Communicating Process Architectures 2008
P.H. Welch et al. (Eds.)
IOS Press, 2008

YASS: a Scaleable Sensornet Simulator for Large Scale Experimentation

Jonathan TATE and Iain BATE
Department of Computer Science, University of York
{jt, iain.bate}@cs.york.ac.uk

Abstract. Sensornets have been proposed consisting of thousands or tens of thousands of nodes. Economic and logistical considerations imply predeployment evaluation must take place through simulation rather than field trials. However, most current simulators are inadequate for networks with more than a few hundred nodes. In this paper we demonstrate some properties of sensornet application and protocols that only emerge when considered at scale, and cannot be effectively addressed by representative small-scale simulation. We propose a novel multi-phase approach to radio propagation modelling which substantially reduces computational overhead without significant loss in accuracy.

Keywords. sensornets, networks, simulation, scalability.

Introduction

Wireless Sensor Networks, or *sensornets*, are an emerging discipline of embedded system and network design. Large numbers of minimally resourced nodes are equipped with sensors to monitor their physical environment. Nodes cooperate to manage ad-hoc wireless networks, within which distributed applications distill voluminous raw data about sensed physical phenomena into meaningful information with utility to end users. Real-time interaction with the real world is not merely a factor to consider in sensornet design; it is the fundamental purpose of the sensornet.

Wireless sensor networks are currently at an interesting point in their evolution. Some real-world deployments have been implemented but at relatively small scale. These small trials have validated the concept as workable and useful. However, despite considerable interest, wireless sensor networks have not yet made the transition from the laboratory to commonplace real-world usage. What is holding back these real-world deployments?

One contributing factor is the lack of confidence that a given network design will function adequately in a given environmental scenario. Few experimental studies employ large numbers of sensornet nodes, and consequently there is relatively little experimental measurement of sensornet protocol scalability [12]. Until such time as there exists a large number of previous sensornet installations from which to draw experience, simulation offers a low-risk and low-cost environment in which to assess the viability of proposed solutions, and to improve the quality and relevance of any putative solutions. Answers offered by simulation can be only as good as the model from which simulation results derive. Nevertheless, simulation is a valuable first stage in weeding out unworkable solutions, and in identifying which options are worthy of further study.

Good practice generally suggests the reuse of existing tools wherever possible to avoid wasteful duplication of effort. However, where existing tools are inappropriate, unusable, or simply unavailable, it is often necessary to develop new tools to address pressing needs. Most existing simulators focus on low-level simulation of small networks [34]. However, this is infeasible where many simulation instances are required to rigorously explore parameter landscapes, for example in protocol evolution or multi-objective optimisation experiments.

Poor performance also precludes the simulation of large sensornets. This latter point is of key importance where interesting behaviours are evident only in large sensornets, and in assessing the capability of candidate protocols at scale. Most existing simulators can handle at most a few hundred nodes before runtime becomes intolerable, as poor time complexity implies lengthy wall time for network problem instances of moderate but realistic scale [16]. Real sensornets may contain thousands of nodes, and proposals exist for sensornets containing millions of nodes. It is clear that simulation scalability cannot be ignored in this context.

The decision was therefore taken to develop *yass*, "Yet Another Sensornet Simulator". *yass* is a high-level sensornet simulator which prioritises speed over accuracy to render feasible the exploration of large sets of scenarios in acceptable time. Our current work uses *yass* to model sensornets for a wide range of purposes including the tuning of existing protocols, the design of new protocols, and multi-objective optimisation of sensornet applications.

The structure of the remainder of this document is as follows. Section 1 discusses related work on sensornet simulation. Section 2 enumerates the research objectives addressed by this paper. Section 3 describes the *yass* simulator and the novel optimisations it implements. Section 4 considers the performance profile of these optimisations, and section 5 assesses the impact on accuracy. Section 6 describes experiments to validate *yass*. Finally, section 8 presents conclusions against the research objectives.

1. Related Work

Sensornet design and evaluation frequently requires *simulation* and *emulation*; large-scale *testbeds* or *field trials* are infeasible and costly [4]. Formal analysis of sensornets may [18] or may not [30] be feasible; regardless of feasibility, it is rarely attempted.

Investigators must determine acceptable accuracy-scalability tradeoffs [31]; simulation-derived results are meaningless if simulated behaviour does not sufficiently match real behaviour [29] and are particularly sensitive to timing discrepancies [23]. Wireless communication models are usually the component with highest computational cost [27] but represent the greatest source of inaccuracy [20].

Discrete event simulators are well suited to computer network simulation [3]. *Simulation models* [4] are constructed, similar to those used in model checking [33], and executed in *simulation engines* [13]. Incorporating real application code, execution environments, hardware, network connections, or other real entities, into *simulation models* yields *emulation models* [13], improving accuracy but harming scalability [35]. Real and simulated entities interact directly in *hybrid simulations* [24].

Numerous sensornet-relevant simulators and emulators exist. Unfortunately, no current examples offer total accuracy or reach desired scalability. *TOSSIM* offers cycle-accurate low-level emulation of Berkeley motes running TinyOS but very simplistic network modelling [25]. More detailed modelling might be required where observable phenomena are very sensitive to minor variation in network conditions [20].

ns-2, the predominant network simulator in sensornet research [25,5], uses highly-detailed network models [5] but is single-threaded [16] and scales only to around 100 simulated nodes [27]. *ns-2* was not originally designed for wireless network simulation, support for which must be added through extensions [5]. Much of the popularity of *ns-2* can be attributed to the breadth of reusable libraries and protocol models developed by numerous researchers. However, the complexity stemming from this lack of focus and the underlying architecture can make working with or extending *ns-2* time-consuming and difficult [16].

J-Sim offers similar facilities to *ns-2*, also providing a component model which can be scripted and customised through the *Tcl* language. *J-Sim* is less widely used than *ns-2* but better suited to sensornet simulation as it was designed for this purpose, and has more detailed support for modelling physical environments and network-environment interaction. It also offers limited multithreading support within a single processing host [2].

The high computation cost of network simulation might be addressed by task parallelisation. Interentity communication across parallel simulation hosts [32] may negate some benefit of additional processors by Amdahl's Law [1,34]. However, this is not to say that parallel processing has no role to play, merely that care must be taken to ensure that simulator designs work harmoniously with the characteristics of a given target parallel processing environment.

Simulating sensornet-scale networks of millions of nodes requires entity concatenation [3] and layer concatenation [5] to reduce memory footprint [6], further sacrificing simulation accuracy. *GloMoSim* exploits parallel execution by multithreaded simulation, scaling to 10000 simulated nodes across 10 processors [38]. However, to achieve this scale *GloMoSim* consolidates many independent entities of the simulated system into single compound entities, necessarily sacrificing low-level accuracy for performance.

2. Research Objectives

Given a set of typical and broadly comparable sensornet configurations, and a typical sensornet application implementing a tuneable networking protocol, we define the following objectives that form the principal contributions of this paper:

Objective 1: *Identify techniques through which to improve upon the performance of existing sensornet simulators*
Objective 2: *Measure the extent to which these optimising techniques improve performance and enable larger-scale simulation*
Objective 3: *Improve the range of simulated network measures made available to the investigator*
Objective 4: *Validate optimised simulation to ensure that performance gains do not sacrifice solution quality*

3. Yet Another Sensornet Simulator

yass (Yet Another Sensornet Simulator) is distributed under the GNU General Public License. Source code and API documentation can be downloaded from the project web pages: `http://www.cs.york.ac.uk/rts/yass/`

3.1. Motivation and Concept

Simulation is an important and accepted tool for network researchers, offering reduced cost, time and risk in comparison to real-world experiments. They represent a sensible first step in development, and are of particular importance where real-world experiments are infeasible.

