BIOANS: BIO-INSPIRED AMBIENT INTELLIGENCE PROTOCOL FOR WIRELESS SENSOR NETWORKS

Michael Breza[#], Richard Anthony*, Julie A. McCann[#]
[#]Department of Computing, Imperial College
*Department of Computer Science, University of Greenwich, London, UK
{mjb04,jamm}@doc.ic.ac.uk, R.J.Anthony@gre.ac.uk

ABSTRACT

This paper describes the BioANS (Bio-inspired Autonomic Networked Services) protocol that uses a novel utility-based service selection mechanism to drive autonomicity in sensor networks. Due to the increase in complexity of sensor network applications, self-configuration abilities, in terms of service discovery and automatic negotiation, have become core requirements. Further, as such systems are highly dynamic due to mobility and/or unreliability; runtime self-optimisation and self-healing is required. However the mechanism to implement this must be lightweight due to the sensor nodes being low in resources, and scalable as some applications can require thousands of nodes. BioANS incorporates some characteristics of natural emergent systems and these contribute to its overall stability whilst it remains simple and efficient. We show that not only does the BioANS protocol implement autonomicity in allowing a dynamic network of sensors to continue to function under demanding circumstances, but that the overheads incurred are reasonable. Moreover, state-flapping between requester and provider, message loss and randomness are not only tolerated but utilised to advantage in the new protocol.

Keywords: service-selection, autonomic, emergence, wireless sensor network.

1 Introduction

A highly efficient distributed protocol to drive autonomicity in sensor network is described. The protocol has been specifically designed to limit the amount of communication that actually occurs during the negotiation of service provision in large-scale self-configuring systems with limited resources (communication bandwidth and battery power).

In particular, this paper is concerned with applications which gather context and/or environmental information from wireless sensor networks (WSN). Applications in this domain have some common requirements which include: robustness, the ability to reconfigure dynamically; scalable deployment platforms; stability despite configuration change; efficiency in the use of systems resources, especially in relation to the number of messages transmitted as scale increases; and low communication latency because the applications can have a real-time aspect.

2 The road to Ambient Computing

In Weiser's vision of ubiquitous or calm computing, technology would be integrated or embedded into our environment, and would be able to adapt to changes in that environment and its user's demands automatically [19]. Therefore introducing autonomicity, the ability to self-manage, to sensor networking is key to the achievement of calm computing. An example low scale application could be medical monitoring in the home where the user patient is unable to carry out technical support therefore it is imperative that systems fully self-manage. The other application extreme is environment monitoring e.g. building ambience or movement sensing of glaciers, which can involve potentially 10,000's of sensor nodes. These nodes must self-configure, self-optimise to maintain application performance and battery life while the glacier is moving and self-heal or degrade gracefully as some nodes will inevitably die.

To achieve application-level self-management in sensor networks we have developed a decentralized, lightweight, scalable yet powerful protocol called ANS. ANS (Autonomic Networked Services) executes on each of the sensor nodes. We assume a degree of redundancy whereby a given node can have many functions e.g. a video sensor might be relaying patient location as its primary function, but also may be capable of analysing the gait of the patient should that be required. Likewise, a sensor node may be used as a gatherer of environmental information at one moment, and then in the next

moment serve as a relay for a node that cannot afford to communicate its data due to dwindling power reserves. This redundancy is key to the function of ANS and based on the well-established principle that sensing is cheap while communication is expensive. Sensed environmental characteristics such as location, gait, temperature etc. are termed *context services*. We use the term Quality of Context (QoC) to indicate the extent to which the sensed data meets application-specific requirements such as resolution, precision, and sample rate (note that QoC is differentiated from QoS in that the latter is concerned with the extent to which service is provided, such as reliability in the provision of sensor data, whereas the former is concerned with the characteristics of the data itself). If an application requires a location service it will initiate a request identifying a level of QoC required (e.g. degree of precision or accuracy of reading as a percentage), and thus accept service from the best-suited device.
When a given device is no longer providing the appropriate context quality the application running ANS uses the functionality of ANS to automatically reconfigure to another context provider that is closest to the appropriate QoC. This self-optimisation can find a new context or service, not available when the initial provider was selected. The ANS node, using a given context, is kept aware of that quality/accuracy though the use of a periodic message, piggy-backed on top of requested sensor data, which describes a given device's current quality of context. When the QoC becomes beyond the range suited to the requester it issues a re-tender request. Alternatively, a re-tender message can be issued periodically. Here, the re-tender is broadcast to the network and the sensors that match the context supply their QoC and a binding is made between the requester and that node's service thus ensuring system self-optimisation. Through this simple mechanism the ANS allows us to build self-configuring, self-optimising, and self-repairing applications for sensor networks and other service oriented systems that require autonomicity.

The ANS protocol by its very nature exhibits engineered emergence and its evolution is bio-inspired, therefore we describe what we mean by this in section three. Quality is described in terms of quality of context (QoC) rather than Quality of Service, therefore Context and Context awareness is described in section four. Section five covers autonomicity and section six deals with emergence aspects of the protocol. Section seven describes the experiments carried out and their respective assumptions. The detailed experimental results are presented in sections eight, nine and ten while related work is discussed in section eleven and then we conclude.

3 Emergence concepts for sensor networks

Emergence describes higher-level states, patterns or other behaviours that arise in systems of numerous lower-level components that have local autonomy to interact with their neighbours. The individual components are typically quite simple and operate with only a local view of the system and yet there are many examples in nature where highly optimized global behaviour results [9]. The higher-level behaviour cannot be predicted by examining the individual components or their behaviour in isolation. The science of emergence is described in [12] [6] [10].

The term 'engineered emergence' describes the purposeful design of interaction protocols so that a predictable, desired outcome is achieved at a higher level (i.e. *emerges*, at the level of systems or applications), although at lower levels the specific behaviour of individual components at any moment cannot be predicted. Typically a small set of rules operate on limited amounts of locally available (cached) state information concerning the node's execution context and its local environment. See for example [4]. Emergence is employed in ANS to achieve simultaneously scalable and robust negotiation in sensor network applications. The negotiation protocol needs to be stable and predictable in terms of its higher-level behaviour (i.e. a suitable context provider needs to be located within a reasonable time-frame), although the low-level behaviour (such as the actual interactions with and between sensor nodes, and the ordering of events such as message transmission) has elements of randomness and can thus not be precisely predicted.

Engineered emergence is a general approach to building systems that benefit from these characteristics (scale, robustness and stability, but that do not require precise knowledge of lower-level activity or configuration. Sensor networks, which contain numerous sensors each having different QoC characteristics (different locations, different accuracy, different levels of battery life remaining etc.), but fundamentally serving as redundant spares for one another, are a highly suitable domain in which applications can take advantage of engineered emergence.

Traditional design of distributed applications focuses on strict protocols, message acknowledgments and event ordering. Each message and event is considered important and randomness is generally undesirable imposing sequenced or synchronised behaviour which is generally deterministic. Such a design paradigm can lead to inefficiency, for example through large numbers of transmitted messages and additional communication latency, especially when some of these messages do not directly contribute to correct application behaviour at higher levels [5].

Natural biological systems however are fundamentally non-deterministic and there are many examples of large-scale systems that are stable and

robust at a global level; the most commonly cited examples being drawn from cellular systems and insect colonies. Though initially the ANS was not designed with emergence in mind, it has become evident that it exhibits and can exploit further emergent properties. ANS requires that a small number of *appropriate* quality bids are elicited from sensor nodes (service providers) in potentially very large systems. In this application domain it is important to minimise the total amount of communication, the latency of service negotiation, and also to preserve the battery power at each sensor node.

The delayed-bid mechanism (described in detail later) purposely introduces randomness into the basic ANS protocol to spread, in time, the high number of bids sent as responses to a QoC request in large systems. The randomness makes the system non-deterministic as exactly which time a particular sensor node will send its reply, or the order in which replies are received; and thus the actual choice of context provider, is not predictable. Yet this non-determinism can be shown to enhance stability and efficiency, whilst simultaneously reducing resource usage. Results presented in sections nine and ten demonstrate that these benefits are achieved without adversely affecting robustness or latency of service negotiation.

