QoS Assurance vs. Dynamic Adaptability for Applications

Saleem N. Bhatti and Graham Knight
Computer Science Department, University College London, Gower St., London WC1E 6BT, UK
S.Bhatti@cs.ucl.ac.uk and G.Knight@cs.ucl.ac.uk

Abstract

Enabling adaptability to network QoS (quality of service) is seen as a key feature for future applications. One way to approach adaptability is to build it into the network and allow applications to signal their requirements to the network. This means that resource reservation mechanisms must be available end-to-end, which is not always the case, especially on the Internet. Also, user preferences affect how the application is used. So the application must be dynamically adaptive, taking into account the application's capabilities, the available network QoS and the user preferences. We suggest that it is possible to build practicable QoS summaries that capture all these inputs and allow dynamic adaptability.

1. Introduction

In today's Internet, applications face operation with a range of connectivity/access network technologies, QoS (quality of service) fluctuations throughout the network and host mobility. This means that today's applications must be adaptable enough to match their mode of operation to the resources they find available in the network. The nature of the network is such that there may be QoS variations during the lifetime of a single application flow. This is especially troublesome for multimedia applications with QoS sensitive flows. While applications show some dynamic adaptation capabilities (e.g. use of elastic buffering to counter jitter on flows), they generally require the user to select the correct configuration or user preferences for the application to function in a given situation. Typically, the user of such applications may lack the expertise or knowledge about network conditions to complete this task adequately. Even if the user has the correct application configuration, the best-effort service offered across the Internet means that applications typically experience QoS fluctuations and so the configuration may need to be changed during the lifetime of any single application instance. We would like one or all of:

- resource reservation mechanisms to ensure that applications receive the quality of service they require
- the network to provide suitable adaptation mechanisms which allow communication to continue when QoS fluctuates
- applications to be dynamically adaptable, being able to configure their flow construction appropriately to match the detected network QoS

2. QoS assurance

In [1], there is an excellent description of the elements of a general QoS architecture for assuring QoS. With QoS assurance mechanisms, applications have to specify what they want the network to do for them, and the network tries to honour this request. The network itself may provide adaptability mechanisms [2] to cope with fluctuating QoS or heterogeneity due to network architecture. [1] suggests that (among other things) the application should be in control of the following information that passes to the network: QoS service-level (e.g. guaranteed-service, controlled-load service) and QoS management policy (how adaptation should take place when there are QoS fluctuations). These affect the way the application is integrated with the network (the separation of functionality between the network and the application), and how heterogeneity is supported (at the user, application and network access level).

QoS assurance mechanisms are based around the use of resource reservation and QoS re-negotiation to achieve the desired QoS for the application. Mechanisms such as RSVP [3] are designed to provide resource reservations in the Internet. However, RSVP can fail [4], and adaptability mechanisms in the network become impracticable and increasingly difficult to support in *ad hoc* or mobile network architectures. Additionally, where resource reservation and QoS re-negotiation are used, they should be available end-to-end for maximum utility. In today's Internet there is very little resource reservation support and network components lack the programmability required to enable adaptability in the network. Furthermore, as such mechanisms do become available, they will not be deployed uniformly, and this will further compound the heterogeneity in the Internet. For many applications, such

QoS assurance or adaptability mechanisms within the network may only be useful if they are available end-to-end. Where these mechanisms do exist, there is great value in being able to use them, but in the "standard" best-effort Internet, other mechanisms are required to address adaptability.

3. Dynamic adaptation decisions

There is currently great interest in making applications adaptable to changes in network QoS. This is of particular importance for real-time media flows or flows that are sensitive to QoS fluctuations. Much of the attention for this work has focused on mechanisms that allow adaptability of the *media flow construction* (which we call the **flow-state**) by scaling (e.g. [5]), filtering (e.g. [6]) or encoding techniques (e.g. [7] [8]). An area that has received little attention, however, is how applications can automatically select the correct flow-state to match the available network QoS as detected during operation. The application must currently rely on the user to set the correct preferences to allow operation in a particular QoS regime. Although application specific mechanisms exist to allow some automatic adaptation (e.g. elastic buffering to combat jitter in audio tools), there is little support for mechanisms that allow applications to automatically and dynamically make decisions about their flow-state, based on user preferences and measured network QoS values for that flow.

