# Global Convergence[1] in Partially Fully Connected Networks (PFCN) with Limited Relays

Azad H. Azadmanesh [2], Axel W. Krings[3], Bahador Ghahramani[4]

*Abstract* – In a distributed system, it is often necessary for nodes to agree on a particular event or to coordinate their activities. Applications of distributed agreement are many, such as Commit Protocols in distributed database systems, selection of a monitor node in a distributed system, detecting an intruder, or agreeing on the malicious behavior of a node.

Among many forms of Distributed Agreement, one form is called Approximate Agreement (AA), in which the nodes, by exchanging their local values with other nodes, need to agree on values which are approximately equal to each other. Research on AA for fully connected networks is relatively mature. In contrast, the study of AA in partially connected networks has been very limited. More specifically, no general solution to the AA problem exists for such networks. This research solves the AA problem for a specific, scalable, partially connected network with limited relays. The research considers the worst failure mode of nodes, called Byzantine, and hybrid failure modes. The results show low communication cost in comparison to fully connected networks. The network is designed to take advantage of the results available for fully connected networks - thus the analysis for obtaining the expressions for Convergence Rate and Fault Tolerance becomes relatively easy.

*Keywords*— Approximate Agreement, Byzantine Agreement, Clock Synchronization, Convergent Voting Algorithms, Fault-Tolerant Multiprocessors, Mixed-Mode Faults.

**Submitted to**: **IJITDM**
First submission: April 30, 2002
Second Submission: Sep 9, 2002

[1]This research has been supported in part by NIST grant number 60NANB1D0116.
[2]University of Nebraska at Omaha, Dept. of Computer Science, Omaha, Ne 68182-0500
[3]University of Idaho, Dept. of Computer Science, Moscow, Idaho 83844-1010
[4]University of Nebraska at Omaha, Dept. of Information Systems, Omaha, Ne 68182-0392

# 1 Introduction

A distributed system can be described as an interconnection of autonomous nodes[5]. Many critical applications make use of such systems to achieve higher reliability and performance. However, distributed systems are more prone to failures due to higher redundancy of components. In such a system, to make an application survivable in spite of component failure or intrusion, the nodes must be able to co-coordinate their activities. This requires the nodes to execute some form of "agreement" protocol, which depends on the application.

One form of agreement is called Approximate Agreement (AA) in that the nodes must decide on final values which are different from each other by a maximum of $\epsilon$ [6, 8]. Some applications of AA are in synchronizing the local clocks of a distributed system, and sensor data management, where the sensors do not necessarily show the same value [4, 7, 10]. The agreement problem is simple when every node behaves correctly, but it becomes notoriously difficult when some nodes do not behave according to their specifications. AA is defined by the following two conditions [5, 6]:

- *Agreement* – The voting algorithms executed by all non-faulty nodes eventually halt with voted values that are within $\epsilon$ of each other.
- *Validity* – The voted value held by each non-faulty node is within the range of the initial values held by the non-faulty processes.

The first condition ensures that all non-faulty nodes agree on almost the same value, and the second condition enforces the agreed upon values to be in the range of good values held by non-faculty nodes.

**Convergent Voting Algorithms** – The vast majority of voting algorithms for AA employ "rounds" of message exchange coupled with voting functions which guarantee that the

[5]Herein, the word "node" and "process" running on a node are used interchangeably.

difference between the values held by non-faulty processes is reduced toward zero in each round [3, 5, 6, 8, 10]. This property is called *single-step convergence*.

In each round, every correct node exchanges its local value with other nodes. Upon the receipt of values from the other nodes, each node executes a function $F$, producing a new value. This new value is used as the voted value for the current round. This process continues until the Agreement condition is met. The Validity condition is met if in each round, $F$ produces a value within the range of the correct values received.

**Contributions** – Much work has been conducted for solving the AA problem for Fully Connected Networks (FCN). However, as the number of nodes increases, FCN becomes very expensive in terms of communication resources. On the other hand, although most systems are partially connected, there has been little focus on Partially Connected Networks (PCN). Therefore, the primary objectives of this research are:

- Design a network model, which is not fully connected, is homogeneous, regular, and easily scalable.
- Devise a methodology that can easily take advantage of currently available results for FCN.
- Employ limited amount of message relays.
- Determine the performance and Fault Tolerance under single mode and hybrid fault models.
- Compare the results with those of fully connected networks. Since there has been no solution for any partially connected network, the results can not be compared to other partially connected networks.

**Organization** – Section 2 provides background information on communication system protocols, different fault types, and the relevant studies of the AA problem. Section 3

explains a network model called *Partially Fully Connected Network* (PFCN), and obtains the performance and fault tolerance of the network under the worst-case single fault mode (Byzantine). It also compares the results with fully connected networks. Section 4 studies the same network model under a hybrid fault model consisting of three fault modes. Section 5 obtains the optimal structure of the PFCN model for a given number of nodes. Section 6 considers the communication cost of the PFCN model and compares it with fully connected networks. Section 7 deals with variations of PFCN under the Ethernet network protocol. This section will show that performance, fault tolerance, and communication cost are further improved. Finally, Section 8 gives a summary and some remarks on future studies of AA.

# 2 Background

## 2.1 Synchronous vs. Asynchronous Systems

Distributed systems can be partitioned into two classes: *synchronous* systems and *asynchronous* systems. In a synchronous system, there are finite bounds on the processing and communication delays of non-faulty nodes [5], in that there exists a point in time by which any node executing a voting algorithm knows that it has received all data from all non-faulty nodes. Therefore, any data arriving after that time must have come from a faulty node.

By contrast, an *asynchronous* system imposes no finite bounds on node operations [5]. Thus, there is no fixed time by which a node is guaranteed to have received a value from all non-faulty nodes. Each node must decide for itself when to stop waiting for messages and begin the voting algorithm for the current round [5]. This paper is restricted to the study of *synchronous systems only*.

