On a Unified Framework for the Evaluation of Distributed Quorum Attainment Protocols

Daniel A. Menascé, Yelena Yesha, and Konstantinos Kalpakis

Abstract— **Quorum attainment protocols are an important part of many mutual exclusion algorithms. Assessing the performance of such protocols in terms of number of messages, as is usually done, may be less significant than being able to compute the delay in attaining the quorum. Some protocols achieve higher reliability at the expense of increased message cost or delay. A unified analytical model which takes into account the network delay and its effect on the time needed to obtain a quorum is presented. A combined performability metric, which takes into account both availability and delay, is defined in this paper, and expressions to calculate its value are derived for two different reliable quorum attainment protocols: Agrawal and El Abbadi's and Majority Consensus algorithms. Expressions for the Primary Site approach are also given as upper bound on performability and lower bound on delay. A parallel version of the Agrawal and El Abbadi protocol is introduced and evaluated. This new algorithm is shown to exhibit lower delay at the expense of a negligible increase in the number of messages exchanged. Numerical results derived from the model are discussed**

Keywords—**Mutual exclusion, performability, performance analysis, fault-tolerance, distributed systems, delay analysis, majority consensus, tree-based mutual exclusion protocols, primary site protocol.**

I. Introduction

One of the most fundamental problems in the area of distributed systems is the mutual exclusion problem, which consists of ensuring that no more than one process can access the same shared object simultaneously. Many distributed mutual exclusion algorithms have been proposed in the past few years [2], [3], [4], [5], [10], [12], [13], [15], [16], [17], [19], [20]. Some of them are resilient to node and network failures [5], [10], [20]. In some cases, the quorum is statically defined [10], [11], while in some other algorithms a quorum attainment protocol has to be executed prior to mutual exclusion achievement [1]. In all cases, the performance of the protocols is measured in terms of the number of messages needed to get the quorum. Some studies also show, as an independent metric, the availability of the protocol, defined as the probability that a quorum is obtained, even in the presence of failures. We argue that these two measures are not the only relevant ones. In fact, counting the number of messages is less important than assessing the time it takes to establish the quorum, called hereafter *quorum attainment delay*. A certain protocol P_1 may send more messages than protocol P_2, but may be able to parallelize the exchange of messages in a more effective fashion than P_2, resulting in a smaller quorum attainment delay. Some protocols may achieve higher availability at the expense of larger number of messages or quorum attainment delay times. So, in order to define a single measure of performance, one should take into account at the same time delay and availability, to make a fair comparison of all existing mutual exclusion protocols. In this paper we present a novel metric to assess the performance of quorum attainment protocols which combines delay and availability simultaneously. This metric, a performability [18] type of measure for quorum attainment protocols, is denoted by $\mathcal{P}$ and is defined as

$$\mathcal{P} = \frac{A}{D'_Q} \tag{1}$$

where A is the availability of the algorithm and D'_Q is the average normalized quorum attainment delay, i.e. the average quorum attainment delay divided by the node to node communications time under zero network load. So, the higher the availability and the lower the delay, the better the performability of the quorum attainment protocol.

A general framework for the evaluation of quorum attainment protocols is developed in the form of a unified analytic model. This model allows one to compute several performance metrics, such as quorum attainment delay, performability, availability, and average number of messages, for distributed quorum attainment protocols. The unified model has two submodels: a network submodel and a protocol submodel. The network submodel computes the average end-to-end delay experienced by a message in the underlying network connecting all nodes. The protocol submodel computes the relevant performance metrics for a specific quorum attainment protocol. An iterative procedure is used to solve the fixed-point equation which results from the unified model. A proof of existence of a unique solution for this procedure is provided in the paper. Performability values are derived for two reliable protocols: Agrawal and El Abbadi's [1] and Majority [5], [20]. Results are also derived for the Primary Site [2] protocol as a lower bound on delay and an upper bound on performability. The unified model was used to study the performance of both the Agrawal and El Abbadi and the Majority Consensus quorum attainment protocols. A parallel version of the Agrawal and El Abbadi protocol is proposed and compared with the other protocols. The parallel version is shown to exhibit a higher performability and lower quorum attainment delay than other protocols at the expense of a negligible increase in the number of messages exchanged,

D. A. Menascé is with the Department of Computer Science, George Mason University, Fairfax, VA 22030-4444. E-mail: menasce@cne.gmu.edu .

Y. Yesha is with the Computer Science Department, University of Maryland Baltimore County, MD 21228-5398. E-mail: yeyesha@cs.umbc.edu

K. Kalpakis is with the Computer Science Department, University of Maryland Baltimore County, Baltimore, MD 21228-5398. E-mail: kalpakis@cs.umbc.edu .

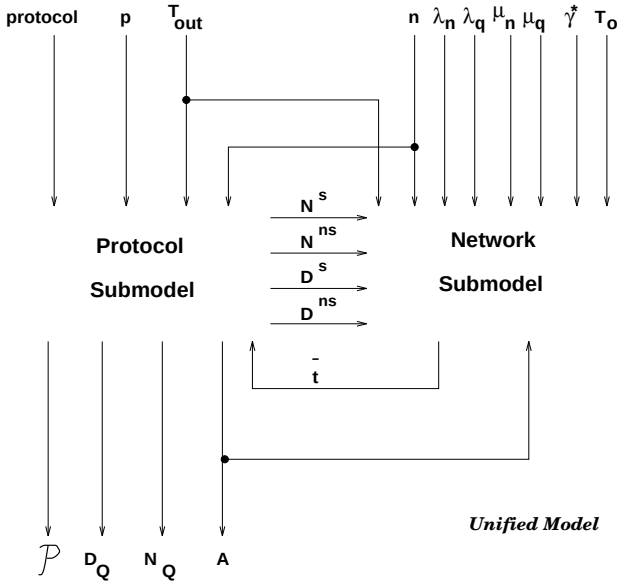


Fig. 1. Graphical View of the Unified Model

while maintaining the same availability of the original protocol.

The rest of the paper is organized as follows. In Section II, we present the unified model. A proof that the iterative procedure to solve the unified model has exactly one solution is given in the Appendix. The network submodel is presented in Section III. Sections IV, V, and VI contain the equations for the submodels for the Agrawal and El Abbadi, Majority Consensus, and Primary Site quorum attainment protocols, respectively. The parallel version of the Agrawal and El Abbadi protocol is discussed in Section VII. Section VIII presents the results of several numerical studies carried out with the use of the analytic model. Finally, Section IX presents concluding remarks.

II. A Unified Evaluation Model

We define here a unified model for the evaluation of quorum attainment protocols. This model can be better understood with the help of Fig. 1. The unified model is composed of two submodels: the *network* submodel and the *protocol* submodel. The network submodel computes the average end-to-end delay expected by a message in the underlying network connecting all nodes. The protocol submodel computes the relevant performance metrics for a specific quorum attainment protocol. Among these performance metrics are: availability, performability, average delay, and average number of messages to obtain a quorum.

The following input parameters are considered by the unified model:

- p: probability that a node fails. We assume that all nodes fail independently with the same probability. [1] A node is said to have failed if it does not reply to messages within a specified timeout. It is also assumed that when a node fails it will have to go through a recovery procedure in order to be considered up again.
- T_{out}: time interval after which a node is declared to be down.
- n: number of network nodes involved in the mutual exclusion algorithm.
- λ_n: average arrival rate of normal traffic messages generated per node, i.e., messages not related to a quorum request.
- $\frac{1}{\lambda_q}$: average time interval elapsed since a node obtains a reply (successful or not) to a previous request to obtain the quorum and the next request issued by the same node. We assume that all nodes issue new requests at the same rate.
- $1/\mu_q$: average size of messages generated by the quorum attainment protocol (in bits).
- $1/\mu_n$: average size of normal traffic messages (in bits).
- γ^*: network saturation traffic (in bps), i.e., the traffic above which the network delay goes to infinity.
- T_0: network zero load delay, i.e., the message delay when the network traffic tends to zero.

The performance metrics computed by the unified model are given below. They all come from the protocol submodel.

- $\mathcal{P}$: performability of the protocol.
- A: availability of the protocol, i.e., the probability that a quorum is attained.
- D_Q: average time needed to successfully obtain a quorum.
- N_Q: average number of messages needed by the quorum attainment protocol in order to obtain the quorum.

The network submodel generates the following metric:

- $\overline{t}$: average end-to-end delay of messages in the underlying network.

The protocol submodel generates the following metrics needed as input parameters by the network submodel:

- D^s: average time needed to execute the quorum attainment protocol given that the quorum is successfully obtained.
- D^{ns}: average time needed to execute the quorum attainment protocol given that the protocol fails in obtaining the quorum.
- N_m^s: average number of messages needed by the quorum attainment protocol each time it is executed, given that the quorum is successfully obtained.
- N_m^{ns}: average number of messages needed by the quorum attainment protocol each time it is executed, given that the quorum is not successfully obtained.

The performance metrics of the unified model may be computed as follows. We assume that if a requesting site fails in its attempt to obtain a quorum, the site resubmits its request until it finally succeeds. The probability, Pr [i failures before success], that exactly i failures occur before a successful quorum is obtained is

$$Pr\ [i \text{ failures before success}] = (1-A)^i\, A \qquad (2)$$

[1] Note that p can be written as MTTR/(MTTF + MTTR) where MTTF and MTTR are the mean time to failure and mean time to repair, respectively.

Each time a failure occurs, the average delay is equal to D^{ns}. Each node waits a time interval equal to $1/\lambda_q$ before submitting a new request. Finally, when a success is achieved, the delay is D^s. So, the average delay D_Q is given by

$$\begin{aligned} D_Q &= D^s + \\ &\quad \sum_{i=0}^{\infty} i\,(D^{ns} + 1/\lambda_q)\; Pr\,[i \text{ failures before success}] \\ &= D^s + \sum_{i=0}^{\infty} i\,(D^{ns} + 1/\lambda_q)\,(1-A)^i\,A \\ &= D^s + \frac{(D^{ns} + 1/\lambda_q)\,(1-A)}{A} \end{aligned} \tag{3}$$

Using the same kind of reasoning one can find the average number of messages needed in order to attain a quorum:

$$\begin{aligned} N_Q &= N_m^s + \sum_{i=0}^{\infty} i\; N_m^{ns}\; Pr\,[i \text{ failures before success}] \\ &= N_m^s + \sum_{i=0}^{\infty} i\; N_m^{ns}\,(1-A)^i\,A \\ &= N_m^s + \frac{N_m^{ns}\,(1-A)}{A} \end{aligned} \tag{4}$$

Finally, the performability is defined as

$$\mathcal{P} = \frac{A}{D_Q/T_0} \tag{5}$$

As it can be seen from Fig. 1, there is a dependency between the network and the protocol submodels. For instance, the protocol submodel needs as an input parameter the value of $\bar{t}$ computed by the network submodel. But the network submodel needs the values of A, D^s, D^{ns}, N_m^s, and N_m^{ns}, computed by the protocol submodel, as input parameters. This dependency is solved by using a fixed-point equation approach by the following iterative algorithm.

1. Set $i \leftarrow 0$; $\bar{t} \leftarrow T_0$;
2. Compute D^s, D^{ns}, N_m^s, and N_m^{ns} using the protocol submodel for a specific quorum attainment protocol.
3. Compute $\bar{t}$ using the network submodel.
4. If $i = 0$ then $i \leftarrow i + 1$ and go to Step 2.
5. If the values of D^s, D^{ns}, N_m^s, and N_m^{ns} obtained in iteration i are sufficiently close (within a specified tolerance) to the ones obtained in iteration $i-1$, generate the final metrics for the unified model and stop.
6. $i \leftarrow i + 1$; Go to Step 2.

A proof of the existence of a unique fixed-point solution for the unified model is given in Appendix A.

III. The Network Submodel

We assume that the nodes are connected by an underlying communication network. In a geographically distributed network, the underlying network may be a packet switching network. In a local network, we may have an Ethernet like or token-ring network. In a massively parallel distributed memory multiprocessor, the underlying network may be a hypercube type of network. We assume that the time to process a quorum request at the nodes is negligible if compared with network transmission time.

