Understanding Behavioural Tradeoffs In Large-Scale Sensornet Design

Jonathan Tate Iain Bate
Department of Computer Science, University of York,
York, YO10 5DD, United Kingdom
$\{$jt | iain.bate$\}$@cs.york.ac.uk

Abstract

When designing a complex system such as a sensornet it is not always practical to build and deploy a realistically sized prototype. At the same time many of the interesting behaviours are hard to observe in small scale simulation and many of the current simulators do not scale well. In this paper we use present an efficient but effective simulator which, when combined with modern day large-scale computational capabilities, allow us to measure the impact of tuning an existing protocol and evaluate the necessary tradeoffs.

1 Introduction

Wireless Sensor Networks, or *sensornets*, are an emerging discipline of embedded system and network design. A consistent thread running through sensornet research is that they must operate under highly-constrained resource availability Whether implemented using specialised discrete motes, or as functionality piggy-backed on existing equipment, it is vital that resource usage is kept to an absolute minimum. This maximises the lifetime of networks whose power sources cannot be replenished, and minimises the cost of necessary hardware.

Of course, it is equally important that the sensornet keeps pace with the real world with which it interacts, and has sufficient redundancy to recover from individual node failures. Contradictory requirements lead us to realise that multi-objective optimisation is essential to ensure that a reasonable compromise can be found. To date there is a relative absence of studies examining sensornet behaviour at the scale necessary to provide confidence of their verity and applicability.

Despite an impressive wealth of theoretical treatments of sensornet protocols, there remains a dearth of practical experience and experimental results with which to validate theoretical results and to provide confidence of their verity and applicability. In particular, there are precious few studies of systems of the scale at which the strengths of the sensornet paradigm are best suited. However, the typical research organisation today has significant computing facilities available that allow realistic simulations and experimentation that would have been impractical or impossible ten years ago.

In this paper we measure the empirical response of network performance metrics to changes in network protocol configuration. By examining the consequent relationships, and similarities between such relationships, we can build models that give us insight into the tradeoffs and compromises inherent in tuning and optimising a protocol. We show that these interrelationships are surprisingly complex even where only one parameter is controlled.

We also categorise and measure types of suboptimal behaviour. Specific protocol modifications can be designed to address these specific weaknesses. This is in contrast to the all-too-common approach of proposing new protocols without any supporting evidence of the problems they are supposed to address. In future work we plan to use the information and insights gained to systematically develop new and improved protocols.

The structure of the remainder of this paper is as follows. Section 2 discusses related work. Section 3 defines our objectives, and section 4 describes how simulation experiments address these objectives. Section 5 considers the compromise necessarily inherent in optimising networking protocols against multiple objectives and identifies weaknesses in existing protocols. Finally, section 6 derives conclusions against our stated objectives.

2 Related work

Flooding is perhaps the simplest possible *network routing protocol*; nodes rebroadcast messages to their neighbours, which in turn rebroadcast to their neighbours, and so on. Most non-geographical routing protocols use flooding in some capacity, for example in route discovery, as it requires few resources and no network knowledge [7].

Gossiping extends flooding by having all nodes rebroadcast with probability $p \in [0,1]$. Higher values of p give higher probabilistic delivery guarantees. Bimodal behaviour in p is observed where packets reach either very few or very many nodes, with sharp transition about some critical probability p_c where often $p_c \in [0.6, 0.8]$ [7]. Selecting p is generally difficult. Typical goals include improve delivery probability by preventing *premature gos-*

sip death [7]. A myriad of flooding and gossiping protocol variants have been proposed but to the best of our knowledge none is based on measurements of the defective behaviours for which they are intended to compensate through detailed modelling or experimentation.

Ni *et al.* [10] propose probabilistic, counter-bounded, distance-based and location-based broadcast schemes but do not combine probability p with other factors. *GOSSIP1* [7] sets $p = 1$ for the first few hops before reverting to the standard network-wide p. *GOSSIP2* [7] extends *GOSSIP1* by raising p to $p_a > p$ at nodes with few neighbours, which is particularly beneficial in sparse networks.

