

# An Introduction to the State of the Art in Approximate Agreement Algorithms

John M. Hall, *Student, University of Idaho*

**Abstract**— This research reviews the current state of the art in approximate agreement algorithms with respect to survivability. From this standpoint, survivability is interested with reaching agreement under an arbitrary fault model. This research found the quite mature fault models to be beneficial to guaranteeing system survivability. Other current research has also improved the applicability of approximate agreement. In particular, the research into partially connected systems is promising.

**Index Terms**— Approximate Agreement, Survivability, Byzantine, Malicious, Convergence, Algorithms, Survey.

## I. INTRODUCTION

**D**ISTRIBUTED agreement was first formulated in [14]. The same group made the problem more colorful and added several solutions in [11]. This paper introduced the Byzantine Generals Problem. In this problem, a set of processes must reach a common agreement. More specifically, one process initiates the communication and is known as the general. If the general sends the same value to each process, then all non faulty processes must agree on that value. Otherwise, all non-faulty processes must at least agree on the same value. This problem is greatly complicated by the fact that there may be malicious processes involved that may behave arbitrarily to try to disrupt the agreement formed. This paper introduced the now classic results under several assumptions that with  $m$  malicious processes, agreement can only be guaranteed if there are a total of at least  $3m + 1$  processes. This research sparked a wide range of associated research. For example, the same group investigated relaxing the requirements in [12]. One of the most interesting research directions investigates approximate agreement.

Approximate agreement was first posed in [7]. This research investigated trying to get a group of processes to agree not on an exact value, but within some error of a common value. This research introduced the idea of an algorithm that would cause all non faulty processes to converge upon a common value. They showed that a provable convergence rate could be found if and only if there were a total of at least  $3m + 1$  processes. The more interesting case was their introduction of an asynchronous algorithm. This is particularly interesting since an asynchronous algorithm is not even possible for the original Byzantine Generals Problem. The approximate agreement algorithm can work in virtually all real world problems because the error can be chosen to be arbitrarily small. The requirements for convergence in this algorithm can

not handle as many malicious processes. There must be at least  $5m + 1$  processes for convergence to be guaranteed.

There are two requirements of any approximate agreement algorithm:

- 1) *Agreement* - All non-faulty processes come to agree on values that are within some defined  $\epsilon$  of each other.
- 2) *Validity* - The value chosen by a non-faulty process will be within the range of values initially proposed by non-faulty processes.

Approximate agreement is often misunderstood. Agreement in this sense is not like typical human agreement. In human agreement, there is meaning attached to the numbers. In approximate agreement, this can be misleading. An approximate agreement algorithm is only guaranteed to converge to a value that is within the range of initial values of non malicious processes. This means that in order to attach any meaning to the resulting values, all non faulty processors must produce an acceptable value, and that it does not matter which value within this range is chosen, as long as the values chosen are close together. Malicious processes could cause the non-faulty processes to come to agreement at any point within this range.

Approximate agreement algorithms have many real world uses. Measuring temperature or a similar phenomenon with redundant sensors is a common application. Each non-faulty reading is an appropriate value, but a faulty or malicious sensor may attempt to cause the sensor group to agree on an inappropriate value. Clock synchronization is another such application. However, in these applications the actual time is not important. It is only important that the clocks are synchronized to the same time. Modifying the Byzantine Generals Problem to trying to agree on an attack time would be another such example. It is important that all commanders attack at the same time, but it is not important what that time is.

An approximate agreement algorithm can be assessed by several factors. The most important factor to take into account when assessing agreement algorithms in general is the fault model that they deal with. This paper is geared to the survivability of the various algorithms. As a result, only algorithms that allow human malicious or Byzantine faults will be considered. Papers such as [1] and [16] do not address such arbitrary faults, and will not be considered. When dealing with approximate agreement algorithms, the convergence rate is also critical. The convergence rate is the factor by which the range of values between non-faulty processes is reduced at each step of the algorithm. This rate has a large impact on the time feasibility of the algorithm in actual applications. The

number of messages passed per round is also important. This can have a large impact on the scalability of the algorithm.

## II. BACKGROUND

In order to describe approximate agreement algorithms and their properties, it is necessary to first construct the mathematical background used. These definitions are similar to many provided in the literature [2], [4], [5], [6], [7], [9], [10], [14], [15]. These variations define the terminology to be used in the remainder of this paper.

