A quality of service framework for dependability in large-scale distributed systems

A Quality of Service Framework for Dependability in Large-Scale Distributed Systems

Peter Bull, Lin Guan, Iain Phillips
Department of Computer Science
Loughborough University
Loughborough, UK
p.bull@lboro.ac.uk, l.guan@lboro.ac.uk,
i.w.phillips@lboro.ac.uk

Alan Grigg
Systems Engineering Innovation Centre
BAE Systems
Loughborough, UK
a.grigg@lboro.ac.uk

Abstract—As recognition grows within industry for the advantages that can be gained through the exploitation of large-scale dynamic systems, a need emerges for dependable performance. Future systems are being developed with a requirement to support mission critical and safety critical applications. 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 takes its properties from the service-oriented computing paradigm and the communication model follows a publish/subscribe approach. While adaptive, such architectures do not, however, typically support real-time levels of performance. There is scope, however, 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. In this paper a QoS based framework is proposed for providing adaptive and dependable behaviour for future large-scale dynamic systems through the flexible allocation of resources. Simulation results are presented to demonstrate the benefits of the QoS framework and the tradeoffs that occur between negotiation algorithms of varying complexities.

Keywords-Quality of Service; Dependability; Network Enabled Capability; Negotiation;

I. INTRODUCTION

There is currently much UK government and industry thinking towards the integration of complex computer-based systems, including those in the military domain. Through this integration large-scale dynamic systems will be formed, creating new possibilities of application and new opportunities for those already existing. 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. This work extends that presented in the author's previous work [1] to include additional and updated framework design details, simulation results and discussion.

In section II of this paper the software architectural needs of future large-scale systems are examined. Sections III and IV 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 V details a set of simulation results from this QoS negotiation framework. Section VI concludes, discussing how simulation results will be validated through the use of a real-time system test-bed.

II. FUTURE LARGE-SCALE SYSTEMS

The following system is an example project that illustrates the objectives driving this work and shows how they apply to the higher level integration of platforms (note that they can also apply to lower level component integration).

Network Enabled Capability (NEC) [2], illustrated in Fig. 1, is a UK Ministry of Defence 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 vehicles, databases or sensors, etc. The system 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 [3].

Research conducted into NEC, such as that produced by the NECTISE (Network Enabled Capability Through Innovative Systems Engineering) project [4], places its focus on Service Oriented Architectures (SOA) as a potential solution to the software architecture needs of NEC. Wang et al. [5] suggest the use of the Data-Centric Publish/Subscribe architecture, the Data Distribution Service (DDS). This paper shall focus on DCPS, and specifically DDS, due to its availability as a mature open standard. Particular consideration must be given to the support of dependability within such architectures.

III. Data Centric Publish/Subscribe and Quality of Service

The Data Distribution Service (DDS), as described by Pardo-Castellote [6], 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.

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. Doing so would be inefficient as it would require designing systems for all worst-case eventualities. It can therefore be expected that periods of high load could occur, causing unpredictable and varying delays. This can create serious problems for delay sensitive real-time applications. It follows that it is necessary to find some form of compromise with regards to resource utilisation.

Quality of Service (QoS) is a 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 setup of data provision, checking for compatibility between the requested and available QoS characteristics, forming a contract between the two entities. Note that DDS currently performs no negotiation as to which subscribers should be provided data from a publisher beyond this compatibility testing. The support for QoS characteristics does, however, greatly increase the suitability of DDS for those dynamic systems requiring predictable performance.

To start developing a framework with which to support dependability in adaptive systems the DCPS software architecture, and specifically DDS has been chosen as a point of focus. 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. DDS 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.

IV. Quality of Service

As Bouyssounouse & Sifakis [7] discuss, there are many elements of a system that must be addressed in order for it to be QoS aware, from application level specification to lower level network protocols. This project has chosen to address these issues largely from the application level and two main elements have been chosen accordingly. These are: the definition of a QoS language with which to communicate and the subsequent negotiation process.

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 the previous example, NEC, 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 at the application level from the perspective of an adaptive and dynamic system, bearing in mind the Network Enabled Capability example from section II.

A. 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 been deemed to have a high impact on network resource usage. It is assumed that the 'Sample Size' can be inferred by the data type and as such is not explicitly specified by the subscriber and is done so here for the publisher to aid clarity.