It could be argued that researchers would be well advised to reuse one of the numerous extant network simulators in their investigative work. However, common practice in the network research community does not follow this strategy. One study of 287 peer-to-peer network research papers [26] finds that approximately 49% obtain experimental results through simulation. Of these 50% do not mention which simulator was used, and 30% use a custom simulator for which there has usually been no attempt to perform empirical validation. Although some cases may be explained by "*Not Invented Here syndrome*", these figures suggest that experimenters are often forced to develop their own tools where no existing tool is adequate for the task in hand.

As the name suggests, *yass* is simulation software for evaluation of sensornets and sensornet applications *in silico*. The main principle underpinning the design of *yass* is that significant performance gains are possible without sacrificing accuracy, simply by avoiding unnecessary work. If the results of a calculation will never be used, then that calculation should not be performed. *yass* addresses two aspects of simulation in which this approach yields useful performance gains; the production of statistical measures, and radio propagation modelling.

3.2. Statistical Measures and Event Traces

Statistics offer a powerful tool with which to understand the behaviour of networks and networking protocols at an appropriate level, without becoming swamped with the great quantities of unnecessary detail pertaining to the mechanisms underpinning the higher-level behaviour. Unfortunately, many existing simulators are weak at revealing statistical information required by investigators [26] such as delivery latency or throughput.

This is particularly true where investigators do not know in advance which statistical measures will provide the answers they require, and hence these required measures are not taken; the experiments must be repeated. Conversely, valuable resources are often squandered in the production of a myriad of metrics in which the investigator has no interest and are not required by the simulator itself.

yass takes a different approach. As the simulation progresses noteworthy network events are recorded in an *event trace* [17]. This event trace records the behaviour of the simulated network elements rather than that of the simulator, and is backed by a relational DBMS to enable large datasets to be handled efficiently and to allow the usage of common data analysis tools. Each event is timestamped by simulation time rather than wall time to allow events to be logged out of their natural ordering, for example when two or more event-generating processes are scheduled unpredictably.

Standard trace operations can be applied during trace analysis. The trace sequence consists of the *interleaving* of all subsequences deriving from individual nodes, such that *restriction* might be applied to filter those events relating to network elements of interest to an investigator, or specific traffic flows. If the desired behaviour of a given network protocol is well-defined it is possible to determine whether a given trace records correct or incorrect network behaviour by determing whether that trace is in the set of all possible correct traces. This offers a powerful non-statistical tool for network analysis by simulation.

Statistical measures are not taken online during simulation but are instead obtained *post hoc* from the event trace offline, allowing data mining across multiple scenarios and extraction of metrics not originally considered by investigators. We demonstrate this principle in section 6 by running sets of simulations, then upon completion analysing multiple recorded simulation traces to derive metrics. These observations address Objective 3's requirements.

The tradeoff for these performance gains is that metrics are not available during simulation. Should investigators require online metric availability for online scenario adaptation it is possible to calculate these from a partial trace at the point of usage by the same method as applied to the complete trace at the end of the simulation.

3.3. Radio Propagation Model

Scalability is a weakness of many existing simulators [4]. Proposed sensornets may involve thousands or tens of thousands of nodes, but most existing simulators struggle with more than a few hundred simulated nodes [16]. 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 which are 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 a 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.

3.3.1. 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 [22]. Lines 1 to 1 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 [11] is addressed by lines 1 to 1 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 [9] but may not in isolation capture all relevant behaviour. It is known that nodes can occasionally exert influence at a surprisingly long distance [12], 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.

