter what happens in round r. By backwards induction, the same argument holds for every round all the way back the first round of the protocol. Backwards induction is problematic when the game played involves fair sharing of information that is initially private and secret for each individual player, i.e. for a rational secret sharing game [39]. However, we note that backwards induction is not a problem for all types of mediators. For example, It is not problematic in the situation where the mediator uses global coin tosses, but the privacy of the global coin must be preserved for only up to a polynomial number of rounds (for an example of games for which this latter type of mediator can be useful, see for example [40]).

The results of Kol and Naor, although non-cryptographic, do depend critically on the assumption of a broadcast channel. In particular, they explicitly assume a non-simultaneous broadcast channel (NSBC) where there is a single sender in each round. Unfortunately, it is non-trivial to implement such a channel in a game theoretic setting where only point-to-point communication exists between the players. Thus, it is a major open problem to remove this assumption from their result.

We contend that one of the major problems in this area is that of scalability. In particular, a scalable algorithm for implementing a mediator could lead to improved algorithms for a large number of problems in modern networks. To illustrate, we list below a few possible applications of a *scalable* approach to mediation.

- *Virus Inoculation:* Ensure that the nodes in a network inoculate against a virus in such a way that few nodes are infected, even when it costs each node to inoculate.
- *Robust Network Creation:* Create a network that is very likely to stay connected if up to a constant fraction of the nodes in the network are deleted, even when it costs each node to create a link.
- *Contention Resolution:* Allow a group of players to send over a single channel with little contention, even when it costs each player to wait for the channel.

While scalable Byzantine agreement is not a sufficient condition for scalable mediation, it does seem to be a necessary one. In addition, recent techniques for developing scalable Byzantine agreement algorithms may have some hope of carrying over to the more challenging problem of developing scalable protocols for secure multiparty computation and hence mediation. See Section 4 below for more discussion of these techniques.

3 What's Possible, What's Not

In this section, we will review different models of the Byzantine agreement problem and describe progress made towards a scalable solution to the problem. For all the models discussed, each processors is assumed to know the ID's of all processors in the network. Moreover, a processor is assumed to know the source of a message sent directly to it. An adversary is said to have *full information* if the adversary's actions, that is,

the choice of bad processors and their actions and the scheduling of message delivery, can depend on the content of all messages sent, even those sent between good processors. The alternative is that the adversary knows the content only of messages sent to bad processors (private channels) or where the hardness of certain cryptographic primitives and a bound on the adversary's computational resources is assumed.

A *nonadaptive adversary* is one that chooses the set of bad processors at the start the protocol. In other words, it can not choose the set of bad processors based on the actual execution of the protocol (i.e., as determined by the random bits). In contrast, an *adaptive* adversary can take over processors at any time during the protocol, up to the point of taking over a set number of processors. The number of bad processors tolerated by an algorithm is refered to as the *resilience* of the algorithm.

Until recently, the need for all-to-all communication with a malicious adversary seemed almost axiomatic, and the most "practical" protocols in the message passing model required this. Let's look at this assumption more closely. We do know, from the 1985 result of Dolev and Reischuk, that any deterministic protocol in a synchronous model requires $\Omega(n^2)$ messages, and hence any randomized protocol will have some nonzero probability of error if it is limited to fewer messages. From the impossiblity result by Fischer, Lynch and Patterson [35], any randomized protocol in the asynchronous model must have some non-terminating runs. And these statements are true even if there are private channels, cryptographic assumptions, and a nonadaptive adversary, since the adversary can be lucky and guess the processor's random bits at the start. Given this, one might still ask for an *expected* low cost of communication and 0 error. However, having an *expected* $o(n^2)$ bits of communication with 0 probability of error is known to be impossible only in the asynchronous model with an adaptive adversary which has full information[8, 10]. There is still the possibility that it may be possible in other models though.

Unfortunately, there is no known protocol with $o(n^2)$ expected bits of communication and 0 probability of error, in any of the models with Byzantine failures. Feldman and Micali in 1985 [33] began a line of work in the private channel model with an expected constant number of rounds; this was extended to the asynchronous model with optimal resilience. However, the protocols are rather complicated and involve significant communication overhead. A recent effort [60] reduces the expected number of bits communicated in the asychronous model with resilience less than $n/4$ to $\tilde{O}(n^3)$ bits.

Finally, regarding resilience, it is impossible to have a randomized protocol with more than 1/3 fraction of bad processors according to an unpublished but often cited result of Karlin and Yao in 1984. All of the known protocols assume that a bad processor cannot spoof some arbitrary number of good processors as this would make any results impossible.

So what can be done with less than all-to-all communication? The authors and their collaborators have designed protocols that succeed with probability $1 - o(1)$ for various models which use $o(n^2)$ bits and take polylogarithmic time, with resilience $n(1/3 - \epsilon)$ for any positive constant ϵ. Most of these protocols involve an *nonadaptive adversary*. This makes it possible to choose and rely on a small representative subset of mostly good processors to coordinate efforts without a high risk that these particular processors will be corrupted.

In this scenerio, the authors and collaborators have shown that even if bad processors can send an unlimited number of bits and messages, it is possible to elect a small set of processors, that is representative in the sense that the fraction of bas processors in the set is not much more than the fraction of bad processors in the general population. Using this technique, it is possible to solve Byzantine agreement in the synchronous model using a polylogarithmic number of bits of communication per good processor and polylogarithmic time to bring all but a $o(1)$ fraction of good processors to agreement with probability $1 - 1/n^c$, even if bad processors are allowed an *unlimited* number of bits of communication [47, 48]. It is also possible, with an average of $\tilde{O}(\sqrt{n})$ bits per processor and logarithmic time, to bring all processors to agreement [45].

Most recently [43], we have found that it is possible to do this in a load-balanced fashion, that is, where the number of bits sent by any one processor do not exceed the average number of bits per processor.