3 Objectives

Given a typical configuration of wireless sensor network hardware and application software, and a typical network routing protocol, we defined the following objectives.

Obj 1: *Build an understanding of the compromises and tradeoffs inherent in attempts to tune a networking protocol against multiple competing objectives.*

Obj 2: *Derive requirements for a new protocol from measured weaknesses in existing protocols.*

Obj 3: *Show how large-scale computing capabilities can be used to satisfy the above in an efficient and effective manner.*

4 Measurement by simulation experiment

In this section we consider an experimental method with which to tune a networking protocol against multiple competing objectives through simulation, as implemented in section 5. We consider the simulated network, the simulation tool, and the simulation environment.

Some interesting effects and behaviours may only become evident in sufficiently large networks. Preliminary simulation experiments considered networks of variable numbers of similar nodes distributed with constant spatial density. Qualitatively different behaviour was observed in networks of 200 nodes and of 500 nodes, with additional features and points of inflection appearing in plotted curves. Increasing node count further to 750, 1000 or 2000 nodes did not yield further features. We conclude that a test network size of 500 nodes is sufficient.

Measurement of network behaviour influenced by protocol tuning would ideally take place in physical testbed networks of realistic scale and composition. Unfortunately, economic and logistical factors preclude the construction of test networks on the order of hundreds of nodes for the experiments described in this paper. All experiments were therefore implemented using the *YASS* sensornet simulation tool. For further details of *YASS* refer to [11], which also addresses validation of the simulator.

4.1 Radio propagation model

Scalability is a weakness of many existing simulators. Proposed sensornets may involve thousands or tens of thousands of nodes, but most existing simulators struggle with more than a few hundred simulated nodes [8]. Scalability problems generally stem from $O(n^2)$ growth in possible node pair interactions, depending ultimately on interacting broadcasts in the shared wireless medium.

YASS implements a three-phase radio propagation model to calculate damage sustained to messages being received at sensornet nodes inflicted by other concurrent transmissions that cannot be prevented by the CSMA mechanism. A corollary is that nodes not receiving messages need not be tested at all. This considerably reduces the computation overhead for lightly-loaded networks.

The phases are ordered by increasing cost such that expensive tests are only applied when strictly necessary. As soon as the simulator has determined that a given packet reception has already been damaged beyond the capability for error detection and correction processes to recover, there is no benefit in applying further checks. This effectively implements a *lazy evaluation* approach explicitly in the simulation model, rather than implicitly through an implementation language which supports lazy evaluation.

Phase one considers random environmental noise not influenced by network activity. Phases two and three apply a clipping strategy to determine nodes posing an interaction risk due to proximity. Phase two considers nearby nodes which are very likely to cause reception corruption, obtaining a fast first approximation. Phase three obtains a better approximation using a more expensive calculation. This multi-phase approach, outlined in Algorithm 1, addresses the requirements of Objective 1.

4.1.1 Three-phase radio algorithm

Consider a sensornet composed of similar nodes distributed in a plane. Assume some node, N, is currently receiving a message being transmitted in the wireless medium by some other node, T. Background *1/f noise* is present at all times but can be rejected at N provided it is sufficiently weak. Inevitably, however, bursts of noise above the rejection threshold will be observed at a predictable rate but at unpredictable times [9]. Lines 4 to 8 of Algorithm 1 model this effect, phase 1 of the algorithm.

Within a circle of radius r, the typical communication range of the node hardware, exist other nodes with which N can reliably detect, receive and send messages. Nodes enclosed by r can reliably communicate with N through the wireless medium, or can refrain from transmitting if the local wireless medium is determined busy by CSMA.

However, it is feasible that N could lie between two other nodes O and P, such that $||\overrightarrow{NO}|| \leq r$ and $||\overrightarrow{NP}|| \leq r$ but $||\overrightarrow{OP}|| > r$. If N is receiving from O at some time, P cannot detect this and may start to broadcast simultaneously. This broadcast by P is very likely to corrupt the unrelated reception at N, as $||\overrightarrow{NP}||$ is within broadcast range.

This *hidden terminal* problem [4] is addressed by lines 9 to 15 of Algorithm 1 which describe this second phase.

