FlavourCast: Context-based Messaging for Ad Hoc Networks

Daniel Cutting, Derek Corbett*
School of Information Technologies
University of Sydney
Australia
{dcutting,dcorbett}@it.usyd.edu.au

Aaron Quigley
Department of Computer Science
University College Dublin
Ireland
aquigley@ucd.ie

Abstract

We propose a novel communication paradigm for ad hoc network middleware called Context-based Messaging and provide evaluation of a simple prototype implementation. The paradigm allows nodes in an ad hoc network to send messages to multiple recipients based on their contextual situation. Unlike protocols which require the sender to specify the names of recipients, or content-based approaches which allow recipients to subscribe to messages based on the message content, our context-based approach allows the source to specify the necessary contextual situation for a node to receive the message. In our prototype implementation, called FlavourCast, the contextual situation of a node is modeled as a set of symbolic attributes which are matched to the attributes specified in the message.

FlavourCast comprises two interacting though independent algorithms. The first uses the metaphor of a landscape to construct a topographical map of the network where clusters of nodes with similar attributes are represented by minima. The second delivers FlavourCast messages from a single source to a number of targets by finding these minima.

Through simulation, we compare FlavourCast to three alternative approaches: flooding, shortest-path multicast and directed random walk. We analyse the results with a variety of metrics and show FlavourCast to be more efficient than both flooding and directed random walk, and comparable to shortest-path multicast.

1. Introduction

Continued improvements in the effectiveness and efficiency of both wireless ad hoc networking and mobile computing devices can support the realisation of what we term massively multiperson impromptu information spaces (MMIIS). Such spaces are overlaid on specific locations (shopping centers, boats, stadiums) or particular events (concerts, cricket matches, football games) where large numbers of people (in excess of 1000) congregate. With large numbers of people carrying mobile devices with suitable wireless communication capabilities, it is conceivable that wireless ad hoc networks can be spontaneously formed without explicit user configuration. Unlike their fixed wireline system counterparts, such networks are particularly problematic for running distributed applications due to their unreliable and unpredictable behaviour. Broadly speaking, our research focuses on building information spaces that offer support for inter- and intra-application coordination in ad hoc networks.

A central tenet for these information spaces is that they do not rely on the provision of a fixed infrastructure. Analogous to the Internet itself, information spaces are not under the control of a single authority but instead are self-forming and self-coordinating. In such dynamic environments, the mobile computing devices act as peers and route information through an ad hoc network in a multi-hop manner. Clearly, a decentralised approach is required due to a combination of problems including device mobility, device constraints and intermittent connectivity.

MMIIS is a form of middleware; software that mediates data-sharing and application coordination between heterogeneous systems. By managing the interaction between disparate platforms and applications, middleware offers a degree of interoperability by abstracting the differences in databases, operating systems, hardware systems and network transport protocols. Commonly referred to as application "glue", middleware is distinct from an application's import or export functions. An example of an established middleware system is the Object Management Group's Com-

*The authors would like to acknowledge the support of the Smart Internet Technology CRC and NICTA, and the helpful collaboration of Gavin Sinclair.

mon Object Request Broker Architecture (CORBA) [16]. Examples of lightweight middleware systems more appropriate to ad hoc networking environments include tuple spaces (LIME [10] and Limbo [3]) and publish-subscribe systems (Elvin [12] and STEAM [9]).

While traditional approaches to the development of middleware for MMIIS are possible, we instead take a "context-aware" approach to the dissemination of data. Context-aware computing attempts to take the current context of the human activity into account when interacting with the users of the system. "Context" can also include information from the sensed environment and computational environment that can be used to alter an application's behavior.

> "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves" [1].

Context includes, though is not limited to, spatial information (location, speed), identity (users and others in vicinity), user models (profile, preferences), temporal (time of day or year), environmental (noise, light), social (meeting, party), resources (printers, fax, wireless access), computing (network bandwidth, login), physiological (hearing, heart rate), activity (supervision, interview), schedules and agendas.

The primary function of our MMIIS is to allow applications and devices to interact with one another without explicitly knowing the names of other devices or the services they make available. Since such networks are very large and spontaneously formed, it can be impractical to discover services and name devices at the application level. Realising that large ad hoc networks are likely to cover significant geographical regions and comprise devices that are very diverse both in capability and contextual situation, we hypothesise that addressing devices by their contextual situation, including available services, current status and location, etc., will open a range of new scaleable applications not currently feasible for MMIIS.

Motivating scenarios where MMIIS are suitable include:

- A military battlefield where fixed infrastructure is typically unavailable. There are potentially many geographic clusters for certain types of context, such as groups of tanks under attack. Such middleware could be useful for delivering contextually appropriate traffic to such groups.
- A sports stadium has low mobility of devices, many inherent clusters (such as groups of team supporters, expensive versus cheap seats, etc.) and would provide a dense ad hoc network of devices. Applications in such a scenario include messaging social groups based on common interests, and requesting photographs of the action from specific areas of the stadium.
- Fighting a forest fire could be aided with an ad hoc network formed by the fire trucks and firefighters along fire fronts. Relevant context could include fire hotspots, trucks running low on water, etc.

1.1. Context-based Messaging

We propose a communication paradigm for a MMIIS called Context-based Messaging, which allows messages to be directed to appropriate nodes in the network without the sender explicitly needing to find or address the recipients. The aim is to address messages not by the names of the recipients, or the content of the message, but by the context of the recipient nodes (the *targets*). This paper describes the implementation and evaluation of a prototype of this paradigm, called FlavourCast. It is assumed that the message does not necessarily need to reach all targets, just "enough", and at a cost far lower than exhaustively flooding the entire network or maintaining global routing table information as with proactive routing protocols.

FlavourCast uses a divide-and-conquer approach, employing two independent algorithms: a topography algorithm, which generates a "height map" over the network based on topological clusters of contextually similar peers, and a delivery algorithm, which delivers the FlavourCast messages from a source peer to as many target peers as possible.

The rest of this paper is organized as follows. Section 2 describes our model, topography and delivery algorithms. Section 3 outlines the metrics, comparative algorithms and the simulation environment used in our experiments and section 3.1 presents and discusses the results. Related work is discussed and contrasted with our approach in section 4. Finally section 5 concludes the paper and discusses future work.