In 2008, the authors and collaborators showed that similar techniques can be extended to give a polylogarithmic time BA protocol in the asynchronous model, though the probability of correctness falls to $1 - O(1/\log n)$. We are currently writing up a version of the algorithm that uses only a polyogarithmic number of bits per processor to elect a small representative set of processors and bring almost all processors to agreement. The result in [45] (but not the load balanced version) will then yield everywhere agreement in the asynchronous model using a total of $\tilde{O}(\sqrt{n})$ bits of communication.

Relying on one small representative set of processors to act as the coordinators is not very robust if a malicious adversary can actively target this small group. To address this issue, the authors have recently found a scalable algorithm that uses secret sharing to circumvent the need to rely on small subsets of mostly good processors [46]. That is, it is possible to solve synchronous Byzantine agreement with an adaptive adversary and private channels using $\tilde{O}(\sqrt{n})$ bits per processor and logarithmic time. Moreover, it is possible to bring all but a $o(n)$ fraction of good processors to agreement using only polylogarithmic number of bits of communication per processor.

We summarize the discussion above in Tables 1 and 2. In these tables, E means to "in expectation", "No Agree" is the fraction of good processors which are not brought to agreement, "Error" is the probability of error, and LB means load balanced.

	Adapt	Full info	Asynch	No Agree	Error	Total Bits
Attiya [10]	Yes	Yes	Yes	0	0	$\Omega(n^2)$ E
FLP [35]	No	No	Yes	0	> 0	Bounded
Dolev [28]	No	No	No	0	0	$\Omega(n^2)$

Table 1: Lower bounds.

	Adapt	Full info	Asynch	No Agree	Error	Total Bits	LB	Time
Bracha [18]	Yes	Yes	Yes	0	0	$\tilde{O}(n2^n)E$	Yes	$\tilde{O}(2^n)$ E
Patra [60]	Yes	No	Yes	0	0	$\tilde{O}(n^3)$ E	Yes	$O(1)$ E
[46]	Yes	No	No	0	$1/n^c$	$\tilde{O}(n^{3/2})$	Yes	$\tilde{O}(1)$
[46]	Yes	No	No	$O(1/\log n)$	$1/n^c$	$\tilde{O}(1)$	Yes	$\tilde{O}(1)$
In prep.	No	Yes	Yes	0	$O(1/\log n)$	$\tilde{O}(n^{3/2})$	Yes	$\tilde{O}(1)$
In prep.	No	Yes	Yes	$O(1/\log n)$	$O(1/\log n)$	$\tilde{O}(1)$	Yes	$\tilde{O}(1)$
[41]	No	Yes	Yes	0	$O(1/\log n)$	$\tilde{O}(n^2)$	Yes	$\tilde{O}(1)$
[47, 48] + [43]	No	Yes	No	0	$O(1/n^c)$	$\tilde{O}(n^{3/2})$	Yes	$\tilde{O}(1)$
[47, 48] + [45]	No	Yes	No	0	$O(1/n^c)$	$\tilde{O}(n^{3/2})$	No	$\tilde{O}(1)$
[47, 48]	No	Yes	No	$O(1/\log n)$	$O(1/n^c)$	$\tilde{O}(1)$	Yes	$\tilde{O}(1)$

Table 2: Upper bounds.

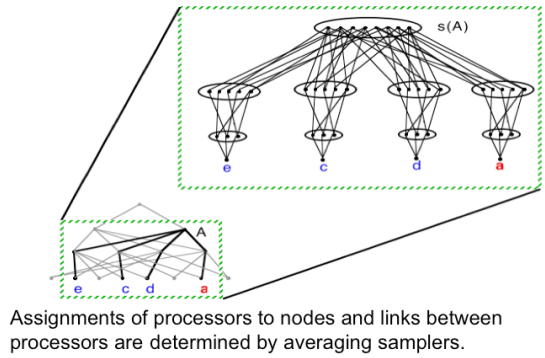

Figure 1: Sparse overlay network for almost everywhere agreement.

4 The Techniques and Their Limits

In this section, we review some of the techniques from the authors' works.

4.1 Resisting Denial of Service Attacks

A difficult challenge when designing scalable BA algorithms is the threat of a denial of service attack by bad processors. That is, a good processor may send a request for information to another good processor, which receives similar requests from every bad processor, and thus cannot afford to respond to all such requests. This is sometimes dealt with in the literature by assuming away most of the problem: bad processors are limited to sending the same number of messages as good processors.[2] We believe this is an unreasonable assumption: during an attack, bad processors are likely to expend more resources than good processors.

Alternatively, a *fixed sparse overlay network* (see Figure 1) for communication can be constructed so that messages from those without direct links to a processor are ignored. (It is assumed that a processor A receiving a message from a processor B can determine that the message is from B.) At the same time, a fixed network has certain disadvantages: bad processors can be chosen so as to surround good processors, cutting them off from any coordination efforts. By using averaging samplers (random-like bipartite graphs) in the design of the overlay network, this problem can be reduced but not avoided altogether.

Also, in such networks, the exact role of each processor is determined by its position in the sparse overlay network. What determines a processor's position? We presume the processors are numbered 1 to n. Alternatively the processors' IDs could have an implicit order, but this would require knowledge of all the IDs. Neither of these assumptions are realistic for the Internet. One might hope to use a hash function to map a larger universe of ID's down to n but this presumes global randomness at the start, or the inability of the adversary to pick the IDs. In any case, such protocols lack the simplicity given by the ID obliviousness of gossip protocols.

A bounded degree fixed overlay network at best achieves almost everywhere agreement for the reason described above. To go from almost everywhere to everywhere seems to require that almost any random pair of processors be able to communicate, yet this opens the door again to a denial of service attack, and appears to result in a $\sqrt{n}$ bottleneck resulting from the Birthday paradox. In particular, if every good processor sends

[2] This still leaves the (much easier) problem of dealing with a few good processors receiving an overload of messages.

