

Engineering Large-Scale Observation Software Systems

DASEL Technical Report 2007/07/DL01

David Lamb, Martin Randles, A. Taleb-Bendiab

School of Computing and Mathematical Sciences,
Liverpool John Moores University, Liverpool, UK, L3 3AF
D.Lamb@2005.ljmu.ac.uk, M.J.Randles@2004.ljmu.ac.uk, A.TalebBendiab@ljmu.ac.uk

Abstract: The ubiquitous centralised feedback control architectural model has been widely adopted in autonomic systems engineering. However, this type of model suffers from a lack of scalability as for instance, the number of managed elements increase. Additionally, the emergent nature of complex systems can quickly render design-based models inaccurate. Alternative approaches based on self-organising systems theory are now increasingly investigated to develop scalable decentralised control models for networks of autonomic systems. However, engineering support for the design and application of these techniques is limited, and their suitability not yet well proven. Based on an ongoing research into model based engineering of self-organising networks, this paper investigates a signature-based method of adaptive observation relevant to existing research in graph theory and complex systems. Primarily, this approach attempts to address the specification and implementation of both the characteristics of the observed structure, and the necessary qualities of a matching and efficient observation overlay. This is tested on a basic topological example in a simulation environment to demonstrate how observers can make efficient use of limited resources while accurately observing a system.

Keywords: Software Engineering, Self-Organising Networks, Observation, Autonomic Systems.

1. Introduction

Autonomic Computing models are intended to provide software with, amongst other aspects, self-managing and configuration capabilities, in order to ensure the system components interact together correctly to meet their design goals and user requirements. The computational cost requirements of this self-* behaviour are dependent on monitoring methods used [1], and further affected by the scale of the managed system. While models exist to allow feedback-based observational control of software, there is still a lack of understanding of the engineering methods required for the scaleable control of systems. Therefore this work takes self-organising system research [2], along with decentralised control and complexity theory, and suggests a monitoring solution that manages complex events through a targeted, resource-constrained observation overlay. We attempt to address some of the software engineering concerns by proposing the basics of

this observation framework, capable of monitoring important limited subsets of large-scale systems.

To introduce the matter of complex system connectivity, we look at a special class of topological emergence observed in real world networks such as the web, and other autonomous and natural systems [3] – namely scale free connectivity. Scale free networks have interesting properties of “resilience and fragility” [4], both characteristics that can be exploited from an observer’s viewpoint. The application of the theory contained in this paper is to combine the research areas described and specify an engineering model for adaptive observation, via metric recognition of topological types. Detecting such structures and behaviour allows system monitoring at a representative sample, reducing complexity and computational load. This provides a step toward behaviour observation and regulation, without the need for exhaustive observation or global information regarding the micro-level systems.

The paper is organised as follows: the next section outlines briefly the principles of complex, self-organising behaviour and associated connectivity, together with an assessment of detection metrics for a common emergent topology - scale-free connectivity. Section 3 presents the basic specification of the programming model, along with identifying areas for expansion. The model is applied in Section 4 in a simulated network environment, written within the open source network simulator Repast [5]. This section includes limited results on the use of the detection metrics and a monitoring strategy based on acquaintance immunisation [6]. Section 5 summarises the research and planned additions.

2. Complex Connectivity

Correct and complete modelling (and therefore monitoring and management) of complex systems can be difficult due to their large scale, and continual evolution in both structure and behaviour. This evolutionary behaviour is often emergent in nature – where overall system (macro) behaviour is the product of many simple interactions at the sub-system (micro) level. Therefore, from a modelling perspective it is helpful to discover these causal micro interactions, and to do so in a way that is scaleable. This section explores the scale-free model’s explanation of topological emergence, and importantly, how to

programmatically detect such structural organisation using relatively simple metrics. Firstly, we present a brief overview of the scale-free connectivity from existing research.