2. Model

The underlying concept of Context-based Messaging is to push messages to nodes in the network that match specified contextual conditions without the source needing to know their names or locations. For example, it may be desirable to send messages to nodes in a sensor network with a temperature above forty degrees or it could be useful to send messages to people

in a sporting stadium who support a particular team. One might imagine that this could even be extended to peer-to-peer (P2P) networks where the types of files peers have to offer becomes the relevant contextual information. Finding files would then be a matter of sending a context-based message to such peers.

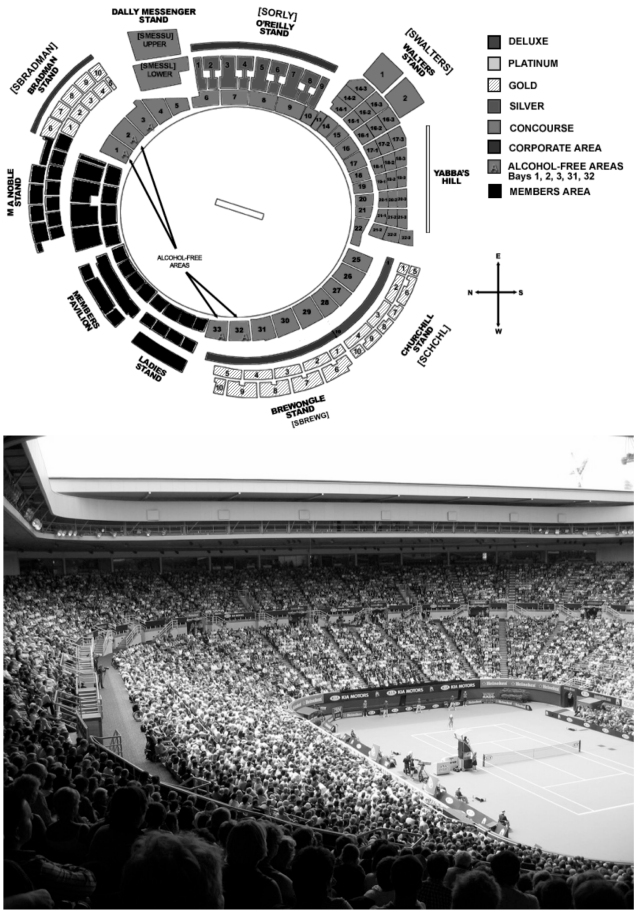

Figure 1. A stadium could host wireless ad hoc networks on the order of tens of thousands of nodes. The Sydney Cricket Ground (top) divides seating according to price and various other criteria.

By considering various scenarios of large ad hoc networks, it is apparent that there are inherent clusters of nodes with similar contexts. For example, in a sporting stadium scenario (figure 1), it is common for supporters of the same team to be seated together. Furthermore, various regions of seating may be classified as "Gold" which might lead one to assume that the supporters in such regions are more affluent than those in other regions, or are more fanatical about the sport: both potentially relevant pieces of contextual information. Similarly, environmental events in sensor networks such as forest fires, will produce clustered high temperature readings in sensors deployed nearby. This premise of inherent contextual clustering is important to our model of Context-based Messaging, as it means that single messages can be routed in the directions of large numbers of target nodes before dispersing upon reaching the cluster.

Our investigation of Context-based Messaging has led us to develop a simplified prototype called FlavourCast. In this model, contextual information is modeled as attributes that are applied to some nodes. In particular we concentrate on a single attribute, RED, that is applied to clusters of nodes in a network. Sending a Context-based Message is thus a case of sending a message to as many RED nodes in the network as possible. The goal of FlavourCast can be summed up as maximising the number of RED nodes reached while minimising the number of transmissions.

FlavourCast comprises two independent though interacting algorithms. The first is based purely on local interaction of nodes and is essentially a network cellular automata. This algorithm uses the metaphor of a landscape to construct a topographical map of the network where clusters of nodes with similar properties are represented as valleys[1]. As the distance (in terms of hops) increases from the clusters, the height rises.

The second algorithm delivers FlavourCast messages from a single source node to a number of target nodes by attempting to find minima in the topography generated by the first algorithm. We discuss these algorithms in more detail below.

2.1. Topography algorithm

The FlavourCast topography algorithm is based on network cellular automata (CA). A cellular automata is a system based on local interactions of "cells" according to a simple set of rules which can result in complex emergent patterns. Often, these systems are arranged as regular grids such that each cell has four or eight neighbours. In a network CA however, cells are nodes in a graph and nodes that share an edge are considered neighbours. In the case of a wireless network, the wireless devices are considered the cells, and the devices within broadcast range are considered the cell's neighbours.

A typical CA has a concept of an universal clock and generations of cells. Each generation is a snapshot

[1] We distinguish the terms *topology*, as meaning the particular node connections in the network (i.e. the graph of nodes), and *topography*, referring to the abstract height map generated by the FlavourCast algorithms with regards to a particular attribute.

of the system as a whole, and moving from one generation to the next is accomplished by applying the rules to each cell in the current generation and updating its state. This synchronous approach is not readily applicable in a wireless ad hoc network. It would be difficult and costly to synchronise the generations across the whole network, particularly if it comprised hundreds or thousands of nodes. Thus the CA model we use in our model is asynchronous. Nodes broadcast beacons periodically to notify their neighbours of their current state. When several beacons have been collected from neighbours, a node can apply the CA rules to modify its state based on the information in the beacons.

CAs have been used to model natural processes such as the spread of infections [14] and heat diffusion [8]. In the case of FlavourCast, we wish to model a landscape based around the locations of clusters of similar nodes. Initially, we based our approach on the heat diffusion CA described in [8] which dictates that the temperature of the cell at the next step will be the average of its neighbours in the current step. If the nodes with the applied attribute begin at a high temperature and the others begin at a low temperature, this CA will diffuse the heat throughout the network producing a topography of hot regions close to the clusters and cold distant nodes. Of course, the problem with this approach is that eventually the temperature of the network as a whole will equalise, leaving no useful topography. One solution to this might be to continually add heat to the system at the clusters, but this would cause the temperature of the network to continually rise and never stabilise, even if the topology of the network were static and reliable. This is a clumsy solution at best and would result in continuous network traffic in the form of beacon updates. A better solution would stabilise the topography when the topology itself was stable.