4 Context Awareness

The perception of context and its quality drives the autonomic behaviour of the ANS. Traditionally context is defined broadly as the circumstances or situations in which a computing task takes place [17]. One of the most common contexts is the location of the user or objects of interest. For example, location can be obtained using a variety of alternative sensor types including ultrasonic badges, RFID tags [11]. The quality of the location information acquired by different sensors will be different. For instance, ultrasonic badges can determine location with a precision of up to 3 cm, while RF lateration is limited to 1-3 m precision. This difference is quantitatively application-specific. The application determines the semantics of perceived quality and how this matches with its requirements.

We can define properties, which we call 'quality of context' (QoC) attributes, which characterise the quality of the context data received. It is very important to differentiate between QoC, and Quality of Service (QoS). QoS is concerned with the sensor's ability to *provide service* (i.e. to sense some aspect of the environment and transmit its data). We assume the sensor nodes can provide sufficient QoS (i.e. they function to their specification). QoC is concerned with the quality of the sensed data itself.

QoC is essential to the ANS for choosing the best-suited service among those available when delivering a specific type of context to an application. While different types of contexts will have QoC attributes specific to them, there are certain attributes that will be common to most contexts. Based on [8], we identify the following common attributes: Precision, Probability of correctness, Resolution, Up-to-datedness (age of information when it arrives at the sink) and Refresh rate (rate of generation of samples). In ANS context providers need to specify QoC attributes for the context information they deliver. These attributes may vary over time and therefore must be updated regularly.

5 Ambient Intelligence Using Quality of Context

Services and quality of context attributes are what drive the ambient intelligence in the ANS. The aim is to provide the following abilities, based on [13]:

- Self-configuration. Applications tender for the services they require automatically.
- Self-healing. Should a service fail then an application merely has to repeat the tendering procedure to find a new service to replace the one that has gone out of use.
- Self-optimisation. Regular re-tendering keeps the network configuration optimal and allows applications to take advantage of new devices joining the network.

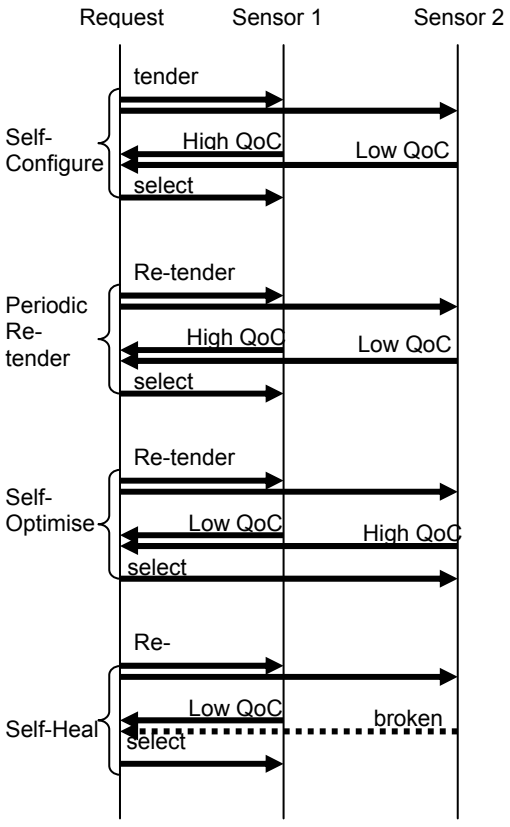


Figure 1. Message diagram of the ANS protocol.

5.1 Tendering and Utility Functions

The ANS uses constructs called 'services' to tie QoC information together with the sensors or actuators that it relates to. A service is a named list of 'commands' and 'events'. Figure 1 is a message diagram showing how the basic protocol is implemented in terms of numbers and types of messages.

A process called 'tendering' is used to select a service to use. When an application wishes to use a service it must broadcast a 'request' command containing the name of the service (such as 'temperature') and its preferences for the QoC attributes. Any devices within range which support the service will use a 'utility function' to calculate how closely they are able to match the requested QoC attributes. Regardless of the number of QoC attributes the result of the utility function will be a single signed integer called 'closeness'. The closeness value is sent back to the requesting device so that it may choose the best device to use. Regular 're-tendering' ensures that applications are always using the most appropriate device and so take advantage of any new devices joining the network. When a service is defined, the QoC attributes that apply to it are translated and scaled if necessary so that they are all of the same order of magnitude from the perspective of a given application [15].

Essentially the utility function treats the available and requested QoC attributes on a device as two points in an n-dimensional space and returns the distance from the requested point to the available point. The application requesting the service will choose the device with the lowest positive closeness since that device will be at least as good as the application wants. If no positive closenesses' are returned then the least negative closeness will be chosen since, while it will not be optimal, it will be the least bad. By not choosing the highest value, applications do not use sensors that are more accurate than they really need. This leaves those sensors more available for use by applications that do need their higher quality, thus aiming to produce a more optimal network configuration. This can also reduce communication overheads by not causing excessively precise information (more data than is necessary) to be transmitted.

6 ANS as an Emergent System

This paper is primarily concerned with selection of context provider(s) based on the quality of information that can they offer. This is determined by a utility function which expresses the application's preferences amongst characteristics such as accuracy and up-to-datedness. Externally deterministic behaviour is required in the sense that applications must be served with context information of the appropriate quality. It is not important that the lower-level behaviour be deterministic; for example it does not matter *which* sensor provides the information at any moment, or *how* the sensor(s) are selected, so long as *some* suitable sensor does provide the appropriate QoC. This relaxation provides an opportunity to take advantage of cheaper (less communication intensive, less synchronous and self-regulating) non-deterministic communication strategies inspired by biological systems such as insect colonies.

Many natural systems have evolved simple and efficient interaction techniques that have contributed to the success of those systems. These techniques typically have common characteristics such as employing randomness (such as in timing mechanisms) and attributing low-value to individual events, actors and messages (the protocols are robust with respect to the loss of some messages, or if some events go unobserved or are unordered)..

There are a number of issues that pertain to self-adaptive protocols caused by the dynamicity of emergent intelligence and moreover there are a number of bio-inspired improvements that can be made to the basic ANS protocol as presented in figure 1. These are discussed below.

6.1 Variable QoC and State Flapping

Many adaptive systems, e.g. networks, have the potential to exhibit state-flapping behaviour. State-flapping is typically indicative of configuration problems (i.e. thresholds set too low) or issues with the dynamism in the environment either caused by the environment itself or by the devices operating therein. ANS is no different in that it aims periodically to self-optimise and if it continuously switches between services there may be destabilisation potential. One extreme situation particular to the nature of the ANS is where the selection of a service in-turn changes that service's QoS value. This we call variable QoC. For example we have two nodes; node A: a fast powerful node delivering an (application defined) QoC of 100 which can serve many requesters, and node B: a slow node that can only service a single requester at time advertising a QoC of 60. The request may be for a Data Aggregation context, requiring the processing of data on the sensor node before sending it back to the requester node. Two requesters wish to have aggregated data at QoC of 90 and will both select node A, who will then apportion its resource between them and give them an actual QoC of 50 each. At the next re-tender, one requester will see a better QoC at node B and switch to that node. This will cause the QoC at A to return to 100. During the following re-tender the requester will see that node A can provide a higher QoC again and return to that. Potentially this requester will flap between node A and B continuously. This is an example of a potential problem with the self-management nature of ANS

protocol as it scales beyond a large number of sensors which could destabilise the whole system.

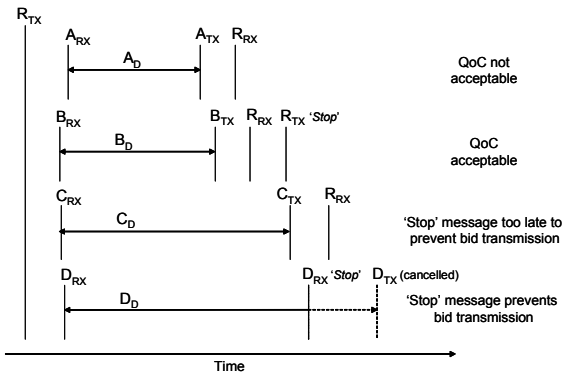
6.2 The Delayed-Bids mechanism

The delayed-bid mechanism [2] introduces a random timing component into the ANS protocol. This breaks the symmetry of behaviour at sensor nodes, spreading out in time the responses to QoC requests. On receiving a QoC request, the sensor nodes locally compute their suitability based on the requested utility function. This is important because the sensors may serve several applications simultaneously. For example, one application may rank precision above resolution and another application may consider resolution to be more significant. Once the node has determined its QoC it transmits a reply (bid) to the requester. The use of a broadcast request is efficient with respect to the simplicity of the protocol and the total number of messages, but introduces a synchronisation point (the receipt of a request implicitly invokes a certain response at each node).