This offers a good first approximation and has been successfully employed in wireless ad-hoc network research [3] but may not, in isolation, capture all relevant behaviour. It is known that nodes can occasionally exert influence at a surprisingly long distance [5], as signals are merely attenuated with distance in the wireless medium rather than abruptly disappearing. On the other hand, if two nodes are sufficiently distant the probability of their interaction is vanishingly small, and the impact on network behaviour is negligible.

Algorithm 1 : Three-phase radio algorithm

```
1:  for each node, $n$ do
2:    determine if $n$ is actively receiving data
3:    if n is currently receiving then
4:      determine if environmental noise corrupts reception at $n$
5:      if reception corrupted at $n$ by noise then
6:        reception at $n$ fails
7:        jump back to line 1 for next $n$
8:      end if
9:      find set of nodes R with $distance < r$
10:     for each node, $m$, in R do
11:       if $m$ is transmitting then
12:         reception at $n$ fails
13:         jump back to line 1 for next $n$
14:       end if
15:     end for
16:     find set of nodes S with $r < distance < s$
17:     for each node, $m$, in S do
18:       if $m$ is transmitting then
19:         apply expensive radio model to find effective received power, $p$, from $m$ at $n$
20:         if $p > sensitivity(n)$ then
21:           determine if error detection + correction at $n$ can nullify influence of $p$
22:           if error correction fails at $n$ then
23:             reception at $n$ fails
24:             jump back to line 1 for next $n$
25:           end if
26:         end if
27:       end if
28:     end for
29:   end if
30: end for
```

We address this by considering nodes falling within an annulus defined by radii r and s, where $r < s$ and s is beyond the communication range of nodes. Consider a node Q falling within this annulus. Reliable pairwise communication between N and Q is impossible as $||\overrightarrow{NQ}|| > r$. However, as $||\overrightarrow{NQ}|| \leq s$, N and Q are sufficiently close that some interaction between N and Q is possible due to random fluctuations in the wireless medium, transmission gain of Q, and reception gain of N.

In other words, should Q broadcast at full power the effective power received at N is below the sensitivity threshold but at times might interfere with an unrelated signal being received at N. For nodes like Q we must consider the distribution function for effective received power at N; sometimes the received power will be above the threshold and at other times below. It is for nodes like Q that the higher cost of sophisticated but expensive radio interference models can be justified. Lines 16 to 28 of Algorithm 1 describe this third phase.

Finally, consider a node X located such that $||\overrightarrow{NX}|| > s$. X is sufficiently distant from N that, should X transmit at full power, the effective received power at N is below the sensitivity threshold even when random fluctuations are taken into account. Transmissions from X cannot be distinguished from background noise at N, and hence need not be considered at all in Algorithm 1. In large networks there may be many such distant nodes, and hence a significant saving can be obtained by this optimisation.

In non-planar networks radii r and s define spheres rather than circles and annuli but the algorithm remains unchanged. For a given point isotropic source the enclosing surface defined by a given radius is a hollow sphere of zero thickness provided that transmission occurs in an empty void. An infinite number of such surfaces can be defined for the continuous range of possible attenuation, with zero radius representing zero attenuation and infinite radius representing full attenuation. If transmission does not occur within an empty void it is appropriate to instead interpret radii r and s as nonspherical surfaces of equivalent attenuation, with equivalent results. Surfaces of equivalent attenuation for complex radio environments may be defined by surveying deployment regions or by specialised propagation models, but only spherical surfaces are considered within the scope of this paper.

4.2 Experimental details

Three simulated networks were defined. Simulated nodes were based on the MICA2 sensornet mote [2] with an IEEE 802.11 MAC layer and radio range of around 150m, although this detail is largely irrelevant as any similarly-equipped nodes will yield similar behaviour. Each network contained 500 simulated nodes. Preliminary experiments showed that this network size is sufficient to reveal features in parameter-response relationships not evident in smaller networks. Networks of more than 500 nodes could be substituted with equivalent results, but with an increased simulation cost overhead.

