Quiescent Consensus in Mobile Ad-hoc Networks using Eventually Storage-Free Broadcasts

François Bonnet
Département Info & Télécom,
École Normale Supérieure de
Cachan, France

Paul Ezhilchelvan
School of Computing Science,
University of Newcastle, UK

Einar Vollset
School of Computing Science,
University of Newcastle, UK

ABSTRACT

We solve the consensus problem using a new class of broadcasts that are very appropriate to ad-hoc networking: every broadcast message is eventually ensured to be garbage-collected, thus freeing buffers in the resource-constrained mobile devices. We identify an impossibility result, the conditions in which a consensus protocol that assumes normal, message-keeping broadcasts can work using the new broadcast, and the adaptation such a protocol would require when these conditions do not hold. The cost of achieving quiescent consensus, estimated through simulations, is shown to be affordable for hosting practical dependable applications.

Keywords

Ad-hoc networking, crash-tolerance, group communication, consensus, quiescence, simulations.

1. INTRODUCTION

A mobile ad-hoc network, MANET for short, consists of small, mobile computing devices (nodes), capable of sending messages to each other over relatively short distances. It is the only medium available for hosting distributed applications when the use of fixed networking infrastructure is not feasible. Any middleware developed to support hosting of applications in a MANET must be adaptive towards: (a) the arbitrary topological changes of the network caused primarily by node mobility, (b) the bandwidth-constrained nature of nodes, and (c) the possibility of a few nodes crashing or leaving the MANET unannounced.

This paper will focus on achieving a principal middleware functionality, commonly known as the *consensus*, which is traditionally built using a *broadcast* facility at the lower level. Our investigations and design efforts will also focus only on solutions that have $\diamond\mathcal{Q}$ and $\diamond\mathcal{R}$ properties described below. These properties guarantee saving of bandwidth and freeing of message buffers respectively, and hence are highly desirable in the MANET context.

SAC '06, April, 23-27, 2006, Dijon, France

Eventual Quiescency, $\diamond\mathcal{Q}$. In any execution of a consensus or broadcast protocol, there is a time after which transmission of messages for that execution stops permanently.

Eventual Relinquishing of Messages, $\diamond\mathcal{R}$. In any execution of a broadcast protocol, there is a time after which nodes permanently give up retaining the broadcast message m. (*Eventually storage-free broadcasts.*)

A broadcast protocol with $\diamond\mathcal{R}$ property, denoted here as $\diamond\mathcal{R}$-broadcast, is the most useful dissemination primitive: each node eventually gives up retaining m and also, by implication, gives up transmitting m. Whereas a broadcast protocol with $\diamond\mathcal{Q}$ property, denoted as $\diamond\mathcal{Q}$-broadcast, ensures only eventual cessation of transmissions of m; in particular, it may leave the nodes retaining m for ever. The seminal paper [1] on quiescent broadcasting observes (in Section 11) that $\diamond\mathcal{Q}$ does not necessarily imply $\diamond\mathcal{R}$.

We would ideally prefer that the consensus middleware system be built using *only* $\diamond\mathcal{R}$-broadcasts. Unfortunately, it is not feasible (for reasons to be explained shortly), and we settle for a *maximum* use of $\diamond\mathcal{R}$-broadcast and use $\diamond\mathcal{Q}$-broadcast only where necessary. Consequently, the consensus problem is solved through two activities:

Activity 1. Using only $\diamond\mathcal{R}$-broadcast, develop a protocol, called *Cons*, that solves only a weaker version of the consensus problem: at least one correct node is guaranteed to decide. (Of course, if more than one node decide, decisions are identical as in the original consensus specification.)

Activity 2. Using a $\diamond\mathcal{Q}$-broadcast, a deciding node disseminates the decision to all other nodes so that those that could not decide by *Cons* receive this $\diamond\mathcal{Q}$-broadcast and decide.

