A Quality of Service Framework for Adaptive and Dependable Large Scale System-of-Systems

Peter Bull
Department of Computer Science
Loughborough University
Loughborough, UK
p.bull@lboro.ac.uk

Alan Grigg
BAE Systems,
SEIC
Loughborough, UK
a.grigg@lboro.ac.uk

Lin Guan
Department of Computer Science
Loughborough University
Loughborough, UK
l.guan@lboro.ac.uk

Iain Phillips
Department of Computer Science
Loughborough University
Loughborough, UK
i.w.phillips@lboro.ac.uk

Abstract *- There is growing recognition within industry that for system growth to be sustainable, the way in which existing assets are used must be improved. Future systems are being developed with a desire for dynamic behaviour and a requirement for dependability at mission critical and safety critical levels. These levels of criticality require predictable performance and as such have traditionally not been associated with adaptive systems.*

The software architecture proposed for such systems is based around a publish/subscribe model, an approach that, while adaptive, does not typically support critical levels of performance. There is, however, the scope for dependability within such architectures through the use of Quality of Service (QoS) methods. QoS is used in systems where the distribution of resources cannot be decided at design time. A QoS based framework is proposed for providing adaptive and dependable behaviour for future large-scale system-of-systems. Initial simulation results are presented to demonstrate the benefits of QoS.

Keywords: Adaptive Systems, Network Reliability, Publish/Subscribe, Quality of Service.

1 Introduction

There is currently much UK government and industry thinking towards the integration of complex computer-based systems, including those in the military domain. Such systems include applications of high safety criticality and must, therefore, be capable of providing the necessary predetermined levels of performance. Current systems requiring such assurances of performance are mostly based on parameters and system states decided during design time, thus allowing a predictable estimate of performance. The ability to dynamically reconfigure systems at run-time would, however, lead to increased flexibility and adaptability. These properties would allow for the better use of existing assets and more sustainable expansion of system functionality.

In section 2 of this paper the software architectural needs of future large-scale systems are examined. Sections 3 and 4 investigate how through the choice of software architecture and use of Quality of Service methods a framework can be developed that supports the objectives of both adaptability and dependability. Section 5 concludes by detailing initial simulation results from this QoS negotiation framework.

2 Future large-scale systems

The following two system-of-systems are examples of projects that illustrate the objectives driving this work and show how they apply to both the higher level integration of platforms and lower level component integration.

2.1 Network Enabled Capability

Network Enabled Capability (NEC) [1], illustrated in Figure 1, is a UK Ministry of Defence (MoD) project aimed at the integration and collaboration of assets through the exploitation of modern networking technologies. At a basic level this refers to the networking of every vehicle, database or sensor, etc. forming a system-of-systems that can then be exploited to achieve new or enhanced functionality, only possible as the product of such collaboration.

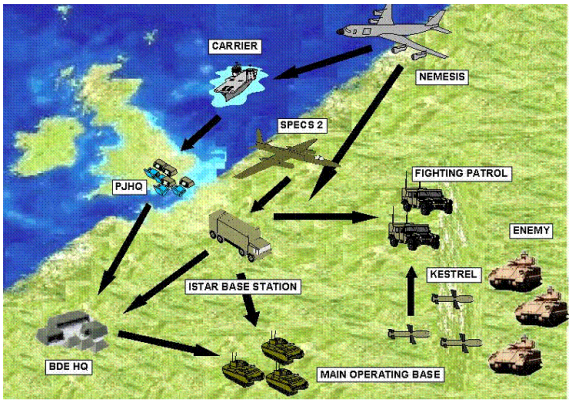

Figure 1. Illustration of an NEC system [2].

Research conducted into NEC, such as that produced by the NECTISE (Network Enabled Capability Through Innovative Systems Engineering) project [3], places its focus on Service Oriented Architectures (SOA) as a potential solution to the software architecture needs of NEC, while Wang et al. [4] suggest the use of the Data-Centric Publish/Subscribe architecture, the Data Distribution Service (DDS).

2.2 Integrated Modular Avionics

The Integrated Modular Avionics (IMA) architecture, [5] is a safety critical, reconfigurable, modular approach to avionics systems used in both civil and military domains.