The emergent nature has been described as growth by preferential attachment, which results in a power law degree distribution [7]. The network evolves by the addition of nodes that connect to other nodes with greater preference according to their degree. This tendency to connect to nodes already having large numbers of links is also known as a rich-get-richer model. This power law distribution, as predicted by the scale-free model is observed in complex systems such as software dependency networks, and is specified as:

$$P(d) = cd^{-\lambda} \text{ for } d=m, \dots, M$$

Where d , the degree of the node is in the range ($m \dots M$), and c is a normalisation factor [8]. This differs from the random graph model where the node degree distribution is typically Poisson. Conversely, the scale-free distribution suggests that most nodes will have low connectivity, but crucially, in the tail of the distribution, a small number of nodes with a very high degree are expected. Given the large numbers of low-degree nodes, the network is resistant to the random removal of nodes. Equally, as there are relatively few hubs (well connected nodes) the network's overall connectivity is susceptible to targeted attacks. While these systems are understood in particular areas of research, they have not been precisely defined, with some mathematical results contradicting heuristic results [3]. Therefore, networks thought to follow the scale-free power law distribution can be difficult to detect without exhaustive monitoring. However a number of additional works [6, 8] have shown signatures for scale-free networks more specific than simple power law degree distributions.

In order to detect this connectivity, it is necessary to provide suitable identifying metrics. Metrics ought to provide the system with information as to when a scale-free phase transition is occurring or has occurred. Additionally, given the crucial nature of hubs in such systems it is necessary to define measures that identify these important network areas. Accordingly, software engineers can use these metrics as triggers for system observers, configured to send events describing organisational change. The following subsections describe some metrics for identifying scale-free topologies and related properties.

2.1 Mean Shortest Path

As the scale-free topology often exhibits small world properties, identifying this occurrence may be a useful measure. In small world graphs, most nodes are a few hops (intermediate nodes) from one another, despite not being directly connected themselves. A measure useful in calculating just how "small world" a given network graph appears is the mean shortest path:

$$S = \frac{1}{N} \sum_i^N S_{i,j}$$

S produces an average shortest path for the entire graph,

where $S_{i,j}$ is the shortest path between nodes i and j . Networks with small-world properties will have relatively low shortest paths when compared to similarly sized regular networks. However, random graphs also tend to have short paths – when compared to regular graphs [9]. Additionally, route-calculating paths for many node-pairs in a non-trivial network can be computationally expensive and may be inappropriate for use in an often-checked signature.

2.2 Clustering Coefficient

Watts and Strogatz [10] refined the above metric to detect small-world properties, and introduced a coefficient to measure clustering, C , in a graph:

$$C = \frac{1}{N} \sum_i C_i \text{ where } C_i = \frac{2\{e_{jk}\}}{d_i(d_i - 1)}$$

C_i is the coefficient for a given node, i , and is based on its surrounding nodes and connections. In the C_i formula, d_i is the degree of node i , and e_{jk} is an edge between nodes j and k . Nodes j and k represent the neighbours of node i . Therefore, the measure gives the proportion of connections that *exist* within the node's neighbourhood, normalised by those that *could exist*. The clustering metric is effectively a measure of how well a node's neighbourhood is interconnected. Watts and Strogatz found that while both random and small-world graphs had low shortest paths, small world graphs could be distinguished by their relatively high clustering coefficient.

2.3 Acquaintance Nomination

While the previous metrics go some way to identifying properties associated with scale-free graphs, some provide only partial identification or suffer from calculation complexity. A promising related approach, Acquaintance Immunisation [11] involves selecting a small subset of nodes to "immunise" while ensuring good coverage of the network. The immunised nodes are selected by the following steps:

1. For a given network size n , select a random set of nodes, the "Interrogated nodes", of size $p*n$, where p is a probability between 0 and 1.
2. For each interrogated node, select a connected neighbour, adding it to the set of immunised nodes.
3. The nodes in the immunised set are then monitored; this set is likely to contain well connected "hub" nodes.