The left and the right subsystems of Figure 1 perform activities 1 and 2, respectively. It is not possible to do away with activity 2 for the following reason. The authors of [1] also observe that for a broadcast protocol to have $\diamond\mathcal{R}$, nodes should have access to membership information indicating which nodes have crashed. Constructing accurate membership information is not feasible since a crashed node cannot be accurately distinguished from the one that has strayed out of other nodes' wireless reachability for too long [6]. Consequently, the $\diamond\mathcal{R}$-broadcasting comes with a price: in a MANET of n nodes, where at most $f < n$ nodes can crash, a $\diamond\mathcal{R}$-broadcast cannot be guaranteed to reach more than $n - f$ nodes even if less than f nodes have actually crashed. For example, if $n = 50$, f =10, and only 3 nodes have actually crashed, then up to 7 nodes that never crash may be unaware of a $\diamond\mathcal{R}$-broadcast.

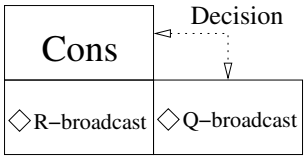

Figure 1: Architecture of a Consensus Middleware System.

When all operative nodes cannot be ensured to receive a $\diamond\mathcal{R}$-broadcast, a consensus protocol working with such a facility cannot ensure that *all* operative nodes decide, but only $n-f$ nodes decide. Since it is possible for at most f decided ones to crash later, only $n-2f$ nodes that never crash can be guaranteed to decide. An implementation of a consensus middleware system cannot therefore aspire to use *just* $\diamond\mathcal{R}$-broadcasting, but only to maximize the use.

The three components of Figure 1 make up a consensus system that will be quiescent if *Cons* is guaranteed to terminate (somehow circumventing the well-known impossibility [5]). On $\diamond\mathcal{Q}$- and $\diamond\mathcal{R}$-broadcast protocols, we point out in Section 2 that such protocols are feasible for MANETs and refer the reader to [7] for details. Main contributions are on *Cons* and consist of the following theoretical, design and simulation results. A quiescent *Cons* is not possible when $n \leq 3f$ (Section 3), though the original consensus is solvable for $2f < n \leq 3f$ [2]. The consensus protocol of [4] can be *Cons* unmodified when $n > 4f$ and appropriately modified when $3f+1 \leq n \leq 4f$ (Section 4). Through simulations (in Section 5), the cost of consensus is shown to be affordable. Next section briefly describes the preliminary concepts and the $\diamond\mathcal{Q}$- and $\diamond\mathcal{R}$-protocols taken from [7].

2. PRELIMINARIES AND THE PROBLEM

The system is a group $\mathcal{G}$ composed of $n \gg 0$ mobile nodes and at most $f < n$ of them may crash at any moment. (Initialising such a $\mathcal{G}$ is not trivial and a solution similar to [3] is assumed.) A node that never crashes is said to be *correct*. We assume that the MANET meets the following *liveness* property for any given application. Let $\mathcal{P}$ be any proper subset of correct nodes of $\mathcal{G}$ at time t; by time $t+I$, $0 \leq I \neq \infty$, some node(s) of $\mathcal{P}$ and some not of $\mathcal{P}$ have direct connectivity between them for a duration of at least B time. I is unknown and B is application specified: the minimum connectivity duration which the application requires for doing 'useful' work over a wireless link. (The application of interest here is message broadcasting.) The liveness property ensures that any partition that may occur does heal eventually, i.e., within some finite duration I.

Though the liveness property obliges direct or indirect connectivity between a pair of correct nodes, different links of an indirect connectivity may not exist *contemporaneously*. Consider, for example, a 2-hop path from node i to node j via node k. When k enters the radio range of j, it may have already lost the connection it had with i (for a minimum period B); that is, the wireless link between k and i and that between k and j may not exist simultaneously. Consequently, message dissemination involves each node (here, k) acting as a store-and-forward device for others' messages: (node k) retaining the m received from one node (here, i) for a while and transmitting it to another node (here, j) which is already in, or later comes into, its wireless range. This also means that a message from i to j can take an arbitrary amount of time, as in the classical asynchronous communication model [5].