The IMA software architecture [6] is comprised of "Application", "Operating System" and "Hardware" layers, forming a three layer model. The separation of the architecture into these layers allows for abstraction and transparency between components, be it hardware or software based. The abstraction found within this architecture aids the assurance of safety critical operation through the spatial and temporal partitioning of elements.

Where the military IMA architecture concept [7] differs to the civil is in the addition of blueprints to the model. Blueprint documents are used to configure the system state (e.g. communication channels and which applications are running) and switch it between operational modes (e.g. standard flight and enemy engagement). These documents are currently created during design time due to the extensive verification and validation required to ensure their correctness. This means that in practice only a small number of blueprints exist for each aircraft and as such the system is only capable of switching between these few predefined configurations.

Investigative work, detailed by Ford et al. [8], is being conducted into how IMA could be made more adaptive while maintaining safety critical levels of performance. This includes an assessment of open software architectural approaches and particular focus is given to the Data Distribution Service (DDS) with future work said to focus on its use within highly dependable systems.

3 Publish/Subscribe architectural approaches

The example systems in section 2 both suggest the use of a publish/subscribe software architecture model as a means of supporting adaptive behaviours. The two approaches suggested for these are Service Oriented Architectures (SOA) and Data-Centric Publish/Subscribe (DCPS). These approaches differ in that SOA places focus on the invocation of functionality whereas DCPS is centred on the sharing of data. This paper shall focus on DCPS due to the availability of mature open standards. Particular consideration must be given to the support of dependability within such architectures.

3.1 Data Centric Publish Subscribe

DCPS architectures follow the publish/subscribe model closely. This model, as discussed by Gehlot et al. [10], has the following stages:

1. A publisher announces itself to the middleware, and its details are recorded.
2. A subscriber requests the fulfilment of a service from the middleware.
3. Wherever possible the middleware matches this request to the details of a publisher held within its records and replies with the location and interface details of this publisher.
4. The subscriber contacts the publisher to request service fulfilment.

The Data Distribution Service (DDS), as described by Pardo-Castellote [9], is an Object Management Group (OMG) standard for a real-time DCPS system architecture.

A client application places a subscription to a topic of information (for example temperature readings or GPS coordinates), which is then matched to a publisher capable of dispersing data relevant to that topic.

Each node within the system maintains a record of the available publishers and the subscriber information relevant to them. Data is separated into domains in order to minimise the amount of data held by each node and increase scalability.

3.2 Publish/Subscribe and Quality of Service

Within an adaptive system where system elements join and leave in an ad-hoc manner it will not always be possible to provision adequate resources for all situations and therefore, periods of high load will occur causing unpredictable and varying delays. This can create serious problems for delay sensitive applications. It is therefore necessary to find some form of compromise with regards to resource utilisation. Quality of Service (QoS) is a blanket term used to describe the specification and process of ensuring an acceptable level of performance between two parties.

DDS makes use of QoS methods during the set up of data provision. Data readers declare their interest in a topic and the associated QoS properties that they require. The data writer then checks for compatibility between this request and the stored record of QoS characteristics available to form a contract between the two entities. The support for QoS characteristics greatly increases the suitability of DDS for those dynamic systems requiring predictable performance.

3.3 Framework design choice

To start to develop a framework with which to support dependability in adaptive systems it is necessary to choose an underlying software architecture to focus on. For this purpose DCPS, and specifically DDS, has been chosen. This is due to the fact that it has been suggested for use in the types of systems that this project is investigating and is one of few such standards that have been developed with dependability in mind.

Following a specification such as DDS in the development of supporting methods would provide a well tested and evaluated means to base the design on. This shall, however, only be used as a reference given that the proposed framework will need to go beyond the functionality that currently exists.

To investigate further into the development of an adaptive and dependable system framework a discussion is necessary as to the Quality of Service methods that will be employed and the issues that such systems might face.

4 Quality of Service

For a system to make use of Quality of Service methods there are three main elements that must be addressed. These are: the definition of a QoS language with which to communicate, the subsequent negotiation process and system wide optimisation.

