Investigating the Effects of Core Packet Behaviour between Sensors in Autonomic Networks

Investigating the Effects of Core Packet Behaviour between Sensors in Autonomic Networks

Kevin Curran, Matthias Baumgarten, Maurice Mulvenna, Kieran Greer, Chris Nugent

Faculty of Computing and Engineering, University of Ulster, Northern Ireland, UK
Email: {kj.curran, m.baumgarten md.mulvenna, krc.greer, cd.nugent}@ulster.ac.uk

Abstract

A core goal for autonomous systems such as that proposed here is automated collaboration in order to perform tasks or share information. The system is always distributed by default and frequently on a large-scale. It can be argued that robustness and economy demand the deployment of a tested autonomic supporting infrastructure whenever possible. Industrial systems middleware is generally modular and adaptive, however it can be demanding of bandwidth and indeed computational resources. Ideally we would like to map cleanly designed real-time resource constrained embedded systems onto autonomic smart-world supporting infrastructures. This paper is an attempt to investigate the peculiarities of node behaviour in such a network.

1 Introduction

Typically, network topologies are no longer static, due to the increasing mobility of users. Ubiquitous computing is a term often associated with this type of networking (Bischoff and Kortuem, 2006). The Internet is built on the DARPA protocol suite Transmission Control Protocol/Internet Protocol (TCP/IP) (RFC761, 1980), with IPv4 (RFC791, 1981) as the enabling infrastructure for higher-level protocols such as TCP and the User Datagram Protocol (UDP) (RFC768, 1980). It can be argued that the protocols underlying the Internet were not designed for the latest cellular type networks with their low bandwidth, high error losses and roaming users, thus many 'fixes' have arisen to solve the problem of efficient data delivery to mobile resource constrained devices (Saber, 2003). Mobility requires adaptability meaning that systems must be location-aware and situation-aware taking advantage of this information in order to dynamically reconfigure themselves in a distributed fashion (Solon, 2003). Situations, in which a user moves an end-device and uses information services, can be challenging. In these situations the placement of different co-operating parts is a research challenge. The heterogeneity is not only static but also dynamic as software capabilities, resource availability and resource requirements may change over time. The support system of a nomadic user must distribute, in an appropriate way, the current session amongst the end-user system, network elements and application servers. In addition, when the execution environment changes in an essential and persistent way, it may be beneficial to reconfigure the co-operating parts. The redistribution or relocation as such is technically quite straightforward but not trivial. On the contrary, the set of rules that the detection of essential and persistent changes is based on is a challenging research issue. This problem is the focus of this paper. A middleware framework has been developed that provides uniform access to remote services and device-specific capabilities, the decoupling of the application communications model and the underlying interoperability protocols alongside dynamic extensibility supporting a range of devices from small-embedded systems to full-fledged computers. This opening chapter provides an overview of the subject, outlines the problems in streaming media to mobile devices, depicts some of the traditional solutions and presents proposed solutions for the addressed problem area.

One of the challenges for future smart environmental infrastructures is the need for them to need to reason about their situation and to understand their own behaviour. To do this they are required (both at the level of individual components and as a whole) to be introspective and reflective, and to feed back the results of these processes to be used to improve performance. While this provides the knowledge with which they can, eventually, manage and configure themselves it does also make them more self-aware or in short it makes them smarter. However, in order to get 'smarter', the environment, its entities and services need some form of

properly represented, well correlated and widely accessible repository that leads to the concept of knowledge network.

Within the concept of a knowledge network there is a basic need for the expressive and flexible means to promote context-awareness. Smart environments, their components and services need to have awareness of situations with differing degrees of granularity (Sterritt et al., 2004). There is a requirement for some form of computational model of context processing as presented in (Balkenius & Moren, 2000) that orchestrates context stimuli and components in a coherent representation. Additionally, there is a requirement for some way of gauging the quality of contextual information objectively as it is gathered, as from the Quality of Context mechanism of (Buchholz et al., 2003) where any contextual information has associated with it parameters including precision of information, correctness probability, trust worthiness, resolution and regency. Simply said, contextual information cannot be reduced to a trivial set of data to be accessed by components, but requires some higher-form of organization. Knowledge networks have to provide a *virtual view* of the environment they are to operate in to allow the *concept of interest* to adapt to changing conditions.

2 The Need for a Smart World Infrastructure

Pervasive computing environments consist of multitudes of heterogeneous devices, both stationary and mobile, with different and dynamic changing capabilities and specific ways to access them. One crucial device capability is the ability to communicate and interact with other devices such as in spontaneous networks with changing members due to the communication range (Weis, 2006). The network interfaces are highly heterogeneous ranging from infrared communication to wired connections. Interoperability protocols are tailored to specific requirements as well, e.g. a sensor does not need to implement a complex interoperability protocol but can simply emit its data periodically as events. To summarise, devices interact by forming spontaneous networks using different network interfaces and interoperability protocols. Membership in these networks is temporary and network related properties like communication cost and bandwidth change dynamically. Distributed applications in this scenario are structured into application objects, or services, interacting with each other. Services in turn use device capabilities or further services, which are provided by either the local device, or by remote interaction with other devices. From the application's point of view, one of the main challenges is to use services and capabilities with changing availability. In addition, even a service that is both functional and reachable can become unavailable. Take for example a video presentation system integrated into a Bluetooth equipped PDA. If the user leaves the Bluetooth base-stations wired link area, the video stream becomes unavailable, because the user cannot get a signal. Existing middleware platforms typically address portability of applications via standardised interfaces for remote service interaction, e.g. via stub and skeleton objects, and interoperability of applications across different middleware platforms via interoperability protocols. An autonomic middleware would overcome the more difficult coding against the raw communications APIs provided by generic middleware, thus time to market is reduced and the user experience is improved. Three distinct attributes which can be derived from the core of this body of work are as follows:

