Ross Shannon · Graham Williamson · Aaron Quigley · Paddy Nixon

Visualising Network Communications to Evaluate a Data Dissemination Method for Ubiquitous Systems

Received: 11 June 2007 / Accepted: 3 July 2007

Abstract Ubiquitous computing systems which include wireless devices in their networks rely on sometimes fragile *ah-hoc* communication channels between heterogeneous devices to operate. To maintain stability and robustness, the contextual information contained on each node present in the system needs to be disseminated between nodes, so that in the event of a node failure, the data is still available to the wider network. One approach to disseminating information around topologically unstable networks is a type of epidemic algorithm called *gossiping*. However, developing an efficient, resilient algorithm to operate effectively in such an environment is an ongoing challenge. We have developed a visualisation tool that supports the evaluation of gossiping methods by enabling the algorithm designer to view the visual evolution of a network of nodes, from which they can monitor the stability of information within the network. By using this tool the designer can watch as information is propagated throughout their network, and see the effects of events such as node failures, so that they can make informed changes to the design of their algorithms.

1 Introduction

Ubiquitous computing systems require a large amount of rich contextual information to be available in order to support adaptive, context-aware applications. System designers face the challenge of delivering information from contributing sensors to all points where it is required by ubiquitous applications. Furthermore, because of the nature of the information and the applications which use it, it must be delivered in a timely manner. Traditional communication networks generally have well-defined topologies and stable communication between participating devices. However, when developing communication protocols for ubiquitous systems, we must take into account the system's inherent instability, as wireless devices come and go, and links between nodes are frequently disrupted.

These characteristics have encouraged us to seek a more appropriate method of disseminating information throughout a ubiquitous system. We are using a non-deterministic mechanism called gossiping [6, 16] to underpin the communications. Gossiping algorithms support fully-decentralised system designs by requiring far fewer guarantees about network structure, reliability and latency than traditional approaches. Though still an experimental technique, we feel that gossiping promises to be efficient, robust and scalable.

Gossiping algorithms aim to maximise the reach of individual messages throughout the network while minimising communication overheads. Through gossiping, sensors or other devices which generate relevant contextual data attempt to mirror as much of their data as possible to other nodes in the network, so that in the case of a node dropping out of the system, the information it was carrying is still available from another point in the network. The designers of such algorithms need to balance a wide range of factors to develop efficient algorithms which are robust to change.

Alternatives to gossiping include unbounded flooding of messages throughout a network, which comes with performance penalties due to large transmission overheads. Disseminating messages using distributed hash tables, spanning trees or other forms of structured overlays such as CAN [11] or Chord [13] require continuous maintenance as the network evolves, making them more difficult to use in *ad-hoc* systems. Another option is IP multicast, though this does not give delivery guarantees and may not always be available. An exemplar ubiquitous system, Gaia [12], communicates using CORBA dis-

This work is partially supported by Science Foundation Ireland under grant numbers 03/CE2/I303-1, "LERO: the Irish Software Engineering Research Centre" and 04/RPI/1544, "Secure and Predictable Pervasive Computing."

Systems Research Group,
School of Computer Science and Informatics,
UCD Dublin, IE
Tel.: +353-1-7165352
Fax: +353-1-7167262
E-mail: ross.shannon@ucd.ie

tributed objects, but is designed to operate on smaller scales than the other systems mentioned here.

Ubiquitous systems are generally evolutionary in nature, with enormous state spaces and complex entanglements of components and devices. The complexity of the interactions within the dynamic component population in these systems is difficult to appreciate, particularly as they are ever-changing. When working with communication schemes such as gossiping, many of the assumptions made in the evaluation of data dissemination methods for static networks no longer apply. As the system's behaviour is emergent, it can be difficult to evaluate the effectiveness of a certain algorithm, or the effect that a small optimisation to that same algorithm will have. We require an abstraction to allow designers to see the high-level behaviour of their systems in operation, and for this, we propose the use of visualisation. In the following sections we present our approach to the development of a visualisation framework to support system designers tasked with creating intelligent gossiping algorithms.