In large systems, near-simultaneous reply-message generation behaviour presents a problem as the communication channel is temporarily congested. This may possibly deny communication service to another, maybe higher-priority, application. In addition, typically only one or a small number of sensors are required to provide information to a particular application, so much of the communication, the battery power consumed at sensor nodes, and the processing of replies at the requester, is wasted.

The delayed bids mechanism directly reduces the bottleneck network congestion problem through the injection of a random delay (locally determined at each sensor) which spreads out the replies. It also provides an opportunity for significant reduction of the number of messages. This is because the response messages are dispersed in time and the requester node can process some messages (e.g. the bid from sensor 'A' in figure 2) before others have been sent. Once sufficient responses with the appropriate QoC parameters have been received a Stop-Bids message is sent (e.g. in figure 2, after the bid from sensor 'B' is received). This has the effect of cancelling all unsent responses at sensor nodes (e.g. in figure 2, the bid from sensor 'D' is cancelled). Some unwanted messages may already be in transmission (e.g. in figure 2, the bid from sensor 'C'), but if the system is tuned appropriately, the large majority of unnecessary messages can be avoided.

A further optimisation can be applied if sensor nodes can eavesdrop on the replies of their neighbours. When transmissions are temporally dispersed there is the opportunity for some nodes to analyse their own operating context before deciding whether to place a bid; i.e. is it a poor quality sensor with better quality neighbours, or perhaps it can offer the best QoC? In this way, if it 'hears' a relatively high quality response from a neighbouring node it need not transmit its own reply.



Key:

R	Requester
A,B,C,D	Sensor nodes capable of providing the requested service
R_{TX}	Requester sends (re)tender request
R_{RX}	Requester receives bid reply from a sensor
R_{TX} '*Stop*'	Requester sends Stop-Bids message
A …D_{RX}	Sensor receives (re)tender request
A …D_{TX}	Sensor sends QoC bid
A …D_{RX} '*Stop*'	Sensor receives Stop-Bids message
A …D_D	Bounded local random hold-off delay before sending QoC bid

Figure 2. Delaying and cancelling bids.

Engineered emergence applications share many of the beneficial characteristics of the natural systems which inspire them. In the delayed-bids enhanced ANS mechanism all messages are deemed to have low-value, and the protocol tolerates the loss of any individual message: if a request receives no replies (within the maximum random time delay for replies) the request message is deemed lost and is repeated; if individual sensor nodes do not receive the request message they simply do not participate in the bidding, this aspect of reliability is actually more beneficial as the scale grows, as it can be actually helpful if the pool of respondents can be diluted in such a free and randomised way. Likewise individual response messages are of low value. If the Stop-Bids message is lost the protocol still functions correctly, it just looses efficiency as the reply-quenching savings are lost.

6.3 Communication Complexity

Let *n* represent the total number of sensor nodes in communication range of the requestor, and *m* represent the number of messages transmitted in a re-tender. All calculations assume a single service requester and no message loss. In large systems the Stop-Bids message reduces responses in a way which can only be treated probabilistically, as it depends on: the specific tuning of the application (e.g. how many responses are required); the tuning of the protocol (e.g. the range of random timeout values); the number of sensors in range of a particular requester at a given time, capable of providing the required service; and the actual random timeouts chosen at each sensor. Let the number of responses cancelled by the Stop-Bids message be *c*, where:

$$0 \le c \le n - 1 \qquad (\text{where } n \ge 1) \qquad (1)$$

(1) states that in the best case all bar one response will be cancelled (this requires that the responses are suitably spread out in time), and that in the worst case (the Stop-Bids message comes too late) no messages are cancelled.

$$m = 1 + 1 + (n - c)$$

$$\text{and thus} \qquad (2)$$

$$3 \le m \le 2 + n$$

(2) calculates the total number of messages required for a single re-tender, taking account of (1), plus the actual re-tender and Stop-Bids messages.

The overall communication complexity depends on the frequency of re-tenders (f), which is application specific. Let the number of messages generated per second be M:

$$3f \le M \le (2 + n) f \qquad (3)$$

(3) scales up the result from (2), taking into account the frequency at which re-tenders occur.

Significantly, (3) shows that the communication complexity is linear in n in the worst case, and thus the protocol will scale well, with predictable worst-case degradation in terms of communication costs.

As an example, in a system in which a requestor has 100 in-range sensor nodes to choose from, and in which a re-tender message is generated every 10 seconds (f = 0.1), the mean number of messages generated per second by the BioANS protocol would be between 0.3 (best case) and 10.2 (worst case) depending on actual tuning.

7 Modelling the ANS to examine bio-inspired optimisations.

Taking the discussions above into account we wish to examine initially how ANS scales to large numbers of sensors, how well it copes with extreme state-flapping and how we can use bio-inspired techniques to improve the overheads and ultimately the performance of the protocol.

The ANS protocol has already been implemented on sensor networks for two applications; patient monitoring in the home, and building usage monitoring. However the numbers of nodes have yet to exceed 5. To further understand ANS under scaling conditions we modelled it as a discrete event simulation, using observed performance parameters obtained from our physical prototype. Our primary aim is to examine the trade off between protocol overhead and performance. Packets sent to the sensor for readings and packets with sensor data are counted as work packets. We consider all communication that is not concerned with performing work as overhead. We define performance as the ability of a requester to receive its desired QoC, and the percentage of overall run time the requester receives that QoC.

The experimental results presented in the paper are built up in a number of stages, evaluating the algorithm's performance as the various stages of operation are incrementally added and tuned. The metrics have been chosen to highlight the cost of autonomicity.

The first set of experiments (reported in section 8) assume that every sensor node can hear all requests and all can service the request. This is an extreme condition in that all devices are in the service pool and active, simulating an environmental monitoring situation with high powered radio or multi-hop functionality. This allows us to determine whether state-flapping will destabilize the system, and to what extent overhead impacts negatively on performance under these extreme conditions. The second set of experiments (reported in section 9) includes reducing the scope of the radio transmission and thus reception radius, thus limiting the number of sensors that can hear a given requester. This change adds a realistic set of constraints that the system will face in deployment in low-radio or building environmental settings. The third set of experiments (reported in section 10) stress test an optimised variant of ANS, which we name BioANS.

7.1 Assumptions

The duty-cycle between re-tenders is an important consideration in ANS; a large duty-cycle between re-tenders lowers protocol overhead at a cost of resilience to sensor failure. In these simulations the duty-cycle is set at an interval of 10 queries (i.e. 10 data requests between re-tender requests).

Variable QoC arises in services where the sensor's ability to deliver its information degrades with each additional concurrent requester. As each requestor binds to that service it reduces the QoC by 33%. A node does not advertise any QoC values that are below 1.

To faithfully simulate the dynamic nature of the sensor network, sensor failure and replacement is built into the experiments. The sensor failures are exponentially distributed with a mean of one failure every 5000 time units [1], and a failure triggers a replacement of one or more new sensors with a replacement time lag exponentially distributed with a mean of 10000 time units. When the sensors are replaced, the number of new sensors is geometrically distributed with a mean sensor node count of 1.6.

[1] One time unit in the simulation model approximates one second of wall-clock time.

When a sensor used by a requester fails, the requester immediately (as soon as it notices the failure) starts the re-tendering process. The QoC of the new sensor is completely random, and it has a ten percent chance of having a variable QoC.

The advertised QoC of the sensors is assumed to be correct[2]. All sensors serve data that is of interest to all of the requesters, but different requesters want different QoC. Each requester requires only one sensor at a time.