Each network was identical in all regards other than node spatial distribution. By averaging or otherwise compositing results across these three test networks we ensure that the characteristic quirks of any given network do not exert undue influence. Node spatial distribution within a bounding volume was random and even, with constant spatial density measured in units of *nodes per cubic metre* (*node* m^{-3}). A density of 1.5×10^{-7} *node* m^{-3} was employed throughout the experiments.

In the simulated application each node serves as a packet source and packet sink, utilising the unicast paradigm throughout. Each node generates packets sporadically with a single randomly selected destination to model a general distributed and decentralised process control application. Simulated packets have length randomly selected in the interval $[128, 1024]$ bits, including header. With the MICA2 radio having a transmit speed of $38.4Kbs^{-1}$ [2] this gives per-packet transmit times in the interval $[3.33 \times 10^{-3}, 2.67 \times 10^{-2}]$ seconds.

When packet transmission begins the local wireless medium is occupied for some duration in this interval. Nodes implement CSMA such that, if attempting to broadcast a packet, an exponential backoff procedure is implemented should the nearby wireless medium be occupied. A waiting node will implement up to 8 sense-wait cycles, doubling the wait period on each iteration, before giving up and dropping the packet. Note that although this greatly reduces packet broadcast collisions it does not avoid the *hidden terminal* problem [4], which is faithfully recreated in the simulation environment.

4.3 Computing resources

Simulation of large networks is a computationally intensive task [8]. Exploring parameter landscapes sampled at many points to understand behavioural tradeoffs and compromises implies a greater computational cost, as the evaluation of each sampling point implies the execution of at least one large simulation. In practice, the cost is higher still; each parameter landscape sampling point may be evaluated several times in several simulated networks.

The obvious remedy to this problem is to distribute the work among multiple computation hosts. Division of work might be achieved by running multiple simulations in parallel, or distributing a given simulation between multiple hosts. In this paper we apply the first approach as each test case can be executed in isolation from the remainder of the test suite. This avoids the significant coordination overhead implicit in the second approach; as any sensornet node can interact with any other node, it is notoriously difficult [13] to divide the problem into smaller subproblems with low mutual dependency.

Ideally, the computation work associated with each simulation could be divided between an arbitrary number of processing hosts to exploit high performance multipurpose servers, low cost single-purpose resources such as hosts implemented using multiple FPGAs, and the unused capacity of end-user workstations. The high resource demands of the current generation of network simulation tools implies that this will be difficult unless this goal is considered throughout the software design, and it will be difficult to retrofit existing simulation tools [8]. The YASS simulator [11] used in these experiments was designed with this goal in mind but does not yet offer support for division of a simulation instance between multiple hosts.

Given finite resources and a potentially infinite search space, there is necessarily a tradeoff between that which we would like to evaluate by experiment, and that which we can realistically evaluate within reasonable bounded time. We implemented a principled search method [12] which sampled that parameter space at a finite set of points, then applied statistical model fitting techniques to interpolate between these sampled points. Our method does not require the individual simulation experiments to be conducted in any particular order. It is not necessary to wait for all planned experiments to complete before analysing data; it is possible to use the partial set of completed simulation instances to obtain preliminary results to assess whether it is worth continuing to completion. Of course, the more data points that are available for analysis, the greater the accuracy of results.

The experiments implemented for this paper required over 100 days of computation time, but were performed in a few days with resources that were already available. Subsets of the computation job set were packaged for execution and managed automatically by suitably prepared scripts. The allocation of computation job sets to computers could be managed automatically by tools such as *BOINC* [1] or *Sun Grid Engine* [6] to any desired level of granularity. Owing to the lack of interdependency between any given pair of simulation experiment instances, however, it was not necessary to employ these tools.

Simulation results from all hosts were combined into a single results set for analysis. The *YASS* simulator is implemented in Java; simulations can be executed on any platform capable of hosting a Java Virtual Machine without the need to recompile, and without consideration of problematic issues such as architecture endianness.

The set of simulation experiments was divided among a heterogenous set of computation hosts. The majority of these experiments were assigned to a set of high performance hosts specifically intended for heavy workloads, but other hosts were utilised including normal end-user workstations and laptops. Hosts employed the Linux 2.6.26.5 and 2.6.26.5-x86_64 kernels, OpenSolaris 2008.05 and Microsoft Windows XP.