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

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
```

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 1 to 1 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.

3.3.2. Cost Analysis

From the definition of Algorithm 1 it is evident that the number of computational steps is dependent on the size of various sets of nodes defined for each message-receiving node. Assume all network nodes are found in the set N, and that all nodes are alike. For each node $x \in N$, the following sets are defined:

- A_x: all other nodes within radius r with which reliable communication is possible
- B_x: all other nodes within radius s with which interaction is possible, where $A_x \subseteq B_x$
- $C_x = B_x \setminus A_x$: all other nodes with which interaction is possible but reliable communication is impossible
- $D_x = N \setminus B_x$: all other nodes with which no interaction is possible

A_x, B_x and C_x can either be defined implicitly by deciding set membership at the point of use, or precalculated and cached for improved efficiency under a *staged simulation* approach [36]. Note that the cost of maintaining these cached sets is non-trivial for networks containing mobile nodes, being $O(n^2)$ in total node count to rebuild. Membership of D_x need not be defined explicitly as members cannot interact with x and play no role in Algorithm 1.

The cardinalities of A_x, B_x and C_x are independent of total network size, and are dependent only on spatial density in the region around x. Clipping spheres of radius r and s centred on a given node, x, enclose the volumes v_r and v_s respectively. Multiplying these volumes by the local node density (measured in *node* m^{-3}) yields the number of nodes m_{xr} and m_{xs} falling within clipping spheres of radii r and s respectively.

From these node counts we see that $|A_x| = m_{xr}$, $|B_x| = m_{xs}$, and $|C_x| = |B_x| - |A_x|$. Each is $O(d)$ in local node density d and $O(1)$ in overall network size. Assuming uniform spatial node distribution and homogenous nodes throughout the network we can ignore the identity of node x, yielding $|A|$, $|B|$ and $|C|$. These set cardinalities, and the corresponding computational cost per individual node, are independent of total network size, a highly desirable property for scalable simulation of large networks.

3.3.3. Complexity Analysis

Consider the total cost for all nodes passively monitoring the wireless medium, waiting to begin receiving data. For a given node $x \in N$ we need check only members of A_x for starting transmissions as other nodes cannot establish communications with x. Assuming each pairwise check completes in constant time, uniform $|A|$ for all nodes, and i idly listening nodes, the total cost is $i|A|$. If i is $O(n)$ in network size and $|A|$ is $O(d)$ in spatial density d, total cost is $O(nd)$ and hence linear in total network size $|N|$. A similar cost is observed for nodes establishing that the local wireless medium is free prior to packet transmission as this entails similar local passive monitoring.

Consider the total cost for all nodes actively transmitting to the wireless medium. Trivially this is zero, and hence $O(1)$ in network size $|N|$, because after transmission of the packet body begins the transmitting node makes no further checks on the local wireless medium.

Now consider the total cost for all nodes actively receiving from the wireless medium. As described in Algorithm 1 in the worst case this requires for each node $x \in N$ to perform pairwise checks between x and all potentially interfering nodes $y \in B_x$. In the worst case the radii r and s are such that for each node $x \in N$ a significant proportion of N is represented in A_x and B_x, and every ongoing message reception must be tested against every ongoing message transmission.

Assume at some time t that some proportion α_t of nodes are actively transmitting and some proportion β_t of nodes are actively receiving, with $\alpha_t, \beta_t \in [0,1]$ and $\alpha_t + \beta_t \leq 1$. The set of transmitters, F, has $|F| = \alpha_t|N|$, and the set of receivers, G, has $|G| = \beta_t|N|$. Under the optimised model the number of pairwise interaction tests required is no greater than $|F||G| = \alpha_t\beta_t|N|^2$, as opposed to the $|N|^2$ tests required under the unoptimised model. Optimised cost remains $O(n^2)$ in network size $|N|$ if total network activity is dependent on network size as described above. However, as $\alpha_t, \beta_t \in [0,1]$ optimised cost is generally lower, and never higher, than unoptimised cost.

As $\alpha_t, \beta_t \rightarrow 0$, network activity decreases and cost savings under optimisation increase. Total cost under the optimised model only reaches that of the unoptimised model when every node is actively participating in network activity, and every node potentially interacts with every other node. This is of particular relevance to the simulation of sensornets, in which network traffic is minimised in order that energy consumption resulting from network activity be minimised. Energy-efficient network applications accordingly require less real time for simulation to complete.

3.4. Parallelism

yass is a time-driven rather than event-driven simulator, in which any simulated entity can interact with any other and simulated time progresses in small increments. There is no requirement that consistency must be maintained during the calculations which progress from one consistent state to another, but each simulated time increment transforms the simulation model from one consistent state to another consistent state. A dependency exists from any given state to the preceding state from which it was derived, implicitly requiring that the sequence of simulated periods must be addressed sequentially. However, there is no requirement that calculation within a simulated period be conducted serially.

The radio model described in section 3.3 enables the most costly element of the simulation to be addressed using parallelism. Within each simulated period the three phases of the radio model must be executed serially. Within each phase the work can readily be divided between any number of processing units, and the results combined prior to commencing the next phase. This is feasible because each calculation considers only a single pairing of nodes which can be performed independently of all other pairings, and can be performed without large volumes of shared state. However, this implies the existence of a single *master* which delegates work to numerous *slaves*, regardless of whether this delegation uses multiple CPU cores, multiple clustered hosts, or any other similar architecture. The delegation of node pairings need only be performed once if the set of simulated nodes and the set of processing units does not change, though fresh data would need to be distributed for each simulation quantum.

Although there is no fundamental impediment to concurrent execution of sections of the simulation model, the current implementation of *yass* uses a purely sequential approach as the bottleneck is I/O throughput rather than CPU speed. Recording the event trace from which statistical measures and network metrics are derived (see section 3.2) can generate data at a rate orders of magnitude greater than typical hard disks can store it. It follows that no real performance gain would be observed by splitting the processing work between multiple processing units unless atypically high-performance storage resources are available. This is a consequence of prioritising offline processing over online processing in the simulator design.