1. **Uniform Reduced Instruction Set Programming API:** while classical middleware addresses uniform access to remote services the additional heterogeneity of specialised device capabilities requires similar abstractions, e.g. proxy objects, in order to access different device capabilities in a uniform way independent of the underlying platform. It is also the intention of this autonomic framework to provide more control over an application's adaptive behaviour during its entire life-cycle. Many different sources of information are available to help decide how a system should adapt but this information is spread out over multiple places, times and people making it difficult to retrieve and is usually ultimately ignored. Our architecture intends to organise and disseminate this information between parties in order to enhance the effectiveness of adaptive decisions.

2. **Flexible Protocol Support**: the service model of an autonomic middleware, e.g. remote procedure call or events, is typically reflected in its underlying interoperability protocol, e.g. using request/response messages or emitting event messages. Current devices and systems frequently require the integration of a variety of such service models that are reflected by their correspondent interoperability models. A decoupling of the service model from the interoperability model used by the middleware can help to bridge these interoperability domains. Additionally, this allows different communication paths for the incoming and outgoing messages. For example, in the case of two devices communicating via infrared in order to save energy. If the infrared link breaks due to obstacles or distance and a wireless radio link still exists, communication can continue. This can be either achieved by providing one interoperability

protocol over different network interfaces or by the abstraction of different interoperability protocols that allows flexible usage of existing technologies.

3. **Tailorable:** To be useable on all kinds of devices found in future scenarios, the autonomic middleware has to be tailorable to the device at hand, a mobile device as well as a desktop. The core functionality should be small enough to be executed on a mobile micro-device platform, but easily extensible to use the capabilities of resource richer devices. Two additional requirements similar to tailorable are to accommodate traffic heterogeneity and application heterogeneity. To accommodate traffic heterogeneity, the middleware should provide support for non-streaming protocols and streaming protocols at a continuum of rates. To accommodate application heterogeneity, the middleware should not force a particular application style (such as synchronous over asynchronous) but should be flexible to accommodate varying styles. Adaptation on the fly is also feasible on today's hardware. It is more efficient to compute a different representation on demand rather than storing a set of pre-recorded video representations and switching between them during transmission.

The above requirements can be summarised as *Operational Transparency* and *Performance Transparency*. Operational Transparency means that no action needs to be conducted by users due to host movement. This can be achieved by detecting host movement and performing actions that ensure service continuation at the mobile host's new location. Operational Transparency by itself does not make any quality of service guarantees. Performance Transparency ensures that protocol service should continue with a similar performance should a mobile host migrate to a new location. Methods of ensuring that a framework has performance transparency include optimal routing of packets to and from the mobile hosts, efficient and robust migration procedures and efficient use of network resources such as transmission bandwidth and processing.

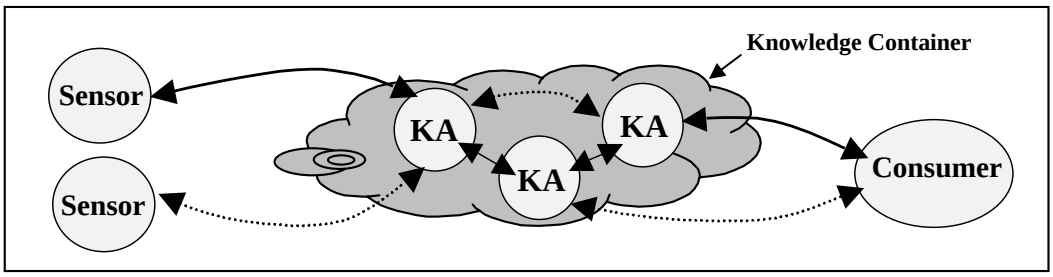

Figure 1: The role of Knowledge Atoms in the infrastructure

One key idea behind our smart world infrastructure is the uniform abstraction of services as well as device capabilities via knowledge atoms and aggregate containers. Consequently, the middleware delivers requests to either device services in the middleware or transport protocols. The infrastructure should allow the flexible integration of new transport plug-ins and device capabilities by simply registering a new entity, which accepts an invocation. This allows the provision of access to all features available on resource-rich computer systems. A key characteristic of the infrastructure is that it should evolve and adapt to changing environments similar to living things in their struggle for survival.

3 Knowledge Networks

A knowledge network is a generic structure that organises distributed knowledge of any format into a system that will allow it to be retrieved efficiently. The rationale of the knowledge network is to act as a middle layer that connects to a multitude of sources, organises them based on various concepts and finally provides well-structured, pre-organised knowledge to individual services and applications. To use the knowledge network we need a querying mechanism to be able to retrieve information. The knowledge network will organise itself in an autonomous manner and it is possible to use the querying mechanism also as part of the knowledge organization mechanism, to autonomously create temporary views that reflect the use of the system. Autonomic service components autonomously achieve self-organization and self-adaptation towards the provision of adaptive and situated communication-intensive services. To achieve this, we need to identify a fundamental, uniform abstraction for situated and autonomic communication entities, at all levels of granularity. This abstraction is called an ACE (Autonomic Communication Element), and it represents the cornerstone of the component model, in which the four driving scientific project principles (situation awareness, semantic self-organization,

self-similarity, autonomic component-ware) may converge (see Figure 1). The main focus of our work is the lightweight organization and request-based provision of knowledge and to this extent the concept of a *knowledge network* has been developed. Figure 2 positions the concept of knowledge networks within the context of the highly distributed knowledge provisioning systems.