Our second approach was to use a metaphor of springs connecting neighbouring nodes. RED nodes would tend to pull down on the springs and the others would tend to pull up, constrained by the springs. This would mean that large clusters of RED nodes would produce valleys in the topography. This approach worked well. It created topographies that had deeper regions where more RED nodes were located. However it took a long time (i.e. many beacons) to converge to a stable topography and the areas just a few hops beyond the clusters were essentially flat, meaning no useful gradient information was available in more remote parts of the network. This could possibly be improved with future research, and is an area we would like to explore more fully.

Our third approach, and the one currently employed

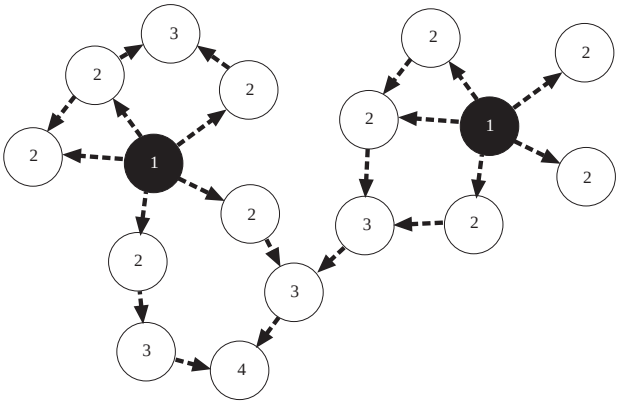

Figure 2. Topography algorithm: if a node has the RED attribute, it sets its height to 1. Otherwise, it sets its height to be the minimum of its neighbours plus one.

by FlavourCast, simplifies the rules greatly. If a node has the RED attribute, it sets its height to one (the global minimum). Otherwise, it sets its height to the minimum of its neighbours plus one (figure 2). These straightforward rules result in gradients that extend far from clusters into the network, and stabilise very quickly (usually a node need only set its height once). The minima in the topography correspond to those nodes that have the attribute applied, and valley walls form around them. When two distinct clusters are nearby, a discontinuity, or ridge, forms at the midpoint. Figure 3 shows such a generated topography for a grid network topology fifteen by fifteen nodes in size.

2.2. Delivery algorithm

The delivery algorithm is quite straightforward. At its simplest, a packet is forwarded to the next hop with the "lowest" height until it reaches a local minimum which by definition is an area where clusters of target nodes reside.

More sophisticated alternatives are possible also. In particular, the FlavourCast algorithm evaluated in section 3 tries not only to find local minima, but traverses ridges in an attempt to find adjacent or more remote minima. This algorithm broadly works as follows: a FlavourCast packet is initially broadcast to all neighbours of the sender. If that neighbour is "lower" than the sender, the packet tries to continue downwards. If it is higher however, the packet tries to head upwards. Each neighbour decides whether to forward the packet

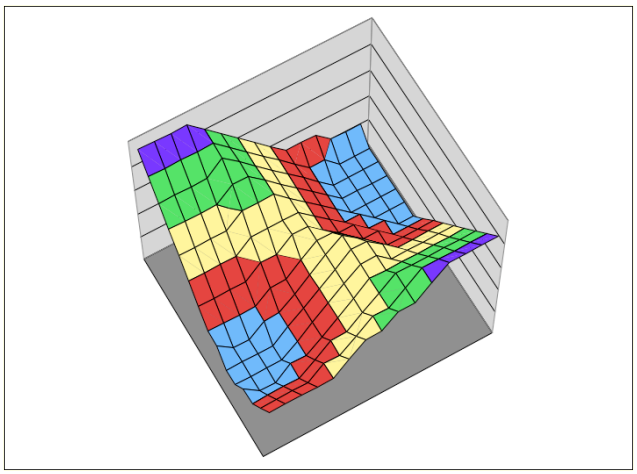

Figure 3. Generated topography for network topology in figures 5 and 6.

```
newNeighbours = neighbours - sender.neighbours
    - sender
lowerNeighbours = newNeighbours[lower]

if #newNeighbours[targets] > 0
        send(newNeighbours[targets])
else if height >= sender.height
        if #lowerNeighbours > 0
                send(newNeighbours[lowest])
        else if #newNeighbours > 0
                send(newNeighbours[random])
        fi
else if #lowerNeighbours > 0
        send(newNeighbours[lowest])
fi
```

```
newNeighbours = neighbours - sender.neighbours
    - sender

if #newNeighbours[targets] > 0
        send(newNeighbours[targets])
else if #newNeighbours > 0
        send(newNeighbours[random])
else
        mostNeighbours = neighbours - sender
        send(mostNeighbours[random])
fi
```

Figure 4. The FlavourCast and DR-walk delivery algorithms.

to subsequent neighbours based in part on whether any next hops satisfy this constraint. Thus some packets are propagated to lower regions of the network where clusters of target nodes are to be found and others are sent over nearby ridges to other minima. Loops are avoided by each node maintaining a cache of FlavourCast packets it has recently seen. If seen again, such packets are silently discarded. Additionally, if a node receives a FlavourCast packet and knows of any immediate neighbours that are targets, it will forward the packet (regardless of the height of the neighbours). In this way, FlavourCast packets tend to bifurcate once a cluster has been reached. The algorithm is presented in pseudocode in figure 4.

3. Evaluation

The evaluation and results described in this section are based on the FlavourCast model described in section 2. As explained above, the goal of FlavourCast is to maximise the number of target nodes reached while minimising the number of transmissions required. To evaluate how well it achieves this goal, we compared FlavourCast to three alternative algorithms using a variety of metrics.

Figures 5 and 6 show how the various algorithms route messages in a sample network. The topography generated by the FlavourCast algorithm for the sample network is shown in figure 3. The alternative routing algorithms we examined were:

- **Flood (figure 5)**. This is the classic flooding algorithm, where each node simply rebroadcasts each message it receives, unless it has received it previously. Flooding is included to provide a "worst case" baseline. Although flooding a message throughout the network means a FlavourCast will reach every target node, it will do so at high cost.