### A. Multi-Sets

Each process performing approximate agreement will use a multi-set to hold the values presented by other processes. A multi-set contains items in non-decreasing order. A multi-set may contain duplicates. In other words, if  $\mathbf{V}$  is a multi-set containing  $n$  elements, then

$$\mathbf{V} = \langle v_1, v_2, \dots, v_n \rangle \text{ such that } v_i \leq v_{i+1} \quad (1)$$

A multi-set has several properties of interest. The range of a multi-set, represented by  $\rho$ , is the range of values between the smallest and largest values in the multi-set. Mathematically this is written,

$$\rho(\mathbf{V}) = [v_1, v_n] \quad (2)$$

In order to measure how big this range is, the diameter is defined to be

$$\delta(\mathbf{V}) = (v_n - v_1) \quad (3)$$

Throughout this paper,  $\mathbf{V}_i$  will refer to process  $i$ 's voting multi-set. The method that is used to populate this multi-set is dependent on the algorithm. Another multi-set of interest is  $\mathbf{U}_{all}$ . This is the multi-set containing all correct values. In this context, a correct value refers to one that a non-faulty process created. From the context of any single process, it is impossible to determine the exact contents of  $\mathbf{U}_{all}$ . However, given constraints such as the maximum number of faults tolerated, some properties of this multi-set can be determined.

The original agreement and validity constraints may be rewritten using this multi-set notation:

- 1) *Convergence* -  $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| \leq C\delta(\mathbf{U}_{all})$  where  $0 \leq C < 1$
- 2) *Validity*  $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{all})$

This notation introduces a couple of new concepts. First of all, the agreement constraint is renamed convergence. This new constraint only says that the results of approximate agreement will reduce the range of values held by non-faulty processes by at least some constant  $C$ . The original agreement constraint required that the processes came to agreement within a value of  $\epsilon$ . In order for our new constraint to satisfy the old constraint, the algorithm need only be repeated until  $\delta(\mathbf{U}_{all})$  is less than  $\epsilon$ . As should now be clear,  $C$  is the rate at which the values held by non-faulty processes converges. Hence,  $C$  is known as the convergence rate. The smaller the convergence rate, the faster the values held by non-faulty processes will

be guaranteed to converge. The convergence rate is actually the worst case rate of convergence. The rate may be faster depending on the values presented by each process.

The function  $F$  which appeared in the above constraints is applied by each process to the multi-set of values received. This function is known as the voting function. There are several classes of voting functions. This function may be egocentric, egophobic, or anonymous. An egocentric algorithm will favor values that are similar to those created by the local process [2]. These are somewhat popular algorithms and have several advantages under certain circumstances. Egophobic algorithms favor values created by remote processes. These are not as popular, but may be found in many clock synchronization algorithms [6]. Anonymous algorithms are the most common and include a family of algorithms known as Mean Subsequence-Reduced (MSR) algorithms [9].

### B. General Algorithms

MSR algorithms are the most common voting function used by approximate agreement algorithms. These algorithms all have the same basic form. If  $F(\mathbf{V}_i)$  is a MSR algorithm then

$$F(\mathbf{V}_i) = \text{mean}(\text{Sel}_\sigma(\text{Red}^\tau(\mathbf{V}_i))) \quad (4)$$

$$\text{mean}(\mathbf{V}) = \frac{\sum v_i}{|\mathbf{V}|} \quad (5)$$

$$\text{Red}^\tau(\mathbf{V}) = \langle v_\tau, v_{\tau+1}, \dots, v_{n-\tau} \rangle \quad (6)$$

This algorithm is made up of several components. The outside function,  $\text{mean}(\mathbf{V})$  is a standard arithmetic mean function. The average of all values in  $\mathbf{V}$  is calculated. The inside function,  $\text{Red}^\tau(\mathbf{V})$  removes the  $\tau$  largest and the  $\tau$  smallest elements from  $\mathbf{V}$  and return the resulting multi-set.  $\text{Red}^\tau$  is used to remove extreme values from a multi-set. Typically, under MSR fault modes, the number of values removed from either extreme is equal to the total number of faults allowed that are allowed by the fault model to transmit invalid values to other processes. The other function,  $\text{Sel}_\sigma$  returns a multi-set containing  $\sigma$  items. However, the implementation of this function defines the specific MSR algorithm used. This function will be defined differently for every MSR algorithm.