## 2.2 Fault Models

In most studies of AA, faults are assumed to behave in the worst-case *Byzantine* manner. A Byzantine faulty node behaves maliciously and arbitrarily, and thus no assumption can be

made about its behavior. For example, it can send an erroneous value to some nodes and no value to other nodes, or transmit conflicting messages to every node. Thus, Byzantine faults have also been called two-faced or asymmetric. In fault tolerant systems, however, it is unlikely that all faults exhibit this worst-case behavior. Thus, the assumption that all faults are worst-case yields overly conservative estimates of fault-tolerance and reliability. Consequently, a more realistic approach to fault tolerant distributed systems design and analysis is to account for the co-existence of different types of fault modes with mixed severities. In one of the early studies, Thambidurai and Park [11] partitioned the faults into $b$ *benign*, $s$ *symmetric*, and $a$ *asymmetric* faults. The total number of faults in the system is $t = a + s + b$. In contrast to Byzantine (asymmetric) faults, the behavior of symmetric faults is perceived identically by *all* nodes, i.e. a node transmits the same erroneous value to every receiving node. A benign fault, also called, manifest fault, is defined as a self-incriminating or self-evident fault to all nodes. It is a fault whose erroneous behavior is detected by all nodes, e.g. transmitting a value outside of a timing-window. Voting algorithms using different fault modes usually produce better fault tolerance [4, 8, 11].

Later Azadmanesh and Kieckhafer [1] extended the three fault model to a five-mode fault model. The five-mode Omissive/Transmissive Hybrid (OTH-5) fault model distinguishes between omissive and transmissive behavior of symmetric and asymmetric fault modes. A transmissive fault is one which delivers erroneous value(s) to one or more receiving nodes. By contrast, an omissive fault fails to deliver a value to one or more receiving nodes. Specifically, these fault behaviors are:

**Transmissive asymmetric** A transmissive asymmetric fault can exhibit any form of arbitrary asymmetric behavior, i.e. it can deliver different erroneous values to different receivers.

**Strictly omissive asymmetric** Such a fault can send a single correct value to some nodes and no value to all other nodes. However, it can not transmit an erroneous value to any receiver.

**Transmissive symmetric** A transmissive symmetric fault delivers a single erroneous value

to all receiving nodes.

**Omissive symmetric** An omissive symmetric fault fails to deliver any value to any receiving node.

Figure 1 summarizes the relationships between the fault models discussed.

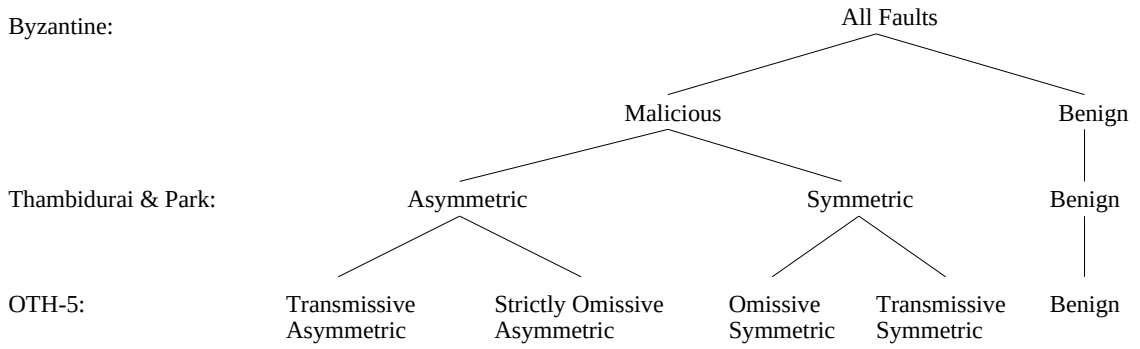


Figure 1: Relationship among fault models

This study considers both single and hybrid fault models. To better understand the behavior of networks discussed, first the analysis considers only Byzantine faults. Later, this restriction is removed and the networks are analyzed under the hybrid fault model consisting of benign, symmetric, and asymmetric faults.

## 2.3 Single-Step Convergence

To define single-step convergence formally, the following definitions are needed:

$\mathbf{V}$ = $\langle v_1, \ldots, v_V \rangle$, where $V = |\mathbf{V}|$, is the multiset of values received and sorted in ascending order.

$\rho(\mathbf{V})$ = $[\,\min(\mathbf{V}), \max(\mathbf{V})\,]$, is the range of values spanned by $\mathbf{V}$.

$\delta(\mathbf{V})$ = $[\,\max(\mathbf{V}) - \min(\mathbf{V})\,]$, is the diameter of $\mathbf{V}$.

$\mathbf{U}_{all}$ = The multiset of all correct values received by non-faulty nodes.

During each round of voting, each non-faulty node $i$ broadcasts its initial value to all nodes

including itself. Then it collects all the values received into the *voting multiset* $\mathbf{V}_i$. Node $i$ then applies a function $F$ to $\mathbf{V}_i$ to attain its latest estimate (voted value) for the round, which it uses as its initial value in the next round. An algorithm is single-step-convergent if both of the following conditions are true following every round of voting:

- $F(\mathbf{V}_i) \in \rho(\mathbf{U}_{all})$, i.e. for each non-faulty node $i$, the voted value is within the range of correct values, and
- $|F(\mathbf{V}_i) - F(\mathbf{V}_j)| < \delta(\mathbf{U}_{all})$, i.e. for each pair of non-faulty nodes, $i$ and $j$, the difference between their voted values is *strictly less than* the diameter of the multiset of correct values received.

Assuming that $\delta(\mathbf{U}_{all}) > 0$, for two arbitrary non-faulty nodes $i$ and $j$, the ratio:

$$C \;=\; \frac{max(|F(\mathbf{V}_i) - F(\mathbf{V}_j)|)}{\delta(\mathbf{U}_{all})}$$

called *convergence rate*, shows the effectiveness (performance) of a convergent voting algorithm. Note that $\mathbf{U}_{all}$ is the multiset of initial correct values before voting, and $F(\mathbf{V}_i)$ and $F(\mathbf{V}_j)$ are the values after voting. If $C < 1$ in each round, it is then guaranteed that after sufficient number of rounds, the system will achieve agreement, i.e. the diameter of correct values held by the non-faulty nodes in the system will be within $\epsilon$.