The average end-to-end delay $\bar{t}$ is, in general, a function of the network traffic, of the size of each type of message injected in the network, and on several characteristics of the network, such as link capacities and routing algorithms in packet switched networks [8], or bus speeds and propagation delays in local networks [6]. In general, the average end-to-end delay in the network can be easily obtained as a function of the network traffic and other network characteristics [6], [8]. As observed in [8], the delay versus traffic curve has a typical shape as a function of the total traffic injected in the network. For low and medium traffic values, the delay stays reasonably constant, and it rises quite sharply when the traffic approaches the network saturation point. This model, called the threshold model, is discussed in detail in [8]. While this threshold model was first introduced to depict the behavior of wide area networks, it is also useful to represent the delay throughputs characteristics of local area networks. In our evaluation of quorum attainment protocols we use a function that captures the threshold type of behavior of communication networks. This function reflects the impact on the average end-to-end delay caused by an increase in the network traffic.

We used curve fitting methods to ARPANET delay versus traffic curves [8] in order to obtain the following expression for $\bar{t}$ as a function of the total traffic γ injected in the network, the network zero load delay T_0, and the network saturation traffic value γ^*. Thus, our network submodel is given by the following expression,

$$\bar{t} = T_0 \left[1 - \frac{1}{1 - \frac{\gamma}{\gamma^*}} + \frac{1}{(1 - \frac{\gamma}{\gamma^*})^2} \right] \tag{6}$$

In order to compute the total traffic γ, some additional definitions are in order:

- γ_n: average total normal traffic injected in the underlying communication network due to normal messages, i.e., not including quorum request messages (in bps).
- γ_q: average total traffic injected in the underlying communication network due to messages generated by the quorum attainment protocol (in bps).

Thus, the total network traffic can be written as

$$\gamma = \gamma_n + \gamma_q \tag{7}$$

The average total arrival rate γ_q of requests to obtain a quorum can be found by observing that

1. Each process which does not have a pending request generates a new request with rate λ_q given that it has not failed.
2. A process that has requested a quorum will not submit a new request until it gets an answer from the previous one. If the process obtains a quorum, then it proceeds to obtain mutual exclusion.
3. As many requests as submitted may be simultaneously being processed.

This situation can be modeled as a queuing system with finite customer population (due to observation 2 above) with infinite number of servers (due to observation 3 above). Let k be the number of requests being processed. Then, the arrival rate $\lambda\ (k)$ of requests to the system given that k requests are being processed is equal to $\lambda_q \cdot (1-p) \cdot (n-k)$. The average service time, $\overline{x}$, of a request is equal to D^s if the process succeeds in obtaining the quorum (which occurs with probability A) and is equal to D^{ns} otherwise. Hence,

$$\overline{x} = A \cdot D^s + (1 - A) \cdot D^{ns} \tag{8}$$

The service rate $\mu\ (k)$ given that k requests are being processed is equal to $k\ /\ \overline{x}$. So, using the results for $GI/GI/\infty//n$ [7], we get that the probability, p_k, of having k requests being processed is given by,

$$p_k = p_0 \cdot \rho^k \left(\begin{array}{c} n \\ k \end{array} \right) \tag{9}$$

where p_0 is given by

$$p_0 = \frac{1}{(1+\rho)^n} \tag{10}$$

and ρ is defined by the following equation:

$$\rho = \lambda_q\ (1 - p)\ \overline{x} \tag{11}$$

So, the average total request rate is equal to the average arrival rate of requests to this $GI/GI/\infty//n$ queuing system and is given by,

$$\begin{aligned}
\lambda_r &= \sum_{k=0}^{n-1} \lambda\ (k) \cdot p_k \\
&= p_0 \sum_{k=0}^{n-1} \lambda_q\ (1-p)\ (n-k)\ \rho^k \left(\begin{array}{c} n \\ k \end{array} \right) \\
&= p_0\ \lambda_q\ (1-p) \sum_{k=0}^{n} (n-k) \rho^k \left(\begin{array}{c} n \\ k \end{array} \right) \\
&= p_0\ \lambda_q\ (1-p) \left[\sum_{k=0}^{n} n\ \rho^k \left(\begin{array}{c} n \\ k \end{array} \right) - \sum_{k=0}^{n} k\ \rho^k \left(\begin{array}{c} n \\ k \end{array} \right) \right] \\
&= p_0\ \lambda_q\ (1-p)\ [n\ (1+\rho)^n - n\ \rho (1+\rho)^{n-1}] \\
&= p_0\ \lambda_q\ (1-p)\ n\ (1+\rho)^{n-1}
\end{aligned} \tag{12}$$

If we substitute the value of p_0 from equation 10 into equation 12 we finally get

$$\lambda_r = \frac{n\ \lambda_q\ (1-p)}{1+\rho} \tag{13}$$

Finally, γ_q can be written in terms of the average arrival rate of requests to the shared resource, the average number, N_m, of messages exchanged per request, as well as their respective size. So,

$$\gamma_q = \lambda_r \cdot N_m \cdot \frac{1}{\mu_q} \tag{14}$$

where N_m is given by

$$N_m = A\ N_m^s + (1 - A)\ N_m^{ns} \tag{15}$$

Note that A, N_m^s,and N_m^{ns} are computed by the protocol submodel.

The normal traffic injected in the network, γ_n, is equal to the average number of up nodes multiplied by the average traffic submitted per node. Thus,

$$\begin{aligned}
\gamma_n &= \frac{1}{\mu_n}\ \lambda_n\ \sum_{k=0}^{n} k \left(\begin{array}{c} n \\ k \end{array} \right) (1-p)^k\ p^{n-k} \\
&= \frac{1}{\mu_n}\ \lambda_n\ n\ (1-p)
\end{aligned} \tag{16}$$

IV. Submodel for the Agrawal and El Abbadi Protocol

In [1], Agrawal and El Abbadi present an efficient and fault-tolerant algorithm for generating quorums in order to achieve distributed mutual exclusion. Their algorithm assumes a logical tree organization of the network. A site attempting to form a quorum sends a Request Quorum (REQ) message to the root of the tree. If the root is up, then it responds with an Acknowledgement (ACK) message. If the root is up then the algorithm continues recursively, trying to form a quorum, with the left or right subtree. If it can obtain a quorum from either of the two subtrees then it returns the quorum together with the root. Otherwise, if the root is down, the algorithm recursively obtains a quorum from both the left and the right subtrees, and then returns their union as the quorum for the whole tree. A pseudocode description of the protocol is given in Fig. 2. A node that wants to form a quorum calls the procedure GetQuorum with the root of the tree as parameter. Note that because of line 4 in the pseudocode the protocol is non-deterministic. That is, the protocol may compute the quorum for the left child first and the quorum for the right child second, or vice versa. We assume that the protocol chooses to compute either quorum first with the same probability. Further, we assume that this choice does not depend on other factors, such as previous information about failures of nodes. The Agrawal and El Abbadi's algorithm achieves fault-tolerance by ensuring that a node requesting mutual exclusion has several alternative quorums.

We derive the necessary expressions, which are needed by our unified model, for the performance analysis of the Agrawal and El Abbadi's protocol. First, some definitions are in order. The *distance between any two nodes* of a tree is equal to the number of edges on the unique path in that tree that connects these nodes. The *level of a node* of a tree is equal to its distance from the root of that tree. The *height of a tree* is equal to the maximum of the distances of any leaf node of that tree from its root. The *height of a node* of a tree is defined to be equal to the height of the subtree of that tree that is rooted at that node.

Hereafter, we assume that a complete binary tree with n nodes is given. This tree has height $h = \log(n+1) - 1$,

```
Algorithm GetQuorum.
Input: The root r of a binary tree T.
Output: A quorum consisting of nodes of T.
Begin
   (* Empty(T) evaluates to true iff T is empty. *)
   (* Up(v) evaluates to true iff node v agrees to*)
   (* be in a quorum. *)
   (* Left(v) returns the left child of a node v *)
   (* Right(v) returns the right child of a node v *)
   (* Both Left(v) and Right(v) return {} if *)
   (* v is a leaf. *)
   1  If Empty(T) Then
   2    Return { }
   3  Else If Up(r) Then
   4          Return ({r} ∪ GetQuorum(Left(r))) or
                     ({r} ∪ GetQuorum(Right(r)))
   5  Else
   6       Q1 ← GetQuorum(Left(r))
   7       Q2 ← GetQuorum(Right(r))
   8       If ( Q1 = ∅ or Q2 = ∅ ) Then
   9         Exit (*Failed to get a quorum*)
   10       Else
   11         Return ( Q1 ∪ Q2 )
End
```

Fig. 2. Agrawal and El Abbadi's Quorum attainment protocol.

has $(n+1)/2$ leaves, and each leaf has height 0 and level h. The number of nodes of that tree at level j (or equivalently at height $h-j$) is equal to 2^j, for any $0 \leq j \leq h$.

We say that a quorum attainment protocol *succeeds* if and only if it returns a non-empty quorum. Otherwise, we say that it *fails*.

We use the following notation. Hereafter, i is assumed to be a non-negative integer less than or equal to h.

- T_i^s: average time to complete the execution of the protocol, when it starts at a node of height i, given that the protocol succeeds.
- T_i^f: average time to complete the execution of the protocol, when it starts at a node of height i, given that the protocol fails.
- M_i^s: average number of messages sent during the execution of the protocol, when it starts at a node of height i, given that the protocol succeeds.
- M_i^f: average number of messages sent during the execution of the protocol, when it starts at a node of height i, given that the protocol fails.
- A_i: probability that the protocol, when it starts at a node of height i, succeeds. Thus, A_i denotes the *availability* of the protocol, when it starts at a node of height i.

Note that if we execute the Agrawal and El Abbadi's protocol on a binary tree of height h, then $D^s = T_h^s$, $D^{ns} = T_h^f$, $N_m^s = M_h^s$, $N_m^{ns} = M_h^f$, and $A = A_h$.

We provide recursive equations for T_i^s, T_i^f, M_i^s, and M_i^f, for all $0 \leq i \leq h$. Agrawal and El Abbadi [1] give an expression for the availability of their protocol. This expression, repeated in terms of our notation, is

$$A_i = \begin{cases} 2(1-p)A_{i-1} + (2p-1)A_{i-1}^2, & \text{for } i = 1, 2, \ldots, h \\ 1-p, & \text{for } i = 0. \end{cases} \tag{17}$$

Consider the execution of the protocol when it starts at a node v of the tree that is at height i. Note that v is the root of a subtree of height i. The average time to execute the protocol and the average number of messages sent depends on whether the root v of this subtree is up or down, on whether the protocol succeeds or fails when it is started at the children of v (if $i \neq 0$), on the probability of failure of the nodes, on the average end-to-end delay, and on the time to detect that a node is down.

Theorem 1: Suppose the protocol starts at a node of height i. Then, the average execution time T_i^s of the protocol and the average number of messages M_i^s sent during the execution of the protocol, given that the protocol succeeds, are

$$T_i^s = \begin{cases} [(2\overline{t} + T_{i-1}^s)(1-p)A_{i-1} + (\mathrm{T_{out}} + 2T_{i-1}^s)pA_{i-1}^2 + \\ \quad (2\overline{t} + T_{i-1}^s + T_{i-1}^f)(1-p)A_{i-1}(1-A_{i-1})]/A_i, \\ \qquad\qquad \text{for i=1,...,h} \\ 2\overline{t} \qquad\qquad \text{for i=0} \end{cases} \tag{18}$$

and

$$M_i^s = \begin{cases} [(2 + M_{i-1}^s)(1-p)A_{i-1} + (1 + 2M_{i-1}^s)pA_{i-1}^2 + \\ \quad (2 + M_{i-1}^s + M_{i-1}^f)(1-p)A_{i-1}(1-A_{i-1})]/A_i, \\ \qquad\qquad \text{for i=1,...,h} \\ 2, \qquad\qquad \text{for i=0} \end{cases} \tag{19}$$

respectively.