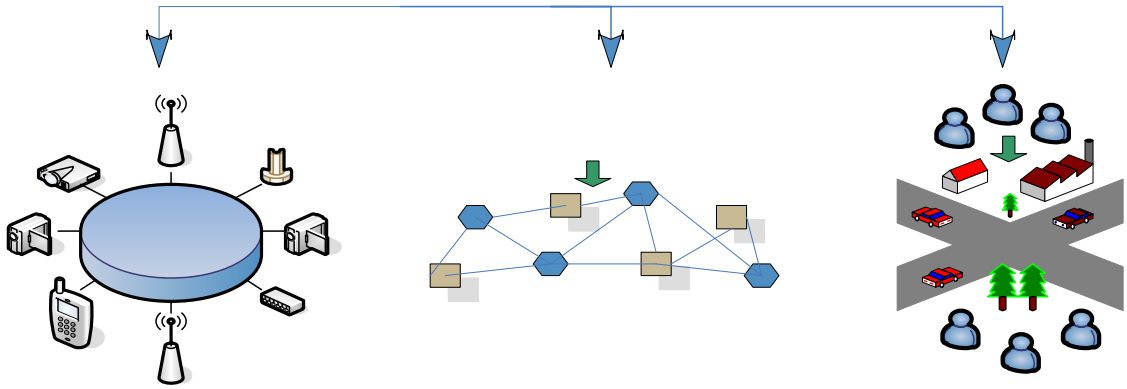

Figure 2: Knowledge Network – A Vision

A knowledge network connects to a data layer that exists, from a knowledge provisioning point of view, below the knowledge network. This layer represents an input layer where knowledge can be registered into the scope of the knowledge network. Once registered, knowledge may be pre-processed, pre-organised, restructured, and enriched with other information. Finally, a dedicated mechanism provides request-based knowledge to individual services and applications. Therefore, the Knowledge Network (KN) itself may be seen as an organizational layer of a more global orientated knowledge provisioning system that is capable of providing well-structured and pre-organised knowledge at different levels of granularity via a dedicated knowledge request layer. The request layer is required to create temporary views of individual parts of the knowledge network without changing any parts of the knowledge network itself. As such, individual views may be generated on a request basis without altering the underlying sources and relationships.

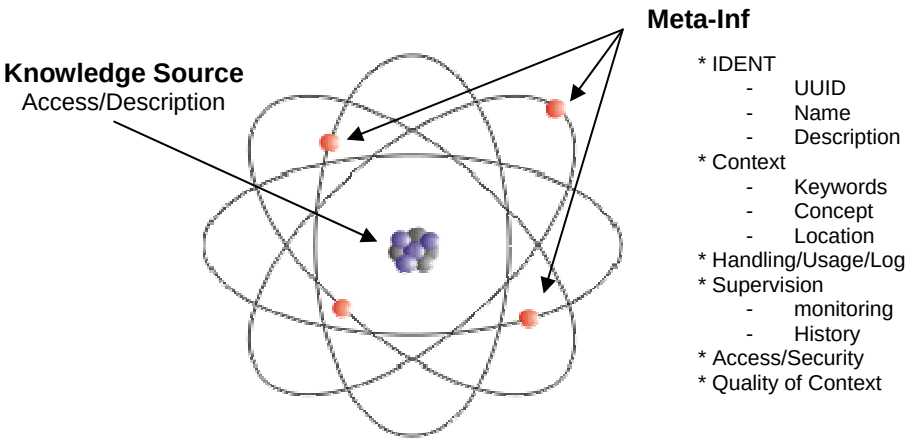

Figure 3: Atom component

The knowledge network itself is made up of only two components, namely Knowledge Atoms (KA), and Knowledge Containers (KC). A knowledge atom is a component that stores a single piece of information, however complex that might be. For example, a KA could represent a primitive type such as an integer value or it could embrace a more complex structure such as the result of a data mining exercise. A knowledge container on the other hand is used to organise knowledge components within the scope of a knowledge network where

such components are all of type knowledge atoms as well as other knowledge containers. Simplified, knowledge atoms provide the knowledge whereas knowledge containers are used to organise the knowledge. A schematic representation of an atom component is depicted in Figure 3 and an atom container component is illustrated in Figure 4.

As shown in Figure 3, KA contains two main components, firstly a knowledge source component, which provides access to distinct knowledge sources independent of their type, location and configuration. Secondly, each KA includes a set of meta-information, which provides additional descriptive information about the knowledge source. Such information may be used for organizational purposes but may also be used for other concepts such as security and access control, supervision methods, location intelligence, context awareness, behavioural analytics as well as a quality control mechanism. In essence the list of meta-information embraced by KA's and KC's is by no means static. Instead the elements itself and their structure should be of dynamic nature based on the atom component itself and the scope it is used for within and without the scope of the knowledge network. Therefore, the set of meta-information available is not strictly specified but seen as a dynamic construct that can be extended at any time.