The time between packet arrivals is affected by the random back-off algorithm used by the radio link layer BNET [1] that ANS was built on. The arrival of responses from requests (non-random arrivals) were normally distributed with the means and standard deviations taken from the packet traces in [14]. Packet loss, collision and traffic management problems were not modelled, because we assume this to be handled by BNET.

Three metrics were measured in these experiments: 1. The average percentage of time in the simulation run that the requesters got the level of QoC requested; measured in the QoC intervals { ≥ 80%, between 60 and 80%, ≥ 0%, no sensor}. 2. The average ratio of work related packets sent and received by the requesters. This is the inverse of the protocol's overhead. 3. Negotiation time, which is defined as the time from when the request packet is sent out to the send time of the select-sensor packet.

The simulations were run for 10,100,000 time units, with measurements taken at equilibrium, i.e. after a settling time of 100,000 time units. Results are generated as an average over all requesters, each over ten runs.

The results are presented as three sets of experiments. The first set does not restrict radio communication range (i.e. sensor locations, and thus the distances between them are ignored). Here we first look at the affects of state flapping on the basic ANS. The basic ANS is then compared with two variants modified to increase the efficiency (i.e. the ratio of work packets). The second set of experiments takes account of sensor location and restricts radio reception distance. It compares the optimizations of ANS and adds the delayed-bids mechanism. The third set of experiments test the effect of different sensor node densities and failure rates on the performance of fully optimised BioANS. The setup and results of each of these experiments are discussed in the sections below.

[2] That is, the number delivered is trustworthy. We do not consider an incorrect QoC value due to malicious intentions or due to the sensor's inability to correctly determine its QoC. This subject has been tackled in our previous work [20].

8 Experiment Set 1: ANS with high-powered radio

These experiments set out to determine whether scaling the ANS beyond 5 sensor nodes adversely affects performance. Further we examine whether or not the state flapping observed in [15] affects the ratio of work packets to overall network traffic (communication overhead), and for what percentage of time the requested QoC is received. State-flapping is only observed when sensors can have a variable QoC and are serving more than one requester, as described in section 6.1. The assumption is that as a higher percentage of the sensors in the network have a variable QoC, there will be more state-flapping among the requesters, and that this will adversely affect performance.

8.1 Effects of increasing network size

This experiment looked at how ANS scaled with respect to the numbers of packets sent and received and the amount of time QoC was met. The number of sensors was increased from 100 through to 10,000. The number of requesters is also scaled, and is always 10% of the number of sensors. The proportion of sensors with variable QoC was fixed at 10%. Sensor failure and replacement rates were fixed to one in every 5000 (failure) and 10,000 (new sensor) time units respectively.

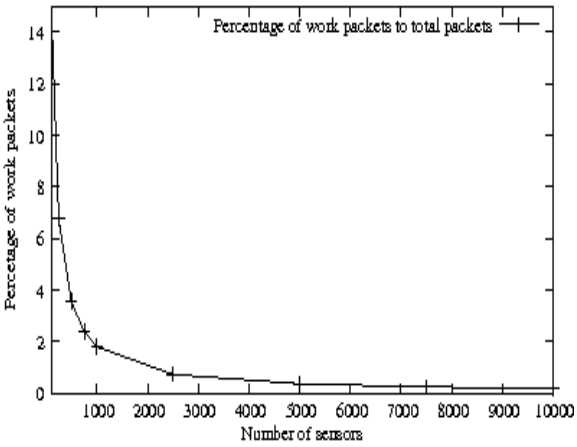


Figure 3. Ratio of work packets to overall packets.

Figure 3 shows the effects of increasing the network size on the ratio of work packets to overall packets sent and received by a requester. We see that only 100 sensors and 10 requesters gave us an average of 15% work packets sent and received per requester (i.e. 85% overhead is observed). Those ratios deteriorate further becoming 2% by 1000 sensors and 1% by 2500 sensors (98% and 99% overheads respectively). However, more positively, at all system sizes the ANS provided the desired QoC to the requesters at a minimum of 94% of the time (not shown). Thus ANS can scale in terms of service provision but this is at a cost. This cost is the number of overhead messages which do not impact

on the speed performance of the system but are very important because for every message we send/receive we consume battery power, therefore shortening the life of the nodes in the system.

8.2 Effects of state-flapping

This experiment was run with 10 requesters and a pool of 100 sensors. The percentage of sensors with variable QoC was adjusted to test the potential effects of state flapping on the number of packets sent and received by an individual requester and the amount of time the QoC requirement is met. The range varied from 0% to 90% in 10% increments.

We observed that as the potential for state-flapping increases, the number of work packets to overall packets stays fairly constant at about 15% (i.e. 85% overhead as before). Figure 4 shows the average time that requesters received their desired QoC decreased as the percentage of sensors with variable QoC increased as one would expect. However, the majority are getting no less than 80% of their desired QoC. The proportion of time for 79% to 60% QoC increases, but the time requesters get 59% and below is always very low.

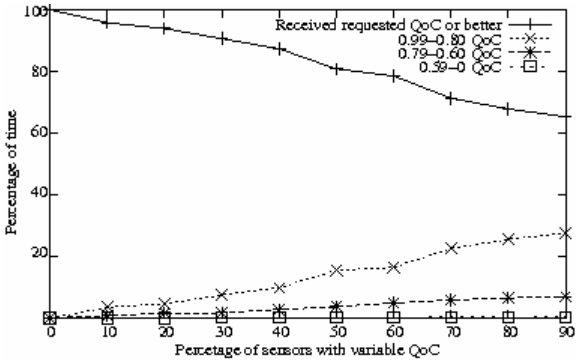


Figure 4: Average time requesters received desired QoC in the presence of state flapping.

These two results show that the adaptation mechanism in ANS provides resilience to state-flapping, without changing the overhead therefore not destabilising the system. There is an associated degradation of QoC received by the requesters as would be expected, but it is not that severe.

Therefore, ANS is certainly adaptive in maintaining a quality of context, but with a very high network traffic overhead. The next set of experiments aims to reduce the unnecessary communication. The first optimisation uses a Stop-bid heuristic. Here we introduce and exploit a random delay before which the sensor node sends its QoC replies. This provides an opportunity for the requester, once it has a suitable response from one or more sensors, to effectively cancel outstanding replies, potentially cutting out a large fraction of the total reply messages. When a sensor receives a Stop-bid message it stops waiting and aborts its reply.

8.3 Optimising ANS to reduce Traffic Overheads

The experiments were conducted in three progressive stages, seeking to improve the communication overhead of ANS. In each case the work packet ratio, and average time from request being sent until the requester receives the last sensor reply was measured. In the first experiment, we ran the basic ANS in systems of between 100 to 1000 sensor nodes. Second, the basic ANS was modified by limiting the amount of time the requestor waits for sensor responses (a timeout of 2 time units was used). This had the effect of reducing the number of responding sensors that were heard. In the third experiment ANS was optimised by examining each sensor response as it was received (this variant is called *first sufficient*). If the requester received a sensor response packet with the requested QoC or greater, then the requester sent a select message (which acts as a stop-bid message), and started using the sensor. All sensors unnamed in the select message cease to send responses. If no sufficient sensors are received in 2 time units, then the requester chooses the best from all the responses already received using the basic ANS selection mechanism. A select message is then sent, and the requester begins to use the selected sensor.

8.3.1 Basic ANS

First the original ANS is tested with sensor populations of 100 to 1000 sensor nodes in 100 node increments. The requester population is always 10% of the number of sensors.

As in the experiments in sections 8.1 and 8.2, the ratio of work packets to total sent and received packets is very small, ranging from 15% at 100 sensors, to 2 percent at 1000 sensors (which concurs with our original experiments in 8.1, see figure 3). As expected, the time a requester waits for all the responses in the basic ANS increases linearly with the number of sensors in the network. This is because the underlying random back-off algorithm in BNet mostly prevents packet collision, and because all sensors have the service that is requested by the requester. The ranges observed ranged from an average of 7.2 time units for 100 sensors, to 72 time units for 1000 sensors.

