A QUALITY OF SERVICE AWARE WEB SERVER

Kristoffer Getchell, Martin Bateman, Colin Allison and Alan Miller
School of Computer Science
University of St Andrews
Fife, Scotland

ABSTRACT
This paper addresses two questions. Firstly, how is it possible for an application to become aware of network conditions and secondly, given this awareness, how can a system be designed that will allow application level adaptation? A framework enabling these adaptations, consisting of three components, is proposed: A network monitor makes available to applications measurements of the Quality of Service (QoS) that the network is providing. A Quality of Service Aware Web server handles the dynamic decisions required in order to determine the optimal version of a site to send to the connecting client. A content adaptation tool allows content providers to generate, from a single high quality pool of resources, Web content that is appropriate for different QoS levels. Currently the Internet relies upon congestion control mechanisms embedded in TCP to prevent congestion collapse and control traffic flows. Yet applications are in a strong position to know how best to react to the onset of congestion, and thus promote the social good.

1. INTRODUCTION
As the size of heterogeneity of the Internet becomes more pronounced, the way in which services are provided to clients becomes relatively more important (Neuman 1994; Brewer, Katz et al. 1998). In the current model, servers strive to provide a uniformed service to all clients, regardless of current operating conditions both within the server and on the network itself. This lack of awareness is both inefficient and unreliable; connecting clients may be on network links ill equipped to deal with the amount of traffic being sent by the server, thus congestion events are likely to occur (Jacobson 1988; Jain and Ramakrishnan 1988; Alman, Paxson et al, 1999), wasting valuable network resources. Worse still, should the server find itself without the internal resources required to service connections, then for a period of time the service may not be available to clients. As the importance of the Internet as an environment for business activity increases, service unavailability becomes a serious financial consideration; for companies like Amazon or Google, even short periods of unavailability are likely to have severe financial implications (Chuang 2001).

At present there is much interest in developing self-managing system architectures. A great deal of research is being undertaken in both the academic and business arenas, with the overall aim of reducing the technical burden associated with the deployment and operation of mission critical services (Various 2004). As the reliance and dependence placed on these services increases, the scale at which they are able to be supplied is also placed under pressure, prompting yet further research into the most suitable manner in which to provision services using highly scalable architectures. In terms of the Internet, there is little consensus as to the most appropriate way forward, with the concepts of geographical dispersion (Colajanni, Cardellini et al. 1998), server load-balancing (Colajanni, Cardellini et al. 1998; Bunt, Eager et al. 1999) and various methods of content adaptation (Abdelzaber and Bhati 1999; Andersen, Bansal et al. 2000; Badrinath, Fox et al. 2000; Zhang, Chen et al.

Figure 1. Quality of Service Aware Server Farm
This paper addresses two questions. Firstly, how is it possible for an application to become aware of network conditions and secondly, given this awareness, how can a system be designed that will allow application level adaptation? In response to these questions, a framework enabling these adaptations, consisting of three components, is proposed and discussed in section 2.

2. FRAMEWORK ENABLING APPLICATION RESPONSE (FEAR)

There is not a one to one mapping between network level metrics and the way in which users perceive QoS. Here we identify three factors that contribute to whether a user perceives a Web browsing session as being satisfying or not.

1. **Fidelity**: The quality of data sent to clients is vitally important – if the technical quality is too low then the user is likely to perceive poor quality.

2. **Delay and Feedback**: When a user is interacting with a Web site, they require feedback for their actions. Closely linked with feedback are high delays, which can also detract from user perception – if a page takes too long to display or a video too long to buffer on the connecting client machine, then the users will be disappointed and have a poor perceived level of quality. (Abdelzaber and Bhati 1999).

3. **Consistency**: If there is consistency of presentation within a session then the user will be more likely to associate a higher utility with their browsing session, and thus be left with a higher perception of quality (Bouch and Sasse, 2000).

QoS is usually quantified in terms of network metrics. At this level there are two challenges: How to discover the QoS that is being experienced by specific traffic flows, and how to communicate this information in a timely and useful way to the application.

1. **Round Trip Time**: The length of time to send a packet to a remote host and back to originator. Most users will not unduly notice delays of around 100ms (Abdelzaber and Bhati 1999) although RTT is only one component of the total delay (Ruddle et al. 2003) and so other factors, such as the delay introduced by the server preparing its response, should also be considered.

2. **Jitter**: The variation in the RTT measurements.

3. **Bandwidth**: The amount of data that is transferred per unit time. Together with RTT this impacts on the delay experienced by the user.

4. **Packet Loss**: The proportion of packets lost on a link in a given time period. Loss is used as an implicit indication of congestion. It is easy to measure and as discussed below it is possible to map from loss to fair bandwidth utilisation.