5 Offline static protocol tuning

In this section we consider the various dissimilar types of relationship which exist between network protocol configuration and network performance metrics.

Protocol configurations were evaluated in which only a single parameter, the static gossip rebroadcast probability p, was changed. 20 values of the parameter p were evaluated, distributed evenly in the interval $[0, 1]$. Graphs were plotted of p against observed metric values. Each simulation continued for 600 simulated seconds to allow network behaviour to settle into stable patterns. In packet-centric metrics, where multiple copies arrive at the destination we consider only the first.

Owing to paucity of space it has proved necessary to consolidate multiple related plots within figures in section

5. Where measured response values have incompatible units or scales, the observations have been normalised for each observation type such that the observed value range $[min, max]$ is mapped onto the range $[0, 1]$.

5.1 Local versus global traffic effects

Figure 1 traces *A* and *D* show that the probability of successful point-to-point transmission, and to a lesser extent reception, declines with increasing p. As network utilisation increases, so does congestion and the possibility of overlapping transmissions interfering. The former causes nodes waiting for the wireless medium to become clear for transmission to preempt exponential backoff when another transmission of potential interest must be received (trace *C*), in some cases timing out as the backoff procedure is exhausted (trace *B*). The latter causes nodes currently receiving packets to experience data corruption where multiple signals of sufficient strength interfere and interact beyond the recovery capacity of error detection and correction mechanisms.

As network activity increases with p we find that all individual network actions become less reliable, an effect which must be countered by protocols which can recover from failed individual actions. Flooding and gossiping can achieve this goal, albeit somewhat crudely by simply repeating many redundant copies of each attempted action.

In figure 2 trace *C* we observe the anticipated bimodal behaviour in which either most nodes receive a packet, or very few do, along multi-hop delivery paths. Increasing p increases delivery probability, with a sharp transition at the critical probability p_c [7]. Increasing network size brings a sharper transition. For $p > 0.6$ the probability of a packet being delivered is steady at around 92%.

In contrast, traces *A* and *B* indicate that individual node-to-node pairwise message exchanges become less reliable as p increases as a result of increased network utilisation, and hence increased congestion and interfering transmissions. Single-hop communications become less reliable, and yet the multi-hop communications composed of single-hop communications become more reliable.

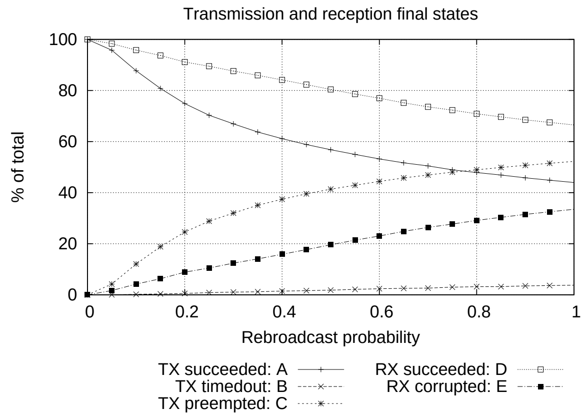

Figure 1. Transmit/receive final states

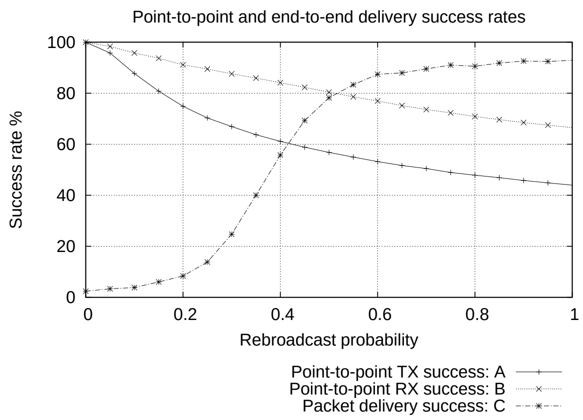

Figure 2. Delivery success rates

This would seem paradoxical as the success probability of any given multi-hop path is the product of all single-hop component probabilities. However, as p increases the number of potential delivery paths explored by packet copies increases rapidly. The success probability of each potential path decreases, but the rapid growth of path count greatly outweighs this effect. Sensornet designers must consider the expected distribution of path lengths in determining suitable p. Where long multi-hop paths dominate a high p is beneficial, but where single-hop or short paths dominate then a lower p may work better.