Proof: Let v be a node of height i. The protocol starts executing at node v. If v is of height 0 then, since the protocol succeeds, the protocol sends two messages (one REQ message plus one ACK message) and its average execution time is $2\overline{t}$. Hence, $T_0^s = 2\overline{t}$ and $M_0^s = 2$. Otherwise, v is of height $i \geq 1$. Let v_L and v_R be the left and right children of v. Nodes v_L and v_R are of height $i-1$. We introduce notation for the following events:

E_{ru} : root v is up.
E_{sl} : the protocol succeeds if it starts at v_L.
E_{sr} : the protocol succeeds if it starts at v_R.
E_{ps} : the protocol succeeds if it starts at v.

If E is an event then E^c denotes its complementary event. For example, E_{ru}^c denotes the event that node v is down, while E_{sl}^c denotes the event that the protocol fails if it starts at node v_L.

There are three cases to consider:

Case 1: node v is up and the protocol succeeds if it starts at the child it chooses to try first. By symmetry and without loss of generality, we assume it will start at node v_L. Since we assume that the protocol succeeds at node v, the probability of this case is $\Pr[E_{\mathrm{ru}} \ \& \ E_{\mathrm{sl}} \mid E_{\mathrm{ps}}]$. Further, in this case, the average execution time of the protocol, when it starts at

v, is $2\overline{t} + T^s_{i-1}$ and the average number of messages exchanged is $2 + M^s_{i-1}$.

Case 2: node v is up and the protocol fails if it starts at the first chosen child and it succeeds if it starts at the other child. By symmetry and without loss of generality, we assume that the protocol starts with node v_L first, and then with node v_R. Since we assume that the protocol succeeds at node v, the probability of this case is $\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}]$. In addition, in this case, the average execution time of the protocol, when it starts at v, is $2\overline{t} + T^f_{i-1} + T^s_{i-1}$ and the average number of messages sent is $2 + M^f_{i-1} + M^s_{i-1}$.

Case 3: node v is down. Given that the protocol succeeds when it starts at node v, the protocol must succeed when it starts at nodes v_L and v_R. The probability of this case is $\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}]$. Further, the average execution time of the protocol, when it starts at v, is $\mathrm{T_{out}} + 2T^s_{i-1}$ and the average number of messages sent is $1 + 2M^s_{i-1}$.

Therefore, the average execution time T^s_i of the protocol and the average number of messages M^s_i sent, when the protocol starts at node v, are

$$\begin{aligned} T^s_i &= (2\overline{t} + T^s_{i-1}) \cdot \Pr[E_{\rm ru}\ \&\ E_{\rm sl} \mid E_{\rm ps}] + \\ &\quad (2\overline{t} + T^s_{i-1} + T^f_{i-1}) \cdot \Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] + \\ &\quad (\mathrm{T_{out}} + 2T^s_{i-1}) \cdot \Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] \end{aligned} \tag{20}$$

and

$$\begin{aligned} M^s_i &= (2 + M^s_{i-1}) \cdot \Pr[E_{\rm ru}\ \&\ E_{\rm sl} \mid E_{\rm ps}] + \\ &\quad (2 + M^s_{i-1} + M^f_{i-1}) \cdot \Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] + \\ &\quad (1 + 2M^s_{i-1}) \cdot \Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] \end{aligned} \tag{21}$$

respectively.

Next, we compute formulas for the probabilities that appear in the above two equations. We start with the computation of a formula for $\Pr[E_{\rm ru}\ \&\ E_{\rm sl} \mid E_{\rm ps}]$. Using Bayes' theorem, it follows that

$$\Pr[E_{\rm ru}\ \&\ E_{\rm sl} \mid E_{\rm ps}] = \frac{\Pr[E_{\rm ps} \mid E_{\rm ru}\ \&\ E_{\rm sl}]\Pr[E_{\rm ru}\ \&\ E_{\rm sl}]}{\Pr[E_{\rm ps}]} \tag{22}$$

Observe that $\Pr[E_{\rm ps} \mid E_{\rm ru}\ \&\ E_{\rm sl}] = 1$ and that $\Pr[E_{\rm ps}] = A_i$. Since the events $E_{\rm ru}$ and $E_{\rm sl}$ are independent, we have that $\Pr[E_{\rm ru}\ \&\ E_{\rm sl}] = \Pr[E_{\rm ru}] \cdot \Pr[E_{\rm sl}] = (1-p)A_{i-1}$. Therefore,

$$\Pr[E_{\rm ru}\ \&\ E_{\rm sl} \mid E_{\rm ps}] = \frac{(1-p)A_{i-1}}{A_i} \tag{23}$$

The formula for $\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}]$ can be computed similarly as follows. We have that

$$\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] = \frac{\Pr[E_{\rm ps} \mid E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr}] \cdot \Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr}\ ]}{\Pr[E_{\rm ps}]} \tag{24}$$

Note that $\Pr[E_{\rm ps} \mid E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr}] = 1$ and that $\Pr[E_{\rm ps}] = A_i$. Since $E_{\rm ru}$, $E^c_{\rm sl}$, and $E_{\rm sr}$ are independent, we have that

$$\begin{aligned} \Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr}\ ] &= \Pr[E_{\rm ru}] \cdot \Pr[E^c_{\rm sl}] \cdot \Pr[E_{\rm sr}] \\ &= (1-p)A_{i-1}(1-A_{i-1}). \end{aligned} \tag{25}$$

Hence, we get that

$$\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] = \frac{(1-p)A_{i-1}(1-A_{i-1})}{A_i}. \tag{26}$$

Finally, we compute a formula for $\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}]$. From Bayes' Theorem, we have that

$$\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] = \frac{\Pr[E_{\rm ps} \mid E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr}] \cdot \Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr}\ ]}{\Pr[E_{\rm ps}]} \tag{27}$$

Since $E^c_{\rm ru}$, $E_{\rm sl}$, and $E_{\rm sr}$ are independent, we have that

$$\begin{aligned} \Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr}\ ] &= \Pr[E^c_{\rm ru}] \cdot \Pr[E_{\rm sl}] \cdot \Pr[E_{\rm sr}] \\ &= pA_{i-1}A_{i-1}. \end{aligned} \tag{28}$$

Further, since $\Pr[E_{\rm ps} \mid E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr}] = 1$ and $\Pr[E_{\rm ps}] = A_i$, we have that

$$\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] = \frac{pA^2_{i-1}}{A_i}. \tag{29}$$

Note that $\Pr[E_{\rm ru}\ \&\ E_{\rm sl} \mid E_{\rm ps}] + \Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] + \Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E_{\rm sr} \mid E_{\rm ps}] = 1$.

Substituting the formulas for the corresponding probabilities in equations (20) and (21), and using the fact that $T^s_0 = 2\overline{t}$ and $M^s_0 = 2$, equations (18) and (19) follow. ■

Theorem 2: Suppose the protocol starts at a node of height i. Then, the average execution time T^f_i of the protocol and the average number of messages M^f_i sent during the execution of the protocol, given that the protocol fails, are

$$T^f_i = \begin{cases} [(2\overline{t} + 2T^f_{i-1})(1-p)(1-A_{i-1})^2 + \\ \quad (\mathrm{T_{out}} + T^f_{i-1})p(1-A_{i-1}) + \\ \quad (\mathrm{T_{out}} + T^f_{i-1} + T^s_{i-1})pA_{i-1}(1-A_{i-1})]/(1-A_i) \\ \qquad\qquad \text{for i=1,...,h} \\ \\ \mathrm{T_{out}} \qquad\qquad \text{for i=0} \end{cases} \tag{30}$$

and

$$M^f_i = \begin{cases} [(2 + 2M^f_{i-1})(1-p)(1-A_{i-1})^2 + \\ \quad (1 + M^f_{i-1})p(1-A_{i-1}) + \\ \quad (1 + M^f_{i-1} + M^s_{i-1})pA_{i-1}(1-A_{i-1})]/(1-A_i), \\ \qquad\qquad \text{for i=1,...,h} \\ \\ 1 \qquad\qquad \text{for i=0} \end{cases} \tag{31}$$

respectively.

Proof: Let v be a node of height i. The protocol starts executing at node v. If v is of height 0 then, since the protocol fails, the protocol sends one message (one REQ message) and its average execution time is $\mathrm{T_{out}}$. Hence, $T^f_0 = \mathrm{T_{out}}$ and $M^f_0 = 1$. Otherwise, v is of height $i \geq 1$. Let v_L and v_R be the left and right children of v. Nodes v_L and v_R are of height $i-1$. Recall the following events (from Theorem 1):

$E_{\rm ru}$: root v is up.
$E_{\rm sl}$: the protocol succeeds if it starts at v_L.
$E_{\rm sr}$: the protocol succeeds if it starts at v_R.
$E_{\rm ps}$: the protocol succeeds if it starts at v.

If E is an event then E^c denotes its complementary event. There are three cases to consider:

Case 1: node v is up and the protocol fails if it starts at node v_L or v_R. Since we assume that the protocol fails when it starts at node v, the probability of this case is $\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]$. Further, in this case, the average execution time of the protocol, when it starts at v, is $2\overline{t}+2T^f_{i-1}$ and the average number of messages exchanged is $2+2M^f_{i-1}$.

Case 2: node v is down and the protocol fails if it starts at the child of v it chooses to try first. By symmetry and without loss of generality, we assume it chooses node v_L first. Since we assume that the protocol fails when it starts at node v, the probability of this case is $\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}]$. In addition, in this case, the average execution time of the protocol, when it starts at v, is $\mathrm{T}_{\rm out}+T^f_{i-1}$ and the average number of messages sent is $1+M^f_{i-1}$.

Case 3: node v is down, the protocol succeeds if its starts at the node chosen first, and it fails if it starts at the other child. By symmetry and without loss of generality we assume it chooses node v_L first and node v_R second. Since the protocol fails when it starts at node v, the probability of this case is $\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]$. Further, the average execution time of the protocol, when it starts at v, is $\mathrm{T}_{\rm out}+T^s_{i-1}+T^f_{i-1}$ and the average number of messages sent is $1+M^s_{i-1}+M^f_{i-1}$.

Therefore, the average execution time T^f_i of the protocol and the average number of messages M^f_i sent, when the protocol starts at node v, are

$$\begin{aligned} T^f_i &= (2\overline{t}+2T^f_{i-1})\cdot\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]+ \\ &\quad (\mathrm{T}_{\rm out}+T^f_{i-1})\cdot\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}]+ \\ &\quad (\mathrm{T}_{\rm out}+T^s_{i-1}+T^f_{i-1})\cdot\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] \end{aligned} \tag{32}$$

and

$$\begin{aligned} M^f_i &= (2+2M^f_{i-1})\cdot\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]+ \\ &\quad (1+M^f_{i-1})\cdot\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}]+ \\ &\quad (1+M^s_{i-1}+M^f_{i-1})\cdot\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] \end{aligned} \tag{33}$$

respectively.