In the absence of performance gain in a typical execution environment the decision was taken to favour a simple serial implementation of the radio algorithm. However, if I/O volume could be reduced by capturing fewer simulated events, or by storing only a representative sample of all simulated events, CPU capacity could become the primary bottleneck. Under this condition it would be highly desirable to re-implement the radio model in *yass* to exploit fine-grained parallelism as described above. In the first stages of experimental work the key challenge is often to establish which of the measurable values yield useful insight. Once this is known it is possible to reduce the volume of data captured to the bare minimum necessary to derive the required measures and discard the remainder, increasing the feasibility of obtaining useful performance improvements through exploitation of parallelism.

The current implementation of *yass* does exploit coarse-grained parallelism. One of the main use cases of the simulator is to efficiently explore parameter landscapes through large numbers of simulations, each test case evaluating some fixed point of the landscape. Several such test case simulations can be executed in parallel in separate threads as simulations do not interact, allowing simple test suite coordination with no interthread coordination overhead other than managing access to disks constituting the database backing storage.

yass has built-in support for scripting large numbers of simulation scenarios and scheduling *n* such simulations to be running at all times until the set of test cases is exhausted. Usually *n* would be defined to be equal to the number of CPU cores available on the host; setting it lower allows these cores to be devoted to other work, whereas setting it higher yields no further performance gains. Again, I/O throughput is the limiting factor on overall simulation throughput, but more efficient use of I/O is possible as other threads can continue to make progress while any given thread performs I/O activity.

3.5. Implementation Details

yass is a time-driven rather than event-driven simulator. The times at which simulated events occur are quantised to the boundaries of discrete periods during which the simulation model transitions from one consistent state to another. Shorter quanta yield greater accuracy but slower progress. The length of all such quanta need not be equal, such that a smaller simulation quantum could be applied for network periods of particular interest to the researcher or during which network behaviour is particularly sensitive to inaccuracy.

The simulator is written in the Java language and as such can be executed without recompilation on any host with a Java Virtual Machine capable of supporting at least version 5.0 of the Java language. The event trace data is managed by a DBMS accessed through JDBC. The Apache Derby DBMS is used by default to minimise end-user configuration, but any DBMS which supports at least SQL-92 and is accessible through JDBC should be compatible.

All simulated entities are represented as objects which are responsible for managing their own state. As a consequence there is very little global state to be managed. Nodes are composed of hardware modules, each of which can adopt a number of states to balance performance and energy efficiency. Hardware module state can be changed by other modules or by the application, either immediately or through a built-in scheduler. Applications are composed of a set of packet producers, a chain of packet consumers, and a packet dispatcher.

4. Evaluating Performance

A set of timing experiments was run to assess the performance of *yass*. Each experiment was repeated nine times on the same machine to minimise influence of outliers and to smooth the effects of the stochastic simulated application. The GNU *time* command was used to measure CPU time allocated by the scheduler, with timing data given to $\pm 1 \times 10^{-2} s$ precision. Each problem instance reused the same simulated network, to block effects of uncontrolled factors, for each of 21 distinct rebroadcast probabilities distributed evenly in the interval $[0, 1]$ running for a fixed number of simulated seconds. Packet source and destination node was randomly selected for each packet.

The first two experiments considered performance for simulated networks of size ranging between 50 and 1000 nodes, varying in 50 node increments, with constant spatial density of 4.0×10^{-8} $node\ m^{-3}$. Simulated nodes were based on the MICA2 mote [8] with an IEEE

802.11 MAC layer and radio range of around 150m, although this detail is irrelevant to simulation runtime. The first experiment was conducted with the optimisations of Algorithm 1 enabled, and the second with these optimisations disabled. The results are shown in figure 1.

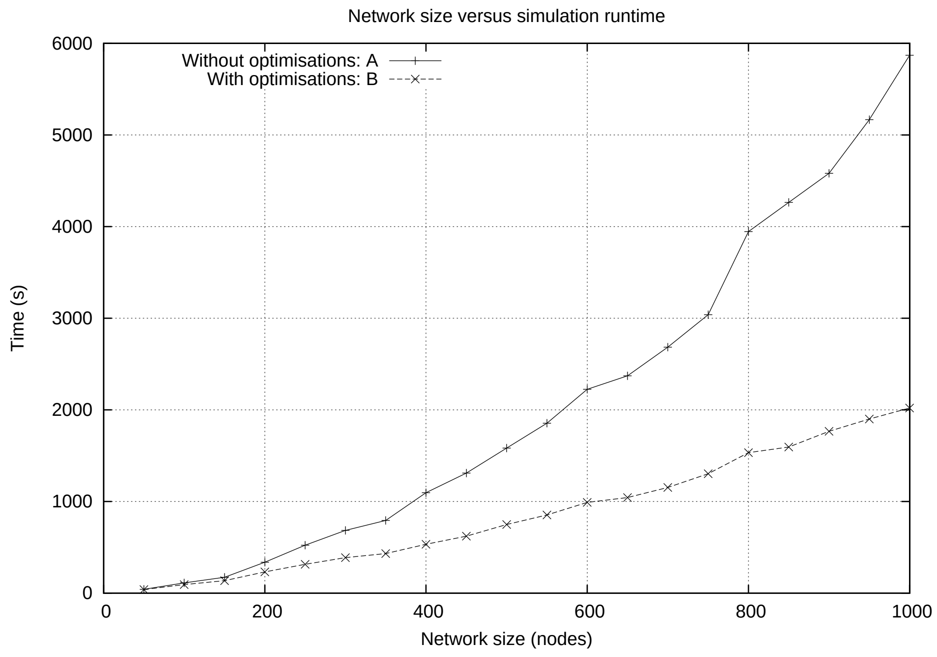

Figure 1. Network size versus simulation runtime

Trace *A* shows the runtime with simulation optimisations disabled, and trace *B* with simulation optimisations enabled. It can be seen that in all cases the runtimes illustrated in trace *B* are less than those of *A*, indicating that in all problem instances considered here the optimisations offer measurable time savings.

Trace *A* approximates a quadratic curve, which is as expected from the $O(n^2)$ behaviour predicted by section 3.3. The plotted curve is not perfectly smooth, an expected consequence of experimental noise induced by the simulated network application's stochastic behaviour.

Trace *B* also approximates a quadratic curve but with much shallower curvature, again as predicted by section 3.3. The curve of trace *B* is sufficiently shallow when considered against trace *A* that a linear approximation is reasonable within this range of network sizes.

Very large networks of the order of tens of thousands of nodes may require adoption of further abstractions in order for simulations to complete in acceptable time [38]. Most such abstractions would adversely affect simulation accuracy, so care must be taken to minimise this effect. Ideally these abstractions would follow the natural structure of the network.

For example, with clusters of multiple cooperating nodes, each cluster could be represented as a single entity when modelling intercluster traffic. However, experiments using simulations of the order of thousands of nodes are entirely feasible with commodity hardware without these additional abstractions and sources of inaccuracy.

The latter two experiments considered performance for networks of density ranging between 1.0×10^{-8} and 1.2×10^{-7} $node\ m^{-3}$, with constant network size of 500 nodes. The third experiment was conducted with simulation optimisations enabled, and the fourth with optimisations disabled. Nodes were based on the MICA2 mote [8] with an IEEE 802.11 MAC layer and communication range of around 150m, although again this detail is irrelevant to runtime. Results are shown in figure 2.

Figure 2 shows two traces for simulations of the same problem instances. Trace *C* shows runtime with simulation optimisations disabled, and trace *D* shows runtime with simulation optimisations enabled. It can be seen that in all cases the runtimes illustrated in trace *D* are

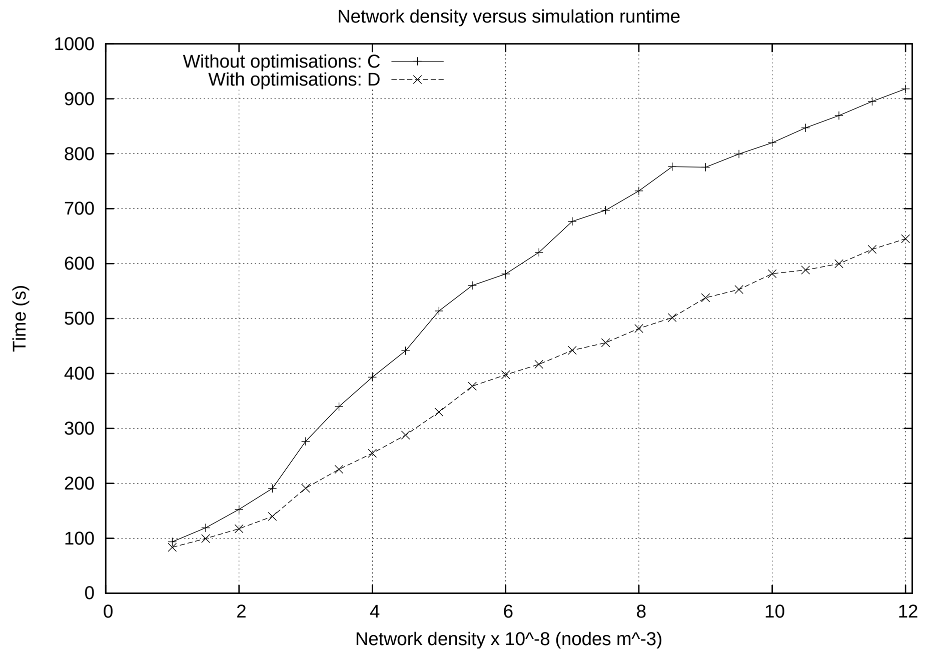

Figure 2. Network density versus simulation runtime

less than those of C, indicating that in problem instances considered here the optimisations offer measurable time savings.

Simulation runtime increases in network density, but traces C and D show runtime asymptotically approaching maximum values as density increases. Potential communication partner count per node grows with increasing density, with commensurate growth in pairwise communication as packets are flooded to more neighbours.