5.2 Route optimality and coverage

Figure 3 trace *A* shows that as p increases from 0 the average distance d_a between randomly selected source and destination nodes for packets steadily increases than levels, levelling out around p_c where $p_c \approx 0.4$ for this network. For $p < p_c$ node pairs separated by distance $d > d_a$ are very unlikely to successfully exchange messages due to packet propagation dying out early. Network applications requiring packets to travel long distances must select a suitably high p; applications in which packet delivery paths are always physically short (e.g. hierarchical aggregation) may cope with lower p.

Figure 3 trace *B* shows that the physical path followed by packets from source to destination is often far from the ideal straight line. In the degenerate case where $p = 0$ each path is perfectly straight as each consists of exactly one node-to-node hop. As p increases to p_c it becomes possible for node pairs located more distantly to exchange packets, though the delivery paths become decreasingly straight (and decreasingly efficient) as few nodes rebroadcast, and those that do so need not be located along the optimal straight path. However, as p increases beyond p_c it is more likely that a node choosing to rebroadcast will lie on, or near, the straight path. As packets encounter delays at each hop, it is more likely that shorter, straighter routes with fewer hops will induce less delay and hence deliver packets earlier than more torturous routes. This effect is reflected in trace *C*.

Figure 4 shows the three-way relationship between p on

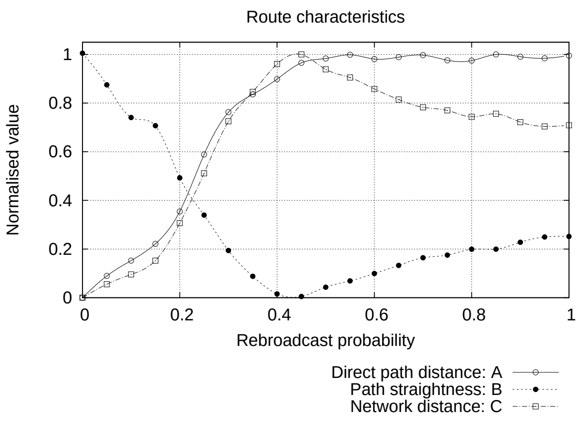

Figure 3. Delivery route characteristics

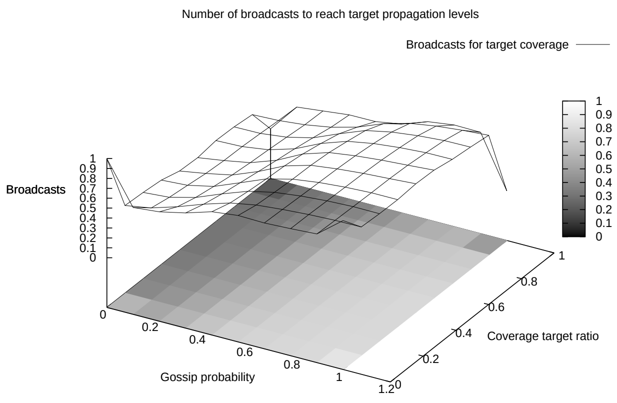

Figure 4. Coverage versus broadcasts

the *x*-axis, the proportion *q* of network exposed to a packet on the *y*-axis, and the total network-wide count *r* of broadcasts of this packet on the *z*-axis, normalised for packets successfully delivered only. *q* is equivalent to the probability that any given packet is delivered, and *r* is a heuristic measure of energy expended in the delivery attempt. Planar slices through the surface parallel to the *yz*-plane give the relationship between delivery probability and energy consumption for given *p*.