8.3.2 Basic ANS with a time-out

A time out is added to the re-tender process. Once a request is broadcast, the requester waits for a period of time, then chooses from the responses received, and the select packet it broadcasts acts as the stop-bids message to the unnamed sensors (and a select to the named sensor). The length of time that a requester waits for responses was a constant 2 time units. This has the effect of limiting the number of heard responses to typically 28 sensors (or less, if fewer sensors are available). The mean per-requester

percentage of work packets (as a fraction of total packets sent and received) was observed to be consistently 34% for all network sizes. This improves over the basic ANS which at best could deliver 15% work-packet ratio (see figure 5). The average amount of time the requester received their requested QoC was always above 98%.

8.3.3 First Sufficient ANS

The final experiment of this section optimized ANS further by adding the immediate processing of the responses. The first sufficient response received where the sensor met or exceeded the requester's QoC needs is selected. The 'select' message acts as a stop-bids signal to any sensors that have not yet sent their reply (bid). We call this version 'First Sufficient' or FS. This gave us more information as to how many sensors were needed, on average, to satisfy the QoC requirements of the requesters. The results show that the average number of responding sensors before a suitable one was found ranged between 1.70 for 100 sensors (with a standard deviation of 1.10) and decreasing to 1.55 for 1000 sensors (standard deviation of 0.83). For each request the number of sensors needed was recorded. In all cases the most frequent (first mode) number of sensors needed was 1, and the second (second mode) most frequent was 2. The largest recorded number of sensors responding before a suitable one was found was 39.

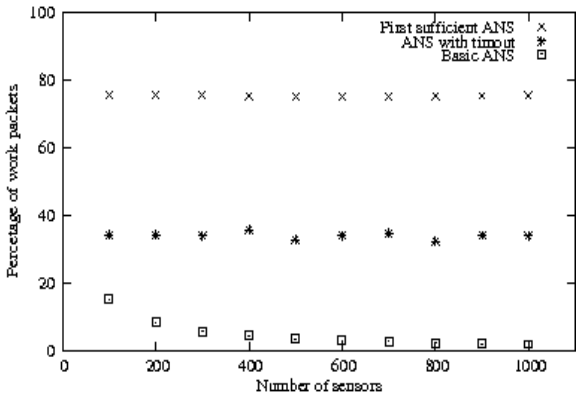


Figure 5. Ratio of work packets to all packets sent and received.

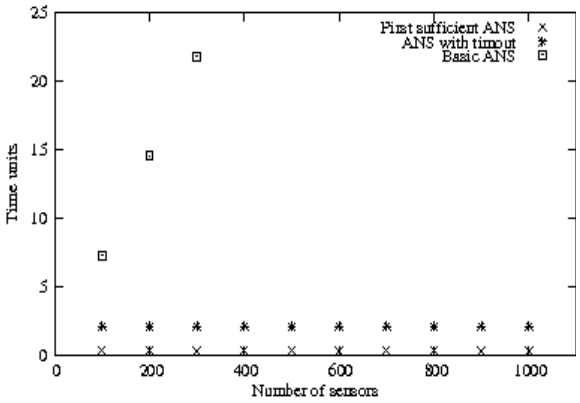


Figure 6. Average negotiation time.
(Note: basic ANS increases linearly off of the graph)

In summary, figure 5 shows us how the ratio of work related packets to all packets sent and received by a requester improves with the various optimisations. By reviewing the response as it is received, we managed to get 75% of all packets sent and received to be work related, thus reducing the overhead of ANS to 25% of the packets sent. This is achieved while providing the requesters with their required QoC 99% of the time.

Figure 6 shows the reduction in average negotiation time after a requester has made a request. The time needed for the basic ANS increases linearly right off of the graph. This illustrates the significance of the ANS with timeout, and the First Sufficient ANS optimisations.

An interesting result of the random back-off communication scheme ANS is built upon is that, in the basic ANS, if the current sensor is still the best sensor, then no change of sensor is made. In ANS with timeouts and optimized ANS, if the current sensor does not have time to respond before another suitable sensor responds, then a change of sensor will occur. In the data we observed an average of 95% of the re-tenders resulting in sensor changes. A simple extension of the protocol, setting delay to 0, resolves this by ensuring that previously used sensor replies immediately. This helps reduce the number of unnecessary sensor changes that occur. As state flapping has not had a significant impact on performance we did not wish to examine this further.

9 Experiment Set 2: Optimizing ANS with Bounded Radio Reception

Location information is now added to the model constraining the number of sensors that can respond to a (re)-tender request. This represents less extreme, more realistic conditions, simulating a smaller number of nodes able to respond to a request due to communication range limitations or restrictions imposed by building artefacts such as thick metal based walls or furniture. A 2-dimensional fine-grained grid of cells is used to describe the area of deployment. The density of sensor nodes in this grid

is measured as the 'density factor'[3]. One or more nodes can reside in a given cell. Consequently, given a constant sensor population, the size of the grid determines the population density of the nodes. For a given node population, a larger grid will have a lower density than a smaller grid. In each experiment (i.e. change in grid size) the nodes are distributed across the grid with a uniform random distribution.

Figure 7 illustrates a typical sensor node distribution with density factor of 0.4 (for clarity, the lines in the diagram are drawn at a distance of 10 cells apart). On average this density factor equates to 5 sensor nodes being in communication range of a requester node.

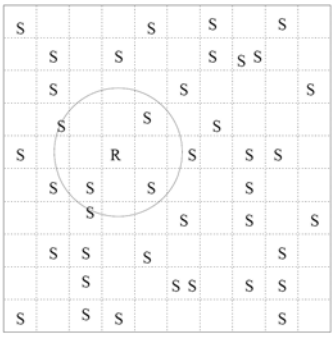


Figure 7. A random sensor node distribution with a density factor of 0.4. Sensor and requestor nodes are represented by the symbols 'S' and 'R' respectively.

The simulation is initialised such that at the beginning each requester can hear at least one sensor. Failures and the constrained number of sensors available can leave a requester in a state with no sensors available.

Once again the goal is to reduce the communication overhead of ANS. The work packet ratio, average number of sensors responding per request, and average time from request being sent until the requester receives the last sensor reply are measured. The experiments compare the performance of four variants of the protocol. The first three of these are the same as used in the section 8 experiments (basic ANS, basic ANS with a 2 time unit time-out, and FS ANS). The fourth variant 'FS with delayed-bid' employs a delayed-bid mechanism (at sensor nodes) as described in section 6.2, in addition to the FS behaviour (which operates at the requestor node). This new variation is further optimised by allowing sensor nodes to choose if they are going to respond to a re-tender request. This is determined by whether they can provide at least 60% of the QoC asked for. If not, the sensor remains silent, therefore further reducing protocol overhead.

The experiments are run in sensor node populations of 100 through 1000 nodes, with a density factor of 5. The results are discussed below and summarized in figures 8, 9, 10 and 11.

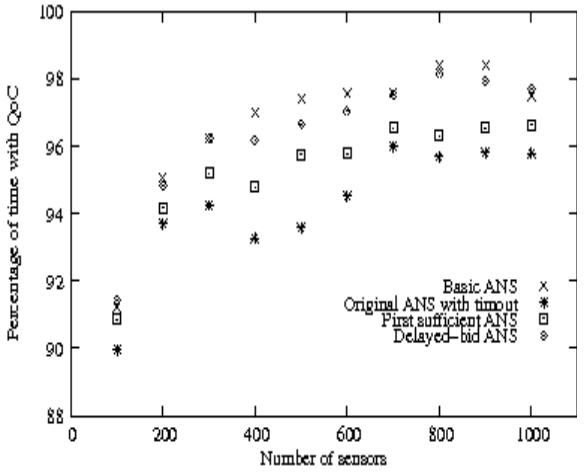


Figure 8. Average percentage of time requesters got QoC.

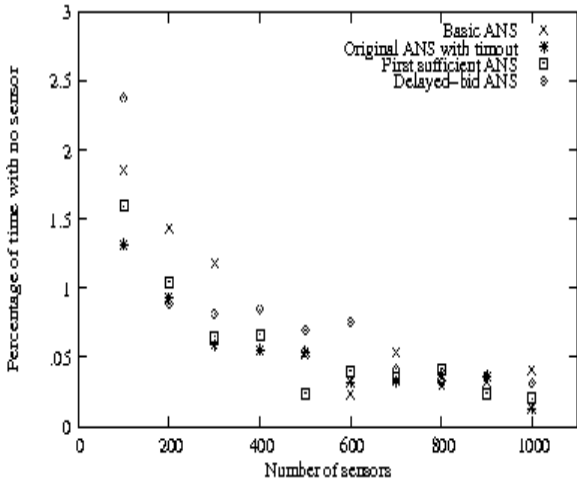


Figure 9. Average percentage of time requesters had no sensor.