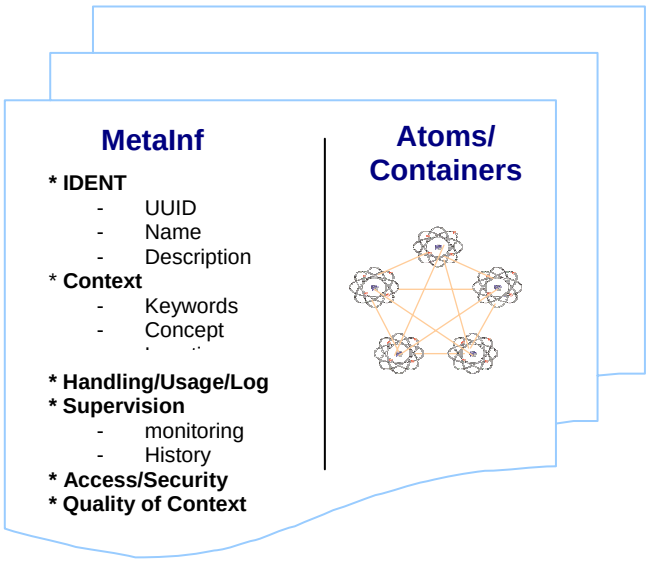

Figure 4: Container Representation

Based on the same underlying concept, a KC also contains two main components. Similar to a KA it also incorporates a set of meta-information, which follows the same concepts as for KAs. As such, individual knowledge can be grouped together based on their value or other concepts as provided through the set of meta-information. Strictly utilising a reference-based approach for this aggregation process allows for the construction of a lightweight network-like structure of any size where individual groups may be further grouped into other concepts. As visualised in Figure 5, this allows for the construction of networks of networks (or super networks), which provide pre-organized knowledge at different levels of granularity.

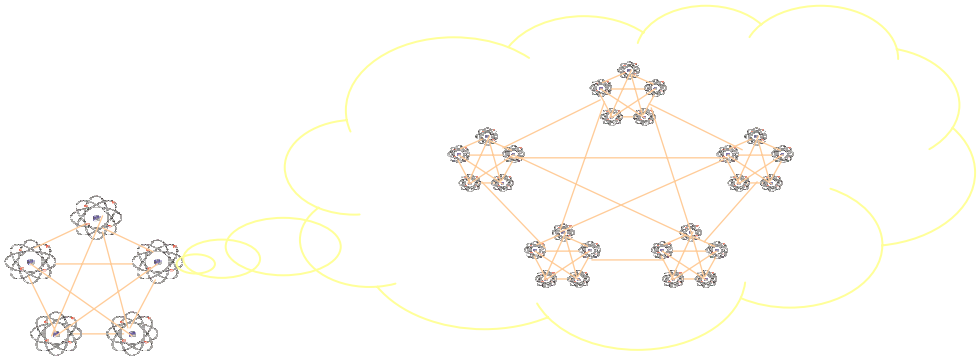

Figure 5: Super Networks

This granularity can be shown by the vertical organization of the network. Information coming from sources may be aggregated together at a higher level in the network to produce new information. This is different to the horizontal organization where temporary views of the source nodes only will be constructed.

4 Knowledge Network Topologies

The goal of an autonomic network is to automatically manage the complexity inherent in the network. Self-configuration within networks is the capability of a system to configure its own network services in response to the environment. Flexibility of location, varied administration control and the requirement to distribute control information globally all contribute to the complexity of the task of autonomic configuration (Melcher & Mitchell, 2004). Thus we can see clearly that routing services are essential to autonomic networks. Designing efficient routing services for autonomic network often needs to consider the properties of specific applications. It is therefore unlikely a fixed routing service in autonomic systems that can effectively meet the requirements of all applications would be implemented.

Ad hoc on-demand distance vector routing (AODV) is a reactive routing protocol designed for ad hoc mobile networks. AODV is capable of both unicast and multicast routing. It is an on demand algorithm, meaning that it builds routes between nodes only as desired by source nodes. It maintains these routes as long as they are needed by the sources. This ultimately leads to lower routing overhead and higher data delivery ratio compared to a table-driven proactive protocol like Optimized Link State Protocol (OLSR) (Das et al., 2000). This paper examines the effects of policies which are suitable for making adaptive decisions upon network packet behaviour at the Knowledge Atom (node) level. It is asserted that both performance gains and necessary functionality of a protocol stack can be achieved by implementing autonomic behaviour in network nodes, but in order to test the validity of this assertion it is necessary to:

- Determine the potential sources of overhead in the implementation of each knowledge atom so as to feed into the design stage of autonomic system level design;
- Determine the benefits of light-weight atoms over monolithic generic atoms in regards to memory space savings in restricted memory mobile devices;
- Determine whether autonomic behaviour provides significant performance gains over one-time network configuration based on initial information;
- Demonstrate that network elements/knowledge atoms can be factored out to achieve performance gains, and synthesis ‘appropriate’ knowledge containers by combining protocol functions;

The primary goal is to provide an infrastructure for building streaming media applications that support interactive mobile multimedia clients receiving optimized synchronized time-based media through light-weight stacks in a heterogeneous distributed mobile environment with fluctuating QoS.

5 Evaluation

The ns-2 simulation environment (NS, 2006) offers flexibility in investigating the characteristics of autonomous systems as it already possesses flexible models for scalable ad hoc autonomic networks. In the ns-2 environment, an autonomic network can be built with many of the same protocols and characteristics as those available in the real world. By leveraging these modules, we added the capability of simulating an autonomic context-aware network. The software includes a library, demultiplexers, multiplexers, broadcast agents and a basic class definition of the context-aware network nodes. The library also includes timer functions, packet manipulation functions, network I/O, buffering and QoS functionality. The basic class defines the interface for the context aware routing service. In this library, the routing service which is one of several packet handlers is implemented by inheritance from the basic class. This routing service must also have an instance of the deployment service in order to send packets.