- **SP-multicast (figure 6)**. Shortest-path multicast finds the shortest route from the source node to each target. This is included as a "best case" algorithm. Each target node is guaranteed to receive the message, and the cost for doing so is close to minimal. It is suboptimal because there will be cases when the shortest path from the source to a particular node could be marginally extended and amortised into a route for a different node (thus saving a number of transmissions overall)[2]. However SP-multicast gives a fairly close approximation to the ideal case. Our implementation of SP-multicast uses global knowledge of the network to calculate the shortest paths; it has not been implemented as a realistic networking protocol as it

[2]An optimal implementation might use Steiner trees to find the minimal subgraph connecting all target nodes and the source node. However. this is a reasonably difficult problem, and it was felt the SP-multicast would suffice for our purposes.

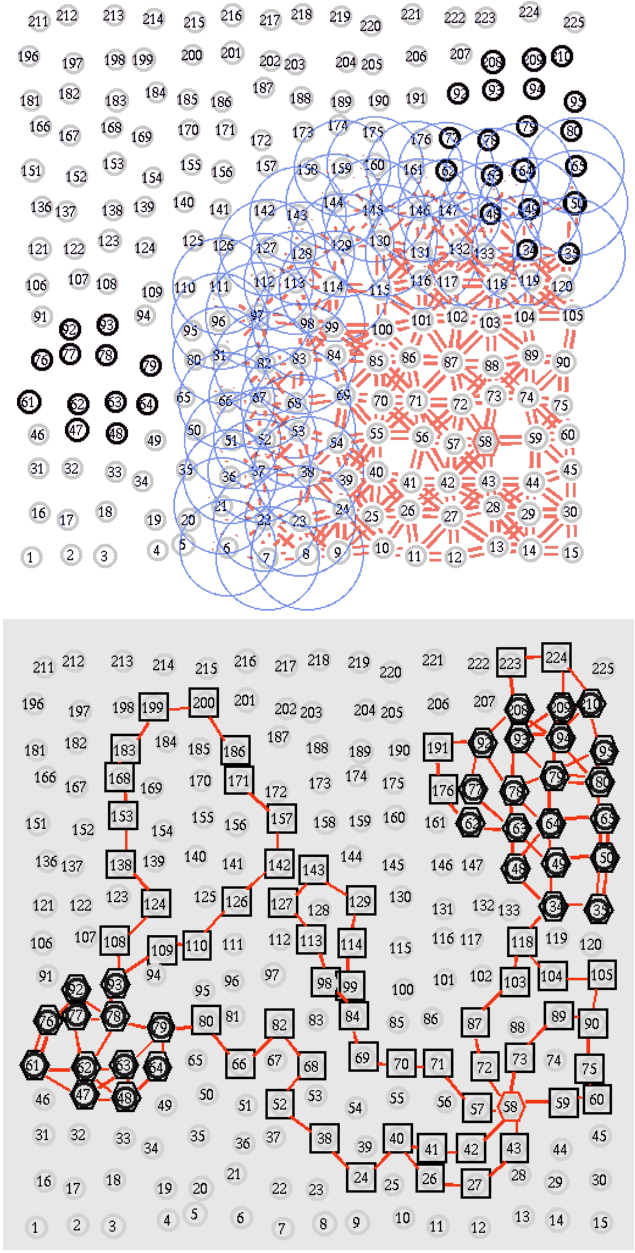

Figure 5. A sample run of Flood and DR-walk.

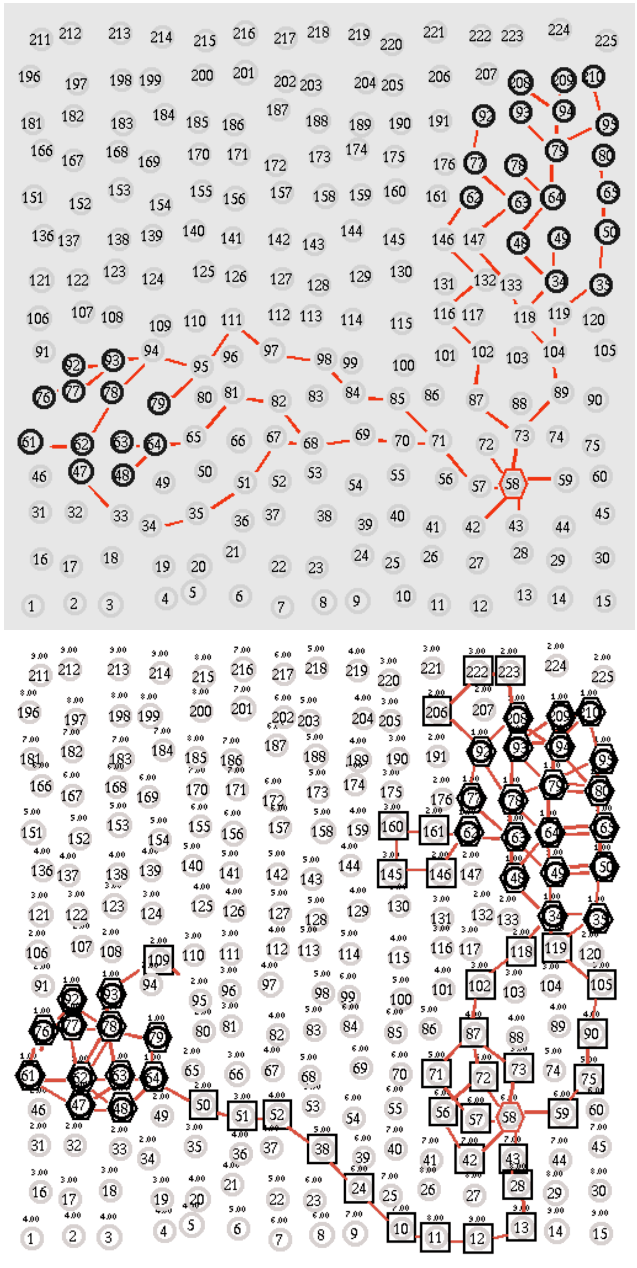

Figure 6. A sample run of SP-multicast and FlavourCast.

is included purely for comparison.