While this alone is a valuable method for selecting observation points, it is also useful for a simple metric with little global knowledge and low computation cost. The proposed Acquaintance Nomination metric follows the steps above, but is only concerned with the number of immunised nodes. Mathematically, this number must lie between 0 and $p*n$, the extremes indicating a disconnected graph or a clique, respectively. Therefore, we can normalise between 0 and 1 as follows:

$$aMetric = \frac{|immunisedNodes|}{|network|*p}$$

Such that $aMetric$ is the size of the immunised node set divided by product of the network size and p . Limited simulation results, following in more detail in Section 4, suggest that this could provide an accurate metric to identify scale-free networks. Scale-free networks produce low $aMetric$ values, as the immunised set is likely to contain few nodes (the important hubs), while random and regular networks tend to produce values nearer to 1, as the acquaintance set is likely to contain many different nodes due to the connectivity of the graph.

3. Engineering Large-Scale Software Observers

In order to monitor large-scale software with sets of observers, it is necessary to deploy monitor instruments in appropriate parts of the system. As discussed in the Introduction, as software engineers, we cannot rely on the luxury of accurate and static model-based system structures for large complex systems. Therefore, any observation overlay must be able to set its coverage according to the present system structure, and to adapt with the structure's changes. Monitors must be specifically targeted within the system, and must be continually reassessed and modified at runtime.

An example design could continually evolve an exhaustive system model, based on current observations, providing a complete overview of the system components and their interconnections and dependencies. However, to provide sufficient detail to allow system management, it is likely that this model itself would grow in size and complexity - to a stage where it would overwhelm any attached diagnostic and reasoning processes. The previous sections have examined scale-free connectivity, with the partial modelling technique in 2.3 being of particular interest, as it suggests a reasonable overview of a scale-free network can be gained from a limited subset of nodes. Our engineering model is similarly based on the notion of partial observation, using a reduced, model-based observation overlay in order to monitor a large, complex and self-organising system structure.

The following subsections discuss some of the design aspects of such an overlay, along with the ways in which the proposed implementation will be tested within a simulated scale-free network environment.

3.1 A Typed Observation Model and Associated Characteristics

The first concern for a software engineer is how to generate and select the most appropriate overlay model in order to monitor an evolving self-organising system. One useful contribution would be the specification of scaleable observation patterns for certain topological features. These features could then be used as triggers for activating the appropriate overlay patterns. The main aim of the model is to use "feature signatures" to aid the classification of features and corresponding patterns to deal with complex and changing topologies. In order to develop and refine the SE model, it is proposed to use the emergence of scale-free connectivity, described in Section 2, as an example set

of features and measures with appropriate observer patterns. These associated measures are used to provide suitable test conditions to specify events that indicate the beginnings of a scale-free topology. As and when certain measures are matched, appropriate observation overlays can be added to the system.

The following subsections attempt to illustrate with basic pseudo-code examples, how various components of the model may function and interact together to form the basic framework. While the unfolding example tests topology identification, it is intended that the model could be extended to identify and observe other self-organisation heuristics.

3.2 Identifying Types: Feature Signatures

The purpose of a feature signature is to quickly identify a particular characteristic set that can then be used by the framework to select an appropriate observer set up or response. This section will concentrate on how the signatures could be structured, and how they can be used to match features. Firstly, it is suggested that the signature is specified as an interface, such that individual signatures subclasses can specify their matching criteria, yet classes interested in checking a signature need only know that it is "a Signature":

```
public abstract class Signature
{
    public abstract boolean isMatch();
}
```

Fig. 1. Signature interface definition

Further example signatures in this paper will be hard-coded, testing for a specific topological feature in a given system structure. However, the use of an interface allows a suitable signature class to externalise match criteria, (for example, to allow the specification of a matching signature in an external logical language) while still being usable in the same manner by dependent classes. Additionally, the inclusion of a Compound Signature (see Fig. 2) class allows one or more signatures to be joined together by a logical operator (e.g. AND/OR) to create more complex signatures that require a combination of several features for identification:

```
public abstract class CompoundSignature
    extends Signature
{
    public enum LogicCondition
    { AND, OR, NOR }

    public CompoundSignature
        (Vector<Signature> signatures,
         LogicCondition condition)

    public boolean isMatch();
    // check signatures according to specified
    // logical join condition
}
```