Next, we compute formulas for the probabilities that appear in the above two equations. We start with the computation of a formula for $\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]$. Using Bayes' Theorem, it follows that

$$\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] = \frac{\Pr[E^c_{\rm ps} \mid E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr}]\cdot\Pr[E_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr}]}{\Pr[E^c_{\rm ps}]} \tag{34}$$

Since the events $E_{\rm ru}$, $E^c_{\rm sl}$, and $E^c_{\rm sr}$ are independent, we have that $\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr}] = \Pr[E_{\rm ru}]\cdot\Pr[E^c_{\rm sl}]\cdot\Pr[E^c_{\rm sr}] = (1-p)(1-A_{i-1})^2$. Further, since $\Pr[E^c_{\rm ps} \mid E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr}] = 1$ and $\Pr[E^c_{\rm ps}] = 1-A_i$, we have that

$$\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] = \frac{(1-p)(1-A_{i-1})^2}{1-A_i} \tag{35}$$

Similarly, we compute a formula for $\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}]$. We have that

$$\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}] = \frac{\Pr[E^c_{\rm ps} \mid E^c_{\rm ru}\ \&\ E^c_{\rm sl}]\cdot\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl}]}{\Pr[E^c_{\rm ps}]}. \tag{36}$$

Note that $\Pr[E^c_{\rm ps} \mid E^c_{\rm ru}\ \&\ E^c_{\rm sl}] = 1$ and that $\Pr[E^c_{\rm ps}] = 1-A_i$. Since the events $E^c_{\rm ru}$ and $E^c_{\rm sl}$ are independent, we have that

$$\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl}\ ] = \Pr[E^c_{\rm ru}]\cdot\Pr[E^c_{\rm sl}] = p(1-A_{i-1}). \tag{37}$$

Hence, we get that

$$\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}] = \frac{p(1-A_{i-1})}{1-A_i}. \tag{38}$$

Finally, we compute a formula for $\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]$. From Bayes' Theorem, we have that

$$\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] = \frac{\Pr[E^c_{\rm ps} \mid E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr}]\cdot\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr}\ ]}{\Pr[E^c_{\rm ps}]} \tag{39}$$

Since the events $E^c_{\rm ru}$, $E_{\rm sl}$, and $E^c_{\rm sr}$ are independent, we have that

$$\begin{aligned} \Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr}\ ] &= \Pr[E^c_{\rm ru}]\cdot\Pr[E_{\rm sl}]\cdot\Pr[E^c_{\rm sr}] \\ &= pA_{i-1}(1-A_{i-1}). \end{aligned} \tag{40}$$

Further, since $\Pr[E^c_{\rm ps} \mid E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr}] = 1$ and $\Pr[E^c_{\rm ps}] = 1-A_i$, we have that

$$\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] = \frac{pA_{i-1}(1-A_{i-1})}{1-A_i}. \tag{41}$$

Note that $\Pr[E_{\rm ru}\ \&\ E^c_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}]+\Pr[E^c_{\rm ru}\ \&\ E^c_{\rm sl} \mid E^c_{\rm ps}]+\Pr[E^c_{\rm ru}\ \&\ E_{\rm sl}\ \&\ E^c_{\rm sr} \mid E^c_{\rm ps}] = 1$.

Substituting the formulas for the corresponding probabilities into equations (32) and (33), and using the fact that $T^f_0 = \mathrm{T}_{\rm out}$ and $M^f_0 = 1$, equations (30) and (31) follow. ■

V. Submodel for the Majority Consensus Protocol

The majority consensus protocol [5], [20] is a relatively simple protocol to achieve mutual exclusion in a distributed system. To guarantee mutual exclusion, a node must receive permission from a majority of nodes in the network. It is obvious, that there can be only one majority at any instant, thus mutual exclusion is achieved. The majority quorum protocol is resilient to both node and link failures. It has been frequently used in replicated databases and as a solution to the distributed commit problem. A brief description of the majority protocol is as follows (see also Fig. 3). $C = \lceil (n+1)/2 \rceil$ messages are sent in parallel to

```
Algorithm MajorityConsensus.
Input: A set L = {v_1, v_2, ..., v_n} of n nodes.
Output: A quorum consisting of ⌈(n + 1)/2⌉ nodes from L.
Begin
   (* Sizeof(X) returns the cardinality of a set X. *)
   (* ResetClock() sets the Clock to the T_out *)
   (* Timeout(): true iff the Clock timeouts. *)
   1 Q ← Empty set of nodes
   2 While Sizeof(L) ≥ ⌈(n + 1)/2⌉− Sizeof(Q) Do
   3      S ← a subset with ⌈(n + 1)/2⌉− Sizeof(Q)
                  nodes from L
   4      Remove from L all the nodes which are in S
   5      Send, in parallel, a quorum request message
                  to each node in S
   6      ResetClock()
   7      While not Timeout() and Sizeof(S) > 0 Do
   8          If a node v ∈ S replies with an ACK
                  message Then
   9              Add node v to Q
   10             Remove node v from S
   11     End–While
   12 End–While
   13 If S̄izeof(Q) < ⌈(n + 1)/2⌉ Then
   14    Exit (* Failed to get a quorum *)
   15 Else
   16    Return ( Q )
End
```

Fig. 3. Majority Consensus Quorum Attainment Protocol.

the C nodes which constitute a majority, where n is the total number of nodes. If all of them acknowledge, the quorum is formed and the protocol stops successfully. If m ($m \leq n - C$) nodes fail, m messages have to be sent to additional nodes and the process repeats itself. If $m > n - C$ the majority cannot be reached and the protocol stops unsuccessfully.

We use the following notation.

- $T^s_{i,j}$: average delay to execute a quorum request given that i messages are sent, that j nodes have not received any quorum request messages yet, and that the quorum will be successfully obtained. Let $M^s_{i,j}$ denote the average number of messages exchanged in this case.
- $T^f_{i,j}$: average delay to execute a quorum request given that i messages are sent, that j nodes have not received any quorum request messages yet, and that the quorum will not be successfully obtained. Let $M^f_{i,j}$ denote the average number of messages exchanged in this case.
- $A_{i,j}$: probability that at least i out of j nodes have not failed.

Observe that the average delay and average number of messages in executing the majority protocol, in the case of successful quorum attainment, are given by $T^s_{C,n-C}$ and $M^s_{C,n-C}$, respectively. Similarly, in case it does not succeed in obtaining a quorum, the average delay and average number of messages are given by $T^f_{C,n-C}$ and $M^f_{C,n-C}$, respectively. Further, the availability of the majority protocol is given by $A_{C,n}$. In what follows, we derive recursive expressions for $T^s_{i,j}$, $M^s_{i,j}$, $T^f_{i,j}$, $T^s_{i,j}$, and $A_{i,j}$.

The probability $A_{i,j}$ that at least i out of j nodes have not failed is given by the following expression:

$$\begin{aligned} A_{i,j} &= \sum_{k=i}^{j} \binom{j}{k} (1-p)^k p^{j-k} \\ &= \frac{j!}{(i-1)!(j-i)!} \int_0^{1-p} x^{i-1}(1-x)^{j-i} dx. \end{aligned} \quad (42)$$

The right hand term of Equation 42 (see [14]) provides a more efficient manner of computing $A_{i,j}$ for large number of nodes. Note that if $i > j$ then $A_{i,j} = 0$.

Theorem 3: Suppose that the protocol selects to send messages to i nodes, that there are still j nodes to which no messages have been sent, and that the protocol successfully obtains a quorum consisting of i nodes. Then, the average delay, $T^s_{i,j}$, and the average number of messages sent, $M^s_{i,j}$, are given by the following two equations:

$$T^s_{i,j} = \begin{cases} [2\overline{t}(1-p)^i + \\ \sum_{m=1}^{\min\{i,j\}} (\mathrm{T_{out}} + T^s_{m,j-m}) \times \\ \qquad \binom{i}{m} p^m (1-p)^{i-m} A_{m,j}]/A_{i,i+j} \\ \qquad\qquad \text{if } j > 0 \\ 2\overline{t}, \qquad\qquad \text{if } j = 0 \end{cases} \quad (43)$$

and

$$M^s_{i,j} = \begin{cases} [2i(1-p)^i + \\ \sum_{m=1}^{\min\{i,j\}} (2i - m + M^s_{m,j-m}) \\ \qquad \binom{i}{m} p^m (1-p)^{i-m} A_{m,j}]/A_{i,i+j} \\ \qquad\qquad \text{if } j > 0 \\ 2i, \qquad\qquad \text{if } j = 0 \end{cases} \quad (44)$$

Proof: The protocol sends messages to i nodes. There are two cases to consider, depending on whether any of these i nodes fail. First, let us introduce the following events:

$\Phi_{i,j,m}$: m out of the i nodes that were sent messages have failed, while there are still j nodes to which no messages have been sent yet.

$\Psi_{k,l}$: at least k out of l nodes are up.

The two cases to consider are:

Case 1: None of the i nodes have failed. The average delay and the number of messages, in this case, are $2\overline{t}$ and $2i$, respectively. This case happens with probability equal to $\Pr[\Phi_{i,j,0} \mid \Psi_{i,i+j}]$.

Case 2: Some of the i nodes fail, say m, where $1 \leq m \leq \min\{i, j\}$. Then, the average delay and number of messages are $\mathrm{T_{out}} + T^s_{m,j-m}$ and $2i - m + M^s_{m,j-m}$,

respectively. The probability of this happening is $\Pr[\Phi_{i,j,m} \mid \Psi_{i,i+j}]$.

Note that when $j = 0$, and since the protocol successfully obtains a quorum of size i, we have that $T^s_{i,0} = 2\overline{t}$ and $M^s_{i,0} = 2i$. Moreover, from the case analysis above, and when $j > 0$, we get

$$T^s_{i,j} = 2\overline{t} \cdot \Pr[\Phi_{i,j,0} \mid \Psi_{i,i+j}] + \sum_{m=1}^{\min\{i,j\}} (\mathrm{T_{out}} + T^s_{m,j-m}) \Pr[\Phi_{i,j,m} \mid \Psi_{i,i+j}] \quad (45)$$

and

$$M^s_{i,j} = 2i \cdot \Pr[\Phi_{i,j,0} \mid \Psi_{i,i+j}] + \sum_{m=1}^{\min\{i,j\}} (2i - m + M^s_{m,j-m}) \Pr[\Phi_{i,j,m} \mid \Psi_{i,i+j}] \quad (46)$$

Next, we compute $\Pr[\Phi_{i,j,m} \mid \Psi_{i,i+j}]$, for $m = 0, \ldots, \min\{i,j\}$. Using Bayes's Theorem, we get that

$$\Pr[\Phi_{i,j,m} \mid \Psi_{i,i+j}] = \frac{\Pr[\Psi_{i,i+j} \mid \Phi_{i,j,m}] \cdot \Pr[\Phi_{i,j,m}]}{\Pr[\Psi_{i,i+j}]}.$$

Since $\Pr[\Psi_{i,i+j}] = A_{i,i+j}$, $\Pr[\Phi_{i,j,m}] = \binom{i}{m} p^m (1-p)^{i-m}$, and $\Pr[\Psi_{i,i+j} \mid \Phi_{i,j,m}] = \Pr[\Psi_{m,j}] = A_{m,j}$, we have that

$$\Pr[\Phi_{i,j,m} \mid \Psi_{i,i+j}] = \frac{A_{m,j} \binom{i}{m} p^m (1-p)^{i-m}}{A_{i,i+j}}. \quad (47)$$

Substituting Equation 47 into Equations 45 and 46, and using the facts that $T^s_{i,0} = 2\overline{t}$ and $M^s_{i,0} = 2i$, and since $A_{0,j} = 1$, we get Equations 43 and 44, respectively. ■

Theorem 4: Suppose that the protocol selects to send messages to i nodes, that there are still j nodes to which no messages have been sent, and that the protocol fails to obtain a quorum consisting of i nodes. Then, the average delay, $T^f_{i,j}$, and the average number of messages sent, $M^f_{i,j}$, are given by the following two equations:

$$T^f_{i,j} = \frac{1}{1 - A_{i,i+j}} \Big[\sum_{m=1}^{\min\{i,j\}} \binom{i}{m} p^m (1-p)^{i-m} \times (1 - A_{m,j})(\mathrm{T_{out}} + T^f_{m,j-m}) + \sum_{m=\min\{i,j\}+1}^{i} \binom{i}{m} p^m (1-p)^{i-m} \times (1 - A_{m,j}) \mathrm{T_{out}} \Big] \quad (48)$$

and

$$M^f_{i,j} = \frac{1}{1 - A_{i,i+j}} \Big[\sum_{m=1}^{\min\{i,j\}} \binom{i}{m} p^m (1-p)^{i-m} \times (1 - A_{m,j})(2i - m + M^f_{m,j-m}) + \sum_{m=\min\{i,j\}+1}^{i} \binom{i}{m} p^m (1-p)^{i-m} \times (1 - A_{m,j})(2i - m) \Big] \quad (49)$$

respectively.

Proof: The protocol sends messages to i nodes. Since we are given that the protocol fails to obtain a quorum consisting of i nodes, it follows that at least one of this nodes fails. First, let us introduce the following events:

$\Phi_{i,j,m}$: m out of the i nodes that were sent messages have failed, while there are still j nodes to which no messages have been sent yet.

$\Psi^c_{k,l}$: less than k out of l nodes are up.