The surface can be approximated as two plateau parallel to the *xy*-plane, divided by a curve of shape similar to trace *C* in figure 2. This configuration illustrates that successful delivery attempts tend to induce a similar high number of broadcasts, and unsuccessful delivery attempts tend to induce a similar low number of broadcasts. If the surface had been closer to an inclined plane this would have indicated that sensornet designers could more finely tune the tradeoff between delivery success probability and energy consumption to suit application requirements.

5.3 Timing and latency effects

Figure 5 traces *A* and *B* indicated the speed at which packets traverse the network. Trace *A* shows the virtual packet speed *v*, as would be demonstrated by a packet travelling on a perfectly straight path between source and destination. This measure is useful in predicting time-bounded reachability for real-time applications. Packets with source-destination distance *d* and deadline *t* are likely (though not guaranteed) to arrive on time if $\frac{d}{t} \geq v$.

Figure 5 trace *A* shows that, as *p* increases, *s* declines rapidly due to network congestion until $p \approx p_c$ at which point the general increase in path straightness (see figure 3 trace *B*) allows *s* to increase. As *p* increases from 0 to p_c successful delivery becomes more likely but slower; for $p > p_c$ both speed and success probability increase in tandem. This is despite the actual physical speed decreasing with increasing *p* in figure 5 trace *B* for $p > p_c$. As *p* increases delivery routes become straighter (see figure 3 trace *B*), and for distance *d* contain proportionately fewer nodes.

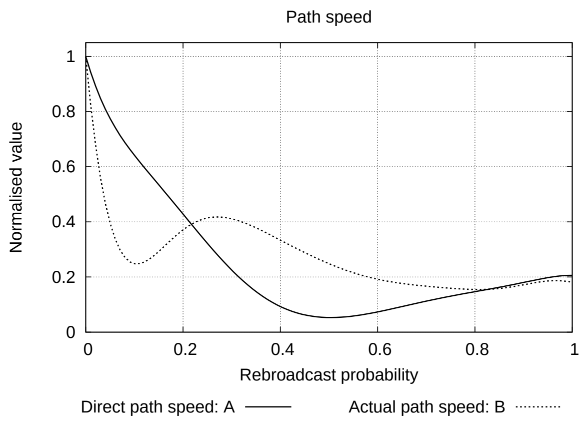

Figure 5. Path speed

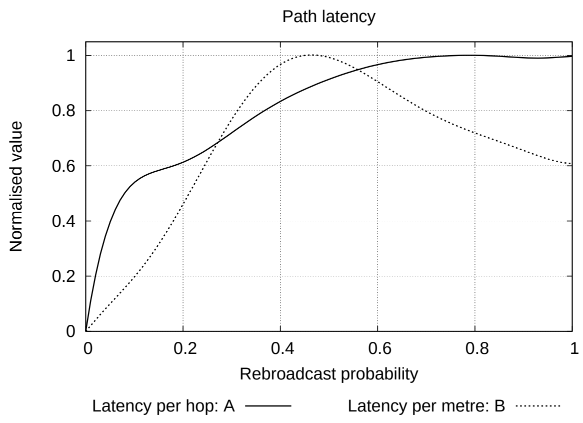

Figure 6. Path latency

However, for $p > pc$ the delay at each node does not increase appreciably, as illustrated in figure 6 trace *A*; nodes have limited buffer space and packets have limited lifespans, so packet queues cannot grow without bound. As figure 6 trace *B* illustrates, if the virtual packet speed *v* increases but routes become straighter as *p* increases where $p > p_c$, then the time taken to travel some constant proportion of the route becomes smaller. Consequently, network designers may wish to decrease buffer sizes and packet lifetimes where newer packets are of greater value than older packets.

Figure 7 shows the three-way relationship between p on the x-axis, the proportion q of network exposed to a packet on the y-axis, and the average latency for delivered packets on the z-axis. Planar slices through the surface parallel to the yz-plane give the relationship between delivery probability and likely delivery latency; expected latency grows almost linearly in delivery probability, with some slope s_p. In other words, this relationship gives the probability that a packet will be delivered within a given time, assuming that it will be delivered at all.