## 2.4 Definition of MSR Algorithms

MSR voting algorithms are a particular family of voting algorithms with the following general form [6, 8]:

$$F(\mathbf{V}) = \text{mean}\left[Sel_\sigma\left(Red^\tau\left(\mathbf{V}\right)\right)\right]$$

The Reduction function $Red^\tau$ removes the $\tau$ largest and $\tau$ smallest elements from multiset $\mathbf{V}$, which produces the following new multiset:

$$\mathbf{M} = Red^\tau(\mathbf{V}) \;=\; \left\langle v_{(1+\tau)},\ \ldots\ , v_{(V-\tau)} \right\rangle$$

The Selection function $Sel_\sigma$ then selects a submultiset of $\sigma$ elements from $\mathbf{M}$:

$$\mathbf{S} = Sel_\sigma(\mathbf{M}) \;=\; \langle s_1, \;\ldots\;, s_\sigma \rangle$$

The final voted value is the arithmetic mean of the selected multiset. Thus, the final voted value $F(\mathbf{V})$ is the *Mean* of a *Subsequence* of the *Reduced* (MSR) multiset. Some examples of MSR algorithms are the Fault-Tolerant Midpoint and Fault-Tolerant Mean [5], Binary Mean, and Binary Suboptimal algorithms [8].

MSR algorithms require that the voting multisets for non-faulty processes be of the same size ($V_i = V_j$). So, if an expected value is not received, an arbitrary default value is chosen for the missing value. All nodes then apply the same voting algorithm to equal-sized multisets.

**Convergence Rate of MSR Algorithms** – Each element of $\mathbf{S}$ corresponds to one unique element of $\mathbf{V}$. Let $g$ be the index of any element of $\mathbf{S}$, and let $k(g)$ be the index of the corresponding element in $\mathbf{V}$, so that $s_g = v_{k(g)}$. Now given two indices into $\mathbf{S}$, $g$ and $h \in \{1, \ldots, \sigma\}$, where $g \leq h$, define $\Delta k(g,h) = k(h) - k(g)$ as the number of elements in $\mathbf{V}$ *spanned* by $(s_g, \ldots, s_h)$ in $\mathbf{S}$.

Finally, for any non-negative integer $z$, define $\gamma_z$ as the minimum value of $(h - g)$ which ensures that $k(h) - k(g) \geq z$, for any values $g$ and $h$, $g \leq h \leq \sigma$. By this definition, $\gamma_z$ exists only if $|\mathbf{V}| > z$. It has been shown [8] that the convergence rate $C$ of any synchronous MSR algorithm is given by the following simple expression:

$$C \;=\; \frac{\gamma_z}{\sigma} \tag{2.1}$$

where $z$ is the number of asymmetric faults.

## 2.5 Local and Global Convergence

The vast majority of research in convergent voting has considered only completely connected systems [5, 6, 10, 12, 13]. Thus, any node can receive direct messages from all other nodes. If the physical connectivity of the system is not complete, then it is assumed that messages are

relayed by intervening nodes to achieve complete "logical" connectivity. As a system grows large, so does the number of its communication links, or the traffic required for message relays. Thus, the assumption of complete connectivity restricts the application of convergent voting to relatively small systems.

There are two types of convergence without relays: *local* convergence over a specified subgraph, and *global* convergence over the entire network. Local convergence implies global convergence in FCN, since every node is connected to every other node. But this is not necessarily true in PCN. While local convergence is a prerequisite to global convergence in PCNs, it does not guarantee single step global convergence. More specifically, two immediate neighbors $i$ and $j$ may receive values from their respective neighbors that are not shared by $i$ and $j$. Since these values change with every round, nodes $i$ and $j$ may diverge with respect to the previous round. Hence, during the course of global convergence a cluster of local nodes may go through a period of convergence and divergence before they finally converge. As a result, global convergence is *asymptotic* rather than monotonic [9], which makes the proof of global convergence without message relays for any particular PCN very complex.

Some attempts have been made to show global convergence for restricted forms of PCN under simple fault models. For instance, [2] shows global convergence for an octagonal network torus under omission faults only. This study, however, achieves global convergence for a PCN called PFCN (discussed in the next section). The study considers the Byzantine and hybrid fault models. PFCN takes advantage of the benefits and the results already obtained for FCN [8].

# 3 The PFCN Model

## 3.1 System and Communication Models

As indicated in the previous subsection, the relay of messages in a PCN becomes prohibitive as the number of nodes increases. In addition, a PCN design must have the capability of achieving global convergence. Consequently, our network model combines fully connected

and partially connected subnetworks in such a way to limit relays of messages and to guarantee global convergence.

The network model groups the nodes into clusters. Each cluster has the same number of nodes, which are fully connected by point-to-point intra-cluster communication links. Each node in a cluster is connected to its corresponding node in each of the other clusters. Inter-cluster communication is performed by the corresponding nodes in each cluster. The motivation for partitioning nodes into clusters is to localize heavy traffic to the intra-cluster communication[6].

One way to allow clusters to communicate with each other is to elect a member of each cluster as a "gateway" node. The gateway node will then be responsible for exchanging the messages of its own cluster with other clusters. This however makes the design more centralized, making the gateway nodes to be points of failure for their own clusters, preventing convergence at the global level. Therefore, every node is treated the same and has the same function and responsibility as the other nodes.

As an example, the following figure shows a network model with 9 nodes. The nodes are partitioned into three different clusters (groups). Each cluster contains 3 nodes.