A reliable broadcast ($\mathcal{RB}$) protocol guarantees the following delivery property, **Delivery**($\Sigma_{\mathcal{RB}}$), despite transient partitions, paths of non-contemporaneous hops and node crashes:

Delivery($\Sigma_{\mathcal{RB}}$). If a broadcast m is initiated or delivered by a correct node, then a set $\Sigma_{\mathcal{RB}}$ of nodes deliver m exactly once, where $\Sigma_{\mathcal{RB}}$ is some subset of $\mathcal{G}$ which contains *all* correct nodes.

A $\diamond\mathcal{R}$-broadcast protocol guarantees **Delivery**($\Sigma_{\diamond\mathcal{R}}$) and $\Sigma_{\diamond\mathcal{R}}$ is some subset of $\mathcal{G}$ containing at least $(n-f)$ nodes. The lower bound on the size of $\Sigma_{\diamond\mathcal{R}}$ cannot be more than $(n-f)$, because: node crashes cannot be detected accurately and any node that receives m must discard it after some time. The $\diamond\mathcal{R}$-protocol we employ here is the PKRM protocol developed by [7], which is briefly described next (omitting optimizations) only for the sake of completeness.

A node i that received m also maintains its propagation knowledge $K_i(m)$ as the set of nodes which it knows to have received m. It also transmits $K_i(m)$ once every β seconds, where $3 \times \beta < B$. (All nodes that are in its radio range can receive its transmission.) If it ever receives a message requesting m, it appends m in the next transmission of $K_i(m)$. Once the size of $K_i(m)$ becomes at least $(n-f)$, node i enters the *realized(m)* state wherein it discontinues its periodic transmissions of $K_i(m)$ and discards m ($\diamond\mathcal{R}$). If it receives $K_j(m)$, it transmits a realized packet indicating 'I HAVE REALIZED(m)'. Any unrealized node that receives this packet, enters the *realized(m)* state. The reactive transmissions of the realized packet leads to eventual quiescence ($\diamond\mathcal{Q}$).

A $\diamond\mathcal{Q}$-protocol guarantees **Delivery**($\Sigma_{\diamond\mathcal{Q}}$) where $\Sigma_{\diamond\mathcal{Q}} = \Sigma_{\mathcal{RB}}$. The $\diamond\mathcal{Q}$-protocol we advocate here is the RDP protocol developed by [7]. This protocol, like the one in [1], uses 'heart-beats' to build a very basic form of the necessary heartbeat failure detector ($\mathcal{HB}$) which only outputs a node's local (1-hop) neighborhood. Consequently, heart-beats can be the periodic 'beacons' naturally emitted by the MAC layer of mobile devices. A node in RDP transmits m whenever it deduces the presence of a neighbor who is not known to have got m. Since n is finite, $\diamond\mathcal{Q}$ is guaranteed. We refer the reader to [7] for a full description, except that a node must realize only when *all* nodes of $\mathcal{G}$, not just $(n-f)$ nodes as described in [7], are known to have got m. (Note: RDP as presented in [7] has $\diamond\mathcal{R}$ property.)

2.1 Description of the Problem (*Cons*)

A consensus protocol satisfies the following, when every operative node proposes an initial value or *initial estimate*:

- **Validity**: If a node decides v, then v was proposed by some node.
- **Agreement**: No two nodes decide differently.
- **Termination**: Every correct node decides.

Even when $n > 2f$, the termination property cannot be achieved if, as in activity 1, $\diamond\mathcal{R}$-broadcast is the *only* dissemination primitive that can be used. It follows from the possibility that f correct nodes receive no $\diamond\mathcal{R}$-broadcast message in a consensus execution when no node in $\mathcal{G}$ has crashed;

these f nodes can neither decide nor terminate the execution. So, we relax the termination property as below and solve the resulting weaker version of the problem.