The primary goal is to provide an infrastructure for building streaming media applications that support interactive mobile multimedia clients receiving optimized synchronized time-based media through light-weight stacks in a heterogeneous distributed mobile environment with fluctuating QoS. A node on an autonomic network needs to be capable of monitoring raw data throughput in order to respond to network variability. Performance measures are generally application dependent, but will typically be a function of responsiveness

and data delivery rate. Hence the simulated knowledge atom component provides the hooks for a monitor stack element to be inserted tailored to an application which has in place means to monitor the data and manage the bandwidth usage of the system when the data requirements of the various media exceed the available bandwidth. Statistics, which can be derived using these classes, are listed in Table 1.

.

• *arrivals this period (pkts)*	• *all arrivals (pkts)*	• *early drops this period (pkts)*
• *total drops (bytes))*	• *drops (pkts)*	• *all arrivals (bytes)*

Table 1: Stream Statistics

Late arrivals of packets may be an indication of some bottleneck in the system, in which case the target can inform the origin about the overload and cause it to scale down the requests. The prerequisite for initiating autonomic led reconfiguration are functions, which allows the system to detect network congestion. This is achieved by monitoring end-to-end delay and packet loss rate. When a packet arrives, a monitor queue object notifies the bandwidth manager object of this event. The bandwidth manager using this information monitors the queue as illustrated in Figure 6. A packet has an expected arrival time and a packet arriving later than expected indicates congestion. We calculate the expected arrival time as the 'logical arrival time' of the previous packet plus the stream period. The logical arrival time is the arrival time observed when bursts are smoothed out, that is, when early packets are artificially delayed such that the stream rate is not exceeded.

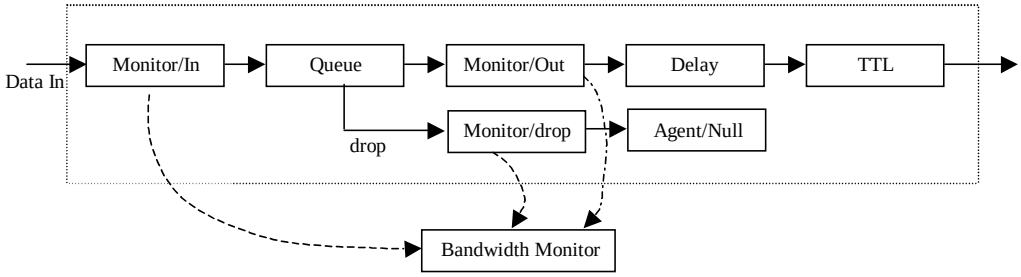

Figure 6: Monitoring requests between Knowledge Atoms

Monitoring packet loss is performed by monitoring the UDP-level latency on each link by sending a 32-byte UDP packet to active participants. If an echo is not received within the maximum ping period or four times the current round trip time estimate, the packet is retransmitted. Using UDP can sometimes lead to loss of messages but this should only lead to a short-term loss of efficiency. A problem arising when only monitoring end-to-end delay is the definition of a threshold value for congestion detection. If a sequence of packets is late (or packets dropped due to buffer overflow), it is assumed that the network is congested. An attempt to minimise the effects of the Hiesenberg Uncertainty Principle (Cassidy, 1992) is done by reading and processing large amounts of data in each cycle with relatively few data rate measurements. This poses the risk of limiting our ability to monitor throughput variations over time thus relying instead on rate averages over more significant time periods. In our case, the effective raw data rate for the system over one read/decode/write cycle is the total amount read (d_T), divided by the sum of the times for the reading (t_r), decoding (t_d), writing (t_w) and monitoring (t_m) of the data. A detailed inspection of packet behaviour in our autonomic network is crucial as requests to knowledge atoms when competing with TCP traffic is severely affected (Widmer, 2000). As the network does not distinguish between normal data packets and knowledge network control packets - beacon messages, and binding requests are likely to be frequently lost when applications/users send requests at a higher rate than the available bandwidth. Knowledge of this is imperative for an acceptable throughput distribution of competing requests. Applying priorities to knowledge atom control packets can be achieved by using two separate queues for data and control packets with a higher priority being assigned to the KA control packet queue.