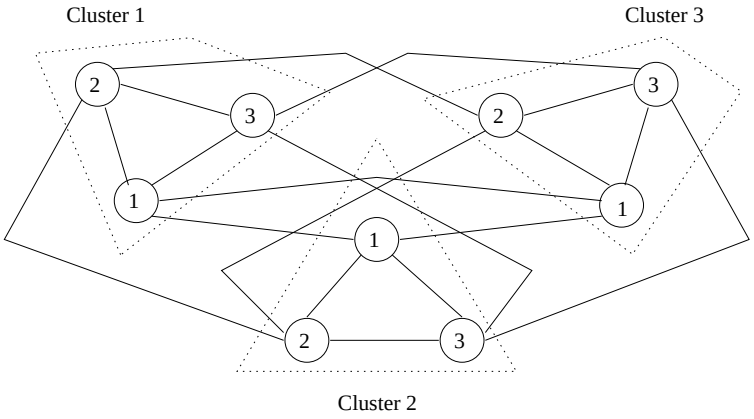


Figure 2: A PFCN model with 3 clusters and 3 nodes in each cluster

Based on this design, we use the following definitions:

[6]Note that the clusters are cliques.

$N$ = Number of nodes in the entire network.

$n$ = Number of nodes in each cluster.

$g$ = Number of clusters (groups) in the network.

$L$ = Number of communication links in the network.

Since each cluster is fully connected, the number of bi-directional links in each cluster is:

$$\frac{n(n-1)}{2}$$

Since there are $g$ clusters, the total number of intra-links is:

$$\frac{gn(n-1)}{2}$$

On the other hand, since each node in a cluster is connected to $(g-1)$ corresponding nodes in other clusters, the total number of inter-connections for each cluster is $n(g-1)$ links. Therefore for $g$ clusters, the total number of bidirectional inter-links is $gn(g-1)/2$. Consequently, the total number of intra- and inter-links in the system is:

$$L = \frac{gn(n-1)}{2} + \frac{gn(g-1)}{2} = \frac{ng(n+g-2)}{2}.$$

Note that the total number of inter-links is the same as if the corresponding nodes in each cluster were fully connected. Both approaches give the same result.

As an example, in Figure 2, $N = 9$, $n = 3$, and $g = 3$. The total number of links is $L = 18$, whereas the total number of links in FCN is $N/(N-1) = 36$. Table 1 shows the percentage of link savings using PFCN in comparison to FCN for different values of $N$, $g$, and $n$. It shows that as the number of clusters increases, the number of links decreases. This issue will be investigated in Section 5.

## 3.2 Implementation of the Voting Algorithm

We assume that the communication system is synchronous and thus all nodes execute the same voting algorithm in lock-steps. Recall that MSR voting algorithms work in rounds.

| PFCN | | | | FCN | Link Reduction |
|---|---|---|---|---|---|
| $N$ | $g$ | $n$ | $L$ | Total Links $N(N-1)/2$ | Percentage |
| 24 | 2 | 12 | 144 | 276 | 48 |
| | 3 | 8 | 108 | | 61 |
| | 4 | 6 | 96 | | 65 |
| 50 | 2 | 25 | 625 | 1225 | 49 |
| | 5 | 10 | 325 | | 73 |
| 100 | 2 | 50 | 2500 | 4950 | 49 |
| | 4 | 25 | 1350 | | 73 |
| | 5 | 20 | 1150 | | 77 |
| | 10 | 10 | 900 | | 82 |

Table 1: Link comparison between FCN and PFCN.

Each round consists of one step, which is the exchange of values with all the neighbors. The voting algorithm for the PFCN model also works in rounds but each round consists of two steps rather than one step.

In the first step, each node in each cluster exchanges its value with all the nodes in other clusters that it is directly connected to. Therefore each node receives $g$ values, including its own, which are stored in an assigned buffer. In the second step of the round, the nodes within the same cluster exchange their buffers. A buffer is treated as a single message, so that each node sends its entire multiset of values received during the first step in $n$ transmissions rather than $gn$ transmissions.

Accordingly, at the end of the second step every node has a multiset of values, with one value from each node in the system. This property enables the PFCN to use the available research results for fully connected networks.

## 3.3 Faults and Their Effects

In order to find the convergence rate, one needs to be concerned with the faults and their effect on convergence. For example, for MSR voting algorithms, the total number of faults is $t = a + s + b$, but the effective number of faults used in the convergence rate expression is the total number of asymmetric faults. Thus, $z_{effective} = a$. Therefore, for MSR algorithms [8]:

$$C = \frac{\gamma_z}{\sigma} = \frac{\gamma_{z_{effective}}}{\sigma} = \frac{\gamma_a}{\sigma} \tag{3.1}$$

which is the same expression as in (2.1).

The same methodology must be applied to the PFCN model. To find the effective number of faults $z_{effective}$, we need to find a scenario with the worst effect on the convergence rate. Assume the maximum number of faults in a cluster is bounded by $f$. Consider one of the clusters. In the worst case, during the first step, each faulty node will receive $(g-1)$ values from the correct nodes in other clusters. These $g$ values, i.e. the $(g-1)$ values received plus the faulty node's own value, are then propagated during the second step to other nodes in the same cluster. Since no assumption can be made about the behavior of the faulty node, these values are considered erroneous. Since there are at most $f$ faulty nodes in a cluster, the total number of erroneous values passed by the faulty nodes in the same cluster during the second step is $fg$.

Furthermore, in the worst case, each good node can receive a bad value from a faulty node in each of the other clusters for a total of $(g-1)$ erroneous values during the first step of a round. In addition, each cluster contains at most $f$ faulty nodes. Thus, the total number of erroneous values received from other clusters by all non-faulty nodes in the same cluster is $f(g-1)$. However, it is possible for the number of non-faulty nodes in a cluster to be less than the number of the faulty nodes, i.e. $(n-f) < f$. As a result, the total number of erroneous values in a cluster is bounded by $r(g-1)$, where $r = \min(n-f, f)$. Therefore, the effective number of erroneous values received by a non-faulty node in the worst case is:

$$z_{effective} = fg + r(g-1) \tag{3.2}$$

The description given above considering the worst scenario for one cluster does not imply a best case scenario in a different cluster. But interestingly, considering a worst case scenario for one cluster indeed generates the exact same number of effective faults for the other clusters.