Let m be the number of the i nodes that fail, $1 \leq m \leq i$. There are two cases to consider, depending on whether $m > \min\{i,j\}$ or not. First, we consider the case where $m > \min\{i,j\}$. Note that since $m \leq i$, $m > j$. In this case, since the protocol fails to obtain a quorum of size i, and since there are less than m nodes to which no messages have been sent ($m > j$), no additional messages will be sent after detecting the failure of any of these m nodes. Note that it must be the case that at least one of the i nodes fails since the protocol fails. Since failure of a node is detected after time $\mathrm{T_{out}}$, it follows that the average delay in this case is equal to $\mathrm{T_{out}}$. The average number of messages is $2i - m$ since the protocol sends i messages and receives $i - m$ acknowledgements back, $\min\{i,j\} < m \leq i$. This happens with probability $\Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}]$.

Second, we consider the case $1 \leq m \leq \min\{i,j\}$. Since the protocol will send m messages to the remaining j nodes, the average delay is $\mathrm{T_{out}} + T^f_{m,j-m}$. The average number of messages sent is $2i - m + M^f_{m,j-m}$. Therefore, since the probability that m out of the i nodes fail given that the protocol fails to obtain a quorum of size i while there are still $j > 0$ nodes to which no messages have been sent is $\Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}]$, it follows that the average delay and number of messages are

$$T^f_{i,j} = \sum_{m=1}^{\min\{i,j\}} (\mathrm{T_{out}} + T^f_{m,j-m}) \cdot \Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}] + \sum_{m=\min\{i,j\}+1}^{i} \mathrm{T_{out}} \cdot \Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}] \quad (50)$$

and

$$M^f_{i,j} = \sum_{m=1}^{\min\{i,j\}} (2i - m + M^f_{m,j-m}) \cdot \Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}] + \sum_{m=\min\{i,j\}+1}^{i} (2i - m) \cdot \Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}] \quad (51)$$

respectively.

Next, we compute $\Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}]$, for $m = 1, \ldots, i$.

Using Bayes's Theorem, we get that

$$\Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}] = \frac{\Pr[\Psi^c_{i,i+j} \mid \Phi_{i,j,m}] \cdot \Pr[\Phi_{i,j,m}]}{\Pr[\Psi^c_{i,i+j}]}.$$

Since $\Pr[\Psi^c_{i,i+j}] = 1 - A_{i,i+j}$, $\Pr[\Phi_{i,j,m}] = \binom{i}{m} p^m (1-p)^{i-m}$, and $\Pr[\Psi^c_{i,i+j} \mid \Phi_{i,j,m}] = \Pr[\Psi^c_{m,j}] = 1 - A_{m,j}$, we have that

$$\Pr[\Phi_{i,j,m} \mid \Psi^c_{i,i+j}] = \frac{(1 - A_{m,j}) \binom{i}{m} p^m (1-p)^{i-m}}{1 - A_{i,i+j}}. \quad (52)$$

Substituting Equation 52 into Equations 50 and 51, we get Equations 48 and 49. ■

VI. Submodel for the Primary Site Protocol

The delay results given for the Primary Site protocol should be viewed as lower bounds on the actual results, since due to its centralized nature, the primary site has the potential of becoming a bottleneck. In this case, the assumption that the quorum request processing time is negligible is not necessarily true. The primary site copy approach [2] is quite simple. A node — the primary site — is responsible for mediating all requests. If it is up, a quorum is formed, at the cost of 2 messages (one for the request and one for the acknowledgement). Otherwise, a quorum is not formed and only one message is sent.

The analysis for the primary site protocol is quite straightforward. The average number of messages, average delay, and availability expressions are as follows: $N^s_m = 2$, $N^{ns}_m = 1$, $D^s = 2\,\overline{t}$, $D^{ns} = T_{out}$, and $A = 1 - p$.

VII. Parallel Agrawal and El Abbadi Protocol

We describe here a modification to the basic Agrawal and El Abbadi quorum attainment protocol. The objective is to decrease the average delay at the expense of a small or no increase in the number of messages while maintaining the same availability. A description of the modified algorithm is given in Figs. 4, 5, and 6.

Figure 4 shows the main body of the protocol while Figs. 5 and 6 show the auxiliary functions SelectPaths and QStatus, respectively. For line 20 of the code in Fig. 4, we need the following definition. Given a tree T and two nodes v and u of T, we say that v is an ancestor of u in T if the path from the root of T to u includes v. Note that v is an ancestor of itself. In this protocol, we send messages, in parallel, to all nodes on paths to leaves of the tree. Further, because the QStatus function is based upon the Agrawal and El Abbadi protocol, it is not difficult to see that our modification returns a quorum as in Agrawal and El Abbadi if one exists. A formal proof of correctness, based on similar arguments, as in Agrawal and El Abbadi, can be easily constructed. However, its presentation would not shed any additional light to the material discussed here and it is therefore omitted.

We give an example of the Parallel Agrawal and El Abbadi (ParAgrAbb) protocol for a complete binary tree T with 7 nodes. We assume that the root of T is numbered 1, and for $i = 1, 2, 3$, the left and right children of node i are numbered $2i$ and $2i + 1$, respectively. Suppose that nodes 1 and 2 of T are down and that all other nodes of T are up.

First, consider the basic Agrawal and El Abbadi protocol. It sends a request to node 1 and after T_{out} it finds that node 1 is down. Then, it requests a quorum for the subtree rooted at nodes 2 and 3. It sends a request to node 2 and after T_{out} it finds it is down. It obtains a quorum for the subtree rooted at node 2 after $4\overline{t}$ time while sending

```
Algorithm ParAgrAbb.
Input: A binary tree T with n ≥ 1 numbered nodes.
Output: A quorum consisting from nodes of T.
Begin
   (* ResetClock() sets the Clock to T_out. *)
   (* Timeout(): true iff the Clock timeouts. *)
   1 UpNodes, DownNodes, WaitFor ⟵ Empty set
                                    of nodes
   2 RootSet ⟵ { the root r of T }
   3 TreeNodes ⟵ The set of nodes of T
   4 While Q̄Status(T, r, TreeNodes – DownNodes)
                                  ≠ Failure Do
   5   WaitFor ⟵ SelectPaths(T, RootSet, UpNodes
                                  ∪ DownNodes)
   6   Send requests in parallel to all nodes in
                                       WaitFor
   7   ResetClock()
   8   While not Timeout() and | WaitFor | > 0 Do
   9     If a node v ∈ WaitFor sends an ACK Then
   10      Add node v to UpNodes
   11      Remove node v from WaitFor
   12  End–While
   13  If QStatus(T, r, UpNodes) = Success Then
   14    Return the quorum computed by
           GetQuorum using UpNodes
           (*Obtained a quorum*)
   15  Add each node of WaitFor to DownNodes
   16  If QStatus(T, r, TreeNodes - DownNodes) =
                                       Failure Then
   17    Exit (* Failed to obtain a quorum *)
   18  RootSet ⟵ Empty set of nodes
   19  For each node v ∈ WaitFor Do
   20    Find lowest ancestor v' of v such that
           QStatus(T, v', TreeNodes -
                          DownNodes) ≠ Failure
   21    Select a child v'' s.t.  v'' ∉ UpNodes and
                                  v'' ∉ DownNodes
   22    Add v'' to RootSet
   23  End–For
   24 End–While
   25 Exit (* Failed to obtain a quorum *)
End
```

Fig. 4. Parallel Agrawal and El Abbadi's Quorum attainment protocol.

Function SelectPaths(T, R, E).
Input: A binary tree T of nodes and two subsets R and E of nodes in T.
Output: A subset of the nodes of T that lie on paths that start at nodes in R and end at leaves of T while they do not use any node in E.
Begin
1 Temp ⟵ Empty set of nodes
2 For each node $v \in R$ Do
3 Find a path in T from v to a leaf of T that does not use any nodes in E
4 Add all the nodes in that path to Temp
5 End–For
6 Return Temp
End

Fig. 5. The SelectPaths function for the Parallel Agrawal and El Abbadi protocol.

Function QStatus(T, v, U).
Input: A binary tree T, a node v of T, and a subset U of the nodes of T.
Output: Success or Failure depending on whether a quorum based on the Agrawal and El Abbadi protocol can be obtained for the subtree of T rooted at v under the assumption that all nodes in (not in) U are up (down).
Begin
(* Left(v) returns the left child of a node v *)
(* Right(v) returns the right child of a node v *)
1 If v is a leaf of T Then
2 If $v \in U$ Then Return Success
3 Else Return Failure
4 If $v \in U$ Then
5 If QStatus(T, Left(v), U) or QStatus(T, Right(v), U) = Success
6 Then Return Success
7 Else Return Failure
8 Else
9 If QStatus(T, Left(v), U) and QStatus(T, Right(v), U) = Success
10 Then Return Success
11 Else Return Failure
End

Fig. 6. The QStatus function for the Parallel Agrawal and El Abbadi protocol.

and receiving a total of 4 messages. Then, it continues to obtain a quorum for the subtree rooted at node 3. It sends a request to node 3, and after time $2\overline{t}$, it finds it is up. At this point, let us assume that it decides to send a message to node 7, the right child of 3. After time $2\overline{t}$ it finds that node 7 is up. Hence, the basic Agrawal and El Abbadi protocol obtains a quorum after $2T_{out} + 8\overline{t}$ time while sending or receiving a total of 10 messages.

Consider now the ParAgrAbb protocol. Suppose that the path selected at line 5 of the code in Fig. 4 consists of the nodes 1, 2 and 4, which becomes the current WaitFor set. The protocol sends requests, in parallel, to all those nodes. Since only nodes 1 and 2 are assumed down, after T_{out} time, the protocol continues with execution of line 13 of the code. Since the set of UpNodes consists of node 4, the protocol continues with line 15, where the set of DownNodes is updated to include nodes 1 and 2. At line 20, the protocol selects an ancestor for each node in WaitFor for which it is not certain the protocol will fail to obtain a quorum for the subtree rooted at that ancestor. For node 1, the protocol selects node 1 and it adds node 3 to RootSet. For node 2, the protocol selects node 2 and it adds node 5 to RootSet. Then, the protocol loops back to line 4. Suppose that the paths selected at line 5 consists of nodes 3, 5, 7. The set WaitFor consists of nodes $3, 5, 7$. The protocol sends requests, in parallel, to all nodes in WaitFor, and after time $2\overline{t}$, since all these nodes are up, all these nodes acknowledge. Then, the protocol continues with line 13. At this point the set of UpNodes consists of nodes $3, 4, 5, 7$. Function QStatus at line 13 returns success and the protocol successfully obtained a quorum consisting of nodes $3, 4, 5, 7$. Note that the quorum is determined in line 14 by invoking the same GetQuorum procedure described in the original Agrawal and El Abbadi protocol with the following modification. The Up(v) function described in GetQuorum, evaluates true iff node v is up and thus agrees to be in a quorum. In that case, this has to be done by sending a message to v and waiting for a reply. Here, the Up(v) returns true if $v \in$ UpNodes. Therefore, no messages need to be exchanged at this point. The use of the GetQuorum procedure guarantees that the quorum obtained by ParAgrAbb conforms to a quorum that would be obtained by AgrAbb. The time taken is $T_{out} + 2\overline{t}$ while the total number of messages exchanged during the execution of the protocol is 10. It should be noted that line 14 is necessary since in the course of running the protocol, there are nodes that are discovered to be up which however are not part of the Agrawal and El Abbadi quorum. For example, consider a tree with 31 nodes and height 4. Suppose that: i) all leaves except the rightmost are down, ii) the left child of the root is also down, and iii) all other nodes are up. Suppose that we explore the tree from left to right; that is whenever we have a choice between left and right child, we choose the left child first. The quorum consists of the path from the root to the rightmost leaf. In this example, the set of UpNodes in the protocol includes many up nodes which are not in the quorum. A similar situation appears in the AgrAbb protocol.

Note that, in this small example, both the original and parallel Agrawal and El Abbadi protocols exchanged the same number of messages, while the delay for the parallel version of the protocol is substantially smaller than for the original version. In general, we expect that the parallel version will exchange more messages than the serial version while the delay of the former will be much smaller than for the later. In section VIII, we compare the performance of both versions of the Agrawal and El Abbadi protocol. Observe that both protocols have the same availability.