There may be many instances of an application used by many different users. Each of the users will have their own preferences for the application. Any decision making process for adaptation must allow the application to apply user preferences to the QoS being experienced by the flow. For example, with an audio tool with numerous encoding schemes, one user may specify that the tool "always uses the best audio quality possible" while another user may specify that the tool aim to "retain stability" in the face of network QoS fluctuations. The application may interpret these preferences as meaning the "encoding with the highest data rate requirement whenever possible" and "do not change flow-state too often", respectively. So the adaptation decision is not only application specific but also application *instance* specific, and controlled by user preferences.

4. Dynamic adaptability in applications

For the provisioning of end-to-end QoS, the user/application is telling the network what is required and asks that the network should adapt/configure itself to comply to the application's requirements, i.e. static adaptability. In enabling dynamic adaptability, the application is constantly (i.e. within application specific time scales) receiving information about the QoS that the network can offer, and then changing its flow-state to comply to the network capabilities.

For assuring QoS, the integration principle says that we need QoS to be "configurable, predictable and maintainable over all architectural layers" [1]. With this in mind, the principle of integration does not have to apply so strongly for adaptable applications. What we require is a well-defined interface between the application and network to receive an application oriented view of the network capabilities; i.e. something that is easily usable and understandable at the application level. What is required is a report that summarises the compatibility between the current network QoS ant the flow-states possible for an application. Our weaker integration requirement is an advantage and a disadvantage. It is an advantage in that it requires looser coupling between the various architectural components giving greater freedom in system design. In fact, this looser integration can be seen as a requirement for Internet applications, as IP is not biased towards to any particular network technology. This latter reason is a disadvantage – we can not directly exploit any network specific mechanisms.

Additionally, the QoS service-level required may be important to the individual users and may determine the cost of the service, e.g. guaranteed-load is a "better" service than controlled-load so may cost more. We argue that the service-level should not be specified by the application. The application should be prepared to be more flexible in its adaptation capability leaving service-level selection to the user. Three reasons for this are:

1. the service-level may determine the cost of the service and users usually wish to control how much they pay.
2. network heterogeneity, lack of resource reservation or network element failure may mean that a particular service-level is not available at a given time at a given point in the network.
3. new, additional service-level definitions may be introduced that are more suitable (in terms functionality or cost) for use in a given situation, e.g. there are two recent descriptions of **adaptive** service-level [9] [10], and differentiated services may be set up that are domain/administration specific [11].

Additionally, if cost-based feedback is available from the network, then cost could be treated as a QoS parameter – although not related to the performance of the flow it acts as a defining constraint in the same way as, say, defining minimum data rate or maximum jitter for a flow. The value of making the cost explicit as a QoS parameter is that it highlights the importance of cost as a feedback control mechanism in future services [12].

QoS management policy will be subject to user preferences and application specific behaviour. Applications may find it useful to have a specification of the QoS management policy before the application starts operating. This would certainly be of value to the network for controlled allocation of resources, and makes sense in the context of trying to assure end-to-end to QoS. However, in our consideration of dynamic adaptability, the use of the application typically requires interaction from the user in order to determine its adaptation requirements, and these may not be known until after the application is running. In [1], the QoS management policy:

captures the degree of QoS adaptation (continuous or discrete) that the flow can tolerate and the scaling actions to be taken in the event of violations to the contracted QoS [9].