Fig. 2. Compound signature interface

3.3 Using Signatures to Select Observation Patterns

The model proposes a factory design to abstract the identification of topology and selection of an appropriate observer. This defers the selection of a particular overlay to another class – the observer factory – which performs the necessary matching of features and returns a suitable observer for the presented structure. From a programmer perspective, example usage of the factory to deploy a suitable observer is shown in Fig. 3.

```
//create or acquire network
NodeSet network=new PrefAttNetwork(500);

//obtain and deploy observer
TopologicalObserver topObs =
    ObserverFactory.getTopoObserver(network);
network.addTopologicalObserver(topObs);
```

Fig. 3. Example deployment code

It is proposed that signature-matching factories associate observation patterns (sets of observer construction algorithms) along with the characteristics (the signature) that the patterns require to provide efficient coverage. At runtime, when the factory is queried for an observer, it simply selects the observer with the closest matching signature. Continuing the scale-free example, should the software determine scale-free features exist in the structure and an appropriate observer is deployed; the observer can monitor and act appropriately. The code in Fig. 4 shows an acquaintance network signature (mathematically specified in 2.3) being tested.

```
DoubleTolerance matchValue = new
    DoubleTolerance(0.1d, 0.1d);

StaticSignature acqSig = new
    AcquaintanceSig(network, matchValue);

Boolean ACQUAINTANCE_MATCH =
    acqSig.isMatch();
```

Fig. 4. Topology detection code fragment

This code fragment shows example usage of the signature. The matching value is a special *tolerance* value, to allow the signature to match a specified value according to a given tolerance (± 0.1 in the example). The signature is constructed using the component undergoing test and the desired matching value. Calling the *isMatch* method tests the network according to the signature's algorithm. In this instance, as a scale-free topology is being tested, should the factory match the signature, it knows that the structure needing observation it is generally robust when subject to random failure, yet fragile when subject to targeted attack of its hubs. Therefore, a scale-free observer response could return an overlay that concentrates on top-level hubs, as they are a weakness in the structure and should be protected. The next section will discuss methods to implement this logical mapping between a given overlay pattern and the individual observer logic.

3.4 Creation and Deployment of Observer Patterns

In order to make up a suitable observation pattern, it must contain a set of observers that can accurately and efficiently monitor the identified structure. If a particular observation target requires a consistent logical response, this reaction could be hard-coded into the observer that is supplied by the process outlined in 3.2. Continuing the scale-free example, if this topological observer's main function was to take action to protect hubs, this simple logic could be encoded directly into either the observer construction or event response code; an example of the latter is shown in Fig. 5.

```
public void processNodeEvent(NodeEvent e){
    if (e.getType()==NodeEvent.NODE_HUB){
        protectAsHub(e.getNode());
    }
}
```

Fig. 5. Observer response code fragment

However, while we may be able to specify certain observation techniques for observed characteristics, in a changing system, this static mapping may not be sufficient. It is quite possible that the observation overlay may need to calculate a suitable compromise of different observation patterns, based on the various matched features. If the target requires this type of reasoned response that may not be available at design time, it would be inappropriate for the observer methods to be hard-coded. Instead, observers must reason about system state using an external logical formalism and evaluation language, partly explored in [12]. For simplicity, to try and illustrate the basics of large-scale management in the proposed framework, the simulation in this paper uses a hard-coded approach. A simple scale-free observer is hard-coded to use acquaintance immunisation to monitor a simulated network. It is, as proposed, deployed via the adaptive identification and selection outlined in previous sections.