2 Evaluating the efficacy of gossiping algorithms

When considering data distribution for ubiquitous systems in general, there are several important measures which allow us to quantify performance. For example, latency is the time it takes for a particular item of data to traverse from a given source to its destination and is important to ensure that data reaches the points where it is needed in a timely manner. Robustness of the algorithm to node and link failures is another important measure: what failure rates can be tolerated before data is lost? A detailed discussion of the evaluation of gossiping algorithms, including the parameters that can affect performance and the measurements that may be taken are discussed by Williamson et al. in [16].

The major strengths of gossiping algorithms stem from their simple, decentralised nature and resulting emergent behaviour, which make them particularly suited to use in ubiquitous systems. However, this is a double-edged sword: the same properties which result in a scalable and resilient protocol introduce complications for evaluation, comprehension and testing.

Through our experiences in building and evaluating gossiping algorithms, it has become apparent that a combination of approaches — simulation, experimentation in global testbeds such as PlanetLab [15], and support via visualisation — is necessary to provide a comprehensive understanding. This is corroborated by the observations of Haeberlen et al. [7] who argue, amongst other things, that we must choose to evaluate at several different points in the evaluation space and avoid tending toward convenient points which suit our purposes.

As designers of these systems, some aspects that a visualisation can be expected to help us understand include: the coverage, or *spread*, of data through the system; the redundancy, or *overlap*, of data distributed across the network; and the interactions and communication pathways between nodes. Graph theoretic measures such as shortest paths, network diameter, or clustering may also be effectively communicated.

As we noted earlier, visualisation forms a component part of a larger evaluation strategy and there are some measures best suited to other forms of evaluation. For example, aggregation and averaging of data over an entire run, or several runs, of an algorithm; determined measurements of data loss; and side-by-side comparisons of the effect of changing parameters of the algorithms.

However, visualisation has several advantages over more traditional analysis. As these systems are based on decentralised, local interactions, behaviour is produced by the collective operation of nodes, rather than the actions of a single node. Visualisation helps us understand collective operations and gives us an insight into the interactions that are occurring between nodes. Rather than trying to interpret a large volume of console output, we are able to visually interpret and analyse the evolution of the network over time and gain an intuitive feel for how the algorithm is working, where it might be going wrong, and how it may be improved.

A key property we expect a viewer to be able to glean from our visualisation is the stability of node membership in the network. Since a node's removal is indicated visually, a person watching the visualisation run will be able to recognise the frequency of node failures and removals, and therefore the amount of data that is likely being lost from the system over time. Viewing a snapshot of the visualisation at any point in its execution will give the viewer an intuitive notion of the robustness of the network. This can be inferred by analysing the link density and distribution of colour throughout the network. A loosely-coupled node means a greater possibility of a node failure causing trouble in the network.

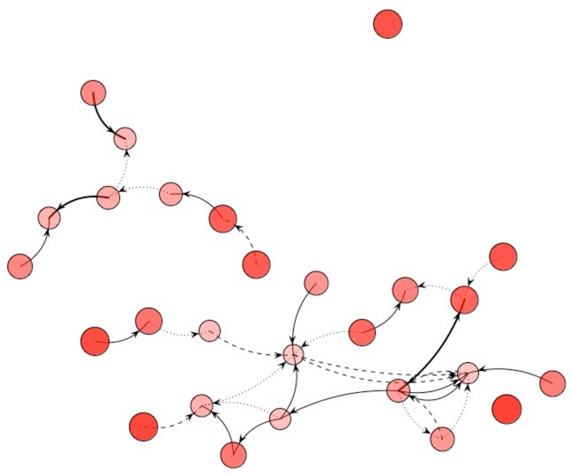

Fig. 1 Two defined clusters of communicating devices with varying link strengths, and an outlying node. Nodes whose data has been mirrored adequately become fainter over time.

3 Visualising gossiped network information

Previous work in the visualisation of network traffic has involved visualising pathways between computers in generally static networks with prescribed topologies [2,3]. Since the interconnections between nodes are known in these cases, the methods presented focus on the data exchange between nodes. A popular technique is to represent data transmission with a flowmap between nodes [10], where the breadth of the link between two nodes corresponds to the amount of data flow along this connection. Methods have been developed for measuring the timeliness of message transmission [14], or routing characteristics of a network [1].