The choice of this  $\text{Sel}_\sigma(\mathbf{V})$  function can have a large impact on the overall approximate agreement algorithm. Some functions result in fast convergence rates and some allow the agreement to diverge. These differences are interesting because they do not make immediate intuitive sense. For example, simply returning the median of  $\mathbf{V}$  will cause the algorithm to diverge. On the other hand, returning the first and last entries in  $\mathbf{V}$  will result in convergence, but returning every other entry in  $\mathbf{V}$  will cause the algorithm to converge even quicker. In a few cases, the definition of  $\text{Sel}_\sigma$  is not the only parameter defining a new MSR algorithm. The method of populating the multi-set before voting may also change between algorithms. Every definition of the  $\text{Sel}_\sigma$  function has an associated  $\gamma$ . This value is the span of the selection method. In a sense, it represents how evenly the function selects values from the

multi-set passed in. All MSR algorithms have the following convergence rate [5]

$$C = \frac{\gamma\alpha}{\sigma} \quad (7)$$

Another somewhat common algorithm type is a Mean of a Subsequence of an Egocentric (MSE) voting algorithm [2]. If  $F(\mathbf{V}_i)$  is an MSE algorithm then

$$F(\mathbf{V}_i) = \text{mean}(Sel_\sigma(\mathbf{V}_i)) \quad (8)$$

In its simplest terms, an MSE voting algorithm will operate on the entire range of values in its voting set. In terms of tolerating faults, either benign or malicious, it is impressive that these algorithms can converge even though they are truly operating with values from malicious processes. In general, an algorithm capable of operating in both MSR and MSE models will converge faster in the MSR model [2].

### III. REVIEW

The original approximate agreement paper [7] includes several algorithms. These algorithms assume that all faults are Byzantine in nature. The fault tolerant midpoint algorithm is an MSR algorithm as described above. The  $Sel_\sigma$  function will simply select the first and last elements of the multi-set passed in. Since the multi-set is sorted, this will result in the largest and smallest of the reduced voting-set to be averaged. The number of rounds required by this algorithm as portrayed is [7]

$$\left\lceil \log_C \left( \frac{\delta(\mathbf{V})}{\epsilon} \right) \right\rceil \quad (9)$$

$$C = c(n - 2m, m) \text{ (synchronous)} \quad (10)$$

$$C = c(n - 3m, 2m) \text{ (asynchronous)} \quad (11)$$

$$c(m, k) = \left\lceil \frac{m-1}{k} \right\rceil + 1 \quad (12)$$

Since a malicious process may cause  $\delta(\mathbf{V})$  to be arbitrary large, the agreement can be delayed. The  $C$  value above is the convergence rate of the algorithm. Since, in this case, the fault model requires  $N \geq 3m + 1$  in the synchronous case and  $N \geq 5m + 1$  in the asynchronous case, the convergence rate is [7]

$$C \leq \frac{1}{2} \quad (13)$$

The next algorithm presented is the fault tolerant mean [7]. This is also an MSR algorithm where the  $Sel_\sigma$  function simply returns all elements of the multi-set passed in. This algorithm will also converge. Its convergence rate is [7]

$$C = \frac{m}{N - 2m} \quad (14)$$

The final algorithm presented is the single-mode optimal algorithm. This algorithm is also an MSR algorithm. The  $Sel_\sigma$  function returns the first and every  $k$ th successive element in the multi-set passed in. This algorithm will also converge. Its convergence rate is [2]

$$C = \left( \left\lceil \frac{N - 2t - 1}{t} \right\rceil + 1 \right)^{-1} \quad (15)$$

These algorithms have been greatly improved upon. The largest improvements have been with regard to the fault model. These original algorithms considered all faults to be Byzantine. Byzantine faults are those that allow for any behavior by a faulty processes [11]. In other words,  $t = m$  where  $t$  is the total number of faults and  $m$  is the number of malicious faults. For many applications, this is over conservative. By refining the fault model, the bounds shown in future papers could be improved.

The first such refinement is shown in [9]. For synchronous systems, Byzantine faults were subdivided into  $a$  asymmetric,  $s$  symmetric, and  $b$  benign faults. The total number of faults  $t$  is  $a + s + b$ . Benign faults are those which are self evident to all other processes. Symmetric faults are those faults that cause a process to send the wrong value to other nodes, but to otherwise perform the protocol correctly. An asymmetric fault may consist of any other behavior. This division of the fault model allowed fewer processors to guarantee convergence if some of the faults expected were symmetric or benign. From a survivability sense, this is important in cases where a malicious user may cause a process to fail in either a benign or a symmetric manner. For this reason, this model is more useful to a system designer trying to harden a system than the previous model. The mixed mode optimal algorithm essentially extends the single-mode optimal algorithm above to handle symmetric and asymmetric faults independently. The resulting convergence rate is [9]

$$C = \left( \left\lceil \frac{N - 2t + b - 1}{a} \right\rceil + 1 \right)^{-1} \quad (16)$$