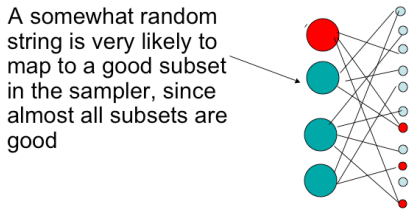

Figure 2: Using averaging samplers.

out $\sqrt{n}$ requests and every good processor responds to each request with probability $1/\sqrt{n}$, then on average one request from each good processor is serviced by a good processor, and no more than $\sqrt{n}$ messages are serviced no matter how many requests a good processor receives from bad processors. This technique presumes that the bombardment of a good processor with many requests does not prevent that processor from randomly choosing a small number to answer. The protocol in [46] requires even more of a good processor under attack: the good processor must compare values sent with each request to decide which to answer. Hence the amount of work done by each processor under a denial of service attack may be large, though the number of messages sent by good processors is relatively small.

4.2 Amplifying Randomness

Another key idea in the work of the authors is the use of a short string with a constant fraction of random bits, where the remaining bits are set by an adversary which can see the random bits. In the full information nonadaptive adversary model, the authors' protocols elect a small set of representative processors which generate an agreed upon logarithmic length string with mostly random bits. Alternatively, in the case of an adaptive adversary and private channels [46], simply the string, without the set of processors, is generated. Such strings can be generated with high probability, and they prove to be very powerful. Generating a single random bit with high probability seems like a much harder problem, especially in the full information model, perhaps no easier that the problem of electing a single good leader (See [32]) .

As in work on pseudorandomness (see [38]), this string known as a *bit fixing random source* can be used as input to an *averaging sampler*, to amplify randomness. See Figure 2. For example, a string whose individual bits are generated by each member of a subset containing a small ϵ constant fraction of good processors can be used to select a comparably sized subset with almost a 2/3 fraction of good processors from a set containing a 2/3 fraction of good processors.

This amplification is needed especially for the asynchronous scenerio where action must be taken despite the fact that bits from good processors are arbitrarily delayed [41]. A single short string with some random bits is sufficient for other interesting purposes, such as providing global coins for Rabin's BA algorithm (as used in [46]), agreeing upon a collection of many small representative sets (quorums) [43], and, in general, choosing from a distribution where a large fraction of choices are good.

4.3 Running Elections

Constructing a small set of representatives (or small string with some random bits) is a gradual elimination process based on Feige's bin selection algorithm [32]. Small groups of processors, most of which contain

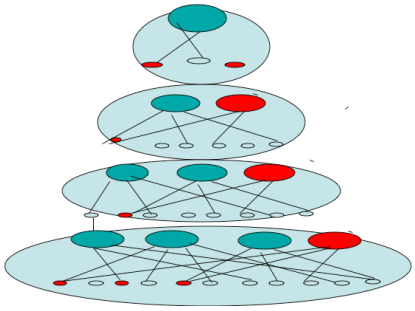

Figure 3: Tournament to elect a small representative set.

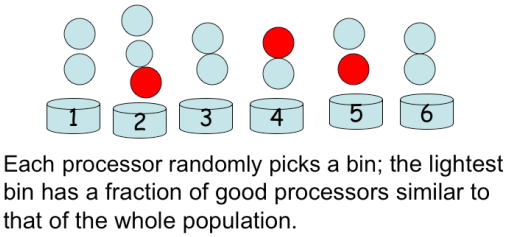

Figure 4: Feige's bin selection.

at least a 2/3 fraction of good processors, each elect a subset of processors (or random bits) to proceed in a tournament-like protocol, so that eventually only a small group of processors (or random bits) survive. See Figure 3.

Feige's bin selection algorithm, originally developed for the broadcast model, is so simple and clever that it is worth mentioning here. See Figure 4.

Each processor announces to the others its choice of a bin number from a certain range. The processors that pick the bin chosen by the fewest processors are elected. Even if the adversary waits to hear the good processors' choices, it cannot bias very much the make-up of the committee. Such an approach may be extendable to a game theory scenerio where there are more than just the two factions of good and bad.

5 Conclusion

In this article, we have argued that Byzantine agreement remains a fundamental and well-motivated problem in distributed computing, and that, moreover, scalable BA is an important problem because of the increasing size of modern networks. We have discussed several recent applications of Byzantine agreement, including a new connection with game theory; have discussed upper and lower bounds on the problem in various models; and have discussed new techniques for designing scalable algorithms for BA. There are many important open problems that remain including the following.

- Is the $\sqrt{n}$ communication cost for doing Byzantine agreement somehow fundamental? More precisely, can we do Byzantine agreement with $o(n^{3/2})$ total bits communication with high probability?

Perhaps an easier problem is: Can we do Byzantine agreement with each processor sending $o(\sqrt{n})$ bits if we limit the number of messages the adversary sends and the channels are public? Note: it is easy to achieve as little as $O(n \log n)$ total bits communication with a communication limited adversary and public channels, the challenging problem is achieving this in a load-balanced sense.

- There are several obvious gaps in Tables 1 and 2, that need to be closed. As just one example, can we do *asynchronous* BA with an adaptive adversary and private channels, where each processor sends $o(n^2)$ bits?
- Can we make any of the scalable almost-everywhere algorithms for BA into Las Vegas algorithms?
- Can we use techniques for scalable Byzantine agreement to design a scalable algorithm for simulating a mediator? A first approach to this is designing an algorithm for scalable BA in the case where all players are selfish.
- Can we use techniques from the scalable BA algorithms to address problems of robustness in networks subject to churn? An idea is to assume that: 1) the number of processors fluctuates between n and $\sqrt{n}$ where n is the size of name space; 2) the processors do not know explicitly who is in the system at any time; and 3) that the number of bad processors in the system is always less than a $1/3$ fraction. In such a model, can we 1) do Byzantine agreement; and 2) maintain small (i.e. polylogorathimic size) quorums of mostly good processors?