- **DR-walk (figure 5)**. Directed-random walk is based on the algorithm described in [2] and given in pseudocode in figure 4. At the beginning of the simulation, each node beacons to its neighbours so they are aware of one another's existence. A node initiates a FlavourCast by broadcasting to all of its neighbours. At each hop, a list of the current neighbours are added to the packet and forwarded on randomly in such a way as to avoid the neighbours of the previous hop. In this way, the packet is generally propagated away from the source, allowing it to reach distant areas of the network more quickly.

The metrics used to compare the algorithms were:

- **Efficacy**. Average percentage of targets reached. This metric measures how well the algorithm delivers the message to all target nodes. Clearly, a high value is preferable.
- **Efficiency**. Average dissipated energy per node. This metric shows how efficient the algorithm is as a whole. More energy dissipation means the algorithm is generally less efficient.
- **Fairness**. Standard deviation of dissipated energy. If the standard deviation is small, then all nodes dissipate energy at a similar rate. If large, then certain nodes are being required to do more work than others, leading to faster battery failure.
- **Latency**. Average number of hops from the source to a target. This metric measures the typical path length for a FlavourCast. Preferably this should be a lower value, so as to keep FlavourCasts timely.
- **Overall**. Average number of transmissions per FlavourCast divided by average fraction of targets reached. This gives an overall indication of how "good" an algorithm is. A low value means that the algorithm reaches a relatively high number of targets for the number of transmissions made.

The simulation environment consisted of a custom packet-level simulator and analyser written in Java, and the NAM visualisation tool. We made no attempt to model the physical, link or MAC layers in detail; these lower layers were generalised into various modules that greatly simplified the simulation. In particular, we assumed the following characteristics for all simulations:

No mobility. We did not investigate how mobility affected the various algorithms. See section 5 for further discussion.

2Mbps reliable, no-latency link. We assumed a 2Mbps wireless network interface capable of reliably transmitting packets between nodes within transmission range. The length of time taken to transmit a packet was directly proportional to its size. We used the power consumption values reported in [11] as measured in [5]. Transmit uses 1327mW, receive uses 966.96mW and idle uses 843mW. Beacon packets for the topography algorithm were always 256 bytes in length, and FlavourCast packets were always 1024 bytes. Thus at 2Mbps, these packets respectively took 1ms and 4ms to transmit.

Limitless energy. Each node was assumed to have infinite energy, and would hence always be available. By assuming limitless energy, we could more easily see how energy usage varied throughout the network over a period of time.

1-hop broadcast and unicast. We assumed the MAC layer provided mechanisms for broadcasting packets to all immediate neighbours or uniquely specifying the recipient neighbour. We did not rely on globally unique addresses for each node: all routing was local.

3.1. Results

We tested the four algorithms across two networks. The first was a rectangular grid-like network of 600 nodes, forty by twenty nodes in size with four clusters of RED nodes dispersed along its length totaling 90 nodes (figure 7). Each node had approximately six to eight neighbours within transmission range. The simulation ran for five simulated minutes, during which 285 FlavourCasts were initiated from random source nodes at a rate of one per second beginning at the fifteenth second. For the FlavourCast algorithm, each node broadcast one beacon per second for the first fifty seconds only.

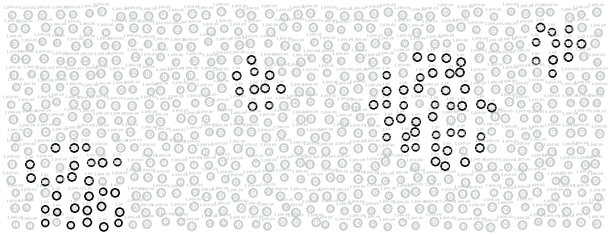

Figure 7. GRID topology: A grid-like network with four major clusters.

The second network topology mimicked a stadium of 1,385 nodes with two regions of expensive seating totaling about 210 nodes (figure 8). Again, each node had approximately six to eight neighbours. This simulation also ran for five simulated minutes, though only 260 FlavourCasts were initiated from random source nodes at a rate of one per second beginning at the fortieth second. FlavourCast beacons were broadcast each second by each node for the first forty seconds only.

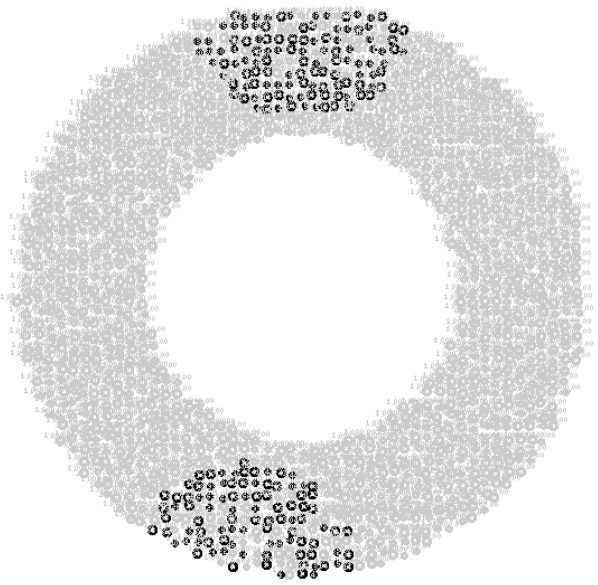

Figure 8. STADIUM topology: A stadium-like topology featuring two clusters of expensive seating (based on figure 1).

Figure 9 shows the average percentage of target nodes reached by each algorithm in each network. As expected, SP-multicast and Flood both reach all target nodes. Interestingly, DR-walk performs much better in the GRID than the STADIUM. This is because the diameter of the STADIUM is much greater than the GRID, meaning the DR-walk has less opportunity to reach a target node before looping back on itself. FlavourCast performs better in the STADIUM than the GRID. This is probably because it more frequently finds 100% of the targets (since there are only two clusters as compared to four in the GRID).

Figure 10 shows the efficiency of the algorithms in terms of the average amount of energy dissipated by each node over the course of each simulation. As expected, flooding is the least energy efficient. Also as expected, FlavourCast is marginally less efficient than SP-multicast, due to its initial beaconing phase to generate the topography. DR-walk scores marginally better than SP-multicast in these simulations. This is because it reaches far fewer target nodes on average, meaning fewer FlavourCast packets are transmitted, whereas SP-multicast will always transmit enough packets to reach every target.

