A Fault and Mobility Tolerant Location Server for Large-scale Ad-hoc Networks

Gareth Owen and Mo Adda

Abstract— **There are many essential applications for quorum systems in ad-hoc networks, such as that of location servers in large-scale networks. Existing research proposes many approaches to the problems, many of which are incomplete, cumbersome, or incur significant cost. We describe and analyse a self-organising quorum system that creates an emergent intelligence to minimise overhead and maximise survivability. We compare our quorum system with ones proposed in the literature in terms of delivery success and find that it performs favourably.**

Index Terms— **Geographical routing, Location server, quorum systems, wireless LAN.**

I. Introduction

Ad-hoc networks are a means of networking computing devices together without requiring any setup or existing infrastructure. We define an ad hoc network as a graph of a set of nodes, *V*, and a set of communication paths, *E*, that vary through time, *t*, as nodes move and fail. A node, $v \in V$, has an identifier v_{ID}, and a co-ordinate v_x, v_y.

$$G_t = (V_t, E_t) \qquad (1)$$

Multi-hop ad-hoc networks allow nodes to communicate with each other without being in transmission range by relaying their messages through intermediate nodes. Routing in small networks is usually achieved through an optimal broadcast mechanism; however, when the network becomes large (>200 nodes), this form of discovery becomes extremely expensive.

Routing in large-scale ad hoc networks can be achieved through using location information from a GPS device. Hou [1] describes Most-Forward-within-Radius (MFR) whereby nodes are aware of their geographical position (from GPS) and progressively route packets closer to the location of the destination. Before a node can route packets in this fashion, it will need to know an accurate location for the destination. Many routing protocols propose a function that maps a node's ID to its location, but whilst this is ideal when networks are static and such a function can be defined, it is not when nodes are mobile and networks span many kilometres. Location servers solve this problem by maintaining location information of a node that sends it frequent updates.

Several types of location server have been proposed to provide this information. Li [2] divides the area into grid-squares of different orders. Starting with the grid-square that the node currently occupies, an order-one square, an order-two square is the grid-square containing four of the order-one squares. Li proposes that a number of nodes in each order-n square should host the location information. Therefore, information about a node's location becomes less densely available the further one is from its location.

The most promising approach to a location server for large ad hoc networks is allocating nodes within a specific radius of a geographical point to serve as a location server [3, 4]. The reasoning is that the load is distributed evenly across the network and setup and query incurs fixed overhead, independent of the number of nodes in the network. Giordano suggested modifying the radius depending upon the node density so that a minimum number of nodes participate; however, they did not address issues such as node mobility, fault tolerance and query success.

Finally, Stojmenovic proposed [5] that all nodes to the north and south in a fixed-width column assume the role of a location server. Location searches are performed by routing packets horizontally so that they will eventually intersect the vertical column. This approach is not feasible for large-scale ad hoc networks because of the large number of nodes. For a more complete review of approaches to location servers, the reader is referred to [6].

The approach shown in this paper is most similar to the work on home-regions but we further develop the system we proposed in [7, 8] by examining performance as a location server for large-scale ad-hoc networks. We compare our results with that of the home-region.

II. A Fault and Mobility Tolerant Quorum System for Ad-Hoc Networks

In this paper, we propose an approach similar to the home region but that addresses many of the problems associated

G. H. Owen is a researcher with the University of Portsmouth's School of Computing, Lion Terrace, Portsmouth, PO1 3HE. (Phone: +442392846782; email: gareth.owen@port.ac.uk).

M. Adda is a principal lecturer with the University of Portsmouth's School of Computing. (Phone: +442392846377; email: mo.adda@port.ac.uk).

with it. Instead of allocating a circular region of nodes as the hosts of a location server, we say that nodes directly adjacent to a particular geographic point should be responsible. This permits a small number of nodes to be used without knowledge of node density.

A. Quorum Deployment

A node sets up a location server by sending an initiation packet toward its home location that will be received by the closest node. If the point is within an area void of nodes, then the closest node will be one on the void perimeter. This node will then assume the role of master of the server, and as it moves the role is transferred to a node that is closer to the home location. Migration is the term we call this process of quorum components moving from node-to-node to remain close to a geographic point.