We chose to make a separation between what "*the flow can tolerate*" and the "*scaling actions to be taken*". We argue that the former is a property of the media and the latter is an application specific requirement that includes interaction with the user. Flow performance specifications can be used to indicate the flow-states that are possible for a flow and can be determined by the application designer. The action to be taken on fluctuations ("*violations*") of QoS is a dynamic adaptation decision and cannot be determined by the application designer *a priori*. It is the difference between the application designer saying "I know what is sensible for the application" and the user saying "I know what is sensible for the application to do for me". Ultimately the application's functional constraints have the final say on which flow-state(s) is (are) functionally *possible*, but this should not dictate *how* the user would like the application to behave, i.e. how adaptation should take place. As an example, consider a remote teaching scenario and the requirements of the audio and video flows. When operating in lecture mode (main part of the teaching session), the conferencing application may tolerate relatively high delay and throughput is asymmetric, but during a question and answer session (at the end of the teaching session), low delay and jitter are required with symmetric throughput for flows.

The application must be able to assess the user preferences and available network QoS in order to make automatic and dynamic adaptation decisions.

5. Application and network compatibility

The adaptation decision for the application is to select a flow-state. For example, given suitable information about loss, achievable data rate, etc. for its flow, an audio application may be able to choose an audio coding that matches the available network QoS by adapting the data rate, or by using forward error correction, etc. Real-time, QoS sensitive Internet applications may use mechanisms such as RTCP [13] (or application specific mechanisms) to distribute such QoS information. In order to select a flow-state that "matches" this information, the application must assess the compatibility between the network QoS, the flow-state requirements and the user preferences.

In [14], we use a mechanism – which we call the **QoSSpace** – that produces a **QoSReport** summarising the compatibility between the network QoS and an application's flow-states. The QoSReport contains a **state compatibility value (SCV)** for each flow-state. The SCV is a number in the range [0, 1] indicating the instantaneous assessment of compatibility between a flow-state and the network, with 0 indicating that the QoSSpace has no confidence that the flow-state cannot be supported by the network, and 1 indicating that the QoSSpace has high confidence that the flow-state can be supported. In order to produce SCVs, the QoSSpace needs QoS parameter measurements for the flow. These can come from any application specific mechanism.

The decision of which flow-state to use depends on network QoS, application capability and user preferences. This decision can be performed in an automatic (no run-time interaction with the user) or semi-automatic fashion (interaction with user before making a flow-state change). The decision mechanism is embodied in an **application adaptation function (AAF)** which takes information about the user preferences, application flow-states and network QoS (QoSReports). In the automatic case, the user indicates user preferences such as "use lowest cost service" at start-up and the AAF automatically selects the most suitable flow-state. In the semi-automatic case, the user can be given the ability of making an informed decision at run-time by the use of SCVs from the QoSReport.

Our QoSSpace model has some similarities to that of a general control system process (Figure 1). However, it has significant differences in its aims and its function.

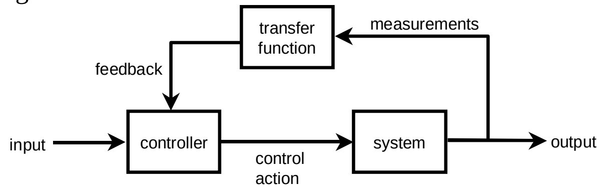

Figure 1: A general control system

Many adaptive mechanisms (e.g. congestion control) rely on the use of a control systems approach to adjust their flows. In a traditional control system (Figure 1), the aim of the *controller* is to generate *control actions* that keep the *system* at a set-point (a defined operating point for the *system*). The *transfer function* uses *measurements* of specific parameters from the *output* of the *system* to produce *feedback* for the *controller*. The *controller* uses the *feedback* information with other *inputs* to decide the *control action* it should take in order to maintain the required *system* behaviour, i.e. maintain the *set-point*.

In our model in, we can identify similarities:

- the *system* is the application and the *output* is the flow as it traverses the network
- the *measurements* from the *output* are a set of QoS parameter measurements for the flow
- the *transfer function* is analogous to the QoSSpace which generates the QoSReports as a *feedback* summary of *system* behaviour
- the *controller* is analogous to the AAF and the *control action* will be to select a suitable flow-state, which may affect the *system* (the application) and the *output* (the flow)

However, our dynamic adaptation scenario differs from the traditional control system in three important ways:

1. the QoSSpace does not model the *system* input/output but provides a summary of the relative **compatibility** of the network QoS offered to the flow and the possible flow-states
2. the contents of the QoSReports from the QoSSpace may be modified by the application and by user preferences
3. the behaviour of the AAF may be modified by user preferences or inputs from other application specific sources

So, overall, while we do have a feedback control process, the *measurements* are processed in a different manner to that of a traditional control system, and so the *feedback* is of a different nature. Essentially, the aim of our model is not that same as that for a traditional control system; instead of attempting to **maintain** a set-point, our model seeks to allow the application to **select** a flow-state. (Our work in [14] is ongoing.)