References

[1] M. Abd-El-Malek, G. Ganger, G. Goodson, M. Reiter, and J. Wylie. Fault-scalable Byzantine fault-tolerant services. *ACM SIGOPS Operating Systems Review*, 39(5):74, 2005.

[2] I. Abraham, D. Dolev, R. Gonen, and J. Halper. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Principles of Distributed Computing(PODC)*, 2006.

[3] I. Abraham, D. Dolev, and J. Halper. Lower bounds on implementing robust and resilient mediators. In *IACR Theory of Cryptography Conference(TCC)*, 2008.

[4] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th Annual Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[5] R. Anderson and T. Moore. The economics of information security. *Science*, 314(5799):610–613, 2006.

[6] R. Anderson and T. Moore. Information security economics - and beyond. In *CRYPTO*, pages 68–91, 2007.

[7] R. J. Anderson. Why information security is hard-an economic perspective. In *ACSAC*, pages 358–365. IEEE Computer Society, 2001.

[8] J. Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. *Journal of the Association for Computing Machinery*, 45(3):415–450, May 1998.

[9] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *Journal of Computer and System Science*, 72(6):1077–1093, 2006.

[10] H. Attiya and K. Censor. Tight bounds for asynchronous randomized consensus. *J. ACM*, 55(5), 2008.

[11] B. Awerbuch and C. Scheideler. Towards a scalable and robust dht. *Theory Comput. Syst.*, 45(2):234–260, 2009.

[12] M. Ben-Or. Another advantage of free choice (Extended Abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30. ACM New York, NY, USA, 1983.

[13] F. Berman, G. Fox, and A. Hey. *Grid computing: making the global infrastructure a reality*. John Wiley & Sons Inc, 2003.

[14] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network. In 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 138–147, 2005.

[15] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network: A simplified characterization. Technical report, CSL, UIUC, May 2005.

[16] V. Bhandhari, J. Katz, C.-Y. Koo, and N. Vaidya. Reliable broadcast in radio networks: The bounded collision case. In 25^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 258 – 264, 2006.

[17] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine resilient random membership sampling. *Computer Networks*, 53(13):2340–2359, 2009.

[18] G. Bracha. An asynchronous [(n - 1)/3]-resilient consensus protocol. In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 154–162, New York, NY, USA, 1984. ACM.

[19] M. Castro and B. Liskov. Practical Byzantine fault tolerance. *Operating Systems Review*, 33:173–186, 1998.

[20] C.-F. Cheng, S.-C. Wang, and T. Liang. The anatomy study of server-initial agreement for general hierarchy wired/wireless networks. *Computer Standards & Interfaces*, 31(1):219 – 226, 2009.

[21] B. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz. Attested append-only memory: making adversaries stick to their word. *ACM SIGOPS Operating Systems Review*, 41(6):189–204, 2007.

[22] A. Clement, M. Kapritsos, S. Lee, Y. Wang, L. Alvisi, M. Dahlin, and T. Riché. Upright cluster services. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 277–290. ACM, 2009.

[23] A. Clement, M. Marchetti, E. Wong, L. Alvisi, and M. Dahlin. Byzantine fault tolerance: the time is now. In *LADIS '08: Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, pages 1–4, New York, NY, USA, 2008. ACM.

[24] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine fault tolerant systems tolerate Byzantine faults. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, 2009.

[25] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira. Hq replication: A hybrid quorum protocol for byzantine fault tolerance. In *In Proceedings of Operating Systems Design and Implementation (OSDI)*, San Diego, CA, USA, 2005.

[26] J. Cowling, D. Myers, B. Liskov, R. Rodrigues, and L. Shrira. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance. In *Proceedings of the 7th symposium on Operating systems design and implementation*, page 190. USENIX Association, 2006.

[27] F. Cristian, H. Aghili, R. Strong, and D. Dolev. Atomic broadcast: From simple message diffusion to Byzantine agreement. *Information and Computation*, 118(1):158, 1995.

[28] D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM*, 32(1):191–204, 1985.

[29] V. Drabkin, R. Friedman, and M. Segal. Efficient byzantine broadcast in wireless ad-hoc networks. In *International Conference on Dependable Systems and Networks (ICDSN)*, pages 160–169, 2005.

[30] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, page 379. ACM, 1986.

[31] Farsite. The Farsite Project. http://research.microsoft.com/farsite.

[32] U. Feige. Noncryptographic selection protocols. In *FOCS*, pages 142–153, 1999.

[33] P. Feldman and S. Micali. Byzantine agreement in constant expected time (and trusting no one). In *FOCS*, pages 267–276, 1985.

[34] M. Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *Fundamentals of Computation Theory*, pages 127–140, 1983.

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

[36] I. T. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In P. Ritrovato, S. A. Cerri, S. Salerno, M. Gaeta, C. Allison, and T. Dimitrakos, editors, *Towards the Learning Grid*, volume 127 of *Frontiers in Artificial Intelligence and Applications*, pages 28–40. IOS Press, 2005.

[37] N. Fultz and J. Grossklags. Blue versus red: Towards a model of distributed security attacks. In *Financial Cryptography*, pages 167–183, 2009.

[38] R. Gradwohl, S. P. Vadhan, and D. Zuckerman. Random selection with an adversarial majority. In *CRYPTO*, pages 409–426, 2006.

[39] M. Humphrey and M. Thompson. Security Implications of Typical Grid Computing Usage Scenarios, 2000. Security Working Group GRIP forum draft.

[40] N. R. Josep Diaz, Dieter Mitsche and J. Saia. On the power of mediators. In *Proceedings of the Workshop on Internet and Network Economics (WINE)*, 2009.