4. Simulation of Adaptive Observer Overlay

The implemented software simulation tool is built on top of an open-source simulation framework, Repast [5]. The simulation is designed to allow the modelling of large-scale network observers, and to assess how they behave with various evolving network configurations. Repast operates simulations on a stepped cycle model, and in our software, an overlaid event-driven observation set is added, resulting in a similar setup to other network simulators, such as the P2P simulator, PeerSim [13]. Our software starts with a construction phase, followed by one or more modification phases until the termination conditions are met. Experiments start with an initial network, which is either: specified as a construction algorithm and generated at runtime, or loaded in a topology description format. At each *step* in the simulation's cycle, the simulation algorithm may change the network's organisation, or cause an interaction between nodes. The software updates the visualisation, and reports network statistics, as shown in Fig. 6.

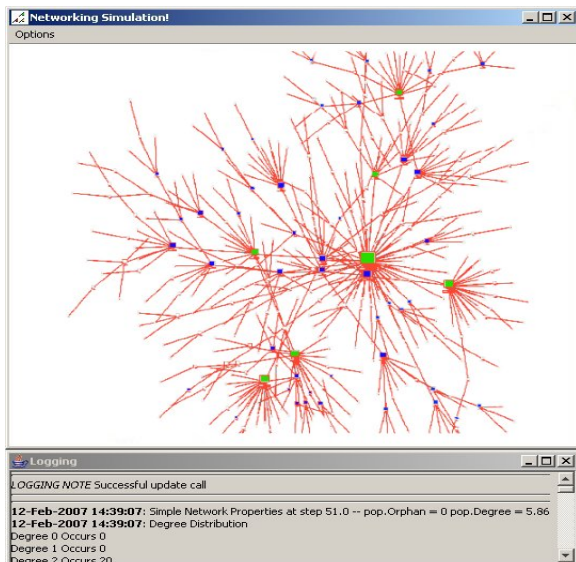


Fig. 6. Example simulation output

During the simulation, observers undergoing testing are deployed on the network, attaching to sets of nodes or individuals. They react in response to changes in the environment on an event model, using a framework of listeners. Feedback takes two main forms, direct or propagated. Direct feedback occurs when the observers make method calls direct on the network's structure, just like the steps in the simulation algorithm, whereas propagated feedback involves generation and sending of semantic events to higher level observers. This approach allows the software to react to changes online, where the detail of each change is represented at code level using *events*. Depending on the problem domain, the changes (runtime events) can be thought of as semantic monitoring events (e.g. a hub discovery), or at lower levels, data exchange between nodes (e.g. a transported datagram).

4.1 Notifying Simulation Changes with Runtime Events

The simulation uses a listener framework to monitor the network under test. A listener framework is a well-established software design pattern [14] whereby interested parties register themselves as listeners on an object. The object must inform all registered listeners when it undergoes a "listenable" change. In order for the listener pattern to work, *listenable objects* must define methods to *add* and *remove listeners*, and specify a *listener interface* that defines the methods the listener must implement (typically of the form *processEvent(theEvent)*). Additionally, suitable *event* classes must be designed, capable of adequately describing changes to others. In this simulation, node-related events are managed by the *NodeEvent* class (see Fig. 7). Instances specify the type of event, along with the node(s) affected. Interested parties must implement the *NodeListener* interface, and then attach to the required node(s), or entire networks. When the item undergoes a listenable change, it calls *processNodeEvent* with a suitably-created *NodeEvent* on registered *NodeListeners*.

```
public class NodeEvent{
    public enum Type {ADDED, REMOVED,
CONNECTED, DISCONNECTED, HUB, NOT_HUB}
    private SimpleNode node, additionalNode;
    private Type type;
}

public interface NodeListener{
    public void processNodeEvent(NodeEvent e);
}
```

Fig. 7. Node event and handler definitions

This event-driven model allows for independent observers with separate goals to exist in, monitor and act on a common environment. The cycle-driven simulation model allows the network to behave and change predictably to test observers in a variety of situations.