Weak Termination: At least one correct node decides.

A solution to weaker consensus that uses only $\diamond\mathcal{R}$-broadcast is referred to as *Cons*. In *Cons*, the correct nodes that do not decide (hence not terminate) may or may not become quiescent; if they do, then *Cons* is quiescent and is denoted as $\diamond\mathcal{Q}$-*Cons*. We here focus only on $\diamond\mathcal{Q}$-*Cons*.

3. AN IMPOSSIBILITY

An execution of all known consensus protocols proceeds in rounds, with each round consisting of two or more phases. In each phase, a node needs to disseminate a value for that phase, either to all nodes or to a (rotating) coordinator. When dissemination is carried out only by $\diamond\mathcal{R}$-broadcasting, we have the following impossibility on solving the weaker version of the consensus problem.

THEOREM 1. *If $f < n \leq 3f$, $\diamond\mathcal{Q}$-Cons is not possible.*

PROOF. By contradiction. Suppose that a *Cons* that is also quiescent exists. We first consider the case where $n = 3$ and $f = 1$, and then generalize the result. Let the three nodes be N_1, N_2 and N_3.

Execution 1*:* N_1 has crashed before the start of the execution; N_2 and N_3 decide by t_d.

Execution 2*:* N_1 does not crash but stays out of wireless range of other nodes until t_d. Nodes N_2 and N_3 decide as in the first execution. After t_d, N_1's $\diamond\mathcal{R}$-broadcast messages are delivered by N_1 and N_3 only, not by N_2. Further, any $\diamond\mathcal{R}$-broadcast by N_2 or N_3 is received only by N_2 and N_3, not by N_1. This is possible because a $\diamond\mathcal{R}$-broadcast message can be delivered by just $n - f = 2$ nodes. As N_1 receives no message from either N_2 or N_3, it can make no progress. Since the protocol is quiescent, there must be a timing instance, say, t_{f1} after which N_1 must stop $\diamond\mathcal{R}$-broadcasting and wait for new messages to be received for decision making to be possible.

Executions $1'$ *and* $2'$*:* They are the same as Executions 1 and 2 respectively, except that N_2 plays the role played by N_1. Let the timing instance t_{f2} in execution $2'$ correspond to t_{f1} in execution 2.

Execution 3*:* The messages $\diamond\mathcal{R}$-broadcast by N_1 (resp. N_2) are received by N_3 and N_1 (resp. N_2), not by N_2 (resp. N_1) before maximum of $\{t_{f1}, t_{f2}\}$. Furthermore, N_3 is so slow that it does not even begin $\diamond\mathcal{R}$-broadcasting until maximum of $\{t_{f1}, t_{f2}\}$ and then it crashes without making any $\diamond\mathcal{R}$-broadcast. N_1 and N_2 are thus (logically) isolated from the rest exactly as they were in execution 2 and in $2'$ respectively: neither node receives any message from the other nor from N_3 until each decides to stop $\diamond\mathcal{R}$-broadcasting. Before maximum of $\{t_{f1}, t_{f2}\}$, no node decides and no operative node is destined to receive a $\diamond\mathcal{R}$-broadcast of another node; also, after that timing instant, no $\diamond\mathcal{R}$-broadcast is made. Hence no node can ever decide; it is a contradiction.

The generalization for any n, $2f < n \leq 3f$, can be done by dividing $\mathcal{G}$ into three sub-groups G_1, G_2 and G_3 of size f, f and $n - 2f$ respectively. When these groups are made to behave as N_1, N_2 and N_3 respectively we get the result. □

4. THE POSSIBILITIES

We show two results: (i) when $n > 4f$, an existing consensus protocol of [4], denoted as the EMR protocol, can be a $\diamond\mathcal{Q}$-*Cons*; and, (ii) the modifications necessary on the EMR for it to be $\diamond\mathcal{Q}$-*Cons* when $3f < n \leq 4f$.