VIII. Numerical Results

We compare the performance of the various quorum attainment protocols discussed in the previous sections. The results for Agrawal and El Abbadi (AgrAbb), Majority (Maj), and Primary Site (PS) are derived from the analytic models, while those for the parallel implementation of Agrawal and El Abbadi (ParAgrAbb) were derived by discrete event simulation. In all graphs shown in this section we assume that the model parameters are the ones given in Table I, unless otherwise specified.

TABLE I
Model Parameter Values

Parameter	Value
T_0	0.1 sec
T_{out}	1.0 sec
γ^*	2 Mbps
λ_q	0.2 request/sec
λ_n	2.0 requests/sec
$1/\mu_n$	1024 bits
$1/\mu_q$	256 bits

Figure 7 shows the variation of the average delay D_Q as a function of the probability of failure for the four protocols. In all cases, the delay is equal to $2\,\overline{t}$ when the probability of failure is equal to zero. For the four protocols, the average delay tends to infinity when p tends to 1. However, Maj tends to infinity faster than AgrAbb and ParAgrAbb, which in turn go to infinity faster than PS. Note however, that the PS curve should be interpreted as a lower bound on delay due to the reasons previously discussed. In the range $0 \leq p < 0.7$, ParAgrAbb exhibits a smaller or comparable delay to all other protocols.

Figure 8 shows the variation of the availability of the four protocols as a function of the probability of failure for a network with 31 nodes. AgrAbb and ParAgrAbb have the same availability for obvious reasons. The PS protocol has the smallest availability for values of probability of failure smaller than 0.5. In this range, Maj is slightly better than AgrAbb and ParAgrAbb. For values of p greater than 0.5, PS presents the best availability, followed by AgrAbb, ParAgrAbb, and Maj. In practice, values of p are not expected to exceed 0.3. So, in the range $p \leq 0.3$, Maj exhibits the best availability. A more detailed comparison between the availability of AgrAbb and Maj can be found in [1].

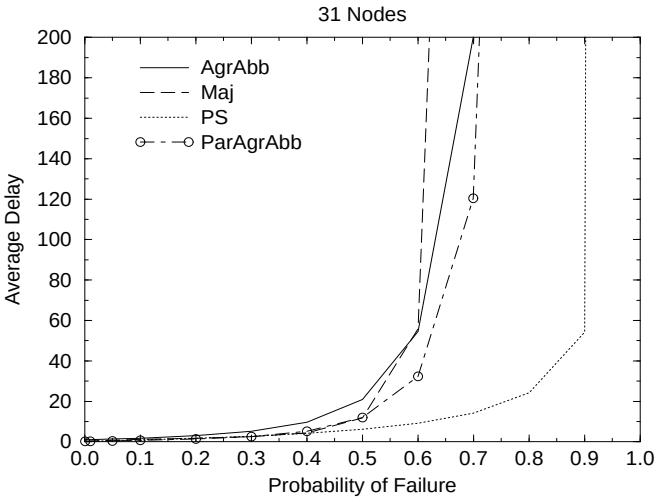


Fig. 7. Average delay vs. probability of failure for fixed n.

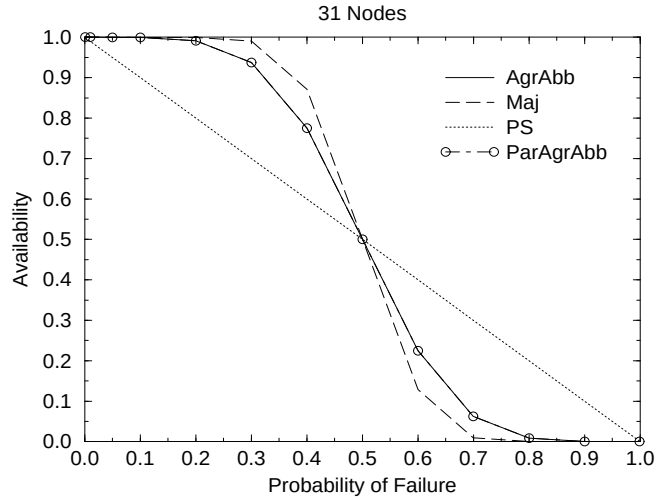


Fig. 8. Availability vs. probability of failure for fixed n.

Figure 9 shows the performability of the four protocols as a function of the probability of failure for a network with 31 nodes. As it can be seen, the performability tends to zero for values of the probability of failure greater than 0.5. This is due to the fact that the average delay for 31 nodes increases very rapidly in this range and tends to infinity as indicated in Fig. 7. In the range $0 \leq p < 0.5$, Maj has a higher availability (see Fig. 8) and lower average delay (see Fig. 7) than AgrAbb. Hence, the performability of Maj is greater than that of AgrAbb in this range. On the other hand, ParAgrAbb has the highest performability of all protocols over the whole practical range of the probability of failure p, $0 \leq p \leq 0.3$, since it exhibits a lower delay than Maj and AgrAbb for any value of the probability of failure in this range. The PS curve in Fig. 9 is accurate as long as the primary site does not become a bottleneck. Otherwise, this curve should be viewed as an upper bound on performability for PS. In the case where it becomes a bottleneck, the assumption that processing time

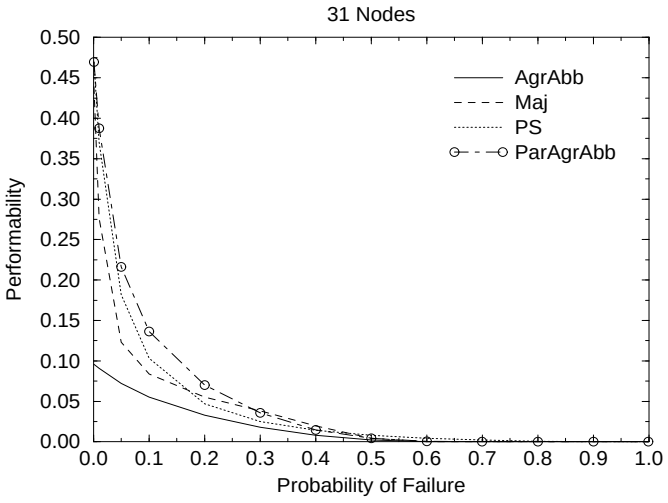


Fig. 9. Performability vs. probability of failure for fixed n.

can be neglected, compared with communication time, does not necessarily hold. Note also, that the theoretical maximum for the performability is 0.5 for any protocol. This is due to the fact that the maximum availability is 1 and the minimum normalized delay is $2\,T_0/T_0 = 2$.

The variation of the performability as a function of the number of nodes for a fixed value of the probability of failure p ($p = 0.01$) is shown in Fig. 10. In general, the performability decreases as the number of nodes increases for Maj, AgrAbb, and ParAgrAbb because the average delay increases faster than the availability increases. It should be noted that Maj and ParAgrAbb are equivalent for $n = 3$. For this reason, they have the same performability for this number of nodes. For values of n greater than 3, ParAgrAbb exhibits a better performability than Maj which in turn is better than AgrAbb. For instance, for 63 nodes and $p = 0.01$, ParAgrAbb has a performability 80% higher than Maj, and Maj has a 170% higher performability than AgrAbb. Thus, ParAgrAbb has a performability 386% higher than AgrAbb for this set of parameters. This trend is expected to continue as the number of nodes increases evenfurther. The PS curve is pretty much insensitive to the number of nodes as expected since the availability for PS is not a function of the number of nodes nor is the average delay per attempt to obtain a quorum. Again, we are not considering the possible bottleneck effect of PS.

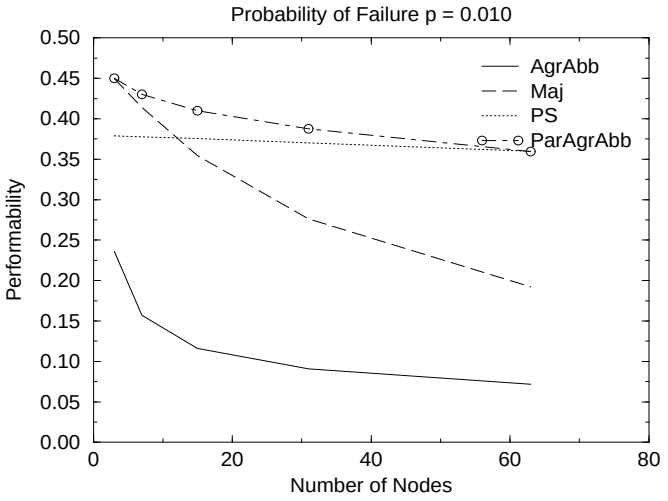


Fig. 10. Performability vs. number of nodes for fixed p.

The performability, as defined in equation 1, does not change if both the availability and the average normalized quorum attainment delay are multiplied by the same factor. Obtaining even a few percent points of improvement in the system availability may be more difficult than reducing the delay by the same factor. Thus, one might want to use alternative performability metrics that give more weight to availability. We propose a *availability-weighted performability* metric defined as

$$\mathcal{P}' = -\frac{\log_{10}(1-A)}{D'_Q} \tag{53}$$

where A, and D'_Q are, as already defined, the availability of the algorithm and the average normalized quorum attainment delay, respectively. Figure 11 shows the variation of $\mathcal{P}'$ as a function of the probability of failure for 31 nodes. Under this new metric and for 31 nodes, Maj has a higher performability than any other protocol, followed by ParAgrAbb, AgrAbb, and PS. If we fix the probability of failure and vary the number of nodes, as indicated in Fig. 12, we observe that the performability for Maj and ParAgrAbb increase initially with n due to the logarithmic effect of the increase in availability as n increases. After some value of n, the availability of both protocols increases very slightly with n but the delay starts to increase significantly with n. At this point, the performability starts to decrease. This occurs first for Maj since its delay increases faster with n than for ParAgrAbb as shown in Fig. 13 discussed next. AgrAbb and PS have very low values of $\mathcal{P}'$ if compared

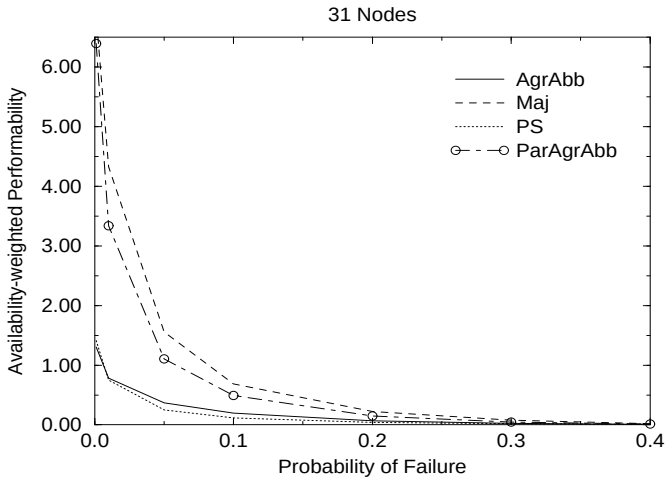


Fig. 11. Availability-weighted performability vs. probability of failure for fixed n.

with Maj and ParAgrAbb. In the case of AgrAbb this is due to its much higher delay (see Fig. 13). In the case of PS, this is due to its lower availability (see Fig. 8).

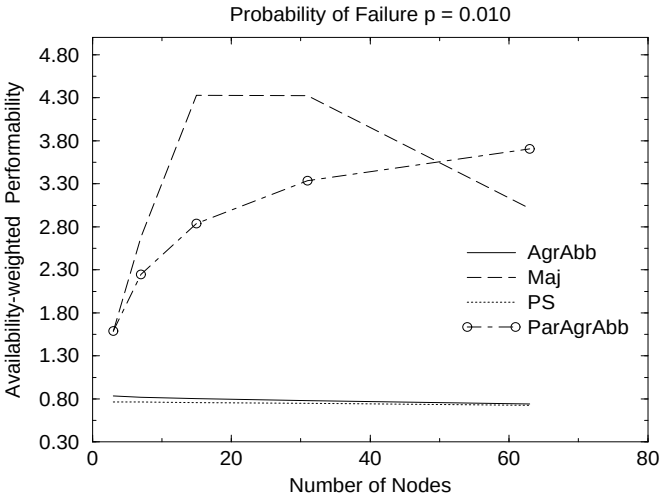


Fig. 12. Availability-weighted performability vs. number of nodes for fixed p.