For MSR algorithms, $\tau = a + s$ represents the total number of malicious faults. In this section, since no distinction is made between symmetric and asymmetric faults, every fault is considered Byzantine, so that $\tau = fg + r(g-1)$. Next section considers PFCN under hybrid fault models.

## 3.4 PFCN Fault Tolerance

Fault tolerance shows the robustness of a voting algorithm, i.e. fault tolerance is the minimum number of nodes $N$ required to guarantee that the voting algorithm is convergent in the presence of a given number of faulty nodes. In [8], it is shown that a voting algorithm can be convergent if $N \geq 3\tau + 1$. Therefore, a PFCN voting algorithm can be convergent if:

$$\begin{aligned} N &\geq 3\tau + 1 \\ &= 3\left[fg + r(g-1)\right] + 1 \end{aligned} \tag{3.3}$$

The following lemma shows that if $n - f = \min(n - f, f)$, then global convergence is not possible. This is because the number of faulty nodes in a cluster would otherwise be greater than the number of the non-faulty nodes in the cluster.

LEMMA **1** : *If $f = \min(n - f, f)$, then a PFCN can be globally convergent.*

PROOF : The proof is by contradiction. If $r = n - f = \min(n - f, f)$, then using (3.3):

$$\begin{aligned} N &\geq 3\left[fg + r(g-1)\right] + 1 \\ &= 3\left[fg + (n-f)(g-1)\right] + 1 \\ &= 3\left[n(g-1) + f\right] + 1 \end{aligned}$$

To ensure that $n - f = \min(n - f, f)$, $f$ can not be less than $n/2$, i.e. $f \geq n/2$. Thus,

$$\begin{aligned} N &\geq 3\left[n(g-1) + f\right] + 1 \\ &\geq 3\left[ng - n + n/2\right] + 1 \\ &= 3\left[ng - n/2\right] + 1 \\ &= 3n(g - 1/2) + 1 \end{aligned}$$

But this value is greater than $N = ng$, which is a contradiction. Accordingly, $f = \min(n - f, f)$ ensures that global convergence will be possible. □

The previous lemma showed that if global convergence is possible then $r = f = \min(n-f, f)$. Thus:

$$\begin{aligned} N_{PFCN} &\geq 3\left[fg + r(g-1)\right] + 1 \\ &= 3\left[fg + f(g-1)\right] + 1 \\ &= 3f(2g-1) + 1 \end{aligned} \tag{3.4}$$

To reach agreement within the entire network, all non-faulty nodes in the network must converge. Therefore, global fault tolerance is achieved through the local fault tolerance of each cluster. It is true that the number of local faults, i.e. faults within each cluster, might vary from cluster to cluster, but the cluster with the maximum number of effective erroneous values will determine the convergence possibility for the whole network. If that particular cluster fails to converge then global convergence can not be guaranteed.

The above discussion shows that by having $r = f$, the effective number of faults obtained in (3.2) becomes:

$$z_{effective} = f(2g-1) \tag{3.5}$$

## 3.5 Convergence Rate of PFCN

As indicated in Section 3.2, at the end of the second step in each round, every node has the same multiset $\mathbf{V}_i$, except for possibly those values transmitted by asymmetric faulty nodes.

This is the same property as that of FCN. Therefore, we can use the same convergence rate expression used for MSR voting algorithms. As a result, using (3.5) yields:

$$C_{PFCN} = \frac{\gamma_{z_{effective}}}{\sigma} = \frac{\gamma_{f(2g-1)}}{\sigma} \tag{3.6}$$

Let us compare the convergence rate and fault tolerance of PFCN against those of FCN. Since $f$ represents the total number of faulty nodes in a cluster, the total number of faults in FCN with the same given $N$ is $fg$. When all faults are considered Byzantine (asymmetric), the convergence rate in (3.1) becomes:

$$C_{FCN} = \frac{\gamma_{fg}}{\sigma} \tag{3.7}$$

and the fault tolerance becomes:

$$N_{FCN} \geq 3fg + 1 \tag{3.8}$$

Accordingly, by comparing (3.4) and (3.8), PFCN needs almost twice as many nodes to have the same fault tolerance of FCN. Also, the rate of convergence for PFCN is slower than that of FCN for the same value of $N$. Assuming that the same selection function is used, this is because $\gamma_{f(2g-1)} \geq \gamma_{fg}$, causing $C_{PFCN} \geq C_{FCN}$.

It must be noted that: 1) this comparison is between a fully connected network and a partially connected network. Partially connected networks are inherently less fault tolerant with slower convergence rate, 2) to date, no results are available for any form of partially connected network that can be compared to PFCN.

# 4 Hybrid Fault Mode Analysis of PFCN

Section 3.4 showed the fault tolerance of PFCN under Byzantine faults. In this section we show the fault tolerance of PFCN under simultaneous presence of *asymmetric*, *symmetric*, and *benign* faults. Since every node is capable of having a complete view of the entire network at the end of second step, and benign faulty nodes, by definition, are self-incriminating, it is

assumed that their values are discarded before the voting process executes, so that for node $i$, $V_i = N - b$.

Assume that each cluster contains at most $a$ asymmetric and $s$ symmetric faults. Similar to the discussion in Section 3.3, the worst case occurs when faulty nodes attempt to relay the values from non-faulty nodes during the second step of each round. Therefore, each asymmetric faulty node will relay $(g-1)$ values during the second step of each round. Hence, the total number of values transmitted during the second step by asymmetric faults, including their own values will be $ag$. Similarly for symmetric faults, the total number of values transmitted during the second step is $sg$. Also, the $r' = \min[n-(a+s), a+s]$ good nodes will transmit $r'(g-1)$ values during the second step. Thus the total number of faults $t$ in the system is:

$$t = (ag + sg) + r'(g-1) + b \tag{4.1}$$

Similar to the single mode Byzantine behavior, for convergence to occur, $r' = a + s$ must be true. As a result:

$$t = (ag + sg) + (a+s)(g-1) + b = a(2g-1) + s(2g-1) + b \tag{4.2}$$

Equation (4.2) shows that the effect of $ag$ asymmetric faults in FCN is equivalent to $a(2g-1)$ asymmetric faults in PFCN. The same is true for symmetric faults.