However, as each message transmission occupies the wireless medium for non-zero time there is also growth in network congestion. Simulated packets have length randomly selected in the interval [128, 1024] bits, including header. With the MICA2 radio having transmit speed of $3.84 \times 10^4 bits^{-1}$ [8] 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.

The tendency for network traffic to increase as more pairwise message exchanges become possible is countered by the tendency for increased congestion to restrict pairwise message exchange, as illustrated in figure 3. As density increases the level of attempted network activity increases, but there is decreasing spare capacity in which to accommodate the additional potential traffic.

Small periods in which the wireless medium remains unused are an expected consequence of the simulated stochastic application, becoming sparser and smaller in the time domain as the network becomes busier. CSMA with exponential backoff defers packet transmission until the wireless medium is free, when it is implicitly claimed for the full transmission period. There is no local or global coordination of traffic beyond this mechanism.

As network utilisation increases there are decreasingly many unoccupied periods fitted between occupied periods because increasingly many nodes attempt to claim the wireless medium per unit volume. Unoccupied periods tend to become shorter as many simultaneously-running backoff procedures attempt to claim the medium.

Smaller packets occupy the wireless medium for a smaller period, accommodating more packets within a given duration. Packets are forbidden from becoming smaller than the minimal header size, an integral number of bits, and hence it is not possible for the network to fill all unoccupied timeslots with sufficiently small transmissions. Eventually the network becomes saturated with the wireless medium occupied in places at all usable times.

This pattern of behaviour in which each subsequent increase in network density yields a smaller increase in simulation runtime suggests that a reasonable approximation to the observed behaviour could be obtained by fitting a logarithmic or harmonic series to the data points over the range of densities considered in the experiment.

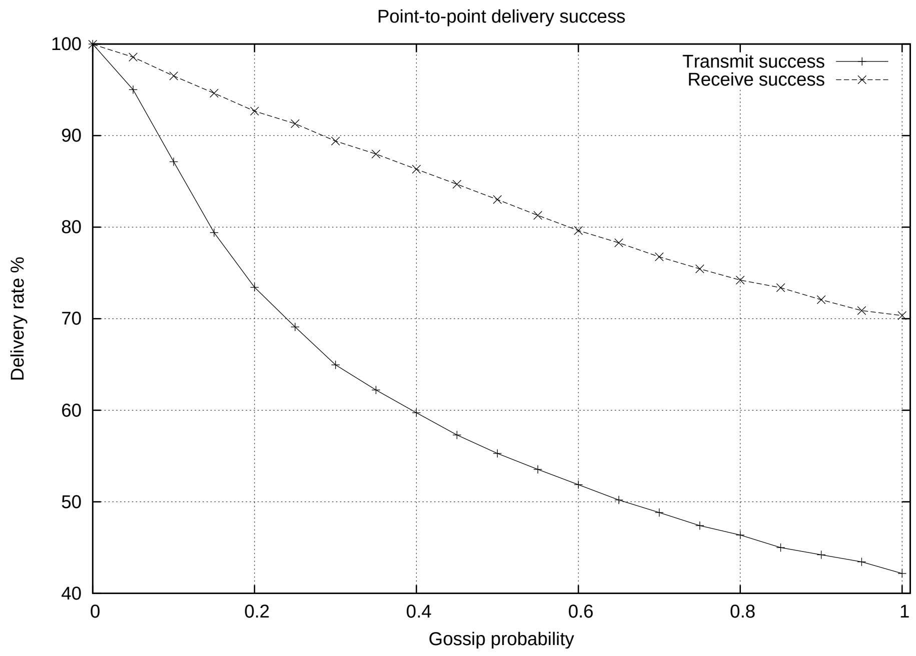

Figure 3. Point-to-point success vs rebroadcast probability

5. Evaluating Accuracy

A set of experiments were run to assess the influence of the proposed simulation optimisations on the accuracy of measured network metrics. Note that the scope of this section is restricted to comparison within the *yass* context, and does not address other simulation environments. General simulation validation is considered in section 6.

Each simulation reused the same 200-node network and simulated application, thus providing blocking of uncontrolled factors. Simulated nodes were based on the MICA2 mote [8] with an IEEE 802.11 MAC layer and radio range of around 150m. A single controlled factor, rebroadcast probability, was varied within the interval [0,1] with 10 intermediate steps. Each simulation was repeated 9 times and the arithmetic mean taken for each of three network performance metrics. The simulation set was repeated both with and without optimisations enabled, and the results plotted on the same axes for comparison.

Figure 4 illustrates the success rates for point-to-point packet transmission, point-to-point packet reception, and end-to-end packet delivery. For each metric there are a pair of traces; one obtained with, and one without, simulation optimisations enabled. It can be seen in each case that the paired traces are very similar across the range of gossip probability values considered in these experiments.

Experimental errors introduced by simulation optimisations are indicated by any divergence between the plots for a given metric. Trace pairs for the point-to-point metrics are so close as to be indistinguishable. The trace pair for the end-to-end metric shows slight divergence, resulting from compounding of multiple smaller point-to-point errors and borderline cases along delivery paths.

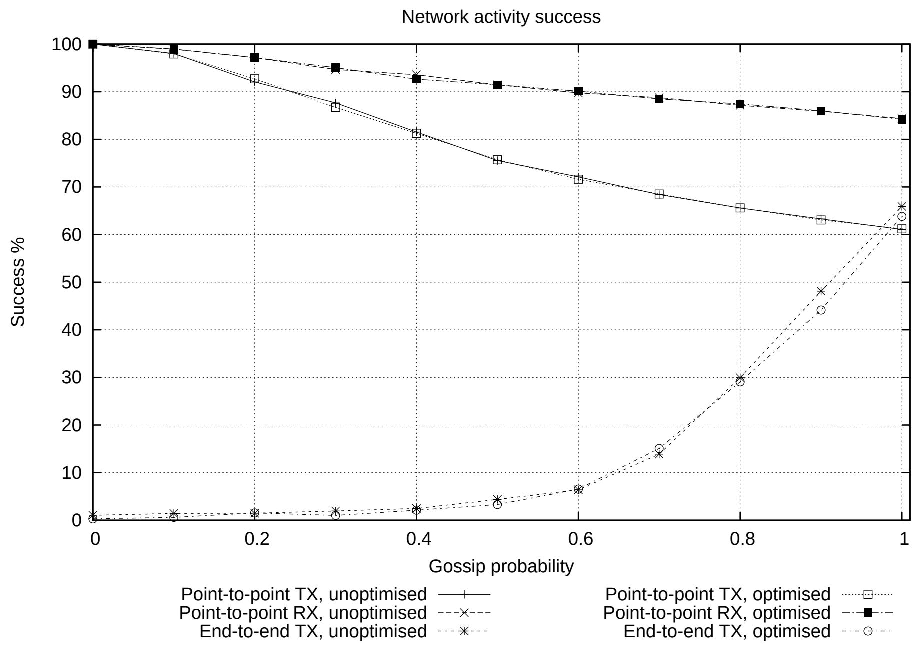

Figure 4. Network metrics with/without optimisation

With each point-to-point packet exchange the abstraction of the simulation model is a source of inaccuracy. A multi-hop delivery path has numerous such point-to-point exchanges, and hence compounds this inherent error for each hop. However, we see from Figure 4 that cumulative experimental error magnitude remains very small in comparison with the magnitude of the measured values.

More importantly, the overall trends and general shapes of response curves are retained when comparing the paired unoptimised and optimised traces. This is significant for experimenters seeking to identify emergent effects in large-scale network behaviour, or to establish the relative merit of two or more candidate protocol configurations. Under these experimental goals it is more important that relative trends and orderings across the parameters are demonstrated than to obtain accurate absolute values for any solution quality metric.

Where accuracy is prioritised, experimenters can employ a two-phase approach. *yass* can be used to quickly establish which input configuration gives the most desirable simulated behaviour. A more detailed but slower simulator can then be used to obtain accurate absolute response measures yielded by this optimal input configuration.

Combining the findings of sections 3.3, 4 and 5 it can be seen that the proposed optimisations allow simulations to be completed in significantly reduced time with only a minimal impact on solution accuracy. This addresses Objective 2 as defined in section 2.

6. Validating Simulation Results

We have demonstrated that the novel optimisations implemented by the *yass* simulator offer real and substantial performance improvements. However, these improvements are of little consequence if the accuracy of simulation results is unduly compromised. We examine the quality of solutions obtained under optimised simulation by validating *yass* against theoretical findings and experimental results published elsewhere, and examining observed trends against expected behaviour. We also consider the advantage conferred by simulating larger sensornets by comparing against results derived from smaller-scale simulations. The experiment designs and results described in this section address the requirements of Objective 4.

All experiments simulate networks employing the *GOSSIP1(p)* protocol defined by Haas et. el. [14] with gossip probability, p, taking 21 values with equidistant spacing in the interval $[0, 1]$. This simple protocol was selected as it has been thoroughly examined in the literature [14,37], which predicts a readily-identifiable and theoretically explicable sigmoid curve when gossip probability is plotted versus end-to-end delivery success rates. Presence (or absence) of this characteristic curve is deemed indicative of the accuracy (or inaccuracy) of the *yass* simulator. Additionally, flooding and gossiping protocols are essential components of many more sophisticated protocols. For example, AODV [28] and DSR [19] use simple flooding for route discovery, but may observe improved performance by substituting probabilistic gossiping [14].

Each value of p is evaluated within an otherwise-identical simulated problem instance for blocking of uncontrolled factors. Each simulation instance is repeated 10 times and the arithmetic mean of the analytical results plotted. Each plotted curve therefore represents the outcome of 210 independent simulation runs. Each simulation continues until 60 seconds of simulated time elapse, allowing performance metrics to settle on stable values.