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 within which it could function. Abdelzaher et al. [8] 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.

1) Service Value Calculation

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 a level of importance that is inconsistent 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.

Burns et al. [9] also suggest calculating value both offline and online. Online analysis amends the reward value based on the performance of the network. A subscriber may have a high priority but if the actual performance it receives falls short of the ideal then its value will be decreased.

a) Offline Value Calculation

A discussion of methods available for calculating the value of a service is given by Burns et al. [9]. This approach, known as value based scheduling, is designed for scheduling processes within an onboard real-time system but the approach would seem to be applicable for inter-platform communication. Where the approach differs to that which is 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) 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 and reliable transmission. 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. Note that as there is an inherent value to running a subscriber, regardless of where within its range of QoS levels it is performing, an additional bonus equal to the value of the subscriber at its minimum QoS level is given to its acceptance.

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

$$U = \frac{1000}{TBF}. \tag{1}$$

It is assumed that the value of the latency is linear and will affect each of the data samples. The reliability (R) is weighted as 1 for 'best effort' and 2 for 'reliable' communications. Given these assumptions the value (V) of a service shall be calculated using (2).

$$V = R.\left(\frac{U_{min}}{L}\right) + R.\left(\frac{U}{L}\right). \tag{2}$$

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 of a value at least great enough to justify its resource usage. This is discussed further in the following sub-section.

b) Resource Allocation

When considering the assignment of value to a reliable service it is also necessary to consider the allocation of resources. Given the previous assumption that a service requiring reliable communication would intrinsically be of a higher value than one requiring only best effort performance then the reward value assigned for fulfilling the service must be enough to justify its acceptance into the system.

If communication is to be as close to reliable as is possible in a real world system where errors can occur then the process for retransmitting missed data samples should take every opportunity to correct errors.

The DDS specification [10] uses a heartbeat message sent periodically from a publisher to its subscribers to check firstly that the subscriber is still there and secondly that it is receiving the data samples sent. A reply is sent containing the details of any data samples that it is missing. If the publisher finds that these samples are still of relevance to the subscriber (based on the latency allowed) then the samples are resent.

Considering the worst case scenario where samples are always required to be retransmitted then the additional resource requirements (R) can be calculated based on the maximum number of samples that are still valid (latency/TBF) and the number of heartbeat messages sent per second (H). This formula is thus given in (3).

$$R = H\left(\frac{latency}{TBF}\right). \quad (3)$$

While it may be that a high frequency of heartbeat messages would allow the maximum chance for samples to be retransmitted this will need to be limited. A high heartbeat rate would increase the overhead of communication and consume additional processing and communication time. It would also require the reservation of additional resources for possible retransmissions.

Resources reserved for possible retransmission could be used as a second class of resource available for negotiation by other subscribers that can tolerate sudden drops in service as these resources are claimed for retransmission purposes. For the purpose of this research project, however, reserved resources shall remain untouched by other subscribers.

c) Online Value Calculation

Observed values for a publisher instance are recorded for the number of timely and accurate data packet transmissions (*a*) the number of transmissions which did not meet the latency allowed (*z*), and the number of timely but inaccurate transmissions (*f*). The values recorded for the different outcomes are based on an assigned weighting.

These values help to give an indication of the actual reward possible given real network conditions. Given that $\mathcal{P}_a$ is the probability of *a* occurring, $\mathcal{P}_z$ is the probability of *z* occurring and $\mathcal{P}_f$ is the probability of *f* occurring, (4) is used as the online value function. The representation of the formula and values used for online value calculation as shown in this paper have been altered from their original form given by Burns et al. [9] to aid clarity in this example.

$$V_i = a\mathcal{P}_a - z\mathcal{P}_z - f\mathcal{P}_f \ . \quad (4)$$

The online value formula (4) results in a reward value composed of the values assigned to accurate, inaccurate and untimely data samples. Considering the offline value calculation formula (2) a more appropriate online value calculation is required for the framework.

Assuming that those data samples that missed their latency (*z*) are of no use to the subscriber no reward should be associated with these. For those subscribers matched with publishers providing them "reliable" communication samples that are received in time but are inaccurate (*f*) could still be of use if received correctly upon retransmission. It is therefore reasonable to assume that a weighting of between 0 and 1 can be applied to the probability $\mathcal{P}_f$. If the reward value received when there are no errors in transmission is C then a formula for online value calculation for the framework (5) can be found.

$$V = C(\mathcal{P}_a + w\mathcal{P}_f) \ . \quad (5)$$

B. 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.

The following are examples of negotiation techniques that could apply to the future large scale systems in question.