In our discussion above, it is assumed that symmetric faults are not as malicious as asymmetric faults. More specifically, it is assumed that a symmetric faulty node will relay (in the second round) the same values received in the first step. Otherwise, symmetric faculty nodes must be treated as asymmetric nodes.

**Fault Tolerance** – In FCN, [8] has shown that a voting algorithm is convergent when

$$V \geq 3a + 2s + 1 \tag{4.3}$$

Thus, $V \geq 3a(2g-1) + 2s(2g-1) + 1$ ensures convergence in PFCN. Since $V = N - b$, fault tolerance of PFCN under the hybrid fault model is:

$$N_{PFCN} \geq 3a(2g-1) + 2s(2g-1) + b + 1 = (3a + 2s)(2g-1) + b + 1 \tag{4.4}$$

Note that fault tolerance of PFCN in (4.4) reduces to fault tolerance of FCN in (4.3) when $g = 1$.

**Convergence Rate** – By observing (4.2), the effective number of asymmetric faults is:

$$z_{effective} = a(2g - 1)$$

Thus, convergence rate of PFCN under the hybrid fault model is:

$$C_{PFCN} = \frac{\gamma_{z_{effective}}}{\sigma} = \frac{\gamma_{a(2g-1)}}{\sigma}$$

# 5 Optimality of PFCN Link Structure

For a given PFCN with $N$ nodes, the least value for $L$ is when the number of clusters $g$ and the number of nodes $n$ in each cluster are as close as possible to each other. Intuitively, this makes sense by observing the formula for the number of links:

$$L = \frac{gn(g + n - 2)}{2}$$

To minimize $L$, $gn$ and $(g + n)$ must be minimized. Obviously, one must find values for $g$ and $n$ such that $(g + n)$ is minimized keeping $N = gn$. This occurs when $g$ and $n$ are very close to each other, if they can not be equal. The following lemma proves this formally.

LEMMA **2** : *Given a PFCN model, with the total number of nodes $N = ng$, the minimum number of links $L$ occurs when $g \approx n \approx \sqrt{N}$.*

PROOF : Assume the number of groups is $x$, so that $g = x$ and $n = N/x$. Hence the number of links is:

$$L = \frac{gn(g + n - 2)}{2} = \frac{N(x + \frac{N}{x} - 2)}{2} = \frac{N}{2}(\frac{x^2 - 2x + N}{x})$$

Since $\frac{N}{2}$ is a constant, to minimize the expression, $(\frac{x^2-2x+N}{x})$ must be minimized. Therefore, let us find the derivative of this expression and set the result to zero to find the value for $x$:

$$\left[\frac{x^2 - 2x + N}{x}\right]' = \frac{x(2x - 2) - (x^2 - 2x + N)}{x^2} = \frac{x^2 - N}{x^2}$$

Now let us set the result to zero and find the value for $x$, i.e.

$$\frac{x^2 - N}{x^2} = 0 \Rightarrow x^2 - N = 0 \Rightarrow x = \pm\sqrt{N}$$

Since $x$ can not be negative, $x$ must be $+\sqrt{N}$. As a result $g = \sqrt{N}$, and $n = \frac{N}{x} = \frac{N}{\sqrt{N}} = \sqrt{N}$. □

# 6 Communication Complexity

Communication cost can be measured in terms of the number of messages sent or received, which is called *message complexity*. It can also be measured in terms of the number of bits transmitted, which is called *bit complexity*.

If we consider message complexity, each node in the PFCN sends $(g-1)$ messages during the first step and sends $(n-1)$ messages to the intra-nodes during the second step of each round. Since there are $n$ nodes in each cluster and there are $g$ clusters, the total message complexity in each round will be:

$$\begin{aligned} MC_{PFCN} &= ng\,[(g-1) + (n-1)] \\ &= ng(g+n-2) \end{aligned}$$

In comparison, the message complexity in FCN with $N = ng$ nodes will be $ng(ng-1)$. Therefore, the PFCN model reduces the number of message transmissions significantly. Figure 3 finds the percentage of messages sent by PFCN with respect to a FCN when $n = 5$. For example, when $g = 5$, the number of messages sent by the PFCN is 33% of the messages transmitted by a FCN with $N = ng = 25$. The figure shows higher percentage of message saving as the number of clusters increases. Similar to the issue of link optimality discussed in Section 5, the best message complexity is achieved when $g$ and $n$ are as close as possible to each other.

In terms of bit complexity, assume the overhead of sending a value is $o$ bits, and on average the number of bits in a value is $v$. During the first step, each process sends its value to $g-1$

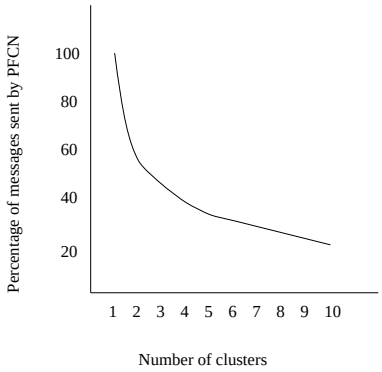


Figure 3: Percentage of messages sent by PFCN as $g$ changes when $n = 5$, in comparison to the number of messages sent by FCN.

nodes in the other clusters. Since each message contains a value and some overhead, the bit complexity in the first step for a single node is $(g-1)(o+v)$. Similarly, during the second step, the bit complexity for a node is $(n-1)(o+vg)$. As a result, the total bit complexity for each round is $ng\,[(n-1)(o+vg)+(g-1)(o+v)]$. In comparison, the bit complexity for FCN is $ng(ng-1)(o+v)$. It is seen that the bit complexity of the PFCN model is again improved mainly because of the saving in the bit overhead during the second step. The bit complexity depends on the protocol because the message size may vary depending on the network protocol. For example, the message size for Ethernet is at most 1.5 KB and for ATM is 53 bytes.