4.2 Adapting Overlays According to Topology

As part of the development, a number of simulations pertaining to the management and observation of large-scale networks have been created. Simulations include modelling graphs using various different construction algorithms, such as growth by preferential attachment, and simulations to organise observer teams, including in a cluster-based hierarchical model. To simulate related work [15] in this research group, the software was used to model an acquaintance observation simulator, using Cohen's algorithm [6] to select a subset of the network for intensive observation.

In this simulation, a network is generated, on which the software arranges an observation overlay. The observer deployment process determines the target topology type, according to its signature, using characteristics outlined in Section 3.2. As part of the experiment, the different signatures described are tested in various configurations and with different match parameters. As a when a satisfactory signature is determined, it is used to differentiate between scale-free and non-scale-free networks. Further information regarding the results of signature testing follows in the next section.

If the tested network appears scale-free, it deploys an observation strategy to monitor the network. As discussed in 3.4, this final deployment logic is presently hard-coded, so the observation responses for different topologies are fixed at design time. In order to provide a test of monitoring large network areas with relatively limited resources, an acquaintance immunisation observer is used as the appropriate overlay for a scale-free situation.

Analysis is then conducted on the observer overlay in order to test its effectiveness at detecting change within the network data from its limited observation viewpoint, which is described in more detail, along with the results in the next sections.

4.3 Experiment Intentions and Setup

The experiment set out to test signature recognition, overlay selection and response, and the effectiveness of the selected observation strategy. As such, it was conducted on a variety of topologies using different generative algorithms and network sizes. The simulation

generated a scale-free graph with the Kumar copy model [16], plus regular and random graphs. Two random graph types were generated to explore the effects of fewer node interconnections; the sparse graph having 2/3 the edges of the denser graph. The regular graphs are circular lattices, with a subtle difference in connections. Lattice 1 is a nearest-neighbour lattice, where nodes are connected to the neighbours on both sides. Lattice 2 is a next-nearest-neighbour lattice, where nodes are connected as Lattice 1, plus their *next* neighbours.

The simulations conducted aimed to assess the effectiveness of the signature approach (concentrating on those in Section 2 as an example) and the use of the specific signatures as a differentiation tool. Additionally, once the chosen observers are deployed, the simulation assesses the effectiveness of acquaintance immunisation as a scale-free observation pattern.

4.4 Experimental Results

The experiments revealed some interesting results comparing the possible scale-free indicators undergoing trial. As expected, the shortest path was relatively computationally expensive to calculate in its simplest form of an exhaustive route search. Additionally, it returned similar values for scale-free and random graphs of various sizes. Owing to the structure of the regular networks, the values were significantly higher in lattices, and as such, shortest paths could serve as an (albeit expensive) indicator between ordered and random or emergent graphs.

The cluster coefficient metric produced fairly consistent results per graph type for graphs of any size, excluding scale-free, where values intersected random graph results, though decayed as graph size increased. Again, the major differentiation in this metric, between any two equally sized graphs of different type, occurred between random and lattice 2 (interconnected neighbourhoods do not occur in lattice 1), with lattices returning higher values than random graphs. As anticipated, the proposed *acquaintance metric* proved an effective measure to differentiate scale-free from other types of graph, with similarly high value results for random and regular graphs, and a significantly lower value for scale free graphs, as shown in Fig. 8.

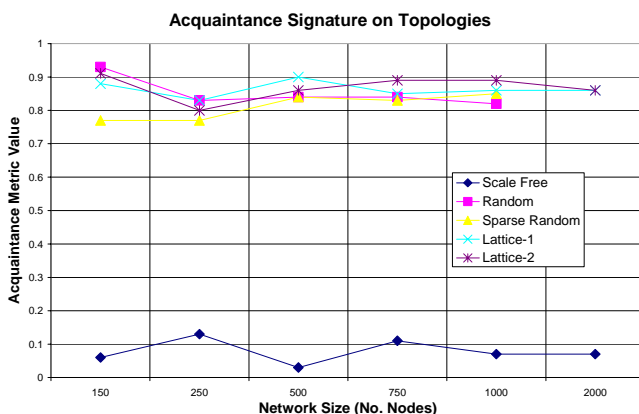


Fig. 8. Acquaintance Signature Results

In using this metric as an identifier, the simulation additionally attempts to assess acquaintance observation,