4.1 The EMR protocol when $n > 4f$

The core ideas of the EMR protocol [4] are as follows. An execution proceeds in rounds, each with two phases. During the first phase of a round, every node broadcasts a value called its *estimate* and waits to receive estimates from a majority of nodes. If all the known estimates are the same, it is adopted as the new estimate; otherwise a default value $\perp$ (different from any possible estimate) is taken as the new estimate. (See lines 1, 2 and 7 in Figure 2.)

During the second phase (not shown in Fig. 2), as in the first, the nodes exchange their estimates and wait to receive estimates from a majority of $\mathcal{G}$. If a node receives the same value from a majority of nodes, it decides on that value and disseminates its decision. Otherwise, if it receives at least one v different from $\perp$, it adopts v as its new estimate; else, it chooses its new estimate randomly from the set of all estimates it knows; it then proceeds to the next round.

Expedited Executions. If an undecided node receives a decision, it decides and disseminates the decision; if it receives another node's estimate for a future phase, it switches to executing that phase with the received estimate as its own estimate. (See lines 3-6 in Figure 2.)

When only $\diamond\mathcal{R}$-broadcasting is used for dissemination, the progress of the EMR protocol execution can be hindered by correct nodes not receiving estimates from a majority of nodes. We show below that this is not the case if $n > 4f$, so long as no correct node has decided. After some correct node(s) has decided, the undecided ones may wait forever for a majority of estimates to be received; i.e., they enter the quiescent state. So, together with the correctness arguments in the original paper [4], the EMR can be a $\diamond\mathcal{Q}$-*Cons*.

LEMMA 1. *When $n > 4f$, the EMR protocol guarantees that at least one node computes a new estimate at the end of each phase, until some correct node(s) decides.*

PROOF. By recurrence. Suppose that no correct node has decided and some correct node, say i, enters a phase with an estimate. (This is trivially true for the first phase of the first round since no correct node has decided and each correct node has an estimate.) Node i $\diamond\mathcal{R}$-broadcasts its estimate, leading at least $n - 2f - 1$ other correct nodes to enter this phase due to expedited executions. Let $n_c \geq (n-2f)$ be the total number of correct nodes that enter this phase. Even if each of the n_c $\diamond\mathcal{R}$-broadcasts was received by f nodes that were to crash soon after reception, $n_c \times (n-2f)$ $\diamond\mathcal{R}$-broadcast messages are destined for n_c nodes. Thus, at least one correct node receives at least $\frac{n_c*(n-2f)}{n_c} = n - 2f$ estimates. Since $n > 4f$, $n - 2f > \frac{n}{2}$, that node computes a new estimate for the next phase, if it cannot decide. □

4.2 Adapting the EMR protocol for $3f < n \leq 4f$.

Lemma 1 shows that at least one correct node receives $n - 2f$ estimates in a given phase, if no correct node has decided. When $n \leq 4f$, $n - 2f$ is not a majority of n. So, the EMR protocol needs to be modified, and this takes the form of a node having to make a few more $\diamond\mathcal{R}$-broadcasts in a given phase. Each of these additional $\diamond\mathcal{R}$-broadcasts serves to diffuse all estimates which the node has thus far received from other nodes, and is made once $\diamond\mathcal{R}$-broadcasts have been received from f other nodes since the last $\diamond\mathcal{R}$-broadcast was made. Additional $\diamond\mathcal{R}$-broadcasting continues