The number and composition of faults that this model can handle in the synchronous case is [9]

$$N \geq 3a + 2s + b + 1 \quad (17)$$

Despite the fact that these algorithms were defined as MSR algorithms, the  $Sel_\sigma$  functions can be placed in an MSE algorithm [2]. The convergence rates of the four modified algorithms we have seen are not as good when voting from the whole voting multi-set. In fact the fault tolerant midpoint algorithm diverges [2]. The convergence rates for the fault tolerant mean, the single-mode optimal, and the mixed-mode optimal modified MSE algorithms are respectfully [2]

$$\frac{3a + 2s}{N - b} \quad (18)$$

$$\frac{3}{\left\lceil \frac{N-b-1}{t} \right\rceil + 1} \quad (19)$$

$$\frac{3}{\left\lceil \frac{N-b-1}{t-b} \right\rceil + 1} \quad (20)$$

The fault model was somewhat expanded by the introduction of the BSO algorithm [15]. The BSO algorithm does not seem to fit cleanly into the MSR category. The fault model

introduced included omissive faults. These were removed from both the asymmetric and symmetric categories. The total number of faults in this model was  $t = a' + s' + o + b$ . Very briefly, this algorithm replaces values that are not received from another process with a default. However, the algorithm manipulates the voting multi-set until the non-faulty values are a majority. It then proceeds very much like a typical MSR algorithm. The resulting convergence rate is [15]

$$C \leq \left( \left[ \frac{N - 2 - 2s - o}{b + o/s} \right] \right)^{-1} \quad (21)$$

In order to converge, BSO requires [6]

$$N \geq 3a' + 2s' + \left\lceil \frac{3}{2}o \right\rceil + 1 \quad (22)$$

A additional level of refinement was shown in [6]. This paper carried the prior fault modes to asynchronous modes and presented a unified model. It also extended asynchronous faults, by dividing both asymmetric and symmetric faults into omissive and transmissive categories. Omissive faults are those where some number of messages are not sent to some processes. This resulted in  $t = (a' + \omega_{a1}) + (s' + \omega_s) + b$   $a'$  is the number of transitive asymmetric faults,  $\omega_{a1}$  is the number of occurring asymmetric omissive faults,  $s'$  is the number of transitive symmetric faults, and  $\omega_s$  is the number of occurring strictly omissive faults. In real-world systems, this division is particularly important. Often it is much easier for an attacker to perform a denial of service attack on a component, then to cause the component to perform in a Byzantine manor. Using this fault model shows that causing such strictly omissive fault behavior is also easier to guard against. This is also particularly important in real-world systems that are not guaranteed tamperproof communication channels. These systems may use cryptographic signatures to guarantee that the communications have not been modified in transit. Since the receiving side will discard a message with the incorrect signature, an attacker who controls only the communication channel could be limited to only creating strictly omissive faults.

The algorithm proposed to deal with omissive faults is also an MSR algorithm [6]. This algorithm works exactly as seen above. However, instead of replacing omissive values, they are removed from the voting set. The prior practice of replacing these values with a default value changed what was originally an omissive fault into a transmissive fault. The modified algorithm is called OMSR. The algorithm is more complex than its predecessors since the voting algorithms on different processors may be using different sized multi-sets. As a result, the convergence rate must be written in two parts [6]

$$C \leq \max \left( \frac{\gamma_{j,a_j}}{\sigma_j}, \frac{(\sigma_i - \sigma_j) + \gamma_{j,a_j}}{\sigma_i} \right) \quad (23)$$

where  $\gamma$  is the span of the  $Sel_\sigma$  function as defined loosely in the general algorithm section above. For more information, see [6]. The total number of processes required is

$$N \geq 3a' + 2s + \omega_{a1} + \omega_s + b + 1 \quad (24)$$

This refinement of the fault model was extended to synchronous systems in [5]. This final fault model represents the state of the art today.