1) Priority based negotiation

The simplest method of differentiating between the criticality of subscribers is through a priority based system. This involves assigning a priority from a finite set of possible values to a subscriber. 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 subscriber would be discarded. This approach is typical for most resource reservation techniques including the network based IntServ and DiffServ models [11].

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, when priorities are assigned statically, analysis can be carried out to predict behaviour or prove certain performance properties.

2) Reward/Penalty based negotiation

An alternative to priority based negotiation is the reward and penalty method described by Abdelzaher et al. [8]. 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 subscriber entering the system while it is running. The negotiation process first adds the new subscriber to the list of running subscribers to determine if there are adequate resources available to meet these new resource requirement. If there are, then the list is used to allocate resources and the process ends. If there are not adequate resources, then the system searches for the subscriber 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 subscriber 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 subscriber and its associated reward now result in a greater 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 subscriber is scheduled.

This approach has traditionally been associated with the selection of system configurations from a set of alternates available. For example, a subscriber for a surveillance

system could make use of a high resolution camera or degrade service to a low resolution radar depending on resources available. Adaptations would be necessary to make it suitable for the applications of future distributed systems where degraded levels of service will be within the application itself.

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 pseudocode in Fig. 2 shows the basic structure of the framework algorithm, assuming for this example that a matching onboard publisher is not available.

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. Note that it will be necessary to employ QoS management to ensure that the agreed levels of QoS are being met. If QoS is not being met then the resources available should be recalculated and the list of running subscribers renegotiated.

V. SIMULATION RESULTS

A simulation has been developed using MATLAB to experiment with the QoS framework proposed within this paper. The simulation is based around an NEC type scenario of nodes physically distributed within an environment, with differing resources and functional capabilities. Network topologies are generated randomly based on a seed input. Nodes within a topology are populated with publishers and subscribers. Each publisher and subscriber has a set of QoS characteristics matching that described within the framework.

```
FOR each connected node
    FOR each publisher on connected node
        IF publisher reliability >= new subscriber reliability
            IF publisher TBF <= new subscriber TBF
                Add publisher to list of potential publishers
            END
        END
    ENDFOR
ENDFOR

Sort list of potential publishers by network link type (wired first) and then by free resources on
link

FOR each publisher in list of potential publishers
    Compile list containing new subscriber and all current subscribers using network link
    Calculate resources required, new subscriber at max QoS levels
    WHILE resources available < resources required
        FOR each subscriber using link
            degraded TBF = current TBF + TBF interval
            IF degraded TBF > maximum TBF
                subscriber must be removed
                reward decrease = current subscriber reward
            ELSE
                calculate new reward
                reward decrease = old reward - new reward
            END
            IF reward decrease < current lowest reward decrease
                Note subscriber with lowest reward decrease
            END
        ENDFOR
        Remove or degrade subscriber with lowest reward decrease
        Calculate resources available
    ENDWHILE
    Calculate new system reward
    IF new system reward > old system reward
        Accept new system configuration, end search
    ELSEIF (old system reward - new system reward) > subscriber    penalty
        Accept new system configuration, end search
    ELSE
        Do not accept new system configuration
ENDFOR
```

Figure 2. Framework Algorithm Pseudocode

An assumption is made that communication between nodes is direct. This is done so as to demonstrate the fundamental functioning of the negotiation algorithms involved. The framework could be adapted in the future to take into account multi-hop algorithms. Adaptations would also be necessary to ensure that the distribution of reward throughout the system is proportional to the amount of resources that are being consumed.

Initial tests have compared the proposed framework to other existing methods of QoS negotiation described within this paper. The two negotiation techniques for comparison (priority based and compatibility testing) are considered to be using one of three fixed QoS levels. These correspond to the maximum level requested, the minimum and a medium level between these points. For the priority based method of negotiation the priority is considered to be the calculated reward value. No penalties are used for the tests as it is useful to show the performance of algorithms while being unbiased by user preference.