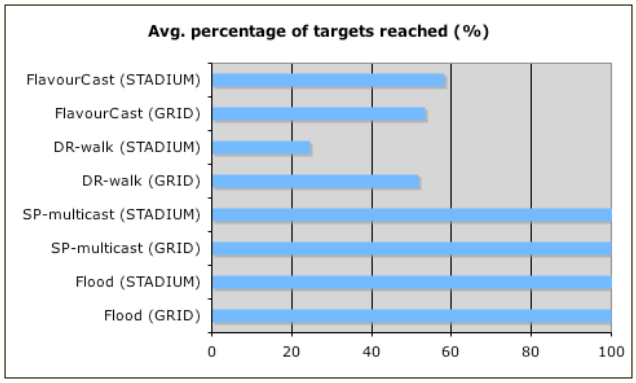

Figure 9. Efficacy. Higher values indicate algorithm finds more targets.

Figure 11 indicates the fairness of each algorithm as defined by the standard deviation of average dissipated energy per node. A low standard deviation would indicate that all nodes have dissipated similar amounts of energy, meaning that no node has been used any more than any other. Obviously since the algorithms are attempting to reach all target nodes, there is going to be some amount of unfairness since the target nodes themselves should be reached more often than the other nodes. The low score for the DR-walk (STADIUM) is due to the fact that few target nodes were reached, thus leaving the system balanced. The slightly higher values for FlavourCast are somewhat expected. It is likely that packets being delivered according to a static topography will traverse the same nodes each time. DR-walk does not have this problem, of course, as it selects randomly at each hop. The high values for Flood are counter-intuitive and unexpected. One would assume that flooding a network would be very fair (though very expensive) as each node would always receive and transmit each packet. It is possible that this is due to a bug in our simulator.

Figure 12 measures the average number of hops from a source to a target node. Those algorithms that deliver to more targets will inevitably have a higher latency, as they will have to travel further to reach more nodes. In particular, the algorithms that perform well in the STADIUM have a high latency simply due to the diameter of the network. Of note is the fact that

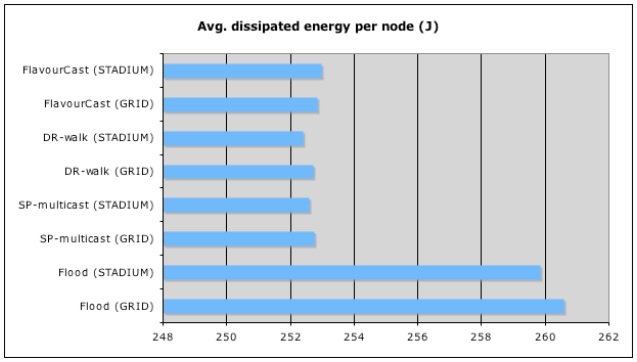

Figure 10. Efficiency. Lower values indicate more energy efficient algorithms.

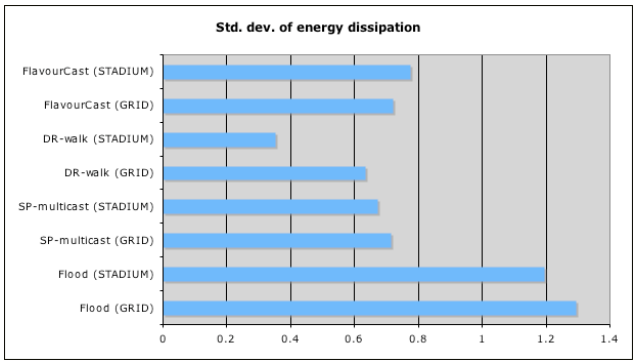

Figure 11. Fairness. Lower values indicate fairer algorithms.

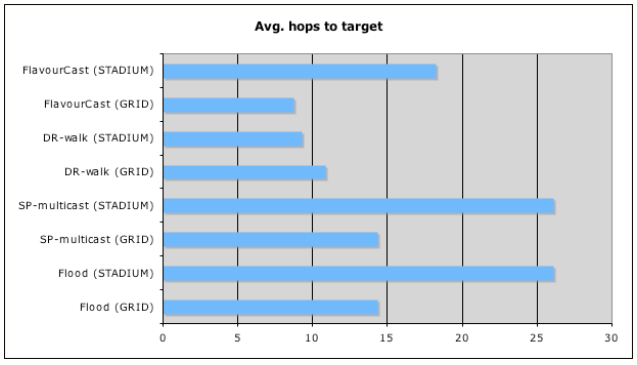

Figure 12. Latency. Lower values indicate fewer hops from source to targets.

FlavourCast has lower latency than DR-walk in the GRID, though it delivers to more targets.

Finally, figure 13 gives an overall indication of how good each algorithm is by finding the ratio of the number of transmissions to the fraction of targets reached. A low value is better meaning the algorithm delivers to a relatively large number of targets at low cost. Not surprisingly, Flood is considered the worst algorithm. Though it always delivers to every target, it does so at high cost. Conversely, SP-multicast has the best score as it always delivers to every target at low cost. According to this metric, FlavourCast performs better than DR-walk on average in both topologies, though the difference is more pronounced in STADIUM where the diameter of the network is greater. FlavourCast also compares favourably to SP-multicast in both topologies.

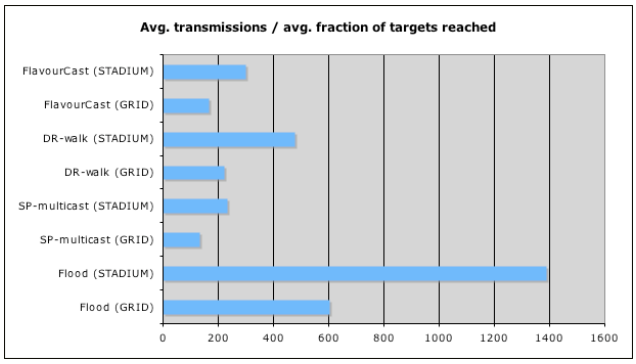

Figure 13. Overall. Lower values indicate "better" algorithms.