(1)	*broadcast* PHASE1(r_i, est_i);
(2)	**wait until** (PHASE1(r_i, est) messages have been received from a majority of processes
(3)	or PHASE2(r, est) received where $r \geq r_i$
(4)	or PHASE1(r, est) received where $r > r_i$);
(5)	**if** (PHASE2(r, est) received **and** $r \geq r_i$) **then** $\{est_i \leftarrow est$; $r_i \leftarrow r$; **execute** phase2 of $r_i;\}$
(6)	**if** (PHASE1(r, est) received **and** $r > r_i$) **then** $\{est_i \leftarrow est$; $r_i \leftarrow r$; **execute** phase1 of $r_i;\}$
(7)	**if** (all received messages carry the same estimate v) **then** $est_i \leftarrow v$ **else** $est_i \leftarrow \perp$;

Figure 2: Phase 1 of round r_i for node i in the EMR protocol.

until estimates from a majority of nodes are received. The details are shown in Figure 3 and explained below.

A node, say i, maintains $\mathcal{E}_i$ as a bag of estimates it comes to know during a given phase, and $\#\mathcal{E}_i$ denotes the size of $\mathcal{E}_i$. At the start of a given phase, $\mathcal{E}_i$ contains node i's own estimate (line 1) for that phase which, as in the earlier Section, is $\diamond\mathcal{R}$-broadcast unconditionally (line 2). Whenever an $\mathcal{E}_j$ is received from node j that is not in $\diamond\mathcal{R}$-Broadcasters set (line 6), the *count* is increased by 1 and the contents of $\mathcal{E}_i$ are modified, if necessary, to include those estimates in the received $\mathcal{E}_j$ (line 6).

Node i waits either for $\diamond\mathcal{R}$-broadcasts to be received from f other distinct nodes (line 8) or for the execution to be expedited (lines 9 - 12). In the case of the former, node i $\diamond\mathcal{R}$-broadcasts its $\mathcal{E}_i$ if $\#\mathcal{E}_i$ is not a majority of n (line 14). Following the $\diamond\mathcal{R}$-broadcast, *count* and $\diamond\mathcal{R}$-Broadcasters are re-initialized, as were done after the first $\diamond\mathcal{R}$-broadcast of the phase (line 4) has been made.

The following lemma asserts that in any phase, so long as no correct node has decided, at least one correct node computes an estimate for the next phase or round. This, together with the correctness arguments in [4], can lead to the conclusion that the adapted EMR protocol is a $\diamond\mathcal{Q}$-*Cons*.

LEMMA 2. *When $3f < n \leq 4f$, the adapted EMR protocol guarantees that at least one node computes an estimate in each phase, until some correct node(s) decides.*

Correctness Hints. Consider a correct node that starts a phase by $\diamond\mathcal{R}$-broadcasting its estimate for that phase (line 2). Of the $n-f$ nodes that receive this broadcast, at most f may crash subsequently; so, we have at least $n-2f$ correct nodes receiving this broadcast and starting that phase as well (due to execution expedition). Note that $n-2f > f$, though $n-2f$ is not a majority of n. It is easy to see that no correct node can ever decide, if the following situation persists for ever in any given phase: there is a sub-group $\mathcal{G}'$, $(n-2f) \leq \mid \mathcal{G}' \mid \leq \frac{n}{2}$, of correct nodes such that (i) all nodes of $\mathcal{G}'$ start the phase, (ii) no correct node outside $\mathcal{G}'$ ever receives the required majority of estimates, and (iii) no node in $\mathcal{G}'$ receives a $\diamond\mathcal{R}$-broadcast of any node that is not in $\mathcal{G}'$. By (iii), the nodes of $\mathcal{G}'$ can only see $\mid \mathcal{G}' \mid \leq \frac{n}{2}$ estimates, even if they find '*count* $= f$' (line 8) and then make a $\diamond\mathcal{R}$-broadcast infinitely often. We can show that both (ii) and (iii) cannot hold for ever; hence the situation characterized above cannot persist forever.