The simulation results given examine algorithm performance in a randomly generated, complex system topology consisting of 15 nodes connected by either wired or wireless means. The network topology can be seen in Fig. 3. The system contains 500 publishers distributed randomly across the nodes. An increasing number of subscribers is entered into the system. Algorithm performance is assessed in terms of system utility, stability and resource utilisation; three areas that have been identified as being of high importance for such future distributed systems as NEC. Note that for clarity the legend for Figures 5, 6, 7 and 8 is given in Fig 4.

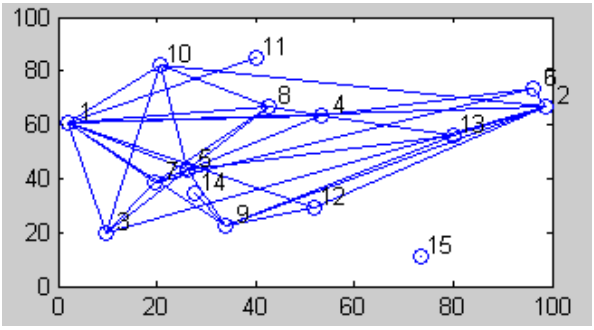

Figure 3. Simulation Network Topology

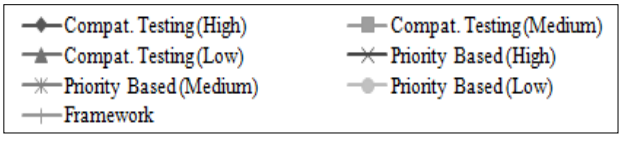

Figure 4. Legend

A. System Utility

The reward values gained from each algorithm are shown in Fig. 5 and the corresponding network utilisation is given in Fig. 6. The final number of serviced subscribers is shown in Table I so as to give context to these measures of algorithm performance. Where the reward value is given this is considered to be in units of "Value" (v), measuring system utility, as determined by equation (2) in section VI.A.1. Note that the network utilisation is shown as a combined percentage from all network connections in the system.

While system resources are largely uncontested algorithm performance can be seen to be fairly similar. As more subscribers enter the system, however, the negotiation methods start to exhibit different behaviour in terms of both reward value gained and network resource utilisation. Considering the compatibility testing method to be the base rate for reward, once all subscribers have entered the system the priority and framework algorithms can be seen to provide an increase of at least 98.8% and 179.9% respectively. This shows the clear advantages of employing selective algorithms when allocating resources. The network resource utilisation itself largely remains consistent among the different algorithms, however, the framework can be seen to use slightly more resources at times which is to be expected given that QoS levels are adapting to both what is required and what is available.

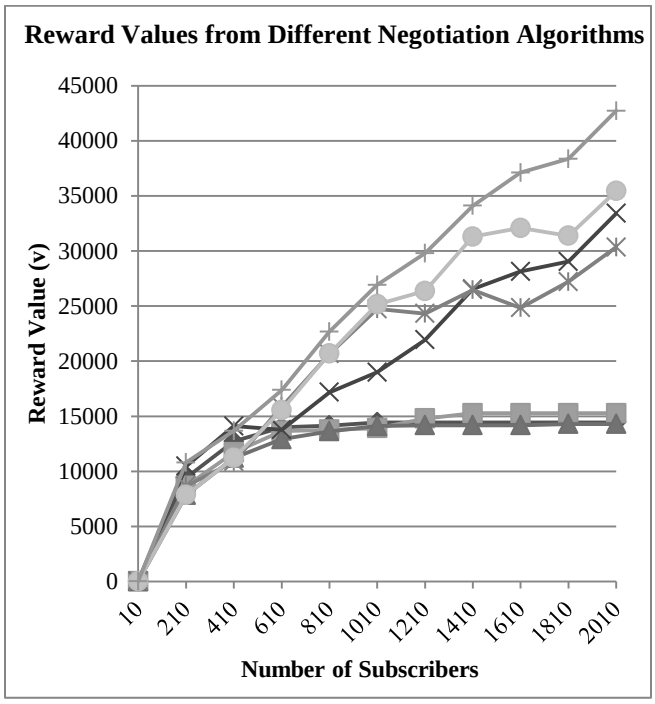

Figure 5. Reward Values for Different Algorithms

TABLE I. NO OF SERVICED SUBSCRIBERS

Negotiation Technique	No. of Serviced Subscribers
Compat. Testing (High)	29
Compat. Testing (Medium)	46
Compat. Testing (Low)	52
Priority(High)	33
Priority(Medium)	44
Priority(Low)	65
Framework (Reward/Penalty)	68