The instability of networks in ubiquitous systems make many of these techniques unsuitable. For instance, the topology and node membership in a traditional network is largely fixed, whereas in a ubiquitous network by their very nature nodes may only join the network for a few moments while they are in range of another device. As such, visualisation algorithms need to be designed to adapt to the dynamic component population at each iteration. Secondly, we are less interested in the volume of data being transmitted and more focused on the stability of certain communication pathways and the ability to route pertinent information from one device to another.

These different priorities call for a departure from traditional network visualisation methods towards a method which can adequately deal with the fluctuations in the underlying data. In our system, being able to view the evolution of the network [4], and the strengthening of ties between groups of devices is of paramount importance.

We have built an evaluation tool in JUNG, the Java Universal Network/Graph Framework [8], which supports modelling and visualising dynamic graphs. In our evaluation tool, seen in Figure 1, we represent devices in the system as nodes in a graph, which expands dynamically as the system evolves. When a device communicates to another, we instantiate a directed edge between them. Each edge in the visualisation has an associated weight, or force [5], which acts upon the edge to draw its two incident nodes closer together. Nodes without adjoining edges will repel each other in much the same way, so that after some time, nodes that are communicating with each other will gravitate together, while unconnected nodes will move further apart. These movements are all smoothly animated, and allow the viewer to see distinct clusters forming naturally in the network.

These properties mean that, when the visualisation begins, the nodes will all spread out evenly over the drawing space, as there are no connections between them and they each contain unique information. Over time, connections are established between them, which are initially rather loose (demarcated as such with a thin dotted line). As the visualisation progresses, nodes that are transmitting information between themselves frequently will begin to converge, and the edge between them will solidify to show this stronger bond.

Ideally, as the visualisation progresses, the nodes will all form a tightly clustered group with dense interconnections (see Figure 2). This means that the data has been disseminated widely, and corresponds to a robust, stable data store. In this state, the network will be able to deal with a node failure, since any information it held has been propagated to other nodes.

Nodes with data that has been acceptably mirrored throughout the network become smaller and lighter as time progresses, indicating that their information is not in danger of being lost to an errant node failure. Outlying nodes, which haven't distributed their content widely or made any connections at all will begin to grow redder and larger over time, as if a pressure was building up within them. If they then make a connection to another node and spread some of their information, this pressure goes down and they return to normal colour and size. On the other hand, if they fail to spread their information before they leave the network, we mark this node black and after a period remove it from the visualisation. Such a removal may leave parts of the graph unconnected, and therefore more susceptible to further problems. This allows the designer to view the visualisation at any time step and identify any obvious problems, and keep a running tally of how much data loss their algorithm has allowed.

This "whole network" view of proceedings allows designers to see the impact various optimisations to the gossiping algorithm's parameters have on the network's evolution, through successive runs of the visualisation.

4 Future work

For our initial evaluations, we have simulated a gossiping algorithm using PeerSim [9] and provided the output to our visualisation tool. In future it will be possible to use the same tool to review the behaviour of deployed algorithms by collecting traces from each instance and collating them into a single source of input. It may even be possible to perform this operation in real-time as the system is operating. Another avenue for future work is to generalise what we have achieved in visualising gossiping algorithms and investigate the application of this visualisation tool to other data dissemination methods.

We aim to perform a user study on this technique with gossiping algorithm designers to validate that our visualisation makes the comprehension of emergent behaviour easier and helps to clarify the real result of performing certain optimisations to a gossiping algorithm. We will also be able to measure how quickly and accurately a subject comprehends what changes in the visualisation correspond to in the underlying network. This study will also give us some indication of what size of network the tool will need to scale to, as at this point we are assuming networks no larger than two hundred distinct devices.