In the search for an optimal set of services that will maximise the possible value within a system, given a set of resource constraints, it could be foreseeable that the computational time required for such a calculation could soon become prohibitively high as the scale of the system increases. Considering the NEC example, the system could potentially be reconfiguring on a frequent basis as new nodes enter or leave and with only a small window of opportunity for communication (for example if a vehicle is passing briefly within range, relaying data). Both of these factors mean that there is an additional objective of keeping the QoS negotiation process as simple and stable as possible. Given the changing scale of future-systems such as those in section II the main resource constraint likely to be experienced is that of the communication bandwidth. This shall therefore be the focus of the QoS process.

The following sections analyse the three main elements of QoS methods from the perspective of an adaptive and dynamic system, bearing in mind the examples from Section 2.

4.1 QoS characteristic definition

The first step necessary for a system to make use of QoS methods is the definition of the required performance characteristics. Applications may be developed across boundaries (be it departmental, organisational, governmental, etc.) and if they are to participate in the same system they need a common language with which to communicate.

For the framework the following QoS characteristics have been chosen. For the subscriber:

- *Latency (L)* – the deadline within which data samples must be received
- *Time Based Filtering (TBF)* – the minimum time between samples received in milliseconds
- *Reliability (R)* – 'best effort' (data is sent unacknowledged) or 'reliable' where data is acknowledged upon receipt and lost packets are retransmitted (providing they are still within the latency allowed)

For the publisher:

- *Time Based Filtering (TBF)* – The amount of time in milliseconds between data samples.
- *Reliability (R)* – as subscriber
- *Sample Size (SS)* – the size in bytes of each data sample transmitted.

With the exception of the publisher 'Sample Size' characteristic these are a subset of the DDS set that have the most impact on network resource usage.

For any negotiation more complex than simply accepting requests if performance criteria match (otherwise rejecting) to take place, applications need to be flexible in their requirements. This means that, where possible, an application should provide a range of performance criteria with which it could function. Abdelzaher et al. [10] give an example of using application developer specified QoS levels. This allows the application a number of predefined levels of operation.

For a greater degree of flexibility over predefined QoS levels, however, and to reduce the overhead of transmitting what could be a high number of levels the framework shall instead use minimum, maximum and interval values. The interval value allows the developer to control the number of levels possible and can be used to specify the sensitivity of the application, decreasing unnecessary network load where possible. For this purpose the TBF subscriber QoS characteristic shall be specified with a minimum, maximum and interval value.

In addition to the definition of QoS characteristics, there is a need for a common understanding or assurance that each application will only request the resources that are actually required. It could be foreseeable that a developer may erroneously view their application at an inconsistent level of importance in relation to others within the system.

As the dynamic behaviour and scale of a system increases the use of a human system for verifying QoS properties becomes increasingly impractical. Solely using a formulaic approach to calculating a services value may, however, not truly reflect its importance as this is found from the result as viewed by the end user, not the level of resources it takes to complete it. Combining a calculated value with a developer defined priority found from a set of subjective guidelines, would provide a potential solution.

A discussion of methods available for calculating the value of a service is given by Burns et al. [11]. This approach known as value based scheduling is designed for scheduling processes within an onboard real-time system but the approach would seem to hold true for inter-platform communication. Where this approach differs to the approach necessary for this work is that it focuses on the selection of service fulfilment from a known set of alternatives (e.g. the service could require a collision avoidance mechanism and the choice could be between an infra-red beam deflection and RADAR). It is assumed for the framework that a subscriber will have one possible data type required from a publisher. Publishers of this data type may vary in their TBF value or reliability but the data received (and sample size SS) will always be of the expected format.

When deciding on a value function for the framework it is necessary to make assumptions about the properties

that a service of high priority would have. A service could be said to be more important if it requires a low latency, high rate of data samples value and reliable transmission. While the sample size will affect the resources required for transmission it is not necessarily a sign of importance in the system. A video stream for example will require more network resources but would not necessarily be more important than a signal from a temperature sensor. A function is required that weights these attributes accordingly. The exact weighting will vary between systems and a very general case has been assumed here.