This reinforces the interpretation in sections 5.1 and 5.2. The more network activity in response to a delivery attempt the greater the chance of success, but the lesser the delivery speed. Note that s_p is not constant; s_p increases with p from 0 reaching a maximum at p_c, decreasing once again as p approaches 1. Network designers requiring minimal variation in delivery latency might select p distant from p_c toward the extremes.

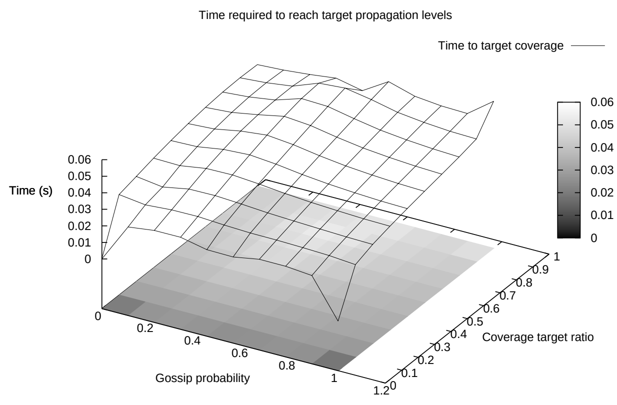

Figure 7. Coverage versus time

6 Conclusions

In section 3 a set of desired research objectives were defined, against which we now state our findings.

Obj 1: *Build an understanding of the compromises and tradeoffs inherent in attempts to tune a networking protocol against multiple competing objectives*

It has been shown that optimising even a single tunable parameter is a delicate balancing act with the results being highly dependent on the specific network and application. As such this form of investigation is a valid and valuable use of sensornet designers' time.

Obj 2: *Derive requirements for a new protocol from measured weaknesses in existing protocols.*

Several undesirable behaviours relating to undelivered traffic were observed and measured. A number of areas for improvement were identified. In particular, the requirement for online adaptation is particularly acute at the beginning and end of delivery paths.

Obj 3: *Show how large-scale computing capabilities can be used to satisfy the above in an efficient and effective manner.*

Efficient simulation and experimental methods were harnessed in achieving objectives 1 and 2. Future work will reuse the infrastructure and methods in systematic design and improvement of protocols for large-scale sensornets. Plans exist to adapt the *YASS* simulator to harness the cheap computation power of commodity GPU devices, and to adapt the compute job scheduling procedure to allocate tasks to shared cloud computing resources.

References

[1] D. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of the IEEE International Workshop on Grid Computing*, November 2004.

[2] Crossbow Technology Inc. MICA2 datasheet, part number 6020-0042-08 Rev A. http://www.xbow.com/Products/Product_pdf_files/ Wireless_pdf/MICA2_Datasheet.pdf. Accessed 09/01/2008.

[3] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *Proceedings of the IEEE International Conference on Communications*, pages 376–380, 1997.

[4] C. Fullmer and J. Garcia-Luna-Aceves. Solutions to hidden terminal problems in wireless networks. In *Proceedings of ACM SIGCOMM'97*, pages 39–49, 1997.

[5] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report CSD-TR 02-0013, UCLA, Feb 2002.

[6] W. Gentzsch. Sun Grid Engine: towards creating a compute power grid. pages 35–36, 2001.

[7] Z. Haas, J. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE Transactions on Networking*, 14(3):479–491, 2006.

[8] T. Henderson, S. Roy, S. Floyd, and G. Riley. ns-3 project goals. In *Proceedings of the International Workshop on NS-2: the IP network simulator*, pages 13–20, 2006.

[9] K. Kundert. Introduction to RF simulation and its application. *IEEE Journal of Solid-State Circuits*, 34(9):1298–1319, September 1999.

[10] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceeedings of the ACM Conference on Mobile Computing and Networking*, pages 151–162, 1999.

[11] J. Tate and I. Bate. YASS: A scaleable sensornet simulator for large scale experimentation. In *Proceedings of Communicating Process Architectures*, pages 411–430, September 2008.

[12] J. Tate, I. Bate, and S. Poulding. Tuning protocols to improve the energy efficiency of sensornets. In *Proceedings of the Fourth IET UK Embedded Forum*, Sep 2008.

[13] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *PProceedings of the Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.