by observing a network for data spreading through it. Ordinary nodes pass data to a random number of neighbours, which do the same, circulating data about the network. The “observed” nodes halt the data and report it; to show the acquaintance technique can observe a network well, data should be detected quickly. Speedy detection would indicate that although a small proportion of nodes are “immunised”, thanks to the topology, the entire network is well protected.

In Fig. 9, the experiment’s network is displayed in a Fruchterman-Reingold [17] visualisation, where hubs tend to gravitate toward the centre and low-degree nodes move outwards. The node size is proportional to its degree, and nodes are coloured according to their state. This visual state is used to show the results of data runs through an observed network from randomly selected nodes. White nodes towards the edges have already received the data, while the observed nodes are the light-coloured nodes at the centre, and the dark-coloured nodes have not yet received information.

At this stage in the experiment, the network has an observation overlay in place, which was set up by the software according to the process described. To recap, the defined scale-free observation pattern (acquaintance immunisation) has been deployed, after the observation framework has used the successful signature (acquaintance nomination) to detect the network topology type and select the appropriate observation strategy.

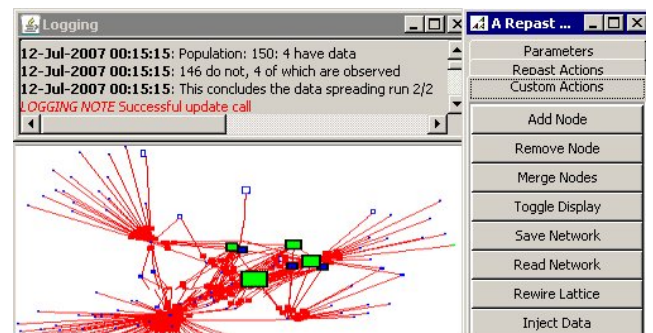


Fig. 9. The Acquaintance Immunisation simulation

The user interacts by clicking the “Inject Information” button, which selects a node at random, and passes data in to it. Data is propagated as described above, and the visualisation is updated. Two such attempts have been performed in the figure above, and as a result, information has only passed onward to 1 node at each attempt before being detected. This is shown in the logging window as 4 nodes of 150; 2 nodes directly given data, 2 nodes had data passed to them.

The experiment has gone some way to showing the implementation of a limited adaptive observation framework, and its value in determining topology type and setting up an appropriate observer overlay. Additionally, owing to the choice of experimental topology, it has shown that acquaintance immunisation (and our minor adaptation to make a useful metric) combine to form a useful observer pattern and feature combination for scale-free type networks. However, it is fair to clarify that the data propagation experiment is limited as the randomly selected insertion point is likely to be a low degree node,

due to the degree distribution. As a result, data propagation may be limited regardless of the presence of an observation overlay.

5. Research Summary

This paper sets out and explains some basic components that combine to make an adaptive observation model for large-scale systems. We intended to take into consideration some of the main software engineering concerns and provide simple designs and pseudo-code implementation in order to show how a system could be developed from suitable system-specific knowledge.

Additionally, experimentation showed that the limited example was successful in both network identification and the assigned observer that deployed as a result. In order to increase flexibility and function properly on systems comprised of unknown or mixed topology, several improvements would need to be considered. The authors have research both existing [12, 15, 18], and currently underway to address some of the areas in which the framework could be extended and fully implemented, including:

- Externalisation of signatures and therefore feature-to-pattern mapping, allowing these aspects to be specified externally in a logical specification language. This would allow for finer-grained detail between individual characteristics and elements of deployment algorithms, along with:
- Reasoning about deployment decisions, to allow runtime adaptation of existing observer pattern algorithms to best serve mixed topology systems
- Sufficient definitions of basic network topologies, identifying characteristics, and associated observation patterns, other than scale-free and the random and regular extremes
- Extension of above definitions to include other large-scale characteristics, in addition to those specific to topology, along with the related observational techniques recommended for the characteristics