Simulated nodes are modelled on the MICA2 mote [8] with IEEE 802.11 MAC layer and radio range of around 150m, and always transmit at maximum available power and maximum available rate. Two networks are considered of 100 nodes and 1000 nodes respectively, with the former being a subset of the latter. Spatial distribution of nodes within a bounding volume is random and even, with constant spatial density of 4.0×10^{-8} *node* m^{-3}.

The simulated application assumes a flat network hierarchy. Each node acts as a packet destination and periodic source, with any source-destination pair being equally likely in a unicast communication paradigm. Offset, period, and per-period production probability differ between nodes, but remain constant throughout and between simulations. Simulated packets have length randomly selected in the interval [128, 1024] bits, including header.

It is assumed that delivery of a given packet fails if it does not arrive at the intended destination node prior to the end of the simulation. Point-to-point packet exchange may fail due to corruption wherever a nearby node is transmitting with sufficient power, for example due to the *hidden terminal* problem [11]. Stochastic failure may also result from background noise with probability of 0.05 over the duration of each transmission.

6.1. Bimodal Delivery Behaviour

Previous results published in the literature predict bimodal behaviour for probabilistic rebroadcast algorithms such as *gossiping* [14,37]. As rebroadcast probability is increased it passes through a critical value. Below this critical value it is expected that packet distribution dies out quickly, and few nodes receive the packet. Above this critical value it is expected that most packets will reach most nodes.

Figure 5 demonstrates the end-to-end delivery ratio versus rebroadcast probability for a typical network and networked application employing the *gossiping* protocol. Graphing experimental results for the 1000-node network yields the sigmoid shape predicted in the literature [14,37], in which the steepest curve section contains the critical probability predicted by theoretical analysis. This demonstrates qualitatively that the relationship between rebroadcast probability and end-to-end delivery ratio is as expected. Quantitatively, rebroadcast probabilities in the interval [0.6, 0.8] are expected to deliver most packets to most nodes [14]. In figure 5 we see that for $p > 0.65$ delivery ratio remains within 90-92%, near-constant within experimental error, matching expected behaviour.

The 100-node network shows similar but less acutely defined behaviour, as discussed in the next section. Note also that end-to-end delivery success is higher for the 100-node network than the 1000-node network for small rebroadcast probabilities. This is because a larger proportion of the network falls within the range of a transmitting node, and hence a greater proportion of the network is directly reachable without any packet routing mechanism.

The proportion of the network covered by packet flooding grows exponentially in continuous time, represented by a number of discrete *rounds* of the flooding algorithm. In each round, every node which received a given packet in the previous round is a rebroadcast candidate. If each node has on average i nodes with which pairwise communication is possible, the initial packet broadcast at the source node will reach up to i rebroadcast candidates in the first round. In the second round, if each candidate is able to rebroadcast, then each can reach up to another i nodes, and so on for each subsequent round.

Assume packet transmission takes negligible time such that network congestion is negligible. Ignoring corrupted broadcasts, and assuming no other network activity, after r rounds of flooding the number of nodes rebroadcasting the packet in the nth round is i^r having received the packet in the $r-1$th round. If the rebroadcast probability is now changed from 1.0 to p we find that the average number of concurrent rebroadcasts is $(pi)^r$. This gives the level of network activity, and hence potential network congestion, due to this packet at this time. It is entirely possible that none of the neighbours of a given node will choose to rebroadcast as there is no local coordination.

At the completion of the rth round, all neighbours of each broadcasting node have been exposed to the packet, as have the outermost nodes reached in previous rounds. As the rth round completes, all neighbours of broadcasting nodes have been exposed to the packet in addition to those exposed in previous rounds. Flooding terminates when there are no further rebroadcast candidates. Disallowing multiple rebroadcasts of any given packet ensures flooding terminates in finite time, preventing packets endlessly circulating within the network.

Take the number of nodes covered by a packet as n_r at the end of the rth round, and $n_0 = 1$ as the source node has already got the packet before flooding begins. Assume that a node which has already been considered as a rebroadcast candidate for a given packet in a previous round is still able to receive that same packet again, even though it will not be considered as a rebroadcast candidate again. In particular, a node will be able to re-receive during the next flooding round after it has itself rebroadcast.

If each node has on average i neighbours, and during the rth round there are $(pi)^r$ concurrent rebroadcasts, up to $i(pi)^r$ nodes can receive a given packet during the rth round. Assume that q rounds complete before flooding terminates, q being $O(x)$ in network diameter x [21]. The total number of possible packet receptions completed at the end of the rth round is given by $\sum_{r=1}^{q} i(pi)^r$.

Each packet reception represents a delivery of a packet to *some* node. It is entirely feasible that a given node will receive a given packet multiple times, but will only be a rebroadcast candidate on the first reception. Provided that at least one of these successful packet receiptions occurs at the intended destination, the packet has been delivered.

Any given node is likely to be a potential communication partner for a number of other nodes. The communication range sphere for any given node is likely to enclose a number of other nodes, such that the neighbour-node sets corresponding to these spheres are unlikely to be identical in the majority of cases. Put another way, it is likely that any given node pairing can be joined by multiple paths through the network graph defined by communication range, and may therefore receive a given packet multiple times along different routes.

In an otherwise quiet network it is therefore very likely that most nodes will receive at least one copy of a given packet even if not all nodes rebroadcast this packet on reception. However, as the expanding ring of network coverage radiates from the source node, it is not guaranteed that all nodes will have received this packet. For example, nodes may have received a corrupted signal, or may have been occupied with unrelated wireless communications, or may simply have not been near enough to a broadcast.

As p approaches 1.0 the delivery success improvement achieved by increasing it further becomes smaller. This is because each node can receive a packet from multiple neighbours but only the first occurrence is necessary for delivery or rebroadcast candidacy. As p grows larger the proportion of nodes which have received at least one copy of a given packet increases, but most additional network activity induced by this growth simply increases repeated reception. The average number of receipts of a given successfully delivered packet per node grows from exactly 1 to just over 3 as p grows from 0 to 1 in the experiments from which figures 5 and 6 derive.