Figure 13 shows the variation of the average delay D_Q as a function of n for a fixed value of p ($p = 0.01$). The average delay increases with the number of nodes for all protocols except for PS. The delay for ParAgrAbb is always less than that of Maj, which in turn is less than the one for AgrAbb. For $p = 0.01$, the delay of AgrAbb ranges from 1.8 to 5 times the delay of ParAgrAbb. Also, the difference between AgrAbb, Maj, and ParAgrAbb tends to increase as n increases.

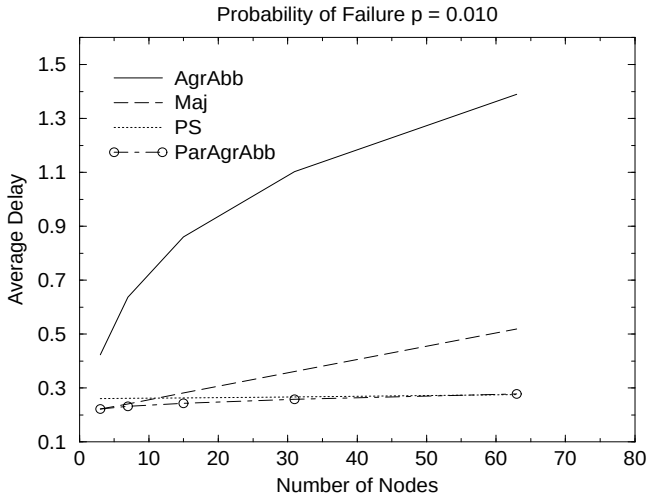


Fig. 13. Average delay vs. number of nodes for fixed p.

Figure 14 shows the variation of the average number of messages N_Q for AgrAbb, ParAgrAbb, Maj, and PS as a function of the probability of failure for 31 nodes. As it can be seen, the number of messages for Maj is larger than for any of the other protocols. For probability of failure in the range $0 \leq p < 0.3$, the number of messages

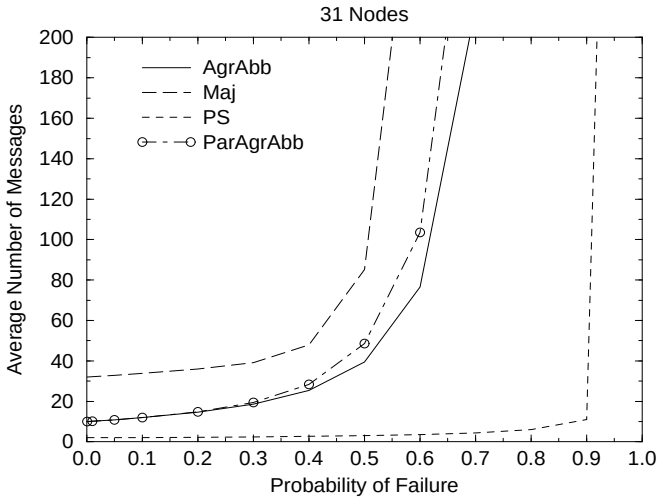


Fig. 14. Average number of messages vs. probability of failure for fixed n.

for AgrAbb and ParAgrAbb are very close (less than 5% difference). On the other hand, the average delay for ParAgrAbb is about half of that for AgrAbb in the same range of p. The average number of messages for Maj in the same range of p is roughly twice as the same number for AgrAbb and ParAgrAbb. The average number of messages for Maj tends to infinity before ParAgrAbb, which goes to infinity before AgrAbb. All protocols saturate before PS. In fact, for Maj, N_Q starts to grow very fast for values of p close to 0.5, while for AgrAbb and ParAgrAbb this happens for p in the vicinity of 0.65. PS shows a very small number of messages until p reaches 0.9. After this point, the average number of messages for PS starts to rise sharply. PS exhibits the smallest number of messages among all three protocols. As shown by the figure, for each protocol there is a different value of p after which the protocol should not be used since the number of messages would increase very fast. The same observations may be derived by looking at the delay curves.

Figure 15 shows the variation of the average number of messages N_Q as a function of the number of nodes for a fixed value of p ($p = 0.01$). For such a low value of p, AgrAbb and ParAgrAbb require an almost identical number of messages to attain a quorum. As it can be seen, the difference in N_Q between Maj and AgrAbb increases sharply with n. For PS, N_Q is not a function of the number of messages.

Figure 16 shows the impact of the rate, λ_q, at which each node generates new requests on the average delay. For this figure we kept the ratio λ_n/λ_q constant and equal to 10. As it can be seen, AgrAbb is clearly more sensitive than the other protocols to the quorum request workload intensity. This is due to the fact that $\overline{t}$ increases with λ_q and the coefficient of $\overline{t}$ in the average quorum attainment delay expression is greater for AgrAbb if compared with the other protocols. Maj and ParAgrAbb are very close to one another, but ParAgrAbb has a better performance than

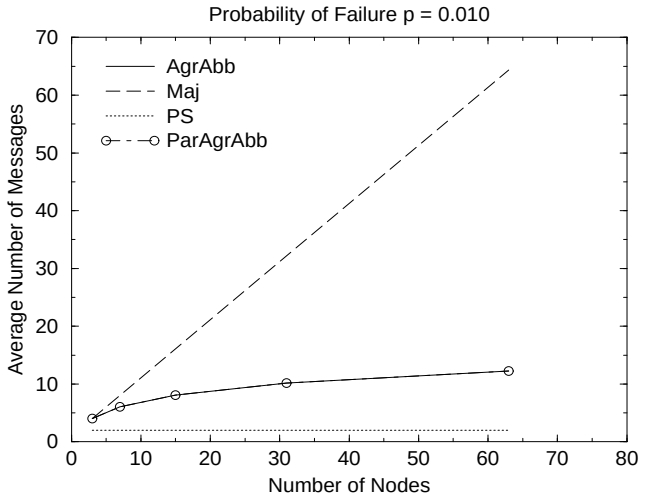


Fig. 15. Average number of messages vs. number of nodes for fixed p.

Maj since it exchanges less messages per quorum request than Maj, thus injecting less traffic into the network. For larger number of nodes, the difference between Maj and ParAgrAbb is expected to increase. Again the PS curve is a lower bound on delay.

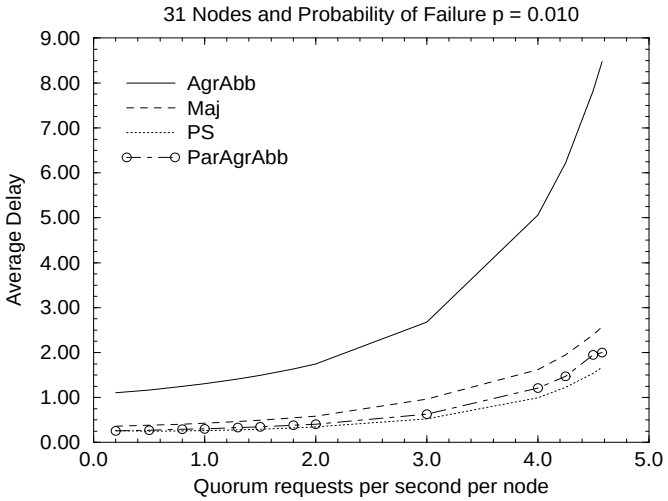


Fig. 16. Average delay vs. arrival rate of quorum requests.

IX. Concluding Remarks

This paper presented a unified framework to evaluate the performance of quorum attainment protocols. The unified model consists of two submodels: one for the underlying communication network and another for the protocol being analyzed. An iterative procedure is given to solve the fixed-point equation that results from the unified model. The procedure is shown to have a unique solution under very general assumptions. The performance metrics computed from the model are: average delay to obtain a quorum, availability, average number of messages needed to obtain the quorum, and performability — a performance metric introduced in this paper to evaluate quorum attainment protocols.

Many mutual exclusion protocols have been proposed in the past. In all cases, the performance of these protocols has been basically assessed in terms of two metrics: number of messages needed to get the quorum and availability of the protocol. Counting the number of messages is less important than assessing the quorum attainment delay. Some protocols may achieve higher availability at the expense of larger number of messages or higher quorum attainment delays. In this paper, we defined a performability metric which is the ratio between the availability and the average normalized quorum attainment delay. So, the higher the availability and the lower the delay, the better the performability of the quorum attainment protocol. The theoretical maximum for the performability for any protocol is 0.5. An alternative definition of performability which gives higher weight to availability than to quorum attainment delay was also discussed.

Analytical expressions were derived to obtain the average quorum attainment delay, average number of messages and performability of the following three protocols: AgrAbb [1], Maj [20], and PS [2]. A parallel implementation of AgrAbb, called ParAgrAbb, was proposed and its performance metrics were obtained through simulation. ParAgrAbb was shown to have a higher performability than all other protocols analyzed over the whole range of values of the probability p of node failure. Even though Maj has a higher availability than ParAgrAbb in the range $0 \leq p < 0.5$, ParAgrAbb has a lower delay than Maj in the whole range of values of p. Therefore, ParAgrAbb provides a better tradeoff than Maj between availability and average quorum attainment delay. The same kind of behavior can be observed for all other values of the number of nodes evaluated. If the availability-weighted performability metric is used, Maj exhibits a better performability than ParAgrAbb for less than 63 nodes. After this point, ParAgrAbb outperforms Maj since the delay for the latter increases faster than for the former protocol.

For all protocols analyzed, the average delay tends to infinity when p tends to 1. However, Maj tends to infinity faster than AgrAbb, which in turn goes to infinity faster than ParAgrAbb and PS. The number of messages for Maj is larger than for any of the other protocols. For small values of the probability of failure ($0 \leq p < 0.3$), the number of messages for AgrAbb and ParAgrAbb are very close (less than 5% difference). On the other hand, the average delay for ParAgrAbb is about half of that for AgrAbb in the same range. The average number of messages for Maj in the same range of p is roughly twice as the same number for AgrAbb and ParAgrAbb. The average number of messages for Maj tends to infinity before ParAgrAbb, which goes to infinity before AgrAbb.

The observations made in the previous paragraph show that a delay analysis is a better method for evaluating protocols than simply counting messages, since the delay does not grow linearly with the number of messages exchanged. Finally, the framework presented here can be applied to

other distributed mutual exclusion and commit protocols as well.

Acknowledgements

The authors would like to thank the anonymous referees for their careful review. In particular, they would like to express their gratitude to one of the referees for pointing out some flaws in the analytical models contained in an earlier version of this paper.

Appendix

Appendix A: Existence of a Unique Solution to the Unified Model

This appendix presents a proof that the unified model has a unique fixed-point solution under the fairly general assumptions given in the following theorem.

Theorem 5: The fixed point equation solved by the algorithm given in Section II has exactly one solution under the following conditions:

- the network is in equilibrium, i.e., $\gamma < \gamma^*$,
- D^s and D^{ns} are strictly increasing functions of $\overline{t}$,
- the probability of a site failure is less than 1 ($p < 1$), and
- the derivatives of D^s and D^{ns} with respect to $\overline{t}$ exist.

Proof: Let us rewrite here some of the equations shown in Section II in a form suitable for the proof. Let D^s and D^{ns} be functions of $\overline{t}$ as defined below.

$$D^s = g_1\,(\overline{t}) \tag{54}$$

$$D^{ns} = g_2\,(\overline{t}) \tag{55}$$

Since D^s and D^{ns} are assumed to be strictly increasing functions of $\overline{t}$, $g_1^{(1)}\,(\overline{t}) > 0$ and $g_2^{(1)}\,(\overline{t}) > 0$. Let $r = 1 - \gamma/\gamma^*$. Since we assume that the network is in equilibrium, it follows that $0 < r \leq 1$. So, according to equation 6, we can write $\overline{t}$ as a function g_3 of r, as follows

$$\overline{t} = g_3\,(r) = T_0\left[1 - 1/r + 1/r^2\right] \tag{56}$$

Function g_3 has the following properties which can be easily derived from expression 56:

- $\lim_{r \to 0} g_3\,(r) = \infty$.
- $g_3\,(1) = T_0$.
- $g_3\,(r)$ is a monotonically decreasing function of r with no saddle point. In fact,

$$g_3^{(1)}\,(r) = \frac{T_0}{r^2}\,\left[1 - \frac{2}{r}\right] \tag{57}$$

which is always negative since $0 < r \leq 1$. Also,

$$g_3^{(2)}\,(r) = \frac{2\,T_0}{r^3}\,\left[\frac{3}{r} - 1\right] \tag{58}$$

which is always positive since $0 < r \leq 1$.