Another interesting direction investigated with respect to approximate agreement algorithms are partially connected graphs [8], [10], [3]. Most research into approximate agreement algorithms assumes that the processes are fully connected. This means that each process may communicate directly to every other process. This is a serious limitation to the applicability of these algorithms in actual systems. The number of needed connections  $k$  increases

$$k = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2} \quad (25)$$

This becomes prohibitive for large  $n$ . Partially connected graphs are a more reasonable model of actual systems. Therefore, this research tries to solve the approximate agreement algorithm where nodes may only talk to neighbor nodes. In some cases, relaying a message from one host to another is allowed. Papers such as [10] and other earlier papers [8] investigated this problem for synchronous systems. This research was extended in [4] to include asynchronous systems. The primary limitation of all research to this point is that it applies only to local convergence, through it also divides local convergence into union and intersection convergence. This paper investigated using partially connected meshes for process communications. This showed that the more connected the meshes were, the more resilient to faults they could be made. This research only shows local convergence. Under a Byzantine fault model, there has been no research that manages global convergence. However, there has been success under limited fault models. For example, in a hexagon mesh, global agreement was shown for omissive faults [3].

The algorithms used in partially connected graphs are so far essentially the same algorithms defined above. The requirements for union and intersection convergence are a little higher. For synchronous and asynchronous networks respectively, union convergence requires [8]

$$N \geq 3a + 2(s + f) + 1 \quad (26)$$

$$N \geq 5a + 4(s + f) + 1 \quad (27)$$

The  $f$  parameter in the above equations is the maximum number of faults from either the processes reachable by a process  $i$  from all those reachable from both processes  $i$  and  $j$ , or the processes reachable by  $j$  from all those reachable from both processes  $i$  and  $j$ . For synchronous and asynchronous networks respectively, intersection convergence requires [8]

$$N \geq 3a + 2(s + \chi) + 1 \quad (28)$$

$$N \geq 5a + 4(s + \chi) + 1 \quad (29)$$

The  $\chi$  parameter is the number of processes reachable by both process  $i$  and process  $j$ . The current results for local convergence in partially connected networks is basically

that intersection convergence is more difficult than union convergence, and that asynchronous is more difficult than synchronous. Also, the more connections that are present, the more types of convergence are possible.

The voting algorithm typically used in approximate agreement is known as an MSR algorithm. There are several types of MSR algorithms. These types and their bounds for optimal performance are discussed in [2]. These types are anonymous, egophobic, and egocentric. An anonymous is the simplest of these. The voting algorithm will treat all values from all sources as equivalent. An egophobic algorithm will favor values that are farthest from the locally calculated value. An egocentric algorithm will favor values close to the locally calculated value. Some algorithms use egocentric algorithms to allow the local process to treat some malicious faults as benign [13]. We will discuss this behavior in more detail below.

Another interesting research area has so far not been successfully applied to full Byzantine faults. The work in [13] incorporates graceful degradation into approximate agreement under an incomplete fault model. This is nonetheless an interesting result that would be even more interesting if it could be extended to Byzantine faults. In this paper, graceful degradation occurred when between  $\frac{1}{3}$  and  $\frac{2}{3}$  of the processes were faulty. In these cases, either the algorithms would work correctly for a non faulty node, or it would be aware of the fact that too many faults had occurred to continue. This would be useful in a Byzantine process to better allow a system to fail safe even if agreement cannot be reached. This algorithm is an extremely egocentric algorithm. All values that are not within an allowed range of the locally used value are ignored. This has many non favorable implications when dealing with Byzantine faults. However, It would seem as though there were no immediate blocks to applying the core failure detection scheme to an algorithm capable of dealing with a Byzantine fault model. I was unable to locate any recent research addressing this point.

#### IV. CONCLUSION

Approximate agreement algorithms are useful for masking faults including malicious behavior. The fault model has become fairly detailed and allows for dealing with many types of faults at provable minimal numbers of processors. The convergence rate has also improved over the evolution of this fault model. However, the most common case in the literature, fully connected systems, are not suitable for many large applications. The large number of communication paths required are prohibitive.

The evolution of the fault model has had several benefits to the survivability field. Even though the survivability field must always be prepared to handle asymmetric faults, the latest fault model allows additional ommissive faults to be handled at a fairly low cost in terms of additional processors. The main problem when trying to build survivable systems using approximate agreement is dealing with common mode failures. Realistically building enough systems to handle a reasonable number of faults, malicious or otherwise, typically requires replicating the same system multiple times. Unfortunately,

since these systems will have the same vulnerabilities, they will not fail independently under a human malicious attack.

The most promising work for the application of approximate agreement algorithms is dealing with partially connected graphs. These graphs are much more accurate representations of how large systems are built. Establishing global convergence under a Byzantine fault model would allow much larger applications to utilize approximate agreement.