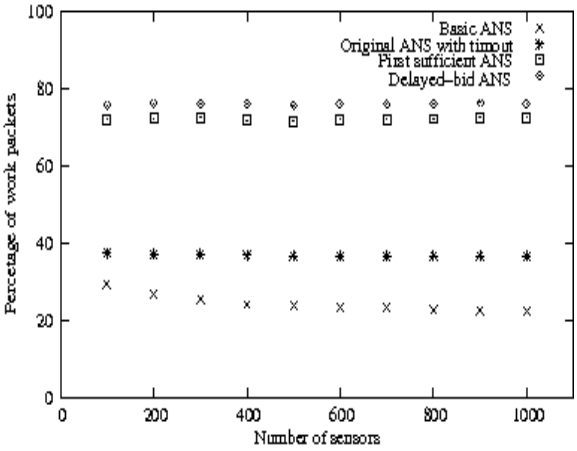


Figure 10. Ratio of work packets to overall packets.

[3] The term 'density factor' is defined as the mean number of sensor nodes within a 100 cell area of the grid; i.e. a density factor of 1 implies that there is an average of one sensor node per 100 grid cells. Each type of node has a wireless range (radius) of 20 cells, with the assumption that there is no interference to limit range. Thus its communication range covers $400\pi \approx 1257$ cells.

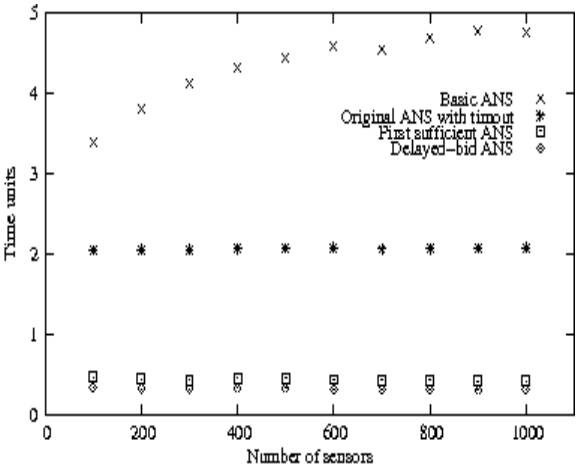


Figure 11. Average re-tender time.

9.1 Basic ANS with location

Figure 8 shows that the basic ANS with restricted communication range consistently provides its requesters with their requested QoC for the highest percentage of time. However, figure 9 reveals periods when requesters were receiving no sensor. All of the variants of ANS showed a similar trend of better average time with the desired QoC and lower average time without a sensor, when the number of sensors increased. Intuitively this is what one would expect given that a requestor has more chance of picking up a service at the required QoC if there are more of them available to it. For basic ANS the percentage of work-packets sent (see figure 10) has increased to 29% for 100 sensors, and 22% for 1000 sensors (i.e. overhead drops from 75% to 71% and from 98% to 78% respectively, compared to the results in section 8). The time taken for a re-tender (shown in figure 11) now starts at an average of 3.3 time units for 100 sensors, and only rises to 4.8 time units for 1000 sensors. Clearly, adding the constraint of location to the model improves the ANS overheads, nevertheless, performance can be improved further.

9.2 Basic ANS with Time-Out and Location

By ignoring bids that arrive after a period of 2 time units, we see the same pattern of results as in the first set of experiments; section 8.3.2. With the location communication restriction added, the time-out enhancement gives the lowest average percentage of time requesters get their requested QoC (figure 8). In figure 9 ANS with timeout gives the lowest average percentage of time with a requester getting no sensor. The percentage of work packets in figure 10 is improved to 38% regardless of the number of sensors. The average negotiation time is confirmed to be 2 time units (see figure 10). This is a direct consequence of stopping waiting for bids after 2 units of elapsed time.

9.3 First Sufficient ANS with location

FS ANS shows the same trend as the other versions of ANS of increasing the average time its requesters get their QoC (figure 8), and reducing average time requesters are without sensors (figure 9) as the sensor population increases. Figure 10 shows that the work packet ratio is similar to the previous section 8 experiments, again significantly better than the basic ANS with or without time-outs. The first sufficient algorithm incurs similar communication overheads regardless of whether or not the transmissions are bounded by location; see figures 5 and 10. The results in figure 11 show very low re-tender times.

9.4 First Sufficient ANS with Delayed-bids and location

The technique of delaying sensor bids improved the performance of ANS with regard to our metrics. The average proportion of time which the requesters got their QoC showed the same trend as the other versions of ANS, as did the average time without a sensor (figures 8 and 9). The work packet ratio (figure 10) shows further improvement of about 3-4% over FS ANS, and shorter re-tender times than FS ANS, see figure 11. Whilst all versions of the protocol are capable of delivering high QoC for high proportions of time, the delayed-bids variant delivers this performance with lower communication overheads and latency.

The experiments in this section confirm the scalability of ANS. As expected all of the versions of ANS perform generally better when the sensor density is increased.

10 Experiment Set 3: BioANS under Demanding Conditions

The optimised variant of ANS, i.e. with the first sufficient optimisation operating at the requester, and the two optimisations at the sensor nodes (i.e. the delayed bid and the sensor determination of whether to provide a bid) is referred to as BioANS in the remainder of the paper. The name reflects the biological systems inspiration for the enhancements to ANS.

The experiments reported in section 9 showed little affect on the QoC. This is because the density for each location range remained high. The experiments reported in this section examine how the density of nodes within a given range affects the QoC, the goal being to establish the break-off point between a density that provides sufficient choice of sensor for a requestor and a density that begins to impact on how the protocol deteriorates. We then select the extreme conditions of a low node density and introduce varying levels of sensor failure to examine how BioANS operates under demanding conditions.

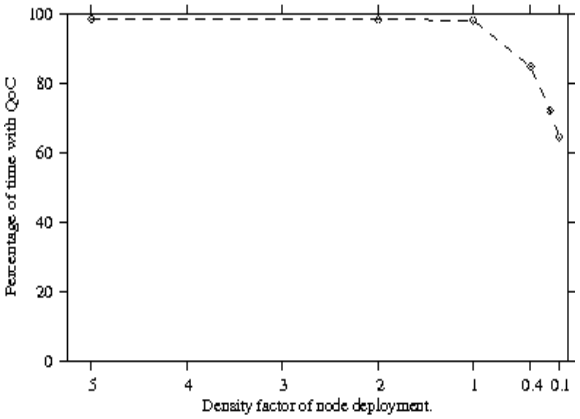


Figure 12. Relationship between density factor and the average percentage of time requesters got QoC.

10.1 BioANS and varying Node Density

Given a communication range of 20 cells, the density of the sensors in the deployment area will affect the performance of the protocol. Figure 12 shows the average QoC received by 50 requesters in a network of 500 sensors.

The average time that requesters got their desired QoC is almost 100% until the density factor falls below 1. At that point the average time that QoC is received falls away. At density factor 0.4 the requesters are only getting their QoC an average of 85% of the time. At a density factor of 0.2 that figure has dropped to 72% and to 64% at a density factor of 0.1.

10.2 BioANS performance at a Low Fixed Node Density

Given the results in section 10.1 above, we decided to test the performance of BioANS as it scales in network size. The density factor was set to 0.4 to examine the performance at the critical density where it begins to deteriorate. The results are summarised in figures 13, 14 and 15.