The scale-free-specific examples reiterated the value of intelligent partial observation, such as acquaintance observation, in reducing monitoring computational load while retaining good coverage of a particular structure. However, it is accepted that this particular example does not address all types of structure found in large-scale systems, as it relies on the particular characteristics of the scale-free distribution. The goal of the presented research is to try and better understand the model of targeted partial observation, and to extend the model to create a complete SE approach. Hopefully, this will assist developers build software that can observe complex systems and react and plan rationally. As such, in order to help expand the model, immediate future work will look to apply and develop this model to real world software problems that suffer from organisational complexities, such as the emergent structure of software components controlled by market economics.

6. References

- [1] J. A. McCann and M. C. Huebscher, "Evaluation Issues in Autonomic Computing," in *Grid and Cooperative Computing – GCC 2004*, vol. 3252, *Lecture Notes in Computer Science*: Springer Berlin / Heidelberg, 2004, pp. 597-608.
- [2] Martin Randles, A. Taleb-Bendiab, Philip Miseldine, "Harnessing Complexity: A Logical Approach to Engineering and Controlling Self-Organizing Systems," *International Transactions on Systems Science and Applications*, vol. 2, pp. 11-20, 2006.
- [3] B. Bollobas and O. Riordan, "Robustness and Vulnerability of Scale-Free Random Graphs," *Internet Mathematics*, vol. 1, pp. 1-35, 2003.
- [4] A.-L. Barabasi, R. Albert, and H. Jeong, "Scale-free characteristics of random networks: the topology of the world-wide web," *Physica A: Statistical Mechanics and its Applications*, vol. 281, pp. 69-77, 2000.
- [5] O. S. SourceForge, "Repast API Documentation," 2005.
- [6] S. H. Reuven Cohen, Daniel ben-Avraham, "Structural Properties of Scale Free Networks," in *Handbook of Graphs and Networks*, S. Bornholdt, Ed.: Wiley-VCH, 2002.
- [7] A.-L. Barabasi and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, pp. 509-512, 1999.
- [8] L. Li, D. Alderson, J. C. Doyle, and W. Willinger, "Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications," *Internet Mathematics*, vol. 2, pp. 431-523, 2005.
- [9] B. Hayes, "Graph Theory in Practice: Part 2," *American Scientist*, vol. 88, pp. 104-109, 2000.
- [10] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440-442, 1998.
- [11] S. H. Reuven Cohen, Daniel ben-Avraham, "Efficient Immunization Strategies for Computer Networks and Populations," *Physical Review Letters*, vol. 91, 2003.
- [12] Martin Randles, A. Taleb-Bendiab, Philip Miseldine, "Scaleable Approaches to Engineering Autonomic Networks using Deliberative Logic and Network Theory," presented at SAACS06, Greenwich, 2006..
- [13] BISON, "PeerSim Sourceforge.net home," 2006.
- [14] R. H. Erich Gamma, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable OO Software*: Addison-Wesley, 1995.
- [15] M. Randles, P. Miseldine, A. Taleb-Bendiab, and D. Lamb, "Using Signatures of Self-Organisation for Monitoring and Influencing Large Scale Autonomic Systems," presented at EASE 07, 2007.
- [16] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. S. Tomkins, and E. Upfal, "Stochastic models for the web graph," presented at FOCS 2000, Redondo Beach, CA, USA, 2000.
- [17] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-directed Placement," *Software - Practice and Experience*, vol. 21, pp. 1129-1164, 1991.
- [18] Martin Randles, Hong Zhu, A. Taleb-Bendiab, "A Formal Approach to the Engineering of Emergence and its Recurrence," accepted, to be presented at EEDAS - ICAC, 2007