Given that the TBF value specifies in milliseconds the amount of time between data samples the sample rate (U) is found in (1).

$$U = \frac{1000}{TBF} \quad (1)$$

Placing exact values on the preference between reliable and best effort service in a real system requires extensive evaluation of the applications that will run within. For this example and for further work it is assumed that a service requiring reliable communication will be twice as valuable to the system as one that requires best effort communication only. A value of 0.5 is therefore assumed for best effort service and 1 for reliable. It is assumed that the value of the latency is linear and will affect each of the data samples. Given these assumptions the value (V) of a service shall be calculated using (2).

$$V = R.\left(\frac{U}{L}\right) \quad (2)$$

To ensure that a sufficient range of integer values exist to reflect the number of services within a system the resulting value V shall be multiplied by a constant k.

Burns et al. [11] also suggest calculating value both offline and online. Online analysis amends this reward value based on the actual performance of the service. A service may have a high priority but if the actual performance falls short of the ideal then its value will be decreased. Observed values for a service instance S_i are recorded for the number of data packets transmitted which did not meet the latency allowed (l_i), the number of timely and accurate transmissions (g_i) and the number of timely but inaccurate transmissions (p_i). Note that l_i and p_i are negative.

These values combined help to give an indication of the actual reward possible given real network conditions. Given that $P_{n,i}$ is the probability of l_i occurring, $P_{c,i}$ is the probability of g_i occurring and $P_{e,i}$ is the probability of p_i occurring, (3) is used as the online value function.

$$V_i = l_i P_{n,i} \,.\, g_i P_{c,i} \,.\, p_i P_{e,i} \quad (3)$$

A similar method of calculating the value of a service is proposed by Liang et al. [12]. This method referred to as robust service selection is used to account for the actual probability of a service being fulfilled given the current system constraints (system size, network performance, etc.).

It is assumed that the value of a service is independent to that of others. This means that the value of two services running is the sum of the individual value of each service. A calculation of the values assigned based on all the different permutations possible would be too complex to calculate within the amount of time available.

Note that it is expected that there will be two classes of applications within the system. A higher, safety critical class, and a lower class requiring varying but non-safety critical levels of service. It is assumed that publisher pairings and resource requirements for the higher class shall be defined offline. Including these services in the QoS negotiation process would likely prove detrimental to their performance.

4.2 The QoS negotiation process

To ensure that resources within a dynamic system are being best utilised in any given state and to provide assurance of performance beyond that of any best-effort method QoS negotiation must take place.

Negotiation can be used at several points within the system, each contributing to the level of dynamic behaviour. Negotiation could occur solely at system start-up, taking into account any changes to the system from its design-time state. This alone would introduce an adaptive aspect not currently seen within most dependable systems. To fully take advantage of an adaptive environment such as that described in the NEC system example, however, this negotiation must also take place at run-time.

The following are examples of negotiation techniques.

4.2.1 Priority based negotiation

The simplest method of differentiating between the criticality of services is through a priority based system. This involves assigning a priority from a finite set of possible values to a service. This assignment can then be used to create an ordered list of services. If the system were to reach a point where the resources available were not sufficient then the lowest priority service would be degraded where possible (and discarded otherwise) in favour of higher priority services. This approach is typical for most resource reservation techniques including the network based IntServ and DiffServ models [13].

The main problem with this approach is with the assignment of priorities. As previously discussed, within future systems there is a need for a method of accurately expressing a services value both subjectively and objectively. They do, however, offer an advantage in that they can be statically analysed to predict behaviour or prove certain performance properties.

4.2.2 Reward/Penalty based negotiation

An alternative to priority based negotiation is the reward and penalty method described by Abdelzaher et al. [10]. This method uses reward and penalty values assigned to each task as a way of ensuring that the maximum utility is provided by the system.

Taking the example of a new service entering the system while it is running. The negotiation process will first add the new service to the list of running services to determine if there are adequate resources available to meet the resource requirements of the new list. If there are, then the list is used to allocate resources and the process ends. If there are not adequate resources, however, then the system searches for the service that is running that when degraded to its next lowest level of QoS would result in the least drop in total system reward (calculated as the sum of the reward values associated with each service running). It then checks to see if this degradation will allow the new process to run. If it will not, then the search continues in the same way until there are adequate free resources to run the new task. If the introduction of the new service and its associated reward now result in a greater or equal new total system reward than was previously seen then the new list is accepted. If it does not, then the system checks to see if the penalty for not including it is greater than the difference of rewards between system configurations. If it is then the service is scheduled.