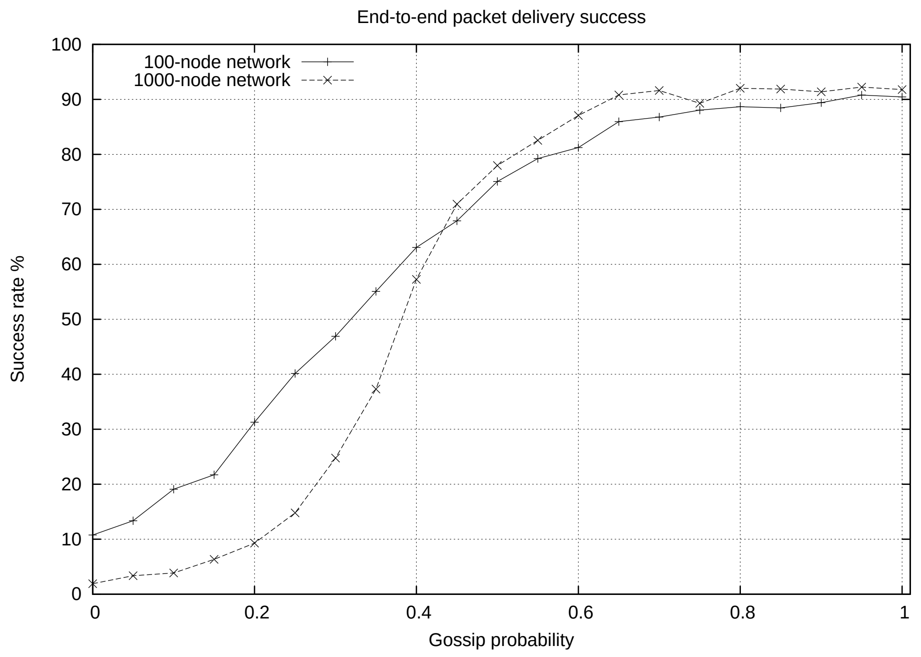

Figure 5. End-to-end packet delivery ratio versus rebroadcast probability for 100-node and 1000-node networks

6.2. *Emergent Effects at Scale*

Further evidence of the merit of larger-scale simulation is found in comparing network performance metrics obtained by simulation of otherwise-similar networks differing in node count. To observe the behavioural differences between small and large networks we generate network performance statistics for two networks. The networks differ in size by an order of magnitude; a 100-node network and a 1000-node network, in which the former is a subset of the latter. We measure *packet delivery success ratio* and *end-to-end delivery latency*, as measured by Das *et al.* in earlier comparative work [10], but omit *routing overhead* as it is always zero for gossip-based networks.

Consider figure 6 in which end-to-end delivery latency is plotted versus rebroadcast probability, normalised against maximum latency observed for the corresponding network size. In both cases we see average end-to-end latency increases until rebroadcast probability reaches around 0.4. For the 1000-node network we see that latency peaks around this rebroadcast probability, and then falls as rebroadcast probability increases further. However, in the 100-node network we do not see a similar fall in latency as rebroadcast probability increases further.

Now consider figure 5 in which end-to-end delivery success rate is plotted versus rebroadcast probability. Both plots follow similarly-shaped sigmoid curves in which delivery success transitions from *subcritical* to *supercritical* state as rebroadcast probability passes a

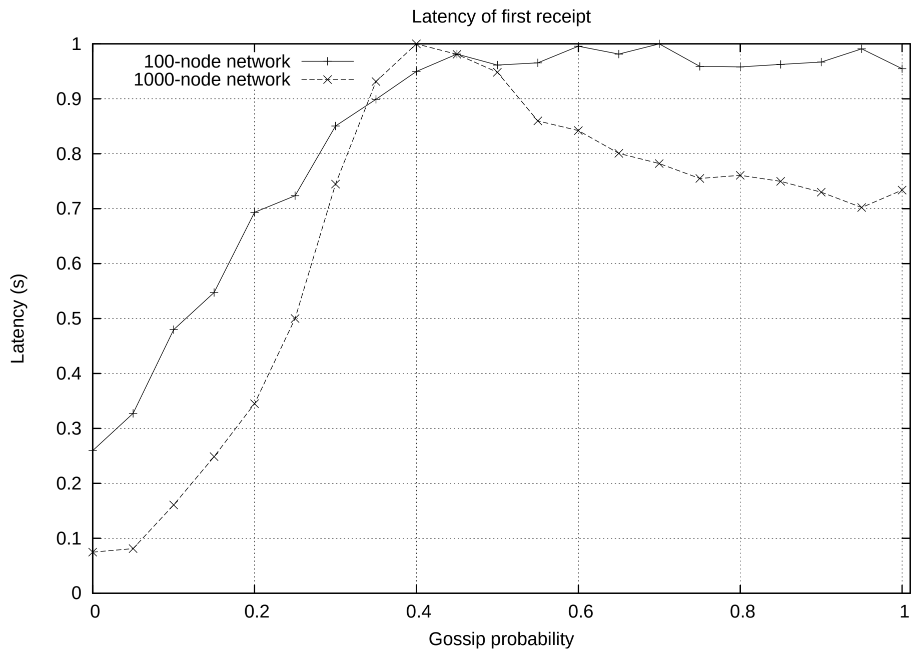

Figure 6. Normalised end-to-end latency versus rebroadcast probability for 100-node and 1000-node networks

critical value. However, the expected sigmoid shape is much more clearly defined in the 1000-node network plot than the 100-node network plot. Increasing network size further yields ever more sharply defined sigmoid curves. This is consistent with *percolation theory* which states that the transition between *subcritical* to *supercritical* states becomes more abrupt as network size increases [37], because the message blocking influence of individual nodes on the greater network becomes lesser [14].

We observe a qualitative difference in simulated behaviour between the small and large networks. This is of key importance in the design of real-time networks carrying time-sensitive data. Although the 100-node network is a subset of the 1000-node network, simulating the smaller network is not sufficient to capture all behaviour demonstated in the larger network. Whereas other behavioural effects might be evident in simulations of varying scale, the extent to which these effects are expressed may also vary. This latter observation is of critical importance where we hope to exploit simulation to discover new effects, rather than to confirm already-known effects.

Sensornet designers utilising simulation techniques must ensure that the scale of the simulated network is comparable to that of the proposed real network. This precludes use of simulators with poor scalability characteristics.

Consider a 1000-node network in which there exists a requirement for average end-to-end latency to be no greater than 0.8s. A designer with access only to data for 100-node networks may determine that this requirement can be met only where gossip probability $p < 0.25$ (figure 6). However, a designer with access to data for 1000-node networks would determine that this requirement can be met with $0.00 \leq p < 0.30$, or $0.65 < p \leq 1.00$. If critical probability p_c at which bimodal transition [14] is observed is around 0.4 (figure 5) it may be greatly preferable to select some value of $p > 0.65$ to achieve acceptable delivery.

7. Future work

The three-phase radio algorithm described in section 3.3.1 is inherently parallelisable because each phase consists of a set of independent pairwise node-node tests. The model requires the preservation of the phase execution ordering only, and does not require any specific ordering of calculations within phases or prohibit these being performed concurrently. However, the current implementation in *yass* performs these tests serially, with coarse parallelism achieved by running several independent simulations in parallel. Plans exist to reimplement the algorithm in a parallelised form, allowing larger individual simulations to be executed in a given time, in which each set of tests is divided between a set of threads which may or may not execute on the same machine. Plans also exist for a GPGPU implementation, exploiting the cheap power offered by commodity many-core GPUs. The algorithm is an excellent candidate for this type of acceleration as it consists of many independent but similar calculations and small shared datasets, mapping neatly onto GPU *kernels* and *textures* respectively [15].

The *yass* tool serves both as a simulation environment and as a testbed for experimental simulator components. It would be feasible to extract the three-phase radio algorithm described in section 3.3.1 and reimplement this for other network simulators such as *ns-2* [5] or *J-Sim* [2]. If successful this would permit similar performance improvement in widely-used simulation tools as in *yass*. However, the work may be non-negligible and require modification of underlying component architecture and interfaces if the data used within the algorithm are not currently exposed to the simulation components responsible for radio modelling.

Section 6 showed that simulation results obtained using *yass* are consistent with theoretical results and simulation results published by other researchers. It would be useful to make further comparisons, across a representative cross-section of typical sensornet scenarios, against other simulators and real testbed networks. Showing the results produced by *yass* to be equivalent, within some acceptable margin of error, would be sufficient validation to accept usage of *yass* in any situation where otherwise a real testbed network or some other less efficient simulator would otherwise have been required. Unfortunately, within the sensornet research community there does not currently exist any broadly accepted validation test suite, the development and acceptance of which is a necessary prerequisite for this work.

Section 3.2 discusses the post-hoc analysis of simulation trace data captured during execution of test cases. Recording this trace data is generally very expensive due to the number of distinct elements and overall volume, such that overall simulation performance is I/O-bound. Reducing I/O overhead would permit more simulated behaviour to be evaluated per unit time, enabling evaluation of longer simulated periods, evaluation of larger simulated systems, or simply to obtain results more quickly.

One approach is to record only a partial trace containing some fraction of simulated events and discard the remainder, with the assumption that the recorded subset is sufficiently representative of the whole. Sampling approaches such as *Cisco NetFlow* [7] record every *n*th packet, hence reducing the volume of recorded data to $\frac{1}{n}$ of the unexpurgated dataset. It would be possible to mandate that simulated time progressed as some multiple of wall time, dropping any trace records which cannot be recorded within deadline, dynamically balancing quality and performance. It remains an open question whether these partial sampling approaches are appropriate in low-traffic wireless sensor networks which are fundamentally unlike the conventional high-traffic wired LANs for which they were developed.