# 7 Variations of PFCN Using Ethernet

This section considers different forms of PFCN using a shared bus medium. One prime implementation of shared bus medium is the Ethernet technology. Ethernet has the following advantages:

- It is physically a broadcast network, rather than point-to-point. This allows a single message transmission to be received by every node.
- Ethernet is restrictive in fault behavior. It is extremely difficult for a node to behave

asymmetrically in a transmissive or omissive mode of failure. This is because to send the same message to every node, a "single" transmission is needed. More specifically:

a. If the transmitting node sends no message, then nobody will receive the message and thus the fault mode is omissively symmetric. Every node will use the same default value to replace the omission, thus transforming the fault behavior into a transmissive symmetric fault.

b. If the transmitting node sends an erroneous message, every node will receive the same value. The fault behavior will again be transmissively symmetric.

With these points in mind, let us look at some variations of PFCN using the Ethernet topology:

1. If $g = 1$, PFCN reduces to FCN. Figure 4 shows the new PFCN model:

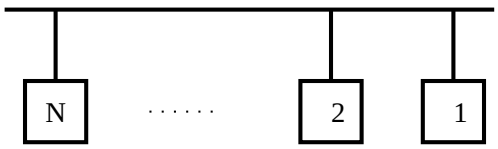


Figure 4: A PFCN, using Ethernet, with $g = 1$.

**Fault Tolerance** – As indicated in (4.4) of Section 4, for a convergent voting algorithm to exist, the following must hold:

$$N_{PFCN} \geq (3a + 2s)(2g - 1) + b + 1 \tag{7.1}$$

With $a = 0$ and $g = 1$, (7.1) reduces to:

$$N_{PFCN} \geq 2s + b + 1$$

**Convergence Rate** – As indicated, the convergence rate of FCN is:

$$C = \frac{\gamma_a}{\sigma}$$

Since $a = 0$, $\gamma_a = 0$. This fact indicates that after one round of broadcast, every node will receive the exact same multiset of values. As a result, every node will end up with the same voted value, which implies perfect convergence, i.e. $C = 0$.

2. If $g = N$, then each group will contain one node. This scenario will reduce the PFCN model to the traditional point-to-point FCN. Thus, the fault tolerance and convergence rate will be:

$$\begin{aligned} N_{PFCN} &\geq 3a + 2s + b + 1 \\ C &= \frac{\gamma_a}{\sigma} \end{aligned}$$

3. If $N > g > 1$, then the groups are point-to-point fully connected, and each group is an Ethernet network. For example, Figure 5 shows the PFCN model when $N = 9$, $g = 3$, and $n = 3$. We observe that whether a node is faulty or not, the values received by the node in the first step will be broadcast symmetrically to other nodes in its own cluster. As a result, no asymmetric behavior takes place in the same cluster.

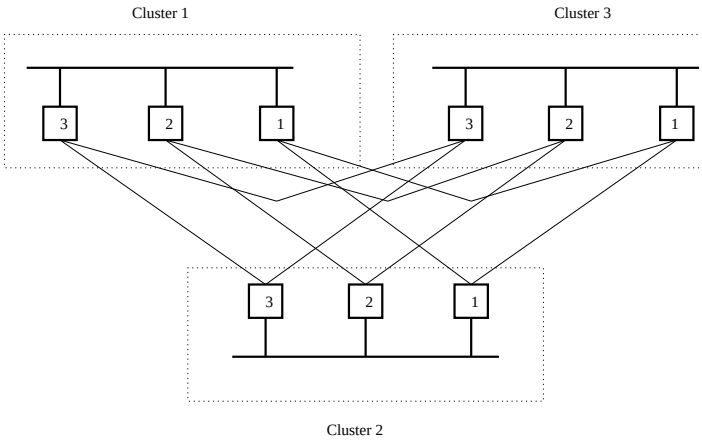


Figure 5: A PFCN model in an Ethernet form, with $n = 3$ and $g = 3$.

**Fault Tolerance** – Although asymmetric values can not propagate in individual clusters, asymmetric behavior is still possible at the inter-link communication level. For example, in Figure 5, assume node 1 of cluster 2 is asymmetrically faulty. In the

first step, node 1 may send conflicting values to nodes 1 in cluster 1 and 3, causing each of the three clusters to have a different value to be used in the second step.

Consequently, since the effect of asymmetric behavior can not be removed, fault tolerance and convergence rate for this form of Ethernet organization stay the same, i.e.:

$$\begin{aligned} N_{PFCN} &\geq (3a+2s)(2g-1)+b+1 \\ C_{PFCN} &= \frac{\gamma_{a(2g-1)}}{\sigma} \end{aligned}$$

## 7.1 Message Complexity (MC)

During the first step, each node sends $(g-1)$ values. During the second step, each node will broadcast one message. Thus, the total number of messages transmitted by one node is $[(g-1)+1] = g$. Since there are $ng$ nodes, the total number of messages transmitted by the entire network, using Ethernet is:

$$MC_{EPFCN} = ng^2$$

Recall that the message complexity of the PFCN, not using Ethernet, is:

$$MC_{PFCN} = ng(g+n-2)$$

This is a large saving of $ng(n-2)$ messages, that is:

$$\begin{aligned} MC_{PFCN} - MC_{EPFCN} &= ng(g+n-2) - ng^2 \\ &= ng(n-2) \end{aligned}$$

Hence, Ethernet topology shows better message complexity if $g > 1$ and $n \geq 3$. Given these lower bounds, the message complexity improves as $n$, $g$, or both increase.

# 8 Summary and Future Research

The problem of AA in partially connected networks is a difficult task to solve and is still open ended. There are a few reasons that make this task difficult. One is that the link

density is less than in a FCN with the same number of nodes, and thus relaying of messages becomes prohibitive. Another reason is that, without message relays, the nodes do not receive values from all other nodes, as in a FCN. Therefore, it is possible for cliques of nodes to be formed, with each clique to converge within its own nodes but show no convergence with other cliques. Finally, having more than one form of PCN makes it very difficult, if not impossible, to devise a general solution that fits all forms of PCN, e.g. irregular networks.