If we combine equations 8, 11, 13, 14, 16, 54, and 55, we can write r as a function g_4 of $\overline{t}$.

$$r = g_4\,(\overline{t}) = 1 - \frac{\lambda_n\; n\;(1-p)/\mu_n}{\gamma^*} - \frac{n\;\lambda_q\;(1-p)\;N_m/(\mu_q\;\gamma^*)}{1 + \lambda_q\;(1-p)\;[A\;g_1\,(\overline{t}) + (1-A)\;g_2\,(\overline{t})]} \tag{59}$$

For fixed values of the model parameters, we can rewrite equation 59 as

$$g_4\,(\overline{t}) = 1 - K_1 - \frac{K_2}{1 + K_3\;g_1\,(\overline{t}) + K_4\;g_2\,(\overline{t})} \tag{60}$$

where K_1, K_2, K_3, and K_4 are positive constants, since we are assuming $p < 1$.

Function g_4 has the following properties:

- g_4 is a strictly increasing function of $\overline{t}$. In fact, its first derivative, $g_4^{(1)}\,(\overline{t})$, is given by

$$g_4^{(1)}\,(\overline{t}) = \frac{K_2\;[K_3\;g_1^{(1)}\,(\overline{t}) + K_4\;g_2^{(1)}\,(\overline{t})]}{[1 + K_3\;g_1\,(\overline{t}) + K_4\;g_2\,(\overline{t})]^2} \tag{61}$$

  Since $K_2, K_3, K_4, g_1^{(1)}\,(\overline{t}), g_2^{(1)}\,(\overline{t}) > 0$, it follows that $g_4^{(1)}\,(\overline{t}) > 0$.
- Also, $\lim_{\overline{t} \to \infty} g_4\,(\overline{t}) = 1 - K_1$, since the delays tend to ∞ when the network delay $\overline{t}$ goes to infinity. Note also that $1 - K_1 \leq 1$ since K_1 is simply the ratio between normal traffic and the network saturation traffic.
- The minimum possible value for $g_4\,(\overline{t})$ occurs for $\overline{t} = T_0$, since T_0 is the minimum possible value for $\overline{t}$ and $g_4\,(\overline{t})$ is a strictly increasing function of $\overline{t}$.

Since g_4 is a strictly increasing function of $\overline{t}$, there is an inverse function $g_4^{-1}\,(r)$ such that

$$\overline{t} = g_4^{-1}\,(r) \tag{62}$$

The solution to the fixed point equation is given by equating equations 56 and 62. Thus,

$$\overline{t} = g_3\,(r) = g_4^{-1}\,(r) \tag{63}$$

Given the properties derived above for functions g_3 and g_4, it is easy to see that they may be plotted as shown in Fig. 17. As it may be seen from the figure, the curves for $g_3\,(r)$ and $g_4^{-1}\,(r)$ intersect at exactly one point. ■

References

[1] D. Agrawal and A. El Abbadi, "An Efficient and Fault-Tolerant Solution for Distributed Mutual Exclusion," ACM Transactions on Computer Systems, Vol. 9, No. 1, pp. 1-20, February 1991.

[2] P. A. Alsberg and J. D. Day, "A Principle for Resilient Sharing of Distributed Resources", *Proceedings of the Second International Conference on Software Engineering*, pp. 562-570, October 1976.

[3] J. M. Bernabéu-Aubán and M. Ahamad, "Applying a Path-Compression Technique to Obtain an Efficient Distributed Mutual Exclusion Algorithm", *Proceedings of the Third International Workshop on Distributed Algorithms*, pp. 33-44, September 1989.

[4] H. Garcia-Molina and D. Barbara, "How to Assign Votes in Distributed Systems,", Journal of the ACM, Vol. 32, No. 4, pp. 841-860, October 1985.

[5] D. K. Gifford, "Weighted Voting for Replicated Data", *Proceedings of the Seventh ACM Symposium on Operating Systems Principles*, pp. 150-162, December 1979.

[6] J. L. Hammond and P. J. P. O'Reilly, Performance Analysis of Local Computer Networks, Addison Wesley, 1986.

[7] L. Kleinrock, Queueing Systems, Volume I: Theory, John Wiley & Sons, 1975.

[8] L. Kleinrock, Queueing Systems, Volume II: Computer Applications, John Wiley & Sons, 1976.

[9] M. H. MacDougall, Simulating Computer Systems, Techniques and Tools, MIT Press, Cambridge, 1987.

[10] M. Maekawa, "A $\sqrt{N}$ Algorithm for Mutual Exclusion in Decentralized Systems," ACM Transactions on Computer Systems, Vol. 3, No. 2, pp. 145-159, May 1985.

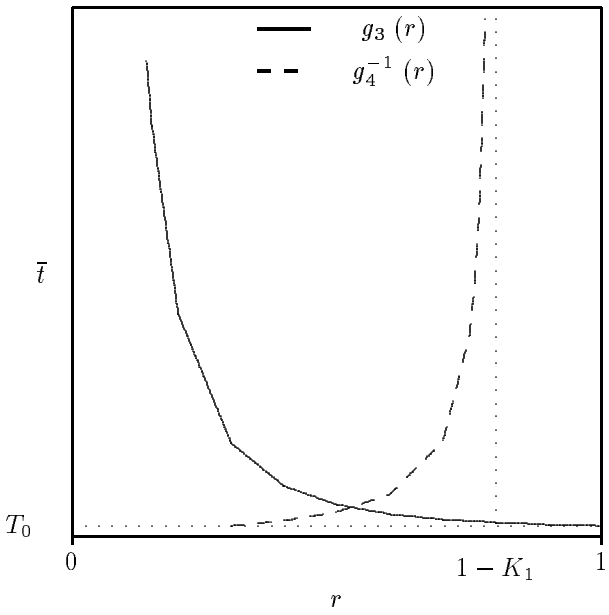


Fig. 17. $g_3(r)$ and $g_4^{-1}(r)$

[11] S. Rangarajan, S. Setia, and S. K. Tripathi, "Fault-Tolerant Algorithms for Replicated Data Management", *Proceedings of the 8th International Conference on Data Engineering*, pp. 230-237, February 1992.

[12] K. Raymond, "A Tree-Based Algorithm for Distributed Mutual Exclusion," ACM Transactions on Computer Systems, Vol. 7, No. 1, February 1989, pp. 61-77, February 1989.

[13] G. Ricart and A.K. Agrawala, "An Optimal Algorithm for Mutual Exclusion in Computer Networks," Communications of the ACM, Vol. 24, No. 1, pp. 9-17, January 1981.

[14] S. Ross, Introduction to Probability Models, *4th* edition, Academic Press, New York, 1989.

[15] B. A. Sanders, "The Information Structure of Distributed Mutual Exclusion Algorithms," ACM Transactions on Computer Systems, Vol. 5, No. 3, 284-299, August 1987.

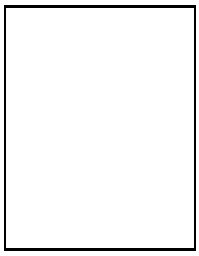
[16] M. Singhal, "A Heuristically-Aided Algorithm for Mutual Exclusion in Distributed Systems," IEEE Transactions on Computers, Vol. 38, No. 5, pp. 651-662, May 1989.

[17] M. Singhal, "A Dynamic Information Structure Mutual Exclusion Algorithm for Distributed Systems, " IEEE Transactions on Parallel and Distributed Systems, Vol. 3, No. 1, pp 121-125, January 1992.

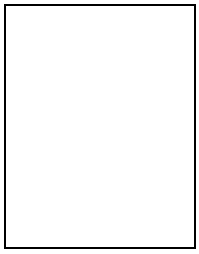
[18] R. M. Smith, K. S. Trivedi, and A. V. Ramesh, "Peformability Analysis: Measures, an Algorithm, and a Case Study," IEEE Transactions on Computers, Vol. 37, No. 4, pp. 406-417, April 1988.

[19] I. Suzuki, and T. Kasami, "A Distributed Mutual Exclusion Algorithm," ACM Transactions on Computer Systems, Vol. 3, No. 4, November 1985, pp. 344-349, November 1985.

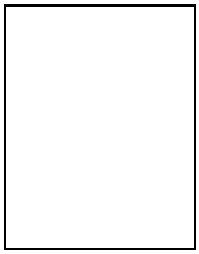
[20] R. H. Thomas, "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," ACM Transactions on Database Systems, Vol. 4, No. 2, pp. 180-209, June 1979.



Daniel A. Menascé received a Ph.D. degree in Computer Science from the University of California at Los Angeles (UCLA) in 1978, a M.Sc. degree in Computer Science, and a B.S.E.E degree both from the Pontifical Catholic University in Rio de Janeiro (PUC-RIO), Brazil in 1975 and 1974, respectively. Since 1992 he has been with the Department of Computer Science at George Mason University where he is a Professor of Computer Science and Associate Director of the Center for the New Engineer. Dr. Menascé was a visiting faculty at the Institute for Advanced Computer Studies (UMIACS) of the University of Maryland College Park from 1991 to 1992, a visiting faculty at the University of Rome, Italy, in 1981, and a full time faculty at PUC-RIO from 1978 to 1992 where he chaired the department of Computer Science from 1981 to 1983. Professor Menascé was the president of the Brazilian Computer Society from 1987 to 1989. His research interests include the areas of performance modeling, distributed systems, and high performance and parallel computing. Dr. Menascé is the author of over 70 refereed articles and is the chief author of three books. His research over the past years has been supported by ARPA, NSF, Hughes Applied Information Systems, IBM Brasil, and the Brazilian Telecommunications Agency.



Yelena Yesha received Ph.D. and M.Sc. degrees in Computer and Information Science from The Ohio State University in 1989 and 1986, respectively, and a B.Sc. degree in Computer Science from York University, Toronto, Canada in 1984. Since 1989 she has been with the Department of Computer Science at the University of Maryland, where she is presently an Associate Professor. Dr. Yesha was a program chair and general chair of the International Conference on Information and Knowledge Management and member of program committees of many prestigious conferences. She was a guest editor of International Journal on Intelligent and Cooperative Information Systems, Journal of Systems and Software and Journal of Computer and Software Engineering. Dr. Yesha was a consultant for Center for Excellence in Space Data and Information Sciences, where she was one of the key people in establishing the digital library program at NASA. Her research interests are in the areas of distributed databases, distributed systems, and performance modeling. She has authored over 40 refereed articles in these areas. Dr. Yesha is presently on leave from University of Maryland and working at the National Institute of Standards and Technology, where she is leading research on digital libraries and electronic commerce at the Information Systems Engineering Division. Dr. Yesha is a senior member of the IEEE Computer Society, member of New York Academy of Science and member of ACM.



Konstantinos Kalpakis received the Ph.D. and the M.S. degrees in Computer Science from the University of Maryland Graduate School, Baltimore, in 1994 and 1992, respectively. He also received the Diploma degree in Computer Engineering and Informatics from the University of Patras, Greece, in 1989. He is currently a Visiting Assistant Research Professor at the Computer Science Department at the University of Maryland Baltimore County. His research interests include parallel and distributed computing, combinatorial optimization, analysis of algorithms, data structures, and databases. Dr. Kalpakis is a member of the Association of Computing Machinery (ACM) and the Society for Industrial and Applied Mathematics (SIAM). He is also a member of Sigma Xi and Upsilon Pi Epsilon.