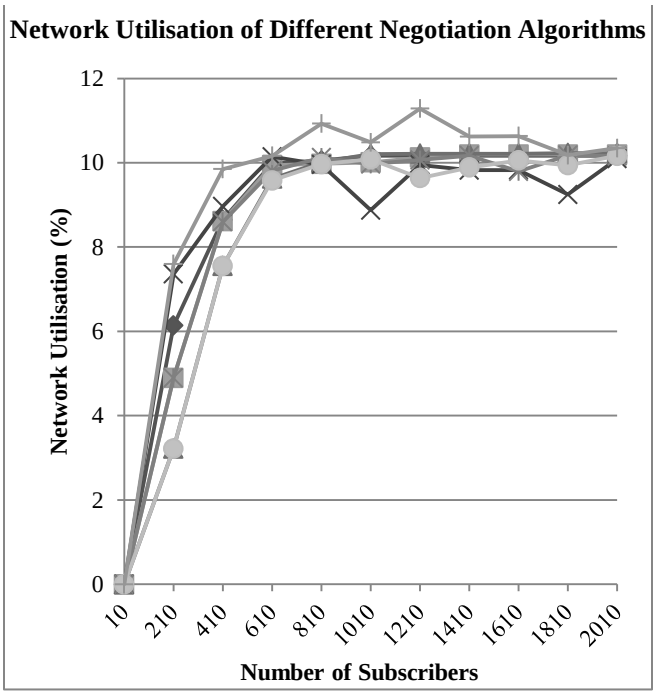

Figure 6. Network Resource Utilisation

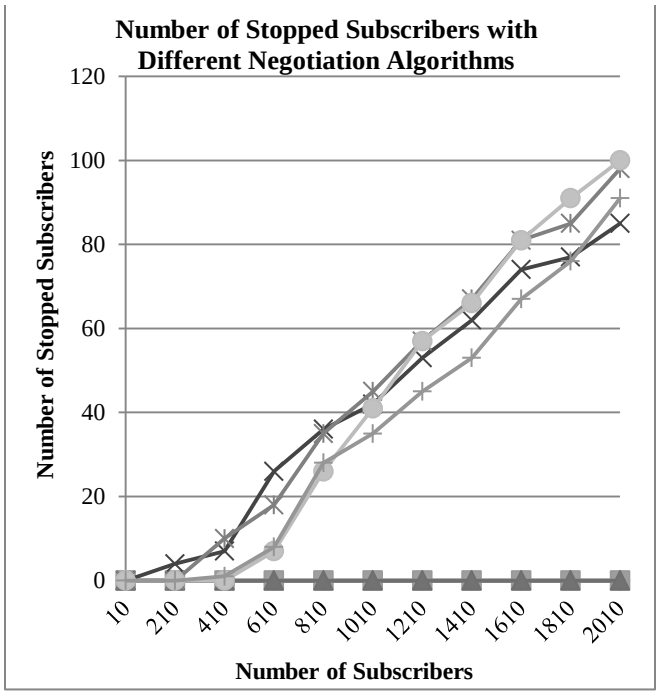

Figure 7. Number of Stopped Subscribers

Note that the network resource use seems to level out around 10% due to the fact that network links are randomly created by the simulation and links may exist between nodes where there are no subscriber and publisher matches so the link will remain unused.

B. System Stability

As a measure of system stability the number of subscribers that were accepted into the system but then subsequently removed in favour of others capable of offering a greater reward was recorded and is shown in Fig. 7.

As the compatibility testing method does not offer the ability to exchange subscribers based this can be seen to remain stable. Performance of the priority based algorithm can be seen to be reasonably similar in terms of number of subscribers stopped at each QoS level. As is to be expected, however, as the "low" level accepts the most subscribers into the system it also experiences the greatest degree of churn. While performance of the framework algorithm does appear very similar to the priority based method, the flexibility of negotiation and graceful degradation possible allows it to perform comparably well given that it also accepts the greatest number of subscribers into the system.