This research has concentrated on a homogeneous, regular, partially connected network, which can take advantage of the results already available for fully connected networks. This network, which is called Partially Fully Connected Network (PFCN), combines fully connected subnets to form a specialized partially connected network. Basically, the nodes are partitioned into clusters, where the nodes within each cluster are connected to the nodes within the same cluster and other clusters in a fully connected fashion. The design of PFCN in such a way has the following advantages:

- The properties of convergence rate and fault tolerance of FCN can easily be extrapolated to PFCN. Thus, fault tolerance and convergence rate can be obtained easily. As a matter of fact, local and global convergence in PFCN are the same.
- The communication complexity is manageable. As shown, the message complexity is improved greatly; specially if Ethernet is used.
- The network is scalable, i.e. as the network grows, nodes and clusters can be easily added, still resulting in a feasible network in terms of fault tolerance, communication complexity, and convergence rate.

Partially connected networks are inherently less fault tolerant than FCNs, due to intermediate faulty nodes for relaying messages. Thus, it is natural for PFCN to be less fault tolerant. Furthermore, this is the first time fault tolerance and global convergence have been shown for a partially connected network under single mode and hybrid fault modes, and thus no comparison can be made to other partially connected network. Consequently, Table 2 below shows the results of PFCN analysis in comparison to fully connected networks.

<table>
<tr><td colspan="2">PFCN</td><td colspan="2">FCN</td></tr>
<tr><td>Single Mode</td><td>Hybrid Mode</td><td>Single Mode</td><td>Hybrid Mode</td></tr>
<tr><td>$C_{PFCN} = \frac{\gamma_{f(2g-1)}}{\sigma}$</td><td>$C_{PFCN} = \frac{\gamma_{a(2g-1)}}{\sigma}$</td><td>$C_{FCN} = \frac{\gamma_{fg}}{\sigma}$</td><td>$C_{FCN} = \frac{\gamma_{ag}}{\sigma}$</td></tr>
<tr><td>$N_{PFCN} \geq 3f(2g-1)+1$</td><td>$N_{PFCN} \geq (3a+2s)(2g-1)+b+1$</td><td>$N_{FCN} \geq 3fg+1$</td><td>$N_{FCN} \geq (3a+2s)g+b+1$</td></tr>
<tr><td colspan="2">$MC_{PFCN} = ng(g+n-2)$<br>$MC_{EPFCN} = ng^2$</td><td colspan="2">$MC_{FCN} = ng(ng-1)$</td></tr>
<tr><td colspan="4">$g$ = Number of clusters<br>$n$ = Number of nodes in each cluster<br>$N = ng$, number of nodes in the system<br>$f$ = Number of faults in each cluster<br>$a$ = Number of asymmetric faults in one cluster<br>$s$ = Number of symmetric faults in one cluster<br>$b$ = Number of benign faults in the system</td></tr>
</table>

Table 2: Summary of PFCN and FCN

One major future study is to investigate the AA problem under a more general partially connected network. As indicated, this is an open problem. Another avenue of study is to analyze the AA problem for PFCN under simple modes of failure such as *omission* faults, without using defaults to replace the omissions.

# References


[1] M.H. Azadmanesh, R.M. Kieckhafer, "Exploiting Omissive Faults in Synchronous Approximate Agreement", *IEEE Trans. Computers*, 49(10), pp. 1031-1042, Oct. 2000.

[2] M.H. Azadmanesh, A.W. Krings, "Network Convergence in the Presence of Omission Faults", *12th Int'l Conference on Parallel and Distributed Computing System*, pp. 416-423, 1999.

[3] M.H. Azadmanesh, A.W. Krings, "A Step toward Global Convergence in Partially Connected Networks", *$10^{th}$ International Conference on Parallel and Distributed Computing Systems*, pp. 234-241, 1997.

[4] M.H. Azadmanesh, A.W. Krings, "Egocentric Voting Algorithms", *IEEE Transactions on Reliability*, Vol. 46, No. 4, pp. 494-502, Dec. 1997.

[5] D. Dolev, N.A. Lynch, S.S. Pinter, E.W. Stark, and W.E. Weihl, "Reaching Approximate Agreement in the Presence of Faults," *Proc. Third Symp. on Reliability in Distributed Software and Database Systems*, Oct 1983.

[6] D. Dolev, N.A. Lynch, S.S. Pinter, E.W. Stark, and W.E. Weihl, "Reaching Approximate Agreement in the Presence of Faults," *JACM*, Vol. 33, No. 3, pp. 499-516, Jul 1986.

[7] B. Hardekopf, K. Kwiat, S. Upadhyaya, "Secure and Fault-Tolerant Voting in Distributed Systems", *Air Force Research Laboratory*, 2001.

[8] R.M. Kieckhafer, M.H. Azadmanesh, "Reaching Approximate Agreement With Mixed Mode Faults", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, pp. 53-63, Jan 1994.

[9] R.M. Kieckhafer, M.H. Azadmanesh, "Low Cost Approximate Agreement in Partially Connected Networks", *Journal of Computing and Information*, Vol. 3, No. 1, pp. 53-85, 1993.

[10] L. Lamport, P.M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults", *JACM*, Vol. 32, No. 1, pp. 52-78, Jan 1985.

[11] P.M. Thambidurai, Y.K. Park, "Interactive Consistency with Multiple Failure Modes", *Proc. Seventh Reliable Distributed Systems Symp.*, Oct 1988.

[12] N. Vasanthavada, P.N. Marinos, "Synchronization of Fault-Tolerant Clocks in the Presence of Malicious Failures", *IEEE Transactions on Computers*, Vol. C-37, No. 4, pp. 440-448, Apr 1988.

[13] N. Vasanthavada, P.M. Thambidurai, P.N. Marinos, "Design of Fault-Tolerant Clocks with Realistic Assumptions", *Proc. Eighteenth Fault-Tolerant Computing Symposium*, pp. 128-133, Jun 1989.