4. Related work

The work related to FlavourCast spans two key areas within the ad hoc networking paradigm, routing and middleware. While FlavourCast is a key component in an ad hoc middleware it also contains a routing component.

The simplest ways of routing in an ad hoc network are flooding and gossiping. Flooding relies on each node that receives a data packet to broadcast it to its neighbours. This process continues until the packet either reaches its destination or the maximum number of hops is reached. Gossiping is similar but sends packets to a randomly selected subset of neighbours instead of broadcasting to all. While both gossiping and flooding are easy to implement they are quite wasteful and gossiping offers no guarantees on delivery. As a result

they are seen as the lowest bound in performance of routing for ad hoc networks. Flooding is used in this paper as the baseline comparison for this reason.

The goal of routing is to deliver messages to their intended recipients with the smallest amount of overhead. This has the dual purpose of conserving power within the network and extending network longevity. This focus has given rise to "data-centric" routing protocols which essentially flatten the networking stack so that nodes are referred to by the data they produce (or consume) rather than an address or global identifier. One reason for this approach in sensor networking is that it is simply infeasible to assign a global identifier to every node due to the large number of nodes making up the network. Typically in data-centric routing protocols for sensor networks, a sink sends a request for a certain type of data to a region of the network and then waits for the data from this region to be returned. FlavourCast can also be considered a data-centric routing protocol, as it uses meta-data in the form of attributes to enable routing.

Sensor Protocols for Information via Negotiation (SPIN) [7] was developed as a novel approach to data-centric routing in sensor networks. In SPIN, data is labeled using meta-data. To facilitate the routing of actual data, meta-data is exchanged among nodes via a data advertisement scheme, similar to FlavourCast's topography generation algorithm. When a node receives a new piece of data it transmits the associated meta-data to its neighbours. Interested neighbours can then request the transmission of the actual data. SPIN is more efficient than flooding as the problems of redundant information passing, overlapping of sensed data and resource blindness are solved. However, SPIN offers no guarantees on data delivery as it is possible that nodes between a source and potentially interested destinations are disinterested and do not request or forward the data. As meta-data is only broadcast to the immediate neighbourhood, local interest affects global delivery. Similarly FlavourCast does not guarantee delivery to all targets nodes but this effect is minimised as the topography at any point in the network will enable messages to be propagated in the correct direction without fear of intermediate nodes choosing not to forward the message.

Directed Diffusion [6] is another proposal for data-centric routing in sensor networks. Directed Diffusion uses a data naming scheme to facilitate the diffusion of data through the network. In Directed Diffusion a sink broadcasts an interest for a certain type of data. As the interest propagates away from the sink, interest gradients are set up along the paths. When a source for the data is found, it is returned along the gradients. The sink can then choose to reinforce certain paths by increasing the gradient along it, or lowering the gradient of other similar paths. This is two-phase pull diffusion. A variant, called push diffusion, reverses the roles of sinks and sources. When there is no geographic routing criterion for pull diffusion, queries are generally flooded to the entire network. This works well if the number of queries is low. Push diffusion works well in the converse case where the number of events is small, as it is these that are flooded to the network instead of the queries. The push diffusion model is the most similar in nature to FlavourCast. The attributes which are distributed locally during the topography generation phase of FlavourCast can be seen as queries and the delivery algorithm which is used to push data to nodes with specific attributes is the data generated by events in answer to the queries. However in FlavourCast, this process is decoupled and distribution of attributes and generation of topology is localised.

Rumor Routing [2] is a variation on Directed Diffusion. Rumor Routing seeks to decrease the number of nodes that are queried by routing the queries to nodes that have observed a particular event, rather than flooding the entire network. To propagate data generated in the network, rumor routing uses long-lived packets called agents. When a node detects an event and data is created, it generates an agent to disperse this data. These agents then travel the network so as to propagate information about the availability of data to distant nodes, leaving a reverse path to the event at each node they visit. When a query is generated for data, the query is propagated in a similar way away from the sink. If this query happens to intersect a path taken by an agent earlier, it is able to then follow a route back to the event. FlavourCast is in some ways similar to Rumor Routing. The delivery algorithm in FlavourCast behaves in a similar way to a Rumor Routing agent and Rumor Routing decouples events from queries with independent algorithms. However instead of creating a reverse route to an event, FlavourCast simply uses the agent based approach as a delivery mechanism. By creating semi-static routes to events in the network, there is the potential that certain nodes will be unfairly overloaded in Rumor Routing.

RUGGED [4] uses natural gradients in the sensor network to route messages. These gradients are inherently formed by the environmental effects being sensed and require no construction phase. This can work well for certain sorts of contextual information, but is not applicable to more abstract forms such as user profiles. Furthermore, the extent to which this information propagates in the network is limited by the environmental effects being sensed, whereas FlavourCast

can potentially propagate this information further.

Constraint-based routing [13] allows constraints to be placed on how a message is routed in addition to objectives the destination must satisfy. To achieve this functionality an A*-like algorithm (CB-LRTA*) is used to search for destination nodes by choosing the next neighbour that best satisfies the specified constraints. Back propagation is used so that a route can be refined over time. The algorithm allows the sender of a message to specify the attributes of a node that should receive the message (the "destination objectives"), as well as constraints and objectives for constructing the route to it. At each point, the algorithm tries to maximise the route objectives and minimise the constraints to find the next best hop. Once a node has been reached that satisfies the destination objectives, back propagation is used to reinforce the path for future messages. This is similar in spirit to FlavourCast, in that it is the sender of the message that specifies the type of node that should receive the message without explicitly addressing any particular node. However CB-LRTA* will cease delivery once it has found any recipient which satisfies the constraints whereas FlavourCast tries to deliver to all such recipients.

Attributed-based routing [15] allows messages to be sent to nodes that have specified attributes. It is suggested that attribute-based routing allows applications to easily implement their own specific routing protocol, and that the simplicity of attribute-based routing means it can be enforced at a very low level in the node architecture. The prototype of FlavourCast presented here uses attributes to model the context of recipient nodes. However, we aim to support richer context models in the future.