Presently, with the master being the only one component, failure of the node hosting it could result in failure of the server so it is important that the data is replicated. Immediate neighbours of the master are sent a duplication packet from the master and assume the role of a slave; however, the server as a whole needs to make sure that enough replicated copies are kept in the event of node failures so that the data is not lost.

B. Quorum failure and mobility tolerance

Formally, we define our location server as a set Q_t where M_t and S_t are the sets of nodes that host master and slave components respectively:

$$Q_t \subseteq V_t = M_t \cup S_t \tag{2}$$

Such maintenance of what is essentially a quorum [9] is very costly process in wired networks let alone wireless, and so a novel approach is needed. Therefore, we draw some of our inspiration from ants in a colony who find their way to food by laying pheromones for each other to follow [10]. These pheromones serve to modify the environment so that other ants may detect these modifications avoiding the need for directed communication. Although ants act independently, their individual behaviour develops an emergent intelligence. Drawing upon this to develop a location server, we suggest that instead of components of a quorum communicating with each other they modify their environment. As every node has to beacon its location to its neighbours for routing to occur, we add to this beacon packet (which we call the environment) an itinerary of location server components held at that particular node. A node's itinerary, $I_{v,t}$, is simply a list of identifiers for quorums which it hosts a component of. Therefore, a beacon frame can be described as the n-tuple:

$$B_{v,t} = \{v_{x,t}, v_{y,t}, I_{v,t}\} \tag{3}$$

Each node maintains a list of all received beacons so that each component of the quorum is able to examine a list of which nodes hold which components. Beacons stored in the list expire after twice the beacon interval.

The master is the only node permitted to duplicate as it is most likely to be in the centre of the quorum and so able to correctly count the number of components. When the number of neighbours hosting components falls below a threshold, the master will begin a duplication phase to raise this number.

We now have a set of quorum components who do not directly communicate with each other, but who act in a self-organising manner forming an emergent intelligence (like ants). Each attempts to get as close as possible to the home location by moving from node to node (except where a destination node is already in possession of one). Each component assumes the role of master and responsibility for duplicating if it finds itself, through observing its environment, to be on the closest node to the home location. If it realises it is no longer the closest then it will automatically remove its responsibility and assume the role of a slave. We illustrate the master and slaves, along with migration in Figure 1, where the master is held on the closest node to the point and has duplicated its data to several slaves on adjacent nodes.

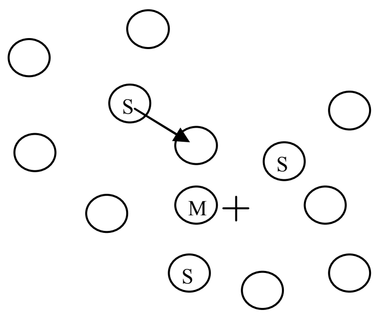

Figure 1: Illustration of quorum and component migration

In addition to the duplication task, the components perform one more tasked based upon information gathered from the beacons. The components self delete themselves to stop a large number of replicated components consuming resources. They do this by each monitoring the number of replicas, and if they rise above a threshold then component will delete itself.

This process is performed on each node at regular intervals for each component held, as described by the ManageComponent(c) function:

```
MANAGECOMPONENT($c$)
1   $n \leftarrow$ closerNeighbor(homeLocation)
2   **if** exists($n$) **and** hasComponent($n$)
3       **then** $c_{role} \leftarrow$ slave
4       **else** $c_{role} \leftarrow$ master
5   **if** exists($n$) **and not** hasComponent($n$)
6       **then** c.migrate($n$)
7   **else if** $c_{role}$=master **and**
                    hasComponent($all$) < $threshold$
```

```
8           then replicate($c$)
9       else if $c_{role}$=slave and
                    hasComponent($all$) > $threshold$
10          then delete($c$)
```

We use multicast for replication of the data. When a node senses that there is less than the threshold number of slaves, then it will create a multicast packet to a number of neighbours defined by the threshold minus the number of slaves. The neighbours chosen will be those who do not already have a component.

```
REPLICATE(C)
1   $needed \leftarrow threshold$ – hasComponent($all$)
2   $destcount \leftarrow 0$
3   while $needed$ > 0 and $neigh$ not equal null
4       if not hasComponent($neigh$) then
5           $c_{dests}[destcount{+}{+}] = neigh$
6           $needed$--
7       endif
8       $neigh \leftarrow neigh_{next}$
9   endwhile
```

C. Update Mechanism

The mechanism to update the quorum is equally simple and based upon a zero knowledge approach to minimise overhead. When receiving a new update packet, a node simply updates itself if it holds a component and then rebroadcasts it. If it does not hold a quorum component already, it simply forwards the update to the next closest node to the centre. This way, each element of a quorum should receive a copy and the number of overhead packets will be equal to the number of components present.