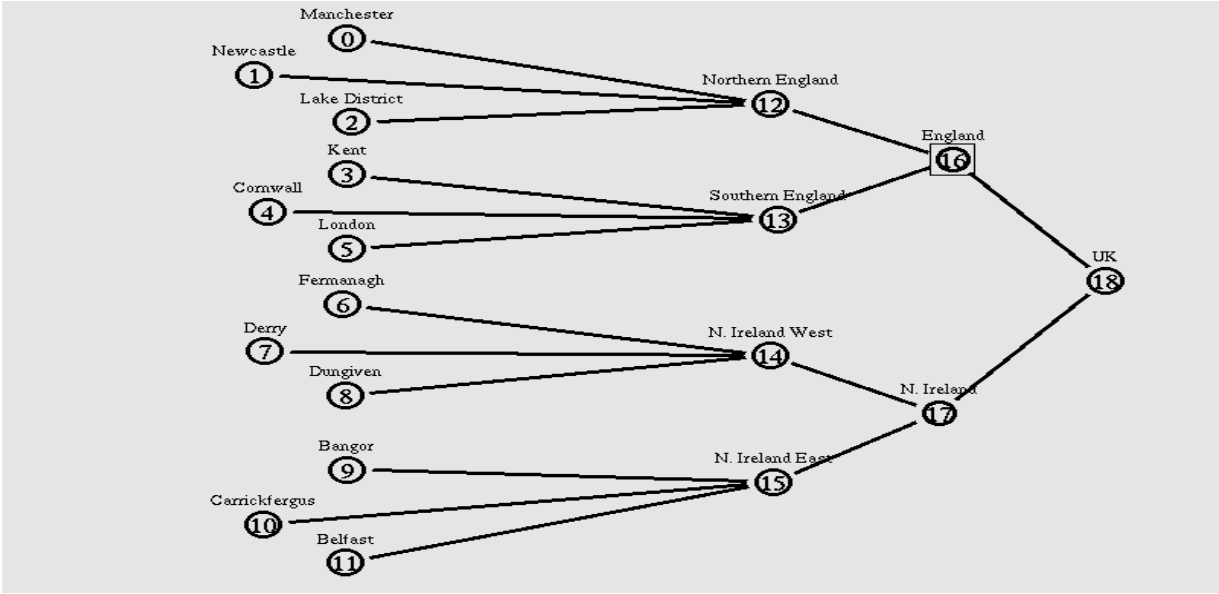

Figure 7: Knowledge Network Weather Example Simulation

Figure 7 illustrates the testbed setup of a simulated knowledge network weather scenario. Figure 8 illustrates the loss of packets when using FTP over TCP sources competing with constant bit rate (CBR) over UDP sources. The duplex-links are defined initially at 2Mb with 10ms delays. The queuing default is droptail. The second series of links (from nodes 12 and 13 to 16; 14 and 15 to 17) are set to 1.7Mb and 20ms. The final links to the higher level weather knowledge atom is 1.1Mb and 15ms.

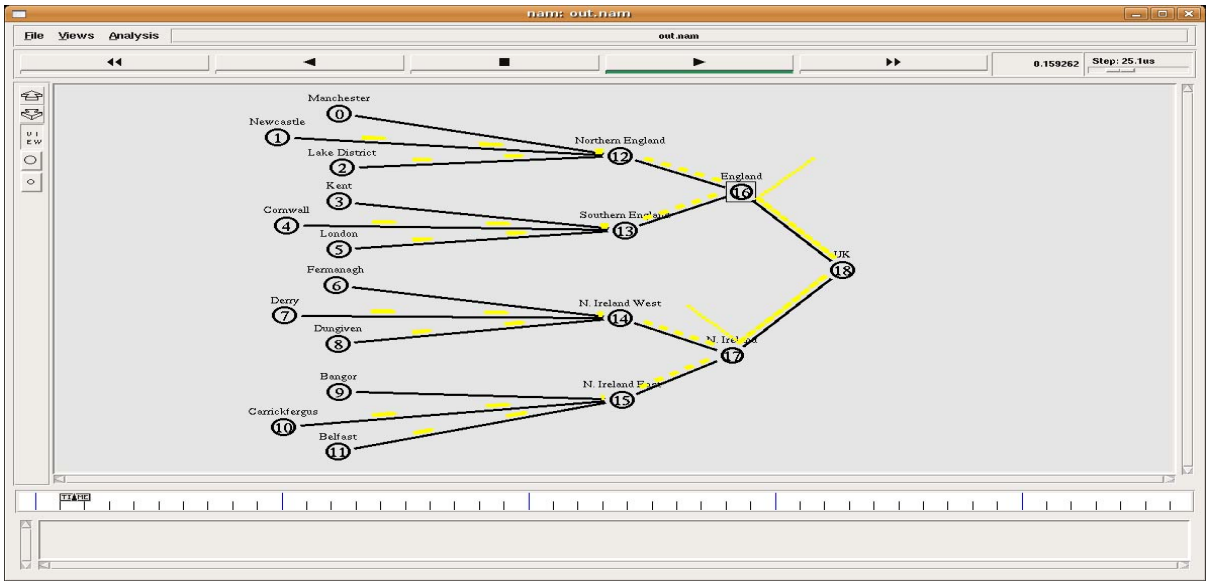

Figure 8: Information flow in Network Animator for Weather Example

Figure 9 illustrates the topology of the smart-world knowledge network where knowledge atoms are mobile. Specific loss characteristics of WLANs include the fact that packet loss rates are highly dynamic and location-dependent. Signal to Noise Ration (SNR) values can quickly drop below the level of 20dB that is generally considered acceptable quality. A major characteristic of a WLAN that distinguishes it from a wired network is that losses occurring on a wired LAN can be generally attributed to congestion and the resultant buffer overflow. In a WLAN, losses are more likely to be as a result of external factors such as interference, alignment of antennae and ambient temperature. The most common problem is arguably fading due to changes in the propagation path as the device moves and interference from other devices in the vicinity. Since the wireless channel is a shared medium, it is important to minimise the amount of feedback from receivers. Simultaneous responses from multiple receivers can cause channel congestion and overload the access point, thereby

hindering the forward transmission of data. Reliability is measured by the bit error rate (BER) and typical BERs on mobile wireless channels range from 10^{-2} to 10^{-3}. Thus achieving acceptable reliability requires powerful error detection and correction techniques. Error control can be applied using forward error correction coding.