Constraint-based Messaging is intended as a communication paradigm for middleware to support MMIIS. Traditional middleware systems such as CORBA [16] are not suited to ad hoc networks as they rely on the availability of central infrastructure such as naming services. As a result of the limitations of traditional middleware solutions in an ad hoc network there has been a movement to develop middleware specifically for ad hoc networks. STEAM (Scalable Timed Events and Mobility) [9] is presented as an event-based middleware for mobile computing and specifically for ad hoc networking. STEAM uses a proximity based algorithm in which the delivery of events is limited to areas geographically proximal to the sender. As a result there is no provision for long distance communications and events are only propagated several hops through the network. By contrast the algorithms in FlavourCast seek to contact all interested parties regardless of their distances from the initiating node.

Another approach to middleware for ad hoc networks has been tuple spaces. A tuple space is a virtual space housed on nodes in the network where different nodes can enter and retrieve data. LIME [10] and Limbo [3] are well-known examples of such tuple space-based middleware for ad hoc networks. Due to the rigorous semantics of tuple spaces however, such a scheme is not well-suited to highly mobile or large multi-hop ad hoc networks.

5. Conclusion and future work

The FlavourCast algorithms described above constitute the first prototype implementation of a more general concept called Context-based Messaging. FlavourCast is an unreliable, message-based multicast communication mechanism where the sender is able to specify the contextual situation necessary for a node to receive the message.

This context-centric approach to message delivery has several benefits in ad hoc networks. In particular, no globally unique identifiers need to be assigned to nodes, and no explicit multi-hop routing tables need to be maintained since all communication is localised to 1-hop neighbours. Senders need have no knowledge of the distant nodes that may receive FlavourCast messages or even the number or general topology of such nodes. The underlying network layers need only provide the ability to broadcast a message to all 1-hop neighbours, and allow such neighbours to be distinguished from one another. No further infrastructure such as landmark beacons or location sensors are required.

Being only a prototype, the current FlavourCast model has several deficiencies that need to be addressed in future work to support realistic Context-based Messaging.

The topography algorithm currently used generates the same gradients leading to a cluster independent of the number of nodes that it comprises. Although this simplifies the topography algorithm and leads to rapid convergence of the topography, it does not allow the delivery algorithm to make informed decisions on whether heading towards a cluster is worthwhile (in terms of the number of targets reached for the number of transmissions made). Other topography algorithms we have examined do allow such a determination but are slow to converge. It would be highly desirable to create a topography algorithm that both converges quickly and provides useful hints to the delivery algorithm as to the potential payoff for heading towards a particular cluster, as this would lead to more efficient FlavourCasts.

We have not explicitly considered mobility in our model for FlavourCast as yet, though this is the cur-

rent focus of our research. The reactive nature of the delivery algorithm (i.e. building routes to nodes on demand) suggests that it will be reasonably tolerant of nodes moving or failing. Of more concern is the topography algorithm which is built on the assumption that the general topology of the network will remain in place for some time, even if the nodes themselves are moving. A high beaconing frequency will allow the topography to stay up to date, but the cost could be very high. We propose that the beaconing rate should fall over time when nodes realise they are in stable areas (by receiving fewer beacons from their neighbours, and not receiving beacons from new neighbours). If a node knows it is moving, or receives beacons from neighbours it did not previously know about, it can beacon more frequently.

At present, FlavourCast only supports the delivery of messages to nodes with a single attribute. We are currently investigating how this can be extended to include multiple attributes such that messages can be addressed, for example, as (RED OR HOT) or (NOT RED AND NOT HOT). A logical language such as this would permit more complex delivery objectives and richer applications. We believe such an extension would be relatively straightforward to implement by maintaining separate topographies for each attribute. When choosing the next hop in the delivery algorithm, these separate topographies could be consulted or juxtaposed as necessary and weighted according to the best choice.

The context model currently used is quite limited, allowing only symbolic attributes to be either applied to nodes or not. A richer model is needed to permit realistic applications. One particularly important factor to consider is that some context is short-lived and some is long-lived. For example, the current CPU load of a device in the network is very short-lived and would be stale by the time it had propagated any distance. However, the favourite soccer team of the owner of a device is long-lived and is unlikely to change during the course of an application. Any model of context that supplants the existing model must be able to identify the longevity of contextual information and act accordingly.

References

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.

[2] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks, 2002.

[3] N. Davies, S. Wade, A. Friday, and G. Blair. Limbo: A Tuple Space Based Platform for Adaptive Mobile Applications. In *Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP '97)*, pages 291–302, Toronto, Canada, May 1997.

[4] J. F. Department. Rugged: Routing on fingerprint gradients in sensor networks, 2003.

[5] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, 2001.

[6] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *ACM/IEEE Transactions on Networking*, 11(1):2–16, February 2002.

[7] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8(2-3):169–185, March 2002.

[8] B. Lowekamp, L. Watson, and M. Cramer. The cellular automata paradigm for the parallel solution of heat transfer problem. *Parallel Algorithms and Applications*, 9:119–130, 1995.

[9] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc networks, 2002.

[10] A. L. Murphy, G. P. Picco, and G.-C. Roman. LIME: A coordination middleware supporting mobility of hosts and agents. Technical Report WUCSE-03-21, Washington University, April 2003.

[11] A. Safwat, H. Hassanein, and H. Mouftah. Energy-aware routing in manets: analysis and enhancements. In *MSWiM '02: Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 46–53. ACM Press, 2002.

[12] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings AUUG97, Brisbane, Australia, September 1997*. Distributed Systems Technology Centre, University of Queensland, Australia, 1997.

[13] Y. Shang, M. P. Fromherz, Y. Zhang, and L. S. Crawford. Constraint-based routing for ad-hoc networks. In *IEEE Intl Conf. on Information Technology: Research and Education (ITRE)*, 2003. IEEE.

[14] H. Situngkir. Epidemiology through cellular automata. Departmental Working Papers wpe2004, Bandung Fe Institute, mar 2004. available at http://ideas.repec.org/p/bfe/wpsbfi/wpe2004.html.

[15] V. Tsiatsis. *Attribute-Based Addressing and Routing Techniques*. PhD thesis, University of California, Los Angeles, 2001.

[16] Z. Yang and K. Duddy. CORBA: A platform for distributed object computing (a state-of-the-art report on OMG/CORBA). *Operating Systems Review*, 30(2):4–31, 1996.