4.2.3 Framework design choice

The reward/penalty method of negotiation is chosen for the framework given its ability to support both the subjective and objective assignments of value. The reward shall thus be calculated using objective data and the penalty shall be assigned by the developer. Some adaptations will still be necessary to make it suitable for the future systems in question. The framework negotiation algorithm will work as follows.

A subscriber sends a request for data to the publish/subscribe middleware instance on its local node. The middleware checks for a local compatible publisher. If one or more are found then the publisher that best matches the QoS requirements of the subscriber is chosen for use. If no compatible publisher is found then the middleware checks nodes connected by network link. A list is compiled of compatible publishers returned. Preference is given to wired links given that they are less prone to interference and any connected nodes are likely to be less mobile. Preference is also given to those publishers on nodes that have the most free resources. Given these two criteria the list is ordered and publishers are checked in sequence to see if the node in which they are based can accommodate them. To check this the middleware manager containing the publisher in question compiles a list containing the new subscriber and all current subscribers being serviced. If it is judged that adequate resources are not available to service this new list then each entry is checked to see which can be degraded to result in the smallest decrease in reward. This is repeated until adequate resources are available. Once this has completed the difference in reward between old and new lists is compared. If the amount of reward has increased then the new list is accepted. If it has decreased less than the penalty value then it is accepted otherwise it is rejected and the next available publisher is checked.

Note that this QoS management will also involve some policing to ensure that QoS is being met. If QoS is not being met then the resources available should be recalculated and the list of running services renegotiated.

4.2.4 System wide optimisation

One of the main advantages of large scale distributed systems is the redundancy provided by having multiple instances of the same services available from multiple locations. For a distributed system to be said to be truly making the best use of resources a level of system wide optimisation is necessary. As described by Abdelzaher et al. [10], this typically involves poling nodes with repeating publishers to see if the total level of reward provided by the system can be increased by changing the node in which a client is receiving its service from. The main problems with this approach are that they introduce yet further renegotiations and therefore disruptions to system operation and in systems where nodes are frequently transient, the swapping of services between nodes can lead to an improvement in performance in the short term but ultimately prove detrimental.

To first address the problem of transient nodes assumptions should be made based on observed behaviour. If a node has been recognised as being present within the system for a predetermined length of time (perhaps purely as a consumer of services) it would be reasonable to assume that its presence will continue for a sufficiently long period for it to be deemed a useful source of services. Determining this type of information requires the sharing of observations amongst nodes.

5 Simulation results

A simulation has been developed using MATLAB to experiment with the framework proposed within this paper. The simulation is based around an NEC type scenario of nodes physically distributed within an environment and with differing resource and functional capabilities. A random network topology is set up based on a seed input and each node is populated with publishers and subscribers. Each publisher and subscriber has a set of QoS characteristics matching that described within the framework.

Network links are either wired or wireless. Wireless links have a signal strength which (along with a small error to account for signal noise) affects which nodes are within communicable distance. An assumption made is that communication between nodes is made directly. This could be adapted in the future as a node relaying data could treat this as a request for service for which a reward would be associated. The level of reward associated would need to decrease as the number of nodes through which the data is passing increases. This is due to the increased consumption in resources in comparison to the reward being gained.

For the purpose of this simulation it is assumed that requests within the system are received consecutively with

only one node dealing with a request at any one time. Future versions of the simulation will adapt this, however, as the order in which requests are received affects the load on a network link and may therefore alter which publisher's receive preference in the negotiation process.

Initial tests have compared the framework described within this paper to one that did not use QoS negotiation and instead matched compatible publishers and subscribers based on their highest possible QoS characteristics. This means that if adequate resources are not available at the time of inquiry then a service is rejected. The result of this is shown in Figure 2.

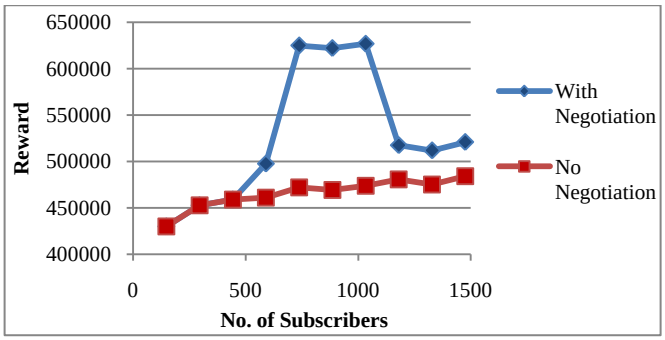

Figure 2. A comparison of reward values gained using reward/penalty negotiation or simple compatibility testing.