8. Conclusions

In section 2 a set of desired research objectives was defined, against which we now state our findings.

Objective 1: *Identify techniques through which to improve upon the performance of existing sensornet simulators*

A novel multi-phase model is presented by which computational costs of the most expensive element of sensornet simulation, modelling effects of radio propagation on low-level communications activity, is reduced substantially.

Objective 2: *Measure the extent to which these optimising techniques improve performance and enable larger-scale simulation*

The proposed optimisations transform the quadratic relationship between simulated network size and simulation runtime to near-linear for networks of the order of 1000 nodes, and that increased time savings are obtained as network size and network density increase.

Objective 3: *Improve the range of simulated network measures made available to the investigator*

Offline analysis performed once at the end of simulation activity can produce any measure that is producable by online analysis performed continually during simulation but with reduced overhead. Offline analysis also enables production of analytical results not foreseen prior to commencement of simulation activity.

Objective 4: *Validate optimised simulation to ensure that performance gains do not sacrifice solution quality*

Simulated results are as predicted by theoretical analysis, and as demonstrated by other simulators. Simulation results obtained under optimised simulation match closely those obtained for unoptimised simulation.

Acknowledgements

The authors would like to thank the anonymous reviewers for the helpful and constructive criticism which was invaluable in the preparation of this document.

References

[1] G. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the AFIPS Spring Joint Computer Conference*, pages 483–485, Atlantic City, April 1967.

[2] A.Sobeih, W. Chen, J. Hou, L. Kung, N. Li, H. Lim, H. Tyan, and H. Zhang. J-Sim: A simulation environment for wireless sensor networks. In *Annual Simulation Symposium*, pages 175–187, 2005.

[3] R. Bagrodia and R. Meyer. PARSEC: A Parallel Simulation Environment for Complex Systems. *IEEE Computer*, pages 77–85, October 1998.

[4] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala. Improving simulation for network research. Technical Report 99-702, USC Computer Science Dept., March 1999.

[5] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *Computer*, 33(5):59–67, 2000.

[6] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of MANET simulators. In *Proceedings of the Second ACM International Workshop on Principles of Mobile Computing*, pages 38–43, 2002.

[7] B. Claise. RFC 3954: Cisco Systems NetFlow Services Export Version 9. Downloaded from http://www.ietf.org/rfc/rfc3954.txt (checked 12/06/2008), 2004.

[8] Crossbow Technology Inc. MICA2 datasheet, part number 6020-0042-08 Rev A.

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

[10] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *INFOCOM (1)*, pages 3–12, 2000.

[11] C. Fullmer and J. Garcia-Luna-Aceves. Solutions to hidden terminal problems in wireless networks. In *SIGCOMM'97*, pages 39–49, New York, NY, USA, 1997. ACM.

[12] 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, February 2002.

[13] E. Goturk. Emulating ad hoc networks: Differences from simulations and emulation specific problems. In *New Trends in Computer Networks*, volume 1. Imperial College Press, October 2005.

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

[15] M. Harris. *GPU Gems 2*, chapter 31: Mapping Computational Concepts to GPUs, pages 493–508. Addison Wesley, 2005.

[16] T. Henderson, S. Roy, S. Floyd, and G. Riley. ns-3 project goals. In *WNS2 '06: Proceedings of the 2006 workshop on ns-2: the IP network simulator*, pages 13–20, New York, NY, USA, 2006. ACM Press.

[17] C. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1985.

[18] P. Huang, D. Estrin, and J. Heidemann. Enabling large-scale simulations: Selective abstraction approach to the study of multicast protocols. In *MASCOTS*, pages 241–248, 1998.

[19] D. Johnson, D. Maltz, and J. Broch. *DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[20] D. Kotz, C. Newport, and C. Elliott. The mistaken axioms of wireless-network research. Technical Report TR2003-467, Dept. of Computer Science, Dartmouth College, July 2003.

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

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

[23] O. Landsiedel, K. Wehrle, B. Titzer, and J. Palsberg. Enabling detailed modeling and analysis of sensor networks. *Praxis der Informationsverarbeitung und Kommunikation*, 28(2):101–106, 2005.

[24] J. Lehnert, D. Gsrgen, H. Frey, and P. Sturm. A scalable workbench for implementing and evaluating distributed applications in mobile ad hoc networks. In *Proceedings of Mobile Ad Hoc Networks, Western Simulation MultiConference WMC'04*, 2004.

[25] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Embedded Network Sensor Systems*, pages 126–137, 2003.

[26] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai. Towards yet another peer-to-peer simulator. In *Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs) 2006*, 2006.

[27] V. Naoumov and T. Gross. Simulation of large ad hoc networks. In *MSWIM '03: Modeling analysis and simulation of wireless and mobile systems*, pages 50–57, New York, NY, USA, 2003. ACM Press.

[28] C. Perkins and E. Royer. Ad-hoc On-demand Distance Vector routing. In *Proceedings of Second IEEE Workshop on Mobile Computer Systems and Applications*, pages 90–100, New Orleans, LA, Feb 1999.

[29] L. F. Perrone and D. M. Nicol. A scalable simulator for TinyOS applications. In *Proceedings of the 2002 Winter Simulation Conference (WSC'02)*, volume 1, pages 679–687, 2002.

[30] D. Rao and P. Wilsey. Modeling and simulation of active networks. In *Proceedings of the 34th Annual Simulation Symposium (SS01)*, pages 177–184, Washington, DC, USA, 2001. IEEE Computer Society.

[31] D. Rao and P. Wilsey. Multi-resolution network simulations using dynamic component substitution. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 142–149, Washington, DC, USA, 2001. IEEE Computer Society.

[32] G. Riley, R. Fujimoto, and M. Ammar. A generic framework for parallelization of network simulations. In *MASCOTS'99*, pages 128–144, 1999.

[33] A. Sobeih, M. Viswanathan, and J. Hou. Check and simulate: A case for incorporating model checking in network simulation. In *Proc. of ACM-IEEE MEMOCODE'04*, pages 27–36, June 2004.

[34] B. Titzer, D. Lee, and J. Palsberg. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of Information Processing in Sensor Networks (IPSN'05)*, pages 67–72. IEEE Press, 2005.

[35] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of the 5th symposium on Operating Systems Design and Implementation*, pages 271–284, New York, NY, USA, December 2002. ACM.

[36] K. Walsh and E. Sirer. Staged simulation: A general technique for improving simulation scale and performance. *ACM Transactions on Modeling and Computer Simulation*, 14(2):170–195, 2004.

[37] A. Schiper Y. Sasson, D. Cavin. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking (WCNC03)*, volume 2, pages 1124 – 1130, March 2003.

[38] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proceedings of the 12th workshop on Parallel and Distributed Simulation*, pages 154–161, Washington, DC, USA, 1998. IEEE Computer Society.