Suppose that some correct node has decided and $\diamond\mathcal{R}$-broadcast its decision. At most f nodes can now be undecided. These undecided ones can receive at most $f-1$ messages and their *count* in line 8 cannot become f. So, they become quiescent.

5. SIMULATIONS

The EMR protocol for $n > 4f$ (subsection 4.1) was simulated using the Jist/SWANS simulator and considering a $\mathcal{G}$ of n nodes, each with 250m wireless range and moving according to the commonly used, Random Waypoint mobility model within a 1000m $\times$ 1000m space. Each node moves at a speed between 1m/s and 5 m/s, with a pause time of 0 seconds; β of the $\diamond\mathcal{R}$-broadcast protocol is set to 5 seconds. To remove any initial bias, the first 1000s of the simulation were discarded.

We measured the cost of weak consensus in terms of the average number of rounds needed for the first node to decide. The measurement was done by varying the number of distinct initial estimates the nodes can have in a group of $n = 50$ nodes (in Figure 4), and by varying n (in Figure 5) with every node proposing a distinct initial estimate. Note that when all nodes propose the same initial estimate, the cost is just 1 round.

To take advantage of the possibility that f may not necessarily be close to its theoretical upper bound $\lfloor \frac{n-1}{4} \rfloor$, the EMR protocol is slightly modified as below: a node waits at line (2) of Fig. 2 for messages to be received from $(n-2f)$ nodes instead of a majority of n nodes, and it checks at line (7) of Fig. 2 if the same estimate v is carried by $\lceil \frac{n}{2} \rceil$ or more messages. Note that when $n > 4f$, $(n-2f) \geq \lceil \frac{n}{2} \rceil$. Fig. 4 and Fig. 5 indicate that the smaller the f, the smaller is the cost, when all other things are the same. This is because when f gets smaller, $(n-2f)$ gets larger, more nodes are consulted and fewer disagreeing estimates cannot prolong the execution. On the other hand, when f reaches its limit $\lfloor \frac{n-1}{4} \rfloor$, a node consults only $\lceil \frac{n}{2} \rceil$ nodes if $n = 4f+1$, and a single dissenter can cause it go to the next round.

The cost never exceeds 4 rounds, and is often close to 2 rounds despite multiple, often a large number of, distinct proposals. The reason for this desirable behavior is that a given $\diamond\mathcal{R}$-broadcast reaches various nodes at widely differing timing instances; consequently, one or two nodes receive a majority of estimates and move on to the next round/phase, thus imposing their estimates on other nodes (see lines 3 and 4 in Fig. 2). This leads to a rapid convergence towards a single estimate, and then the decision.

6. CONCLUDING REMARKS

We have comprehensively investigated an approach for implementing a consensus middleware system with a maximal use of eventually storage-free broadcasting. Investigation includes proposing an architecture, identifying what cannot or can be done, designing protocols for achieving the possible, and estimating the cost of achieving consensus. The average number of rounds needed to achieve consensus is found to be small. This is encouraging for hosting sophisticated

```
(1)   E_i ← (i, est_i);
(2)   ◇R-broadcast PHASE1(r_i, E_i);
(3)   repeat
(4)     count ← 0; ◇R-Broadcasters ← {};
(5)     repeat
(6)       when PHASE1(r_i, E_j) is ◇R-received from j not in ◇R-Broadcasters
(7)       do {E_i ← E_i ∪ E_j; count++; ◇R-Broadcasters ← ◇R-Broadcasters ∪ {j}; }
(8)     until (count = f
(9)             or PHASE2(r, E_j) ◇R-received where r ≥ r_i
(10)            or PHASE1(r, E_j) ◇R-received where r > r_i);
(11)    if (PHASE2(r, E_j) ◇R-received where r ≥ r_i) then {est_i ← est_j; r_i ← r; execute phase2 of r_i;}
(12)    if (PHASE1(r, E_j) ◇R-received where r > r_i) then {est_i ← est_j; r_i ← r; execute phase1 of r_i;}
(13)    ◇R-broadcast PHASE1(r_i, E_i);
(14)  until (#E_i > n/2)
(15)  if (all estimates in E_i are the same value v) then est_i ← v else est_i ← ⊥ endif;
```