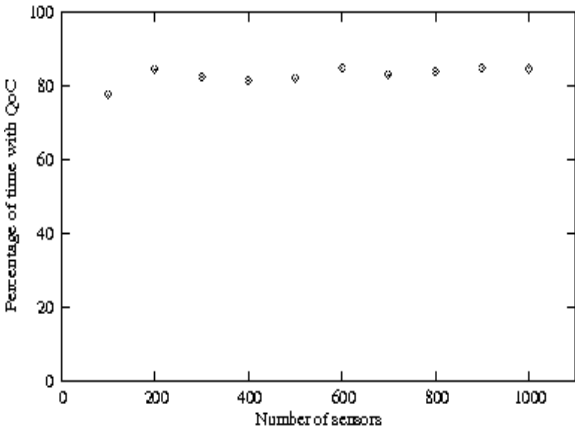


Figure 13. Average percentage of time requesters got QoC as network size increases with fixed density.

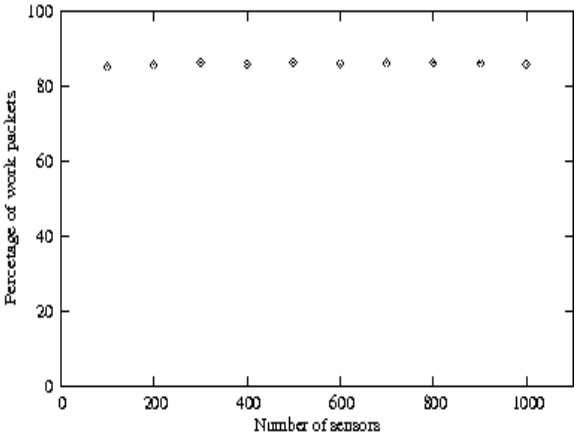


Figure 14. Ratio of work packets to total packets.

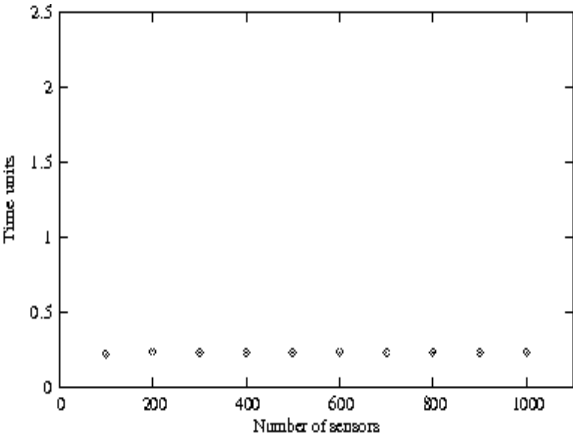


Figure 15. Average re-tender time as network size increases with fixed sensor node density.

These experiments show that BioANS scales to at least systems of 1000 sensors and 100 requesters (the maximum population tested in this experiment). Figure 13 shows that the average time that the desired level of QoC received is almost the same for all network sizes. The average time requesters had no sensor fluctuated a little, but remained consistently below 1% most of the time. The percentage of work packet traffic to total traffic, is constantly high however, see figure 14. Latency, represented here as average time a re-tender took to complete, is also low, and consistent among all of the populations tested, see figure 15.

The fundamental reason for such consistent results is that the sensor density was constant. However the requestors are randomly located and can 'interfere' with each other when they use nodes with variable QoC. Thus the important result is that the effect of such interference changes negligibly as scale increases.

10.3 BioANS performance with failures

To test the robustness of BioANS, we run experiments where the failure and recovery rates of sensor nodes are varied. We also watch the degradation of the network as all of the nodes fail

without replacement, and as the network recovers from a state of no sensors, to a full population of sensors.

All of the previous experiments are run with a failure rate of one sensor node failure every 5000 time units distributed exponentially. The recovery rate is half that, with new nodes being added every 10000 time units, but added in batches (one or more) using a geometric distribution with a mean of 1.6.

The previous experiment shows that the network is stable as the size increases. In this experiment, we increase the failure rate across a range from one in every 5000 time units to one in every 10 time units. The recovery rate is always half the failure rate. The population of the network is always 1000 sensors and 100 requesters, and the density factor is fixed at 0.4 The results are summarised in figures 16, and 17.

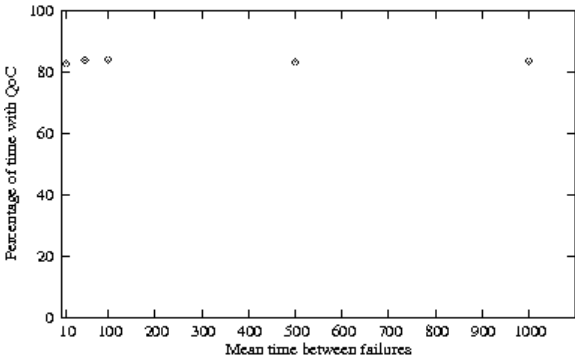


Figure 16. Average percentage of time requesters had QoC in relation to mean time between failures.

Figure 16 shows that the average percentage of time that the desired QoC remains consistently high as the failure rate increases. Only at extremely high failure rates a performance drop is noticed (of course, as failure rate in this experiment was double the recovery rate, the extent of the performance drop would worsen over time). Similar behaviour is observed for the percentage of time a requester has no sensor, the ratio of work packets to overall packets (inverse of the overhead), and the average time for a re-tender to complete. The 'immunity' of BioANS to high sensor node failure rates stems largely from its adaptive re-tender method.

The final experiment looks at how average received QoC degrades as the sensors fail without replacement until the system is completely depleted of sensors[4]. The experiments were run with a density factor of 0.4, varying the sensor node population over the range 0 to 1000. The failure and recovery rates are exponentially distributed with a rate of one every 5000 time units. The results (figure 17) show that BioANS degrades and recovers gracefully.

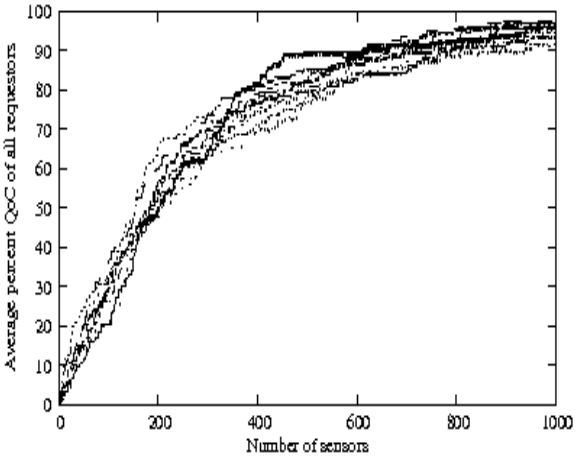


Figure 17. Average percentage QoC as network resumes.

The collective results of the above experiment set show that the bio-inspired optimisations have dramatically reduced the communication overheads and negotiation time, whilst retaining the high robustness, stability and flexibility of the protocol and the high levels of QoC provided to requesters. The optimised protocol is very resilient to highly dynamic network conditions.

11 Related Work

Utility based service selection is gaining interest in the Autonomic Computing community. Some of the current work on this assumes that the utility of services are per application and not shared between applications. The work that allows the sharing of contextual information however, assumes that bulky middleware will buffer this information and drive the self-management of the system therein.

Rajkumar et al. [16] propose a resource allocation model for QoS management within a single system. Resources include CPU utilisation, memory consumption, network bandwidth and latency. Each application delivers to the system the minimum resource requirements it has plus a utility function that returns the increase in performance given additional resources. The system then allocates resources to each application such that the total system utility is maximised.

The Context Toolkit [18] is a framework aimed at facilitating the development and deployment of context-aware applications. It was one of the first projects in this area and is often considered a reference framework which has inspired this work as well as many other projects. The Context Toolkit abstracts context services, e.g. a location service, from the sensors that acquire the necessary data to deliver the service. Thus, applications abstract from the sensors that provide the raw data necessary for

[4] The reverse experiment was also run, i.e. starting from a system of no sensors and incrementally adding them. The results were symmetrical (due once again to the robust adaptive re-tendering used in BioANS), so a single graph is sufficient to show both results – the system can operate continuously along the curve depending on the sensor population.

determining context, and access context data through a network API. Further, the Context Toolkit allows sharing of context data through a distributed infrastructure and collection of storage data to create a history. However, unlike our work this middleware infrastructure is quite bulky thus not suitable for sensor applications where the infrastructure is deployed on the actual sensor nodes. Moreover, it does not provide any autonomicity in terms of allowing applications that enter the distributed environment to discover available services: the location of context services (IP address and port number) has to be known in advance. Also, there is no mechanism that allows context services to adapt and react to failure or degradation of the underlying sensor infrastructure, e.g. by switching to an alternative means of acquiring the same type of context.