[41] B. M. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. In *SODA*, pages 1038–1047, 2008.

[42] M. J. Kearns and L. E. Ortiz. Algorithms for interdependent security games. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[43] V. King, S. Lonergan, J. Saia, and A. Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. In *submitted to ICDCN*, 2011.

[44] V. King, C. A. Phillips, J. Saia, and M. Young. Sleeping on the job: energy-efficient and robust broadcast for radio networks. In R. A. Bazzi and B. Patt-Shamir, editors, *PODC*, pages 243–252. ACM, 2008.

[45] V. King and J. Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{o}(n^{3/2})$ bits. In *DISC*, pages 464–478, 2009.

[46] V. King and J. Saia. Breaking the $o(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. In *PODC*, 2010.

[47] V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *SODA*, pages 990–999, 2006.

[48] V. King, J. Saia, V. Sanwalani, and E. Vee. Towards secure and scalable computation in peer-to-peer networks. In *FOCS*, pages 87–98, 2006.

[49] G. Kol and M. Naor. Games for exchanging information. In *Proceedings of the Symposium on the Theory of Computing*, 2008.

[50] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: speculative byzantine fault tolerance. In *Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, page 58. ACM, 2007.

[51] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, 1999.

[52] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[53] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):401, 1982.

[54] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. In *Proceedings of the 7th symposium on Operating systems design and implementation*, page 204. USENIX Association, 2006.

[55] Y. Mao, F. Junqueira, and K. Marzullo. Towards low latency state machine replication for uncivil wide-area networks. In *Workshop on Hot Topics in System Dependability*. Citeseer, 2009.

[56] Y. Minsky and F. Schneider. Tolerating malicious gossip. *Distributed Computing*, 16(1):49–68, 2003.

[57] T. Ngan, D. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[58] Nisan, Roughgarden, Tardos, and Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[59] Oceanstore. The Oceanstore Project. http://oceanstore.cs.berkeley.edu.

[60] A. Patra and P. Rangan. Brief announcement: Communication efficient asynchronous byzantine agreement. In *PODC '10: Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 243–244, New York, NY, USA, 2010. ACM.

[61] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore prototype. In *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 1–14, 2003.

[62] F. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):319, 1990.

[63] E. Shi, A. Perrig, et al. Designing secure sensor networks. *IEEE Wireless Communications*, 11(6):38–43, 2004.

[64] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[65] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *Proceedings of the eleventh annual ACM symposium on Principles of distributed computing*, pages 83–89. ACM New York, NY, USA, 1992.

[66] V. Vishnumurthy, S. Chandrakumar, and E. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[67] Wikipedia. Anonymous Peer-to-peer. http://en.wikipedia.org/wiki/AnonymousP2P.

[68] J. Yan, S. Early, and R. Anderson. The xenoservice-a distributed defeat for distributed denial of service. *Proceedings of Information Survivability Workshop (ISW)*, 2000.

The Byzantine Empire in the Intercloud

Marko Vukolić
IBM Research - Zurich
CH-8803 Rüschlikon, Switzerland
mvu@zurich.ibm.com

Abstract

The relevance of Byzantine fault tolerance in the context of cloud computing has been questioned [3]. While arguments against Byzantine fault tolerance seemingly makes sense in the context of a single cloud, i.e., a large-scale cloud infrastructure that resides under control of a single, typically commercial provider, these arguments are less obvious in a broader context of the *Intercloud*, i.e., a cloud of multiple, independent clouds.

In this paper, we start from commonly acknowledged issues that impede the adoption of Byzantine fault tolerance within a single cloud, and argue that many of these issues fade when Byzantine fault tolerance in the Intercloud is considered.

1 Introduction

In dependable computing, the term "Byzantine" fault is a catch-all for any type of software or hardware fault. It is frequently associated with malicious behavior and intrusion tolerance; however, it also encompasses spurious and arbitrary faults as well as conventional crash faults. Hence, it is not surprising that Byzantine fault tolerance (BFT) has received a lot of research attention ever since the concept of a Byzantine fault was introduced three decades ago [24, 26], as it promises dependable systems that tolerate any type of misbehavior.

Despite this attention and appealing promises, BFT suffers from very limited practical adoption [21]. Notable exceptions include a modest number of safety critical embedded and real-time systems, e.g., in the aerospace industry [14]. However, in the more conventional distributed systems, BFT remains sidelined.

In particular, the relevance of BFT has been questioned in the context of cloud computing, which is arguably one of the main drivers of the distributed systems agenda in the last few years. Practitioners, whose daily bread-and-butter is dealing with (very) large-scale distributed systems that propel modern cloud services, describe BFT as of "purely academic interest" for a cloud [3]. Such a lack of interest for BFT within a cloud is in sharp contrast to the popularity of mechanisms for tolerating crash faults which are routinely employed in such large-scale systems. Furthermore, the wealth of BFT research that is doing hard work to make BFT practical and to bring the performance of BFT protocols close to that of their crash-tolerant counterparts (e.g., [8, 12, 19]), appears incapable of spurring the interest for BFT applications.

Factors that impede wider adoption of BFT arguably include notorious difficulties in design, implementation, proofs and even simple understanding of BFT protocols, especially when BFT protocols also have to

face asynchrony [17]. However, while relevant, this seems not to be the core of the problem. In fact, it seems that there are strong reasons to believe that even if the BFT community came up with provably optimal and best-possible protocols, such BFT protocols would still fail to make it into the large scale cloud; we discuss these reasons and arguments in more details in Section 2.