Figure 3: Phase 1 of round r_i for node i in the adapted EMR protocol.

dependable applications using MANETs. The problem we solved here is in many aspects complementary to the work in [3] which solves consensus with unknown participants in a crash-free context assuming the existence of a routing protocol. We provide their underlying requirements and their solution, as noted in Section 2, is a pre-requisite for our solution. We are currently investigating ways to avoid the central aspect that gives rise to the impossibility in Section 3: nodes working at the broadcast level and then crashing having done nothing at the consensus level. A cross layer approach, tightly integrating the two layers, seems an appropriate alternative worth investigating.

Acknowledgments

This work was carried out as a part of the UK EPSRC funded research projects *PACE* (Protocols for Ad-hoc Collaborative Environments) and *Networked Computing in Inter-Organisation Setting*. *PACE* also supported Mr François Bonnet's 3-month internship at University of Newcastle.

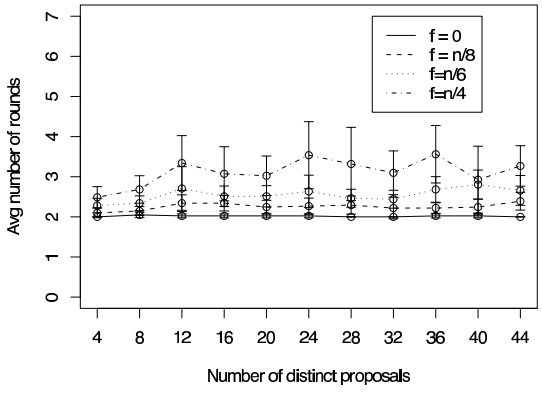

Figure 4: Cost *vs.* the number of distinct initial proposals amongst $n = 50$ nodes.

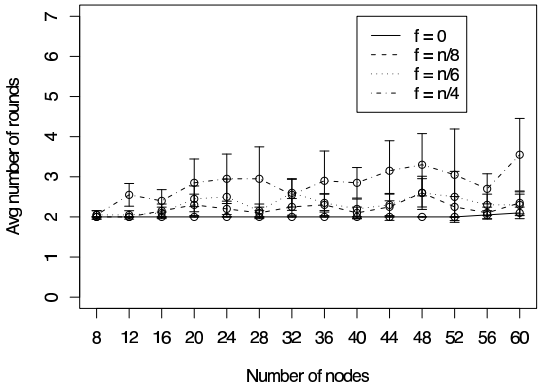

Figure 5: Cost *vs.* n. Initial proposals are distinct.

7. REFERENCES

[1] M. K. Aguilera., W. Chen, and S. Toueg. On quiescent reliable communication. *SIAM J. Computing*, 29(6):2040–2073, 2000.

[2] M. Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *PODC '83: Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30, 1983.

[3] D. Cavin, Y. Sasson, and A. Schiper. Consensus with unknown participants or fundamental self-organization. In *Proceedings of the 3rd International Conference on ADHOC-NOW 2004*, pages 135–148, Vancouver, 2004.

[4] P. Ezhilchelvan, A. Mostefaoui, and M. Raynal. Randomized multivalued consensus. In *Proceedings of the ISORC01*, pages 195–200, 2001.

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

[6] R. Friedman and G. Tcharny. Evaluating failure detection in mobile ad-hoc networks. Technical report, TR-06-2003, CS Department, Technion, October 2003.

[7] E. Vollset and P. Ezhilchelvan. Design and performance-study of crash-tolerant protocols for broadcasting and reaching consensus in manets. In *the Proc. of the 24th Symposium on Reliable Distributed Systems (SRDS)*, pages 166–175, October 2005.