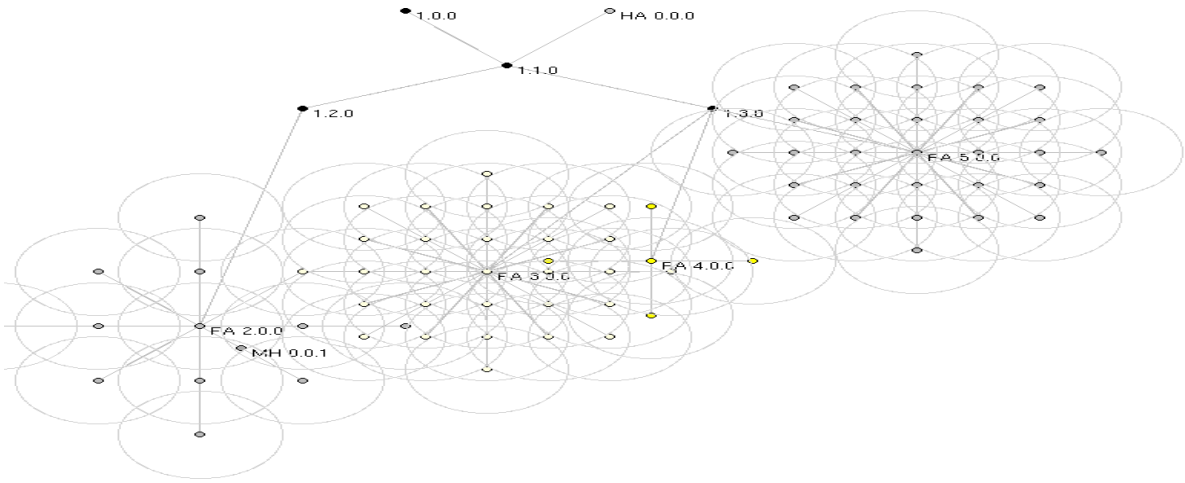

Figure 9: Monitoring node behaviour for mobile knowledge atoms

The goal here is to monitor the knowledge atoms as they are subjected to adverse network conditions. The quality of network packet delivery delivered is quantified in the experiments by using loss rate and adaptive mechanism comparative techniques. The simpler mechanism is the loss rate (meaning both actually lost and packets that arrive) which is a good indication of quality. There is a deliberate policy to intensify interference to emulate behaviour in extremely adverse conditions in order to demonstrate the adaptation mechanism at its most beneficial point. With regards validation of simulated results later on. The NS simulator provides a suite of validation tests that tests TCP congestion control algorithms, queue management, multicast routing and scheduling. In particular, the SACK TCP validation tests -1 was run with the command test-all-sack in ns-2/tcl/test. The TCP validation tests from ns could also be run with the command test-all-sack-v1 in ns-2/tcl/test. Failure rates are simulated through a protocol layer, which duplicates, reorders, and throws away messages. Table 2 shows the default parameters of the NAK protocol. The $_{\text{Retry\_Interval}}$ parameter which is the minimum time interval in between subsequent retransmission requests are sent is set to 2000 ms. The $_{\text{Idle\_Interval}}$ after which the sender starts transmitting idle heartbeats is set to 3000 ms while the $_{\text{Send\_delay}}$ between the sending of subsequent messages is set to 0 ms. These are unchanged throughout with the only varying parameter being the actual epoch message size, which varies from 100 to 900 (see Table 3).

Parameter	(ms)	Parameter	(ms)
Heartbeat_interval: heart beat interval	2000	**Retry_interval**: minimum time interval in between subsequent retransmission requests	2000
Idle_interval: Interval after which the sender starts transmitting idle heartbeats	3000	**Send_delay:** delay set between the sending of subsequent messages	0

Table 2: Default NAK protocol Parameters (unchanged)

Variable Parameter	(bytes)
epochsize: the size of an epoch (default: 500)	100-900

Table 3: Variable NAK protocol parameter

Here we do not filter duplicated request messages as a FIFO layer atop of ACK/NAK performs this. ACK/NAK only makes sure that each message is received at least once. It does not care about duplicates and messages that are out of order or implementation of failure detection. NAK relies only on negative acknowledgements while transmitting an epoch of messages where each epoch has a fixed size of typically 100 to 900 bytes.

	Code Size (bytes)	Code Ratio
Library Code	60460	1.00
Broadcast agent	23468	0.39
Average size of packet handlers of the BA	7823	0.13
Synchronisation agent (4 handlers)	23480	0.40
Average size of packet handlers of the SA	5960	0.10

Table 4: Code size and ratios of classes

An important metric for the framework described here is the sizes of the classes since code size affects communication costs in networks. Table 4 shows the sizes of the compiled classes. We can see that the sizes of the packet handlers from both agents are much smaller than the library code, with a factor of 10. Note that the compiled code of these two agents includes ns-2 code.

6 Related Work

(He et al., 2003) have developed a framework for autonomic routing services for sensor networks. The framework is based on a well-defined parameter space of routing services, and includes a programmable service interface and a deployment module. The programmable interface can implement any routing service that is configured by the deployment module in an automatic and energy efficient manner. The focus however of this framework seems to be on providing a monitoring architecture that is capable of providing information like early warning of sensor node failure along with guidance for incremental deployment rather than automated collaboration in order to perform tasks.

Multicasting (RFC3376, 2002; Phan et al., 2002) has received considerable attention due to the interest in collaboration technologies however; much of the work revolves around either low-level communication protocols or groupware applications, which typically simulate multicast through a series of unicasts. Protocols that build on multicast specially designed for continuous media transmission include RSVP (Siriakkarap et al., 2005). RSVP supports multicasting and resource negotiation appropriate for context-aware computing infrastructures, but no specific implementation mechanisms (e.g. for multicast route set-up or resource reservations) exist thus forcing developers to implement these mechanisms outside the protocol. Group based multicast systems such as described in (Badishi et al., 2006) attempt to address the problems that face mobile devices that are intermittently attached to a network by building a hierarchy as receivers join the multicast group. Servers are connected to other servers higher in the hierarchy, and eventually back to the source with delivery from servers to children through unicast reliable protocols. These systems are more useful for replicated databases or software distribution rather than autonomic computing. Scalability issues also arise, as senders must contact the single sequencer server directly reducing the number of senders that can be supported.