Cohen et al. have proposed iQueue [7], a data-composition framework for pervasive data. iQueue allows applications to create data composers, specify a composer's data sources using functional data specification, and specify a composer's computation. Similar to our Requester, the iQueue run-time system selects data sources satisfying the data specifications, dynamically reselects data sources as appropriate. The goal is very similar to ours, although our approach somewhat different and again their middleware has not been designed in a lightweight fashion. They use a mechanism similar to our periodic re-tender request, in that a data source issues advertisements periodically, but also whenever properties of the data source, e.g. quality of information, change. It would appear that they use Boolean predicates over the values of the properties of the data source. Instead, we present a mathematical model based on application's wishes that evaluates each applications quantitative satisfaction with regard to any particular data source. The aforementioned centralised solutions are not suitable for sensor networks as many of the nodes are too small to carry this burden and it introduces a central point of failure to the system. Therefore we aimed to carry out the same functionality in a more lightweight and decentralised way, hence our bio-inspired approach.

The inspiration for the methods to optimise ANS is stylistically bio-inspired. In [3] an emergent leader election algorithm is given whose communication style is based on the mechanics of pheromone based communication. Pheromone communication is one-way communication without acknowledgment and is implemented as a local broadcast, with no guarantee of delivery. The emergent leader election algorithm uses the inherent non-determinism of unreliable communication to make a very efficient algorithm for large scale distributed systems. A key bio-inspired aspect of the design is to ascribe low-value to individual messages and nodes, i.e. the algorithm is designed to operate correctly at the global level despite high levels of message loss and node failure. A cluster management scheme presented in [5] incorporates the non-determinism of un-reliable communication in a mechanism to recruit idle nodes for distributed computation. Because of the similarity between the style of communication used in these works, and the type of communication we are restricted to in sensor nets, the optimized ANS is heavily inspired by these algorithms. The vast majority of WSN self-adaptation work has concentrated on the scalable and robust routing of packets from a source to a sink in a WSN whereas ANS is an application-level protocol.

12 Conclusions

The autonomic protocol, ANS, describes all services provided by the WSN as contexts and the quality (QoC) of which this context can be delivered. Applications are then composed of sets of calls to the context providers which provide the most appropriate QoC. This paper therefore seeks to measure the trade-off between performance in terms of speed and quality of delivery against the overheads that the addition of an autonomic protocol adds to a highly distributed, resource scarce wireless sensor network application. To this end we took measurements obtained from a smaller scale WSN running ANS and applied them to a simulation model to observe how the protocol would operate under extreme conditions such as failure or very large numbers of nodes, which further allowed us to carry out partial validation of results.

Our results show that extreme scaling and state-flapping do not significantly affect the Quality of Context delivered to the application. However, the overheads incurred in achieving this in ANS were quite significant. This issue was addressed through a series of progressive innovations and the performance was evaluated at each step. Firstly a time-out was used to reduce the requester wait time; and bids received within the time window were considered at the end of this period. Then a mechanism of examining each bid as it arrives and accepting the first sufficient bid was introduced, still operating within the timeout period. If no sufficient bid was received the ‘best’ bid was used. The ‘select’ message was also used as a ‘stop-bids’ signal which cancels any unsent bids at other sensors. Finally a delayed-bid mechanism was added so that sensors do not all respond immediately causing a communication bottleneck. Spreading out the responses from sensors in this way makes the combination of first sufficient and stop-bid more effective, as a greater proportion of unwanted messages are typically cancelled. We call the highly optimised variant of the protocol BioANS. The delayed bid mechanism has scope for yet further improvement, by arranging that the time-delay is

shorter for nodes offering better QoC. We leave this specific optimisation for further work.

The innovations increase the non-deterministic, emergent, characteristics of the protocol. In combination the enhancements have a powerful effect on performance: in BioANS the negotiation time was significantly reduced (thus enhancing responsiveness) and the number of messages was reduced (communication overheads were dramatically cut from 85% down to 25%, thus greatly enhancing efficiency and scalability). The strengths of the original more-deterministic ANS protocol were preserved; i.e. the high levels of QoC received by requesters, and the high probability of getting a sensor (thus BioANS retains high correctness and reliability).

BioANS was tested over a wide range of sensor node densities. Lower density gives less choice of service-provider sensors. Even at low sensor densities the typical received QoC remained high whilst overheads lessened considerably. This result further confirms that the adaptivity of BioANS is both effective and highly efficient.

Finally, an investigation into the effects of node failure demonstrated the robustness of BioANS' adaptive behaviour.

In conclusion the experiments demonstrate that the bio-inspired optimisations of the basic ANS provide a stable, highly scalable and robust protocol that has general applicability to a wide range of applications in sensor networks and similar resource constrained domains.

13 References

[1] Ans over bnet. *http://www.doc.ic.ac.uk/ asher/ubi/ans/bnet.html.*

[2] R. Anthony. Emergence: A paradigm for robust and scalable distributed applications. 1st Intl. Conf. Autonomic Computing (ICAC), IEEE, New York, pp. 132-139, (2004).

[3] R. Anthony. An autonomic election algorithm based on emergence in natural systems. Integrated computer-aided engineering, 13(1): pp. 3-22, (2006).

[4] R. Anthony. Emergent graph colouring. Engineering Emergence for Autonomic Systems (EEAS), First Annual International Workshop, at the third International Conference on Autonomic Computing (ICAC), Dublin, Ireland, pp. 4-13, (June 2006).

[5] R. Anthony. Engineering emergence for cluster configuration. Journal of Systemics, Cybernetics and Informatics, 4(3), (2006).

[6] J. Casti. Complexification: Explaining a Paradoxical World Through the Science of Surprise. Abacus, London, (1994).

[7] N. H. Cohen, A. Purakayastha, L. Wong, and D. L.Yeh. iqueue: A pervasive data composition framework. In Proceedings of the Third International Conference on Mobile Data Management (MDM), pp. 146-153. (2002). URL *http://www.research.ibm.com/sync-msg/MDM2002.pdf.*

[8] K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16: pp. 97-166, (2001).

[9] M. Gell-Mann. The Quark and the Jaguar: Adventures in the Simple and the Complex. Abacus, London, (1994).

[10] R. Genet. The Chimpanzees who would be Ants: A Unified Scientific Story of Humanity. Nova Science Publishers Inc, New York, (1997).

[11] J. Hightower and G. Borriello. Location systems for ubiquitous computing. IEEE Computer, 34(8): pp. 57-66, (2001).

[12] S. Johnson. Emergence: The connected lives of Ants, Brains, Cities and Software. Penguin Press, London, (2001).

[13] J. Kephart and D. Chess. The vision of autonomic computing. Computer, IEEE Computer Society, 36(1): pp. 41-51, (2003).

[14] J. McCann and A. Hoskins. Proof of concept adaptivity and performance benchmark results. Technical report, Imperial College, London, May (2006).

[15] J. McCann, M. Huebscher, and A. Hoskins. Context as autonomic intelligence in a ubiquitous computing environment. International Journal of Internet Protocol Technology (IJIPT) special edition on Autonomic Computing, (2006).

[16] R. Rajkumar, J. L. C. Lee, and D. Siewiorek. A resource allocation model for qos management. Proc. 18th IEEE Real-Time Systems Symposium (RTSS '97), page 298. IEEE Computer Society. ISBN 0-8186-8268-X. (1997).

[17] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. IEEE Pervasive Computing, 1(4), pp. 74-83, (2002).

[18] D. Salber, A. K. Dey, and G. D. Abowd. The context toolkit: aiding the development of context-enabled applications. Proc. SIGCHI conf. on Human factors in computing systems, pp. 434-441. ACM Press. (1999).

[19] M. Weiser. The computer for the 21st century. Scientific American, 265(3): pp. 94-104, September (1991).

[20] M C. Huebscher, J A. McCann, A Learning Model for Trustworthiness of Context-awareness Services, Proceedings of the 3rd International Conference on Pervasive Computing and Communications Workshops (PerSec), (2005).