#### REFERENCES

- [1] Attiya, Hagit, Nancy Lynch, Nir Shavit, *Are Wait-Free Algorithms Fast?* Journal of the ACM, Vol. 41, No. 4, July 1994, pp. 725-763, available at <http://citeseer.nj.nec.com/attiya95are.html>.
- [2] Azadmanesh, M. H., A. W. Krings, *Egocentric Voting Algorithms*, IEEE Transactions on Reliability, Vol. 46, No 4, Dec. 1997, pp 494-502.
- [3] Azadmanesh, M. H., A. W. Krings, *Network Convergence in the Presence of Omission Faults*, 12th International Conference on Parallel and Distributed Computing Systems, August 1999, pp 416-423.
- [4] Azadmanesh, M. H., R. M. Kieckhafer, *Asynchronous Approximate Agreement in Partially Connected Networks*, International Journal of Parallel and Distributed Systems and Networks, Vol. 5, No 1, 2002, pp. 26-34.
- [5] Azadmanesh, M. H., R. M. Kieckhafer, *Exploiting Omissive Faults in Synchronous Approximate Agreement*, IEEE Transactions on Computers, Vol. 49, No. 10, October 2000, pp. 1031-1042.
- [6] Azadmanesh, M. H., R. M. Kieckhafer, *New Hybrid Fault Models for Asynchronous Approximate Agreement*, IEEE Transactions on Computers, Vol. 45, No. 4, 1996, pp. 439-449.
- [7] Dolev, Danny, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, William E. Weihl, *Reaching Approximate Agreement in the Presence of Faults*, Proc. 3rd Symp. on Reliability in Distributed Software and Database Systems, Oct. 1983, pp. 145-154, available at <http://citeseer.nj.nec.com/dolev85reaching.html>.
- [8] Kieckhafer, R. M., M. H. Azadmanesh, *Low Cost Approximate Agreement in Partially Connected Networks*, Journal of Computing and Information, Vol. 5, No. 1, 1993, 53-85.
- [9] Kieckhafer, R. M., M. H. Azadmanesh, *Reaching Approximate Agreement With Mixed Mode Faults*, IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 1, 1994, pp. 53-63.
- [10] Kieckhafer, R. M., M. H. Azadmanesh, *A Unified Approach to Synchronous and Asynchronous Approximate Agreement in the Presence of Hybrid Faults*, IEEE Transactions on Reliability, Vol. 44, No. 4, Dec. 1995, pp. 622-631.
- [11] Lamport, Leslie, Robert Shostak, Marshall Pease, *The Byzantine Generals Problem*, 1982, available at <http://citeseer.nj.nec.com/lamport82byzantine.html>.
- [12] Lamport, Leslie, *The Weak Byzantine Generals Problem*, Journal of the Association for Computing Machinery, Vol. 30, No. 3, July 1983, pp 668-676.
- [13] Mahaney, Stephen R., Fred B. Schneider, *Inexact Agreement: Accuracy, Precision, and Graceful Degradation*, Proc. 5th ACM SIGACT-SIGOPS Symp. on Principles in Distributed Computing, August 1985, pp 237-249, available at [http://www.cs.cornell.edu/fbs/publications/inexact\\_agreement.ps](http://www.cs.cornell.edu/fbs/publications/inexact_agreement.ps).
- [14] Pease, M., R. Shostak, L. Lamport, *Reaching Agreement in the Presence of Faults*, J. ACM Vol. 27, No 2, Apr. 1980, pp 228-234.
- [15] Plunkett, Richard, Alan Fekete, *Approximate Agreement with Mixed Mode Faults: Algorithm and Lower Bound*, LNCS 1499, p. 333, available at <http://citeseer.nj.nec.com/215625.html>.
- [16] Schenk, Eric, *Faster Approximate Agreement with Multi-Writer Registers*, Proc. 36th IEEE Symposium on Foundations of Computer Science, 1995, pp. 714-723, available at <http://citeseer.nj.nec.com/context/80523/176876>.

**John M. Hall** John graduated summa cum laude from Washington State University with a Bachelor's of Science in computer science in 1999. After graduating, John took a job in Boise with Hewlett-Packard. In the fall of 2002, Hewlett-Packard offered John a fellowship to continue his education at the University of Idaho. John expects to graduate with a Master's of Science in December of 2003. John's research interests include network security and

evolutionary computing. John looks forward to returning to Boise to enjoy the ample rock-climbing and kayaking opportunities as well as to apply his recent education.