A range of middleware technologies exist, including the Common Object Request Broker Architecture (CORBA) (Vassilopoulos et al, 2006), SOAP (Davis and Zhang, 2005), Enterprise JavaBeans (Vassilopoulos et al, 2006) and DCOM (Inverardi and Tivoli, 2003). CORBA enables objects to interact in a language and platform independent manner through an Interface Definition Language (IDL). An Object Request Broker (ORB) allows clients to issue requests on an object where the ORB locates the object, transmits the request, prepares the object implementation for receiving and processing the request, and returns results to the client. A problem with CORBA is that the architecture adopts a traditional black box approach such that the platform implementation is hidden from the application.

7 Conclusion

A core goal for autonomous systems such as proposed here is automated collaboration in order to perform tasks or share information. The system is always distributed by default and frequently on a large-scale. It can be argued that robustness and economy demand the deployment of a tested autonomic supporting infrastructure

whenever possible. Industrial systems middleware is generally modular and adaptive however it can be demanding of bandwidth and indeed computational resources. Ideally we would like to map the cleanly designed real-time resource constrained embedded systems onto autonomic smart-world supporting infrastructures. This paper is an attempt to investigate the peculiarities of node behaviour in such a network.

Acknowledgments

This work was supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission.

References

Badishi, G.; Keidar, I.; Sasson, A. (2006) Exposing and eliminating vulnerabilities to denial of service attacks in secure gossip-based multicast, Dependable and Secure Computing, IEEE Transactions on , vol.3, no.1pp. 45- 61, Jan.-March 2006

Balkenius, C., Moren, J. (2000) "A Computational Model of Context Processing", 6th International Conference on the Simulation of Adaptive Behaviour. The MIT Press

Bischoff, U., Kortuem, G. (2006) Programming the Ubiquitous Network, A Top-Down Approach. UbiSys 2006 - System Support for Ubiquitous Computing Workshop - At the 8th Annual Conference on Ubiquitous Computing (Ubicomp 2006), Orange County, California, September 17-21, 2006

Buchholz, T., Kupper,A., and Schiffers, M. (2003) "Quality of context: What it is and why we need it", Workshop of the HP OpenView University Association 2003 (HPOVUA 2003), Geneve (CH)

Cassidy, D. (1992) Heisenberg. Uncertainty and the Quantum Revolution. *Scientific American, 266* (May 1992), 106-112.

Davis, A. and Zhang, D. (2005). A comparative study of SOAP and DCOM, Journal of Systems and Software, Volume 76, Issue 2, Pages 157-169, May 2005

Das, S., Perkins, C., and Royer, E. (2000) Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Tel Aviv, Israel, March 2000,

He, Y., Raghavendra, C., Berson, S. (2003). A Programmable Routing Framework for Autonomic Sensor Networks," *Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services (AMS'03),* Seattle, U.S.A., June 2003

Inverardi, P., Tivoli, M. (2003) Deadlock-free software architectures for COM/DCOM Applications, Journal of Systems and Software, Volume 65, Issue 3, Component-Based Software Engineering, pp. 173-183, March 2003

Melcher, B., Mitchell, B. (2004). "Towards An Autonomic Framework: Self-Configuring Network Services and Developing Autonomic Applications". Intel Technology Journal, Vol. 8., No. 4, 2004.

NS (2006) Network Simulator http://www-nrg.ee.lbl.gov/ns/

Phan, T., Zorpas, G., and Bagrodia, R. (2002). An Extensible and Scalable Content Adaptation Pipeline Architecture to Support Heterogeneous Clients. Proceedings of ICDCS 2002 - The 22nd International Conference on Distributed Computing Systems, July 2-5, Vienna, Austria, 2002

RFC 3376 (2002) B. Cain, S Deering, W. Fenner, I Kouvelas, A. Thyagarajan, Internet Group Management Protocol, Version 3, RFC 3376.

RFC 761 (1980) Transmission Control Protocol (TCP). http://www.ietf.org/rfc/rfc0761.txt?number=761

RFC 768 (1980) User Datagram Protocol (UDP). http://www.ietf.org/rfc/rfc0768.txt?number=768

RFC 791 (1981) Internet Protocol (IP). http://www.ietf.org/rfc/rfc0791.txt?number=791

Saber, M., Mirenkov, N. (2003) A Multimedia Programming Environment for Cellular Automata Systems. DMS'2003 - The 9th International Conference on Distributed Multimedia Systems, Florida International University Miami, Florida, USA, September 24-26, 2003

Siriakkarap, C. Setthawong, P. Tanterdtid, S. (2005) RSVP Based Critical Services Path Recovery in MPLS Network, Information and Telecommunication Technologies, 2005. 6th Asia-Pacific Symposium, 9th-10th November 2005

Solon, T., McKevitt, P., and Curran, K. (2003) Telemorph - Bandwidth determined mobile multimodal presentation. IT&T 2003 - Information Technology and Telecommunications
Letterkenny Institute of Technology, Co. Donegal, Ireland. 22-23rd October, 2003

Sterritt, R., Mulvenna, M., and Lawrynowicz, A. (2004). "Dynamic and Contextualised Behavioural Knowledge in Autonomic Communications", Proc. of the 1st IFIP Workshop on Autonomic Communications. Berlin: Springer-Verlag

Vassilopoulos, D., Pilioura, T., Tsalgatidou, A. (2006) Distributed Technologies CORBA, Enterprise JavaBeans, Web Services — A Comparative Presentation, pp. 280-284, 2006.

Weis, T., Handte, M., Knoll, M., Becker, C. (2006) Customizable Pervasive Applications. PerCom 2006, pp: 239-244