C. Resource Utilisation

Examining the execution times of the algorithms, shown in Fig. 8, starts to show the cost of introducing a flexible and relatively computationally expensive algorithm. At its worst the framework algorithm took an average of around 9.2 seconds to reach a decision as to whether a subscriber should be accepted or rejected (based on a total execution time of 18492 seconds for all 2010 subscribers). This could be reduced through fine tuning of the algorithm and QoS requirements (increasing the sample rate interval, thus lowering the number of QoS levels available for example). The simulation was conducted on a laptop with a 2.2Ghz processor and 2GB RAM, a specification that while not excessive may currently exceed that available on handheld or other such portable devices. Note that if the CPU time allowed for the matching of publishers and subscribers is static then, once initial setup has completed, the framework algorithm places no further penalty on subscriber fulfilment.

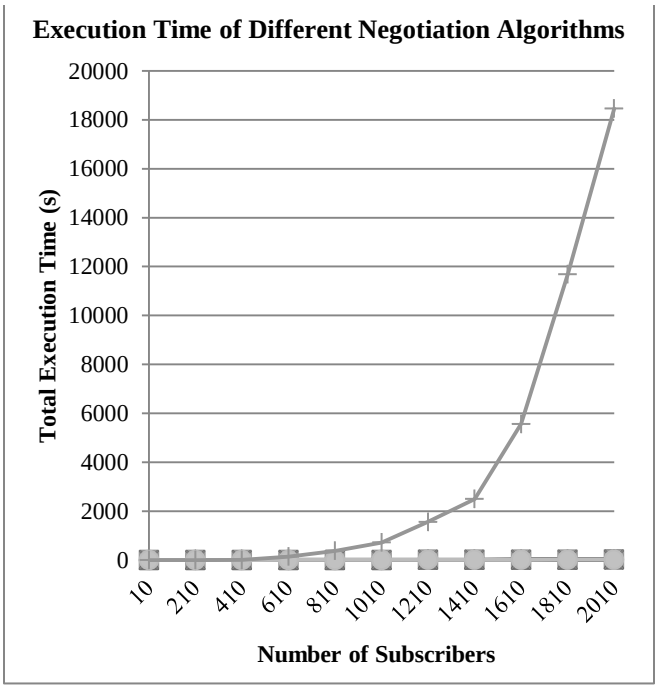

Figure 8. Execution Time of Negotiation Algorithms

VI. CONCLUSION

This paper has discussed the impact of resource allocation on the dependability and utility of future large-scale distributed systems. A Quality of Service framework has been proposed that combines existing methods of providing adaptive and dependable performance, altering these where necessary to suit the requirements of future systems. This includes increasing the flexibility of the system through the introduction of varying levels of QoS and using offline and online system reward calculation as a means of assessing system reward when negotiating resource allocation.

Simulation results have been given for a large-scale distributed system, comparing performance of the proposed QoS framework negotiation algorithm to those already existing in terms of system utility, stability and resource utilisation. The framework algorithm has been shown to consistently outperform existing algorithms in terms of overall system reward, with comparable network utilisation, however, at a cost to the initial setup time required.

Further analysis is now needed on the performance impact of introducing a negotiation algorithm of a higher complexity than that which has conventionally been used in distributed systems. Experimentation is needed to see what exactly the cost to performance will be and when in terms of system size and load the framework shows its benefit. To this end a real-time test-bed implementation of the QoS framework shall be created using the Integrated Modular Avionics software architecture [12] as a distributed system platform.

ACKNOWLEDGMENT

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

REFERENCES

[1] Bull, P., Grigg, A., Lin Guan, and Phillips, I.: A Quality of Service Framework for Adaptive and Dependable Large Scale System-of-Systems. In: Fifth International Conference on System of Systems Engineering (SoSE), pp. 1--6, (2010)

[2] Ministry of Defence: Network Enabled Capability, Joint Service Publication 777 Edition 1 (2005)

[3] 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)

[4] Russell, D. J., & Xu, J.: Service Oriented Architectures in the Provision of Military Capability. University of Leeds (2007)

[5] 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)

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

[7] Bouyssounouse, B., & Sifakis, J.: Embedded Systems Design: The Artist Roadmap for Research and Development. Springer, (2005)

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

[9] 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. In: Journal of Systems Architecture, pp. 305--325, (2000)

[10] Object Management Group. Data Distribution Service for Real-time Systems (DDS), version 1.2, (2007)

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

[12] Ministry of Defence.: ASAAC Standards Part 1. ASSC - Standards & Guidance Support for the UK Military. Interim Defence Standard 00-74 Part 1 Issue 2 (2008)