6. Conclusions

The Internet applications of the future will need to be adaptive in order to cope with network heterogeneity, differing QoS from the service provider, fluctuating QoS during operation, and differing user preferences. Cost of service will play an important part for selecting service-levels and so the QoS available to an application. Decisions regarding cost will be made by the user. While mechanisms for QoS assurance and adaptability in the network may be available, such mechanisms may not be uniformly deployed or available end-to-end. Applications must be prepared to adapt their behaviour as well as use mechanisms within the network. The adaptation capability in applications must be dynamic and automatic, and should not rely on the user to have expert knowledge. It seems possible to create QoS compatibility summaries that can take into account the applications flow-states, the network QoS and user preferences in order to enable dynamic adaptability. We find that there may be conflicts in requirements between QoS assurance mechanisms and dynamic adaptability requirements for the application. These issues are for further study.

7. References

[1] A. Campbell, C. Aurrecoechea, L. Hauw, "A Review of QoS Architectures", Proc. of 4th IFIP International Workshop on Quality of Service (IWQS'96), Paris, France, Mar 1996.

[2] A. T. Campbell, "Mobiware: QoS-Aware Middleware for Mobile Multimedia Networking", Proc. IFIP 7th International Conference on High Performance Networking, White Plains, New York, Apr 1997.

[3] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC2205, Sep 1997.

[4] L. Wolf, C. Gridwodz, R. Steinmetz, "Multimedia Communication", Proceedings of the IEEE, vol. 85 no. 12, pp-1915-1933, Dec 1997.

[5] L. Delgrossi, C. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss, C. Vogt, "Media scaling for Audio-visual Communication with the Heidelberg Transport System", Proc. ACM Multimedia, pp99-104, Jun 1993.

[6] N. Yeadon, F. Garcia, D. Hutchison, D. Shepherd, "Filters: QoS Support Mechanisms for Multipeer Communications", IEEE Journal of Selected Areas in Communication, vol. 14, no. 7, pp1245-1262, Sep 1996.

[7] V. Hardman, M. A. Sasse, M. Handley & A. Watson, "Reliable Audio for Use over the Internet", Proc. INET'95, pp171-178, Hawaii, USA, 27-30 Jun 1995.

[8] E. Amir, S. McCanne, M. Vetterli, "A Layered DCT Coder for Internet Video", Proc. ICIP'96, Lausanne, Switzerland, Sep 1996.

[9] A. Campbell, G. Coluson, D. Hutchison, "Supporting Adaptive Flows in Quality of Service Architecture", ACM Multimedia Systems Journal, 1996, May 1996.

[10] S. Lu, K.-W. Lee, V. Bharghavan, "Adaptive Service in Mobile Computing Environments", in Building QoS into Distributed Systems, (A. Campbell, K. Nahrstedt, Eds), pp25-36, [Chapman & Hall] 1997

[11] K. Nichols, S. Blake (Eds), "Differentiated Services Operational Model and Definitions", IETF DIFFSERV WG, work-in-progress, Feb 1998.

[12] S. Shenkar, D. Clark, D. Estrin, S. Herzog, "Pricing in Computer Network: Reshaping the Research Agenda", ACM Computer Communications Review, vol. 26. No. 2, pp19-43, Apr 1996

[13] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC1889, Jan 1996.

[14] S. N. Bhatti, G. Knight, "Notes on a QoS information model for making adaptation decisions", to appear, HIPPARCH'98, UCL, London, UK, 15-16 Jun 1998.