The reward/penalty negotiation technique and the use of flexible levels of QoS shows a clear advantage over simple compatibility testing. The reward accrued from both techniques is the same until the system starts to reach high load. After this the ability of the negotiation technique to adapt to the restricted resources starts to show benefit. The drop in reward in the negotiation data series after around 1000 subscribers is due to the network links reaching high load and services with high penalties displacing existing subscribers, thus resulting in a perceived drop in reward but an actual increase given the preference expressed by the developer. The slight variations seen in both data series is due to the introduction of subscribers with high sample rates.

6 Conclusion and future work

This paper has described the issues surrounding dependable large scale adaptive system-of-systems. A QoS negotiation framework has been proposed that combines existing methods of providing a flexible software architecture, adapting these where necessary to suit future system-of-systems and increase dependability. This includes: increasing the flexibility of the system through the introduction of varying levels of QoS, offline and online system reward calculation , and adapting negotiation techniques for future large scale systems.

Initial simulation results have been shown that demonstrate the benefit of a negotiation process in the allocation of resources.

Future work shall focus on the further implementation of the simulation, extending it over time to show the performance of the framework in a changing environment. Beyond this simulation results shall be verified through implementation on a test bed.

Acknowledgment

This work was supported by the Engineering and Physical Sciences Research Council, BAE Systems and the Royal Society, UK.

References

[1] Ministry of Defence. *Network Enabled Capability, JSP 777.* Network Enabled Capability: http://www.mod.uk/DefenceInternet/AboutDefence/CorporatePublications/ScienceandTechnologyPublications/NEC/Jsp777NetworkEnabledCapability.htm, 2005.

[2] Ministry of Defence. *OV-1a High-Level Operational Concept Graphic.* Retrieved April 17, 2008, from MODAF: http://www.modaf.org.uk/images/109.gif, 2007, April 4.

[3] Russell, D. J., & Xu, J. Service Oriented Architectures in the Provision of Military Capability. *University of Leeds*, 2007.

[4] Wang, N., Schmidt, D. C., Hag, H. & Corsaro, A. Toward an Adaptive Data Distribution Service for Dynamic Large-Scale Network-Centric Operation and Warfare (NCOW) Systems. *Proceedings of the Military Communications Conference 2008*, pp. 1-7, 2008.

[5] Prisaznuk, P. J. Integrated Modular Avionics. *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference*, pp. 39-45, 1992.

[6] Airlines Electronic Engineering Committee. ARINC Specification 653: Avionics Application Software Standard Interface, 1996.

[7] Ministry of Defence. *ASAAC Standards Part 1.* ASSC - Standards & Guidance Support for the UK Military: http://www.dstan.mod.uk/data/00/074/01000200.pdf, 2005.

[8] Ford, B., Bull, P., Grigg, A., Guan, L. & Phillips, I. Adaptive Architectures for Future Highly Dependable, Real Time Systems. *7th Annual Conference on Systems Engineering Research*, 2009.

[9] Pardo-Castellote, G. OMG Data Distribution Service: Architectural Overview. *23rd International Conference on Distributed Computing Systems Workshops Proceedings*, pp. 200-206, 2003.

[10] Abdelzaher, T., Arkins, E., Shin, K. *QoS Negotiation in Real-Time systems and its Application to Automated Flight Control.* 1997.

[11] Burns, A., Prasad, D., Bondavalli, A., Di Giandomenico, F. , Ramamritham, K., Stankovic, J., Strigini, L. The meaning and role of value in scheduling flexible real-time systems, *Journal of Systems Architecture* , pp. 305-325, 2000.

[12] Liang, Q., Wu, X., & Lau, W. C. Optimizing Service Systems Based on Application-Level QoS. *IEEE Transactions on Services Computing*, pp. 108-121, 2009.

[13] Xiao, X., & Ni, L. Internet QoS: A Big Picture. *IEEE Network , 13* (2), pp.8-18, 1999.