This approach does not guarantee a complete quorum update due to its self-organising approach, but querying multiple quorum components can mitigate this as shown in [8].

```
RECVUPDATEPACKET($p$)
1   $c \leftarrow$ Component($p_{nodeID}$)
2   if not exist($c$) or seenBefore($p$) then ignore
3   if $p_{revision} \leq c_{revision}$ then ignore
4   $c_{data} \leftarrow p_{data}$
5   broadcast($p$)
```

III. Terminode Home-Region Server Implementation

The home-region server was briefly described in the introduction and here we implement it for comparison purposes against our own technique. Due to the lack of technical detail in the original paper we have implemented the home-region server using many assumptions that are detailed below.

First is the issue of deployment of the server. We assume that to deploy the server the node simply sending a packet containing the information to be stored towards the centre of the home region. At each hop the following algorithm is run:

```
PROCESSHR($p$)
1   if distTo($p_x$, $p_y$) < $R$ then
2       broadcast($p$)
3       store($p$)
4   else
5       $n$ = closestNeighbour($p_x$, $p_y$)
6       sendPacket($p$, $n$);
```

The algorithm determines whether the node processing the packet is within radius R of the centre of the home region, and if so the packet is broadcasted and stored, otherwise it is forwarded to the closest neighbouring node.

To update the location server, a node simply sends another deployment packet to the centre location, and all who either contain no data, or contain data that is older than the update will be updated.

IV. Querying the Location Servers

The technique used to query two location servers is similar. A node wishing to determine the destination of the node to which it wants to communicate creates a GET(v_{ID}) request and routes it toward the centre of the home region, or the home location (s_x, s_y). Upon reaching a node, the technique varies slightly according to the location server used:

Quorum system: If the node contains information of the destination's location, a REPLY(v_{ID}) request is sent back to the origin.

Home-region: If the node is within radius R of the centre of the region, and contains information about the destination's location, a REPLY(v_{ID}) request is sent back to the origin.

The two packets are defined as:

$$GET(nodeID) = \{v_{ID}, s_x, s_y\} \tag{4}$$

$$REPLY(nodeID) = \{v_{ID}, v_{x,t}, v_{y,t}\} \tag{5}$$

Whilst the query is being executed, the packet originating the query and any subsequent packets are queued until a reply is received, at which point they are then tagged with the location of the destination and sent to the nearest neighbour. If no reply is received within one second a further GET(v_{ID}) packet is resent, up until a maximum of five, at which point the query is deemed to have failed.

V. Results

A. Simulation description

Simulations are performed in the Jist/SWANS simulator [11] and are repeated 10 times for validity. Packets are routed using Most-Forward-in-Radius [1], except where stated. Five bidirectional flows are set up between random nodes and packets are sent at a rate of 1 per second per direction, after *t*=20s to allow for the servers to deploy. Nodes are able to cache locations of nodes for up to 10 seconds after discovery or extraction from received packets to avoid large numbers of location discoveries.

Parameter	**Value**	**Parameter**	**Value**
Terrain	1000x1000	**Number of nodes**	200
Home-Region: *R* value	100m	**Number of bidirectional flows**	5
Tx/Rx Range	100m	**Quorum: Number of slaves**	5
Mobility Model	Random Waypoint	**Beacon rate**	Every 5 seconds + jitter
Simulation-time	2 minutes	**Manage Component call rate**	Every 6 seconds + jitter

Our simulations simulate node failures by clearing a node's memory with a defined probability every five seconds. One can determine an expected number of node failures per minute (X) given the number of nodes in the network (N) and the failure probability (D):

$$X = 12DN \qquad (6)$$

To simplify simulation we assume the location (s_x, s_y) that a server (s) will reside near is assigned at known globally. The locations are allocated in a grid-like fashion with each node taking the next sequential location, so that two servers have precisely the same location and the load is distributed evenly across the network.