However, one should beware of not seeing the forest for the trees: the cloud computing landscape obviously does not end with a single cloud. The sky is very cloudy these days, with many clouds that come in different shapes and colors, forming the cloud of clouds, or simply, the *Intercloud*[1]. These many shapes and colors reflect different implementations and administrative domains, which are like a dream come true for a designer of dependable and fault-tolerant systems, e.g., in terms of failure-independence. There are already some very early protocols that are designed for and deployed in the Intercloud [1, 4, 6] that leverage multiple clouds to add value to the client beyond the dependability guarantees of any individual cloud. The Intercloud can be simply seen as the second layer in the cloud computing stack and it calls for considerably different protocols than the "bottom" layer, i.e., the *inner-cloud* layer [6].

In this paper, we develop the argument around the observation made in [7], that it is in this second layer, in the Intercloud, where the modern Byzantine Empire might find its place. More specifically, we argue (Section 3) why the Intercloud is suitable for BFT, but also (and perhaps more importantly) why BFT is suitable for the Intercloud.

2 Limitations of BFT for the Inner-cloud

Today's inner-cloud is characterized by large-scale distributed systems, such as Amazon's Dynamo [13], Google's File System (GFS) [16] and BigTable [10], Yahoo's Zookeeper [20] or Facebook's Cassandra [2], running in a single administrative domain and powering some of the most popular services in existence. While these protocols deal with crash-failures routinely, there is no (known) trace of BFT inside them. Let us hypothetically consider the best possible and by all means optimal BFT protocol for a given task. The following arguments would still impede the adoption of such a protocol in the inner-cloud.

Inherent cost. The inherent cost of BFT in terms of number of replicas needed to tolerate failures is often cited as one of its major drawbacks. Consider state-machine replication (SMR) [5, 30] as a classical example: crash and asynchrony-tolerant implementations like Paxos [22], require at least $2f + 1$ nodes to tolerate f failures, whereas in the case of BFT Paxos $3f + 1$ nodes are required [23]. While assuming small trusted hardware components inside nodes can eliminate this overhead (see e.g., [25, 32] for recent contributions), trusted architectures have yet to be embraced more widely.

Failure independence. All fault-tolerant protocols, including BFT, require some level of failure independence [31]. However, in the case of BFT, this level is particularly high. Consider for example a Byzantine failure that results from a malicious exploit of a vulnerability of a given node. To prevent such a failure to propagate to other nodes, one must assume different implementations, different operating systems, separate administrative domains and even different hardware. This should be complemented by the diversity required for independent crash-failures, which includes different power supplies and possibly even geographical diversity. "In-house" maintenance of the level of infrastructural diversity that BFT requires is prohibitively expensive even for the large-scale cloud providers [21].

[1] http://www.google.com/search?q=intercloud
[2] The Apache Cassandra Project. http://cassandra.apache.org/

Does the model match the threats? Clearly, the above issues which are in the end associated with the cost of deploying BFT, would not be as critical if the Byzantine model would be adequate for the threats within a cloud. However, the Byzantine failure model is often seen as too general, allowing for any type of failures, notably malicious ones. Although attacks on well-known cloud providers occur on a regular basis and sometimes have success [3], exploits that would allow the attacker to hijack the critical part of the internal cloud infrastructure are yet to be seen. Such infrastructure chooses intrusion prevention and detection over intrusion tolerance, and is carefully protected and isolated from the "outside" world. Consider, for example, Google's Chubby [9], a distributed locking and low-volume storage service based on Paxos (and a rare example of SMR within a cloud). Chubby is a shielded service offered to very selected clients such as Google's own GFS and BigTable nodes which do not present a realistic misbehavior threat for Chubby. The same is true for other critical parts of the inner-cloud (e.g., Yahoo Zookeeper) — these critical services that could potentially profit from BFT simply seem not to need it since they typically run in protected environments.

Rare non-crash failures in the inner-cloud call for a failure model more restricted than Byzantine, that would ideally come with a lower cost. However, devising such an adequate failure model for such a single administrative domain, that would be more restricted than Byzantine yet allow for some randomized behavior and a few bugs, is not obvious. Other popular models in the BFT community, like BAR (in which nodes can be Byzantine, altruistic or rational) [2] were devised with a different setting in mind (e.g., cooperative services in peer-to-peer networks) and are clearly not applicable here. This model/threat mismatch seems to be one of the main factors that keep BFT out of the inner-cloud picture.

3 BFT in the Intercloud

Despite their high availability goals, individual cloud services are still subject to outages (see e.g., [7]). Moreover, studies show that over 80% of company executives "fear security threats and loss of control of data and systems".[4] One way to approach these issues is to leverage the Intercloud, i.e., the cloud of clouds. The key-high level idea here is essentially to distribute the security, trust and reliability across different cloud providers to improve on the offerings of any individual one. Early examples of such dependable systems in the Intercloud target reliable distributed storage [11] some of which employ a subset of BFT techniques. For example, RACS (Redundant Array of Cloud Storage) [1] casts RAID into the Intercloud setting to improve availability of data and protect against vendor lock-in. HAIL (High Availability and Integrity Layer) [4] leverages multiple clouds to boost the integrity and availability of the data stored in the clouds. ICStore (Intercloud storage) [6] goes a step beyond RACS and HAIL in terms of dependability, and aims to allow *a la carte* modular combinations of individual layers dedicated to boosting confidentiality, integrity, reliability and consistency of the data stored in the clouds. In the following, we argue why the Intercloud is a big opportunity for BFT in general and what BFT applications could appear early-on in this novel context.

Unprecedented failure independence. The Intercloud offers different implementations of semantically similar services along with the unprecedented failure independence. Intuitively, there is much less failure dependence between a virtual machine hosted by Amazon EC2 and another one by Rackspace Cloud Servers than between two machines in a given local cluster [18]. These two virtual machines come in differ-

[3] Official Google Blog: A new approach to China. http://googleblog.blogspot.com/2010/01/new-approach-to-china.html

[4] Survey: Cloud computing 'no hype', but fear of security and control slowing adoption. http://www.circleid.com/posts/20090226_cloud_computing_hype_security/

ent administrative domains with different proprietary cloud architecture and middleware, possibly different operating systems, not to mention different geographical locations and power supplies. And, best of all, this diversity comes to a BFT designer essentially for free — the maintenance of this diversity remains in the hands of individual cloud providers. Practically, BFT designers and developers are left worrying only about careful programming of their own protocols. The need for n-version programming might persist, but this becomes easier with the use of right abstractions and the modular approach to the problem [17, 28, 31]. Overall, the inherent diversity of the Intercloud promises to significantly reduce the costs of deploying and maintaining BFT protocols.