The approach adopted is to alter the fidelity of media in response to feedback on the level of network congestion, thereby achieving the appropriate trade off between technical quality and delay. In order to do
this it is necessary to decide how to dynamically choose which quality version to send to a given client. In order to facilitate this decision mechanism, an integrated, modular framework into which the Quality of Service Aware servers can be plugged has been designed and a Quality of Service Aware web server has been developed as illustrated in Figure 2. A packaging system has been adopted which sees the components in the systems separated into three distinct areas:

1. **Server Realm** – this package contains the server functionality and interfaces directly with the connecting clients, fulfilling their requests as appropriate.
2. **QoS Realm** – this package is responsible for gathering, and providing to the Server Realm, all available QoS information concerning currently active connections.
3. **DataStore Realm** – this is a utility package used by both the Server and QoS Realms and provides the functionality to quickly store and retrieve data on as-required basis. Within this category there are several implementations of DataStores that are customised to specific needs.

Once an interested Quality of Service Aware server has received the relevant QoS information from the framework, it is then equipped to deal with a connecting client in the most suitable way. Assuming access to the required QoS information is possible, the framework introduces localised changes to the model implemented in “standard” Web Servers, with the Connection Handler speaking to the Link Evaluation Mechanism and then the Content Provisioning Mechanism before returning the most appropriate resource to the requesting client. It is at this stage that indirection is used to send to a client the most suitable resource (based on the quality of the client link). At present the framework has been developed to support five distinct quality levels – this is done in order to provide distinct sets into which various links can be graded and thus reduce the possible problems caused by a slightly erroneous quality classification – there is less likelihood that small errors will cause a change in grading and so we protect the perceived user quality by reducing unnecessary quality reclassifications, and ensuring a more consistent resource presentation.

The system makes use of the concept of a human session in meeting the consistency requirements of users. The aim is to provide a consistent fidelity throughout the lifetime of a session. Here we define a human session as an amount of time spanning a series of requests during which a user might alternate between assimilating the information within a page and downloading new pages. For example a user receives Page 1 from a site and spends 10 minutes reading it, he then proceeds to request Page 2; ideally we want to ensure that the quality of Page 2 is similar to the quality of Page 1 that has already been received – this will ensure continuity throughout the browsing session. In this situation the Human Session timeout should not be set to anything below 10 minutes. In the absence of sharp and prolonged changes in network conditions, all pages served should aim to be from the same quality classification throughout the entire session.

Regardless of any Human Session periods that attempt to give a unified feel to the browsing experience, changes in the QoS readings are continually tracked. If an initial link quality evaluation is found to be consistently lacking then it may be necessary for remedial action to be undertaken, resulting in the link quality being re-graded, in order to provide the most suitable browsing experience to the connecting client. However such a decision should not be taken lightly and cannot be based solely on one, or possibly a handful of bad readings. Instead historical data should be considered, with the weight attached to data diminishing as it ages. In order to achieve this data smoothing an Exponentially Weighted Moving Average (EWMA) is used as it cleanly deals with the ageing of historical data whilst smoothing the effects of individual erroneous readings, something that the arithmetic mean is unable to do.

The steady state behaviour of TCP provides a benchmark against which the behaviour of other congestion control schemes have been judged (Mahdavi and Floyd 1997; Floyd 2001). The steady state behaviour of TCP is given by the following equation (Mathis et al. 1997) where MSS is the maximum segment size on a link, C is a known constant, RTT is the round trip time and P is the probability of loss. Estimates of RTT and P are provided by the Location Information Server described in (Ruddle et. al. 2002).

$$T = \frac{MSS \times C}{RTT \times \sqrt{P}}$$

(1)

If the data flow is application limited, low levels of loss may suggest an available bandwidth in excess of the actual physical bandwidth. This can be accounted for by measuring the actual bandwidth utilisation (B) and setting the estimated available bandwidth to the minimum of that estimated from congestion feedback and actual utilisation. The bandwidth category that a connection request falls into (Q) is then given by:

$$Q = \min \left( \left\lceil \frac{T}{RTT \times \sqrt{P}} \right\rceil, \max(B) \right)$$

(2)

In this way the expected available bandwidth for a destination can be calculated based on past measurements. The result may not correspond to the actual access bandwidth, but may rather correspond to the share of bandwidth available on the bottleneck link. Having established the bandwidth available to a destination the server can then choose the appropriate version of the site to transmit. In order to provide the various versions of resources, a series of tools have been provided which, when given a single high quality Web site and a
meta-level description, transform the single site into a number of quality adapted versions. The transformation happens once offline and creates a series of Web sites that are aligned to the quality levels in use with the content provider. With this approach we avoid the situation of having to re-encode media files on the fly and thus control the amount of additional CPU resource required in order to manage the adaptation, whilst also ensuring that the connecting client request is fulfilled as quickly as possible.

3. CONCLUSION

We have presented the design and implementation of a framework for making the WWW Quality of Service Aware. The approach is to use passive monitoring of transport level headers to make predictions about the QoS that is expected to be experienced by a particular location in the near future. A mechanism for translating a single high fidelity Web site into a set of Web sites that are appropriate for different bandwidths has been outlined. When a user session starts the Web server uses QoS information to determine which bandwidth is appropriate, it then serves the most appropriate resource to the client.

REFERENCES


Mahdavi, J. and S. Floyd (1997). "TCP-Friendly Unicast Rate-Based Flow Control."


Ruddle, A. C. Allison and P. Lindsay (2002) "Best Effort Measurement Based Congestion Control" *European Transactions on Telecommunications* Special Issue on Internet Traffic Measurement.