B. Figures

In the figures, our system is termed 'Quorum.' Figure 2 analyses the effect that varying speed and failure probability has on the delivery success of packets in the network. One can see that in all cases the quorum outperforms terminode's home region. The update interval was set to 10 seconds and so the home region is redeployed at 10 second intervals, therefore varying the deletion probability had little effect on the delivery success.

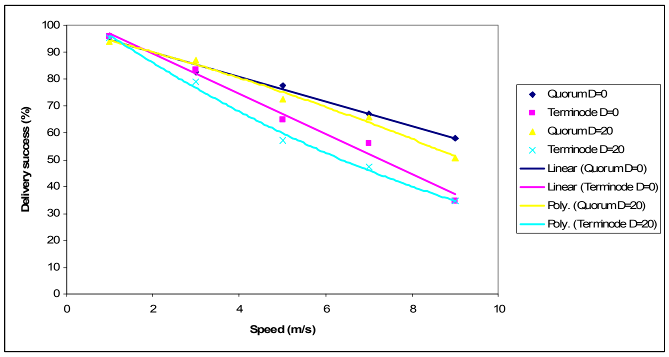

Figure 2: Speed vs. delivery success whilst varying failure probability (update interval = 10s)

The results shown in Figure 3 are obtained in the same manner as the previous graph with the exception the update interval is set at 40 seconds to compare the failure tolerance of both approaches. The performance difference between the two systems widens greatly with the quorum approach performing significantly better. It is worth noting that a certain number of failures will be due simply to the location information being out of date.

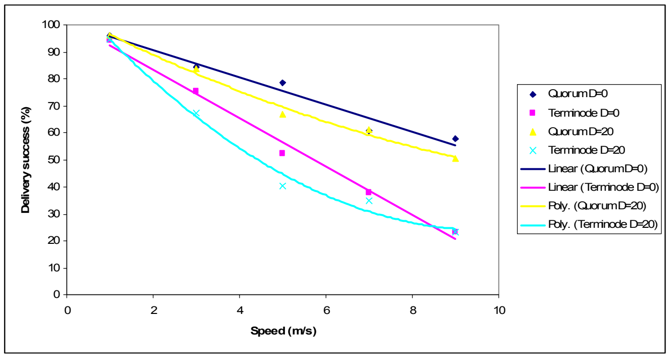

Figure 3: Speed vs. delivery success whilst varying failure probability (update interval = 40s)

Figure 4 examines the effect the update interval has on delivery success whilst varying the failure probability. A certain amount of decline with update interval would be expected with any system due to the increasingly less accurate location information; however, the terminode approach has no means to conquer mobility and node failures at higher speeds and so is outperformed.

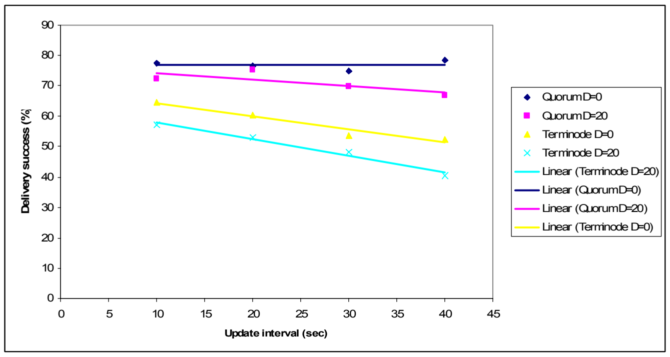

Figure 4: Update interval vs. delivery success whilst varying failure probability (speed=5m/s)

Figure 5 compares only failure probability and delivery success, with a speed of 5m/s and an update interval of 40seconds. The quorum system outperforms terminode's home region approach by significantly under all cases.