The threat calls for BFT. Furthermore, the threat that a virtual machine accessible from anywhere faces is arguably different from the one faced by a Paxos node running deeply within the protected environment of a provider's inner-cloud infrastructure. For example, the increased concern over a virtual machine security stems from the possibility of the compromise of the credentials needed to access the virtual machine remotely, but also from the issues related to cloud multi-tenancy (see e.g., [27]). Hence, the Byzantine model seems to reasonably fit the threat faced by a virtual machine hosted by the cloud; users, including company executives, might sleep better knowing that their (Inter)cloud service is dependable and protected from attack on any individual cloud.

How about inherent cost? On the other hand, the Intercloud clearly does not reduce the inherent cost overhead of BFT compared to crash fault-tolerance in terms of a number of replicas. However, while putting trusted hardware components into virtual machines might not be as simple as putting them into physical machines, there is increasing effort to address those aspects of cloud computing (e.g., Terra [15] and TCCB [29]). The Intercloud and BFT will inevitably profit from these efforts that aim to strengthen the security guarantees of individual clouds; these efforts would allow established BFT protocols that leverage trusted architectures to shine in a new light of the Intercloud. An unresolved issue would still be for industry to embrace such architectures. However, given the level of customers' concerns over cloud security, we might witness market differentiation bootstrapping the process of the adoption of trusted cloud architectures.

What early-bird applications? The Intercloud indeed seems to be a promising candidate for an answer to the question on "where" (in the cloud context) should BFT be deployed [21]. On the other hand, to answer questions "why" and "when" [21], we still need to identify an application that will illustrate the full benefits of BFT in the Intercloud and promote its adoption. This might be BFT state machine replication in an implementation of a very reliable low volume storage service, along the lines of Chubby. This service could be shared among different clients coming from, e.g., small private clouds, and for which the clients would be prepared to pay the premium in order not to deal with complexities and the cost of the maintenance of such a service "in-house". Such a low volume storage application in the Intercloud could in fact need BFT since, unlike Chubby, neither its clients nor its server nodes scattered among the clouds can be fully trusted. Furthermore, BFT could be used in the Intercloud to mask possible inconsistencies inherent to the eventually consistent semantics of a highly available cloud [33]. In this approach an inconsistent reply from an otherwise correct cloud service could be masked as a Byzantine and the Intercloud service could leverage other clouds to improve on the overall consistency. Finally, integrity of data and computation seems like a strong candidate for BFT in the Intercloud and, and as we already discussed, we are already witnessing research efforts in this direction [4, 6]. Clearly, this is not an exhaustive list and we expect to witness an increasing number of ideas for BFT applications in the Intercloud in the near future.

4 Conclusions

In this paper we summarized the limitations of Byzantine fault-tolerance for applications in the large-scale distributed systems of the inner-cloud. We also argued that Byzantine fault-tolerance is more suitable to the Intercloud, a cloud of clouds, which is emerging as the second layer in the cloud computing stack to complement the inner-cloud layer. The Intercloud offers an unprecedented failure independence at a low maintenance cost — this is provided by the diversity of cloud providers and differences in their internal implementations. Moreover, Byzantine failure model appears well suited for the threats that the Intercloud faces.

Acknowledgments. I would like to thank Christian Cachin, Robert Haas and Rachid Guerraoui for their valuable comments and many interesting discussions.

References

[1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. Racs: a case for cloud storage diversity. In *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 229–240, New York, NY, USA, 2010. ACM.

[2] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR fault tolerance for cooperative services. In *SOSP '05: Proceedings of the ACM SIGOPS 20th Symposium on Operating Systems Principles*, pages 45–58, New York, NY, USA, 2005. ACM.

[3] K. Birman, G. Chockler, and R. van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.

[4] K. D. Bowers, A. Juels, and A. Oprea. HAIL: A high-availability and integrity layer for cloud storage. In *CCS '09: The 16th ACM Conference on Computer and Communications Security*, pages 187–198, 2009.

[5] C. Cachin. State machine replication with Byzantine faults. In Bernadette Charron-Bost, Fernando Pedone, and André Schiper, editors, Replication: Theory and Practice, LNCS, vol. 5959, pages 169-184. Springer, 2010.

[6] C. Cachin, R. Haas, and M. Vukolić. Dependable storage in the Intercloud. Research Report RZ 3783, IBM Research, Aug. 2010.

[7] C. Cachin, I. Keidar, and A. Shraer. Trusting the cloud. *SIGACT News*, 40(2):81–86, 2009.

[8] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[9] T. D. Chandra, R. Griesemer, and J. Redstone. Paxos made live: an engineering perspective. In *PODC '07: Proceedings of the 26th annual ACM Symposium on Principles of Distributed Computing*, pages 398–407, New York, NY, USA, 2007. ACM.

[10] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2):1–26, 2008.

[11] G. Chockler, R. Guerraoui, I. Keidar, and M. Vukolić. Reliable distributed storage. *IEEE Computer*, 42(4):60–67, 2009.

[12] A. Clement, M. Kapritsos, S. Lee, Y. Wang, L. Alvisi, M. Dahlin, and T. Riche. Upright cluster services. In *SOSP '09: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, pages 277–290, New York, NY, USA, 2009. ACM.