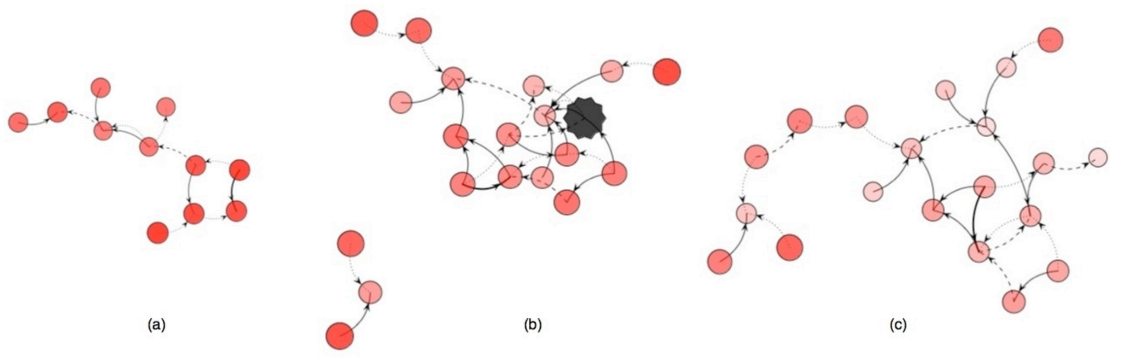

Fig. 2 Three stages in the evolution of a network. In (a), we have a view of the network in its early stages, with low connectivity. In (b) we can see that the graph has become better connected, but there are now two disconnected clusters. Also visible is a node that has failed. In (c) we se a generally well-connected graph after the clusters have joined.

Finally, although the visualisation provides various insights for the system designer, and can be used as an intuitive indicator of correct operation of the system, we see this as one component in a larger data dissemination validation framework. Work is currently continuing on other parts of this evaluation including statistical analysis of simulation output [17], and implementation on PlanetLab [15].

5 Conclusion

We have developed a visualisation framework aimed at designers of dissemination algorithms for ubiquitous systems, which they can use to evaluate the performance of their algorithms. As contextual data is crucial to the operation of a ubiquitous system, we consider the design of these algorithms to be a very important consideration. Our visualisation supports a view of the whole network of interconnected devices. By running the visualisation after changing aspects of an algorithm, designers can examine the effect that a combination of parameters can have on the performance of the network by observing the resulting emergent behaviour.

References

1. S. Au, C. Leckie, A. Parhar, and G. Wong. Efficient visualization of large routing topologies. *Int. J. Network Mgmt*, 14:105–118, 2004.
2. R. A. Becker, S. G. Eick, and A. R. Wilks. Graphical methods to analyze network data. *Communications, 1993. ICC 93. Geneva. Technical Program, Conference Record, IEEE International Conference on*, 2, 1993.
3. R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. In *IEEE Transactions on Visualization and Computer Graphics*, volume 1, pages 16–28, March 1995.
4. U. Brandes and S. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.
5. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42(149160):194–202, 1984.
6. P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic information dissemination in distributed systems. *Computer*, 37(5):60–67, 2004.
7. A. Haeberlen, A. Mislove, A. Post, and P. Druschel. Fallacies in evaluating decentralized systems. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, February 2006.
8. J. O'Madadhain, D. Fisher, P. Smyth, S. White, and Y. Boey. Analysis and visualization of network data using JUNG. *Journal of Statistical Software*, 2005.
9. PeerSim: A Peer-to-Peer Simulator. http://peersim.sourceforge.net/.
10. D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow Map Layout. *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, 2005.
11. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 31, pages 161–172. ACM Press, October 2001.
12. M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002.
13. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.
14. E. Swing. Flodar: flow visualization of network traffic. *Computer Graphics and Applications, IEEE*, 18(5):6–8, 1998.
15. The PlanetLab Homepage. http://www.planet-lab.org/.
16. G. Williamson, G. Stevenson, S. Neely, L. Coyle, and P. Nixon. Scalable information dissemination for pervasive systems: Implementation and evaluation. In *MPAC '06: Proceedings of the 4th international workshop on Middleware for pervasive and ad-hoc computing*, Nov. 2006.
17. G. Williamson, G. Stevenson, S. Neely, S. Dobson, and P. Nixon. An evaluation framework for disseminating context information with gossiping. In *Proceedings of the 1st European Conference on Smart Sensing and Context*, volume 4272 of *LNCS*, 2006.