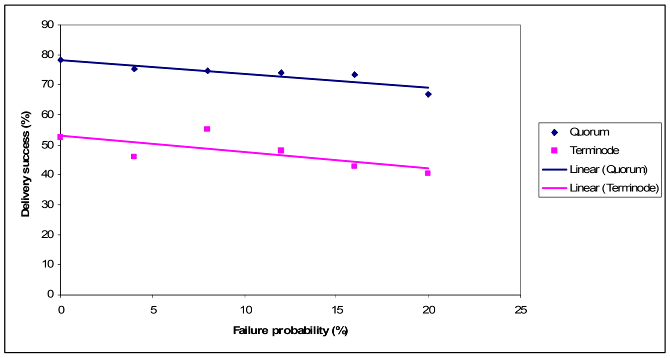

Figure 5: Failure probability vs. delivery success (speed=5m/s; update interval=40s)

Figure 6 illustrates the tolerance to an area void of nodes. Terminode requires a certain region to be populated whereas our system will attach the quorum to the edge of a void perimeter if necessary. In this case, we use the right-hand rule [12] to aid routing, and compare the two systems with a varying sized area void of nodes (*a* x *a*), centred in the simulation scenario. Terminode is again outperformed by a significant margin.

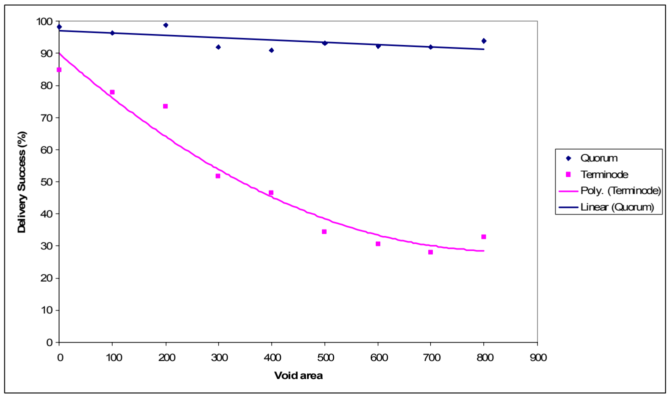

Figure 6: Tolerance of voids (speed=0)

VI. Conclusion

In this paper we described a quorum system that was fault and mobility tolerant for acting as location servers in ad-hoc networks. Our system significantly outperformed the terminode home region approach under all the scenarios tested.

There are many applications other than location servers to which this technique could be applied, with only imagination being a limitation. One idea could be a wiki-based ad hoc network whereby everyone can store and modify information. Another would be the storage of topographical information to aid routing around the network topology where greedy forwarding fails, or where avoidance of traffic hotspots is desirable.

References

[1] T.-C. Hou and V. Li, "Transmission range control in multihop packet radio networks," *IEEE Transactions on Communications*, vol. 34, pp. 38-44, 1986.

[2] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," presented at 6th ACM International Conference on Mobile Computing and Networking (MobiCom), 2000.

[3] S. Giordano and M. Hamdi, "Mobility Management: The Virtual Home Region," EPFL, Lausanne, Switzerland, 1999.

[4] X. Wu, "VPDS: Virtual Home Region Based Distributed Position Service in Mobile Ad Hoc Networks," presented at 25th IEEE international conference on distributed computing systems: ICDCS 2005, Columbus, OH, 2005.

[5] D. Liu, I. Stojmenovic, and X. Jia, "A Scalable Quorum Based Location Service in Ad Hoc and Sensor Networks," presented at IEEE International Conference on Mobile Ad-hoc and Sensor Systems MASS, Vancouver, 2006.

[6] T. Camp, J. Boleng, and L. Wilcox, "Location Information Services in Mobile Ad Hoc Networks," presented at IEEE International Conference on Communications, 2002.

[7] G. H. Owen and M. Adda, "Quorum based geographically static data storage in ad-hoc networks," presented at International Network Conference, Plymouth, UK, 2006.

[8] G. H. Owen and M. Adda, "Self organizing quorum systems for ad hoc networks," presented at International Conference on Communication, Network, and Information Security, MIT Faculty Club, Cambridge, Massachusetts, USA, 2006.

[9] D. Malkhi, M. K. Reiter, A. Wool, and N. Wright, "Probabilistic Quorum Systems," *Information and Computation*, vol. 170, 2001.

[10] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities and Software*: Scribner, 2001.

[11] R. Barr, "An efficient, unifying approach to simulation using virtual machines.," vol. PhD: Cornell University, 2004.

[12] B. Karp, "Geographic Routing for Wireless Networks," in *The Division of Engineering and Applied Sciences*, vol. PhD. Cambridge, MA: Harvard, 2000.