[13] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *SOSP '07: Proceedings of the ACM SIGOPS 21st Symposium on Operating Systems Principles*, pages 205–220, New York, NY, USA, 2007. ACM.

[14] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg. Byzantine fault tolerance, from theory to reality. In *SAFECOMP '03: Proceedings of the 22nd International Conference on Computer Safety, Reliability, and Security*, pages 235–248, 2003.

[15] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 193–206, New York, NY, USA, 2003. ACM.

[16] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *SOSP '03: Proceedings of the ACM SIGOPS 19th Symposium on Operating Systems Principles*, pages 29–43, New York, NY, USA, 2003. ACM.

[17] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić. The next 700 BFT protocols. In *Eurosys '10: Proceedings of the 5th ACM SIGOPS/EuroSys European Conference on Computer Systems*, pages 363–376, 2010.

[18] R. Guerraoui and M. Yabandeh. Independent faults in the cloud. In *LADIS '10: Proceedings of the 4th ACM SIGOPS/SIGACT Workshop on Large-Scale Distributed Systems and Middleware*, pages 12–16, 2010.

[19] J. Hendricks, G. R. Ganger, and M. K. Reiter. Low-overhead Byzantine fault-tolerant storage. In *SOSP '07: Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, pages 73–86, New York, NY, USA, 2007. ACM.

[20] F. P. Junqueira and B. Reed. The life and times of a zookeeper. In *PODC '09: Proceedings of the 28th annual ACM Symposium on Principles of Distributed Computing*, page 4, 2009.

[21] P. Kuznetsov and R. Rodrigues. BFTW^3: Why? When? Where? Workshop on the theory and practice of Byzantine fault tolerance. *SIGACT News*, 40(4):82–86, 2009.

[22] L. Lamport. Paxos made simple. *ACM SIGACT News*, 32(4):18–25, 2001.

[23] L. Lamport. Lower bounds for asynchronous consensus. *FuDiCo '03: Future directions in distributed computing*, pages 22–23, 2003.

[24] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[25] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda. TrInc: Small trusted hardware for large distributed systems. In *NSDI '09: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 1–14, 2009.

[26] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[27] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS '09: Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 199–212, New York, NY, USA, 2009. ACM.

[28] R. Rodrigues, M. Castro, and B. Liskov. BASE: using abstraction to improve fault tolerance. In *SOSP '01: Proceedings of the ACM SIGOPS 18th Symposium on Operating Systems Principles*, 2001.

[29] N. Santos, K. P. Gummadi, and R. Rodrigues. Towards trusted cloud computing. In *HotCloud '09: The 1st USENIX Workshop on Hot Topics in Cloud Computing*, 2009.

[30] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.

[31] F. B. Schneider and L. Zhou. Implementing trustworthy services using replicated state machines. In Bernadette Charron-Bost, Fernando Pedone, and André Schiper, editors, Replication: Theory and Practice, LNCS, vol. 5959, pages 151-167. Springer, 2010.

[32] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves, and P. Veríssimo. Highly available intrusion-tolerant services with proactive-reactive recovery. *IEEE Trans. Parallel Distrib. Syst.*, 21(4):452–465, 2010.

[33] W. Vogels. Eventually consistent. *Commun. ACM*, 52(1):40–44, 2009.

Announcement: Synthesis Lectures on Distributed Computing Theory

Nancy Lynch, MIT

Morgan Claypool is publishing a new monograph series on Distributed Computing Theory, edited by Nancy Lynch of MIT. This series will publish 50- to 150-page monographs on topics pertaining to any aspect of distributed computing theory, including distributed algorithms and lower bounds, algorithm design methods, formal modeling and verification of distributed algorithms, concurrent data structures, and other topics.

This series is the latest addition to the Synthesis Digital Library of Engineering and Computer Science. Synthesis is intended to fill the gap between graduate textbooks and journals, providing focused coverage of fast-moving areas in an approachable, pedagogical fashion.

Use of these books as course texts is encouraged; and the texts may be downloaded without restriction at licensing institutions, or after a one-time fee at non-licensing schools. These books can also be purchased in print directly from the Morgan & Claypool Bookstore, or from Amazon and other booksellers worldwide. Contact info@morganclaypool.com for desk copies.

Contact lynch@csail.mit.edu or info@morganclaypool.com to inquire about contributing to this series as an author or reviewer.

In the Distributed Computing Theory series, two books are already available:

- Communication and Agreement Abstractions for Fault-Tolerant Asynchronous Distributed Systems (2010), by Michel Raynal.
- The Mobile Agent Rendezvous Problem in the Ring (2010), by Evangelos Kranakis, Danny Krizanc, and Euripides Markou.

Upcoming titles include:

- Algorithmic Foundations of Communication in Radio Networks, by Dariusz Kowalski.
- Complexity of Cooperation in Distributed Systems, by Alexander A. Shvartsman and Chryssis Georgiou.
- Fault-Tolerant Agreement in Synchronous Message-Passing Systems, by Michel Raynal.
- Knowledge in Distributed Systems, by Yoram Moses.
- k-set Consensus and the Consensus Hierarchy, by Soma Chaudhuri and Jennifer Welch.
- Link-Reversal Algorithms, by Jennifer Welch and Jennifer Walter.
- Lower Bounds for Distributed Algorithms, by Faith Ellen and Hagit Attiya.
- Randomized Smoothing Networks, by Marios Mavronicolas and Thomas Sauerwald.
- Reductions and Simulations in Distributed Computing, by Sergio Rajsbaum and Corentin Travers.
- The Theory of Timed I/O Automata, Second Edition, by Dilsun Kaynar, Nancy Lynch, Roberto Segala, and Frits Vaandrager.
- Transactional Memory, Second Edition, by Tim Harris, Jim Larus, and Ravi Rajwar.
- Transactional Memory: The Theory, by Rachid Guerraoui and Michal Kapalka.