Quality of Service
Richard E. Schantz
BBN Technologies

One side effect of the rapid growth and deployment of communication-based distributed applications, particularly Internet-based applications, is the generally recognized need for more attention to usability and control of the communications functions in service of these applications. Environments such as these where it is difficult or impossible to predict ahead of time even approximate configurations or load mixes makes it mandatory to develop approaches toward varying but predictable behavior at different times during an application’s life cycle. Quality of Service (QoS) is the term that is being used to loosely organize the collection of activities and technology initiatives that have emerged to improve and control network oriented resource management based on mounting experience with distributed, Internet applications.

While the quality of the service provided has always been a factor influencing effective use, it is only recently that Quality of Service as a named attribute has become the focus for directed distributed infrastructure enhancements. This new focus stems largely from previous Internetwork limitations in areas such as unsynchronized media streams in distributed multimedia applications, or in lacking assured communication resources for high priority users during high-demand periods, with the side effect of diminishing available resources to others. QoS concerns have led to a number of implemented and suggested improvements to the distributed computing infrastructure.

Taken in their narrowest form, the QoS improvements are concerned with the capabilities and control mechanisms available within the communications network itself. Recent work has extended the QoS viewpoint to include not only an end to end application picture, but also the integration of other properties affecting quality of service beyond timely communication services. In its largest sense, QoS involves the multitude of properties beyond the application specific functional behavior of a particular distributed application. Examples include performance characteristics, dependability, behavior and adaptability under various changing environments, and security. This article is about Quality of Service (QoS) in this larger context as an organizing concept for integrated resource management for distributed computing infrastructures and applications. Much of this material is work in progress, with emphasis on emerging trends and directions.

Dimensions

There is now common recognition that effective development of large-scale distributed applications requires utilizing off the shelf infrastructure and service components. Furthermore, the usability of the final product is heavily dependent on the properties of the whole as derived from its parts. In this type of environment, visible, predictable, flexible, and integrated resource management strategies within and between the pieces are required. Quality of Service concerns itself with the non-functional aspects of distributed application development. In its most useful forms the non-functional properties extend end to end and thus have elements applicable to the network substrate, the platform operating systems, the system services and the programming system in which they are developed, the applications themselves, as well as the integrating middleware that ties all these elements together. Two basic premises are that 1) different levels of service are possible and desirable under different conditions and costs, and that 2) the level of service in one dimension
must be coordinated with and/or traded off against the level of service in another to achieve the intended overall result.

We separate the non-functional requirements from the functional requirements for two reasons: 1) to allow for the possibility that these will change independently e.g. over different resource configurations for the same applications and 2) based on the expectation that the non-functional properties will be developed and managed by a different set of people than those customarily responsible for the functional properties of an application. There is as yet no consensus on the precise set of dimensions which quality of service encompasses. Much of the current effort centers on providing assurances for attributes such as cost, timeliness (e.g. response time, jitter), volume (throughput), precision, accuracy, synchronization, availability, reliability, and security. Each of these dimensions may be viewed from the different perspectives of service user and service provider, or the individual user vs. the “system,” which aggregates behavior over many users.

Experience with large scale Internet environments shows that the computing and communication environment underlying the distributed applications are undergoing constant change, both in short term resource availability and in longer term configuration change. These changes often require cost-effective adaptation to the new environment. Implementing application-level awareness of changes to expected and delivered Quality of Service is a current trend in the introduction of adaptive behavior into distributed applications. Adaptation can occur at any and all of the various layers of the system, including customized approaches in the application itself and standard service reconfigurations within the supporting infrastructure. An example of an application level adaptation might be moving from full video to audio and still imagery to text only interactions. An example of a service level adaptation might be acquiring additional bandwidth by preempting a lower priority user or automatically instantiating additional resource replicas when another one becomes unreachable. The keys to success in these adaptations are to effectively coordinate the otherwise independent activities so they provide maximum utility to developers and users without conflicting behavior that would might result from a series of independent actions.

Phases

Certain parts of a Quality of Service implementation are independent of the particular quality that is being managed. For each part, there are a wide variety of techniques to accomplish QoS objectives in a complex networked environment. In different implementations, each part may be present to a lesser or greater degree depending on the requirements of the particular environment. Typically, all are present, although not always accessible to end users. Below we enumerate the major phases of resource management associated with QoS.

- Specification
  - Specify normal operating ranges for clients, objects, and system resources
  - Specify desirable reconfigurations and adaptations

- Negotiation
  - Mediate between desirable or expected behavior and available resources

- Enforcement
  - Activities undertaken to control resource acquisition or consumption toward meeting commitments.
Varying degrees of enforcement can be applied in a QoS implementation. One option is not to incorporate enforcement. In this case, notification and adaptation are the primary benefits of a QoS-aware implementation.

Major classes of enforcement mechanisms include resource reservation and admission control.

- Detection
  - Detect when processes or system resources operate outside their negotiated ranges
  - Gather information that will help determine the source of the problem, the extent of the problem, and proper reconfiguration/adaptation strategies

- Notification
  - Information must filter back through each of the systems affected and up through the layers of the protocol stack for each participant
  - Notification may stop at the lowest layer at which successful adaptation is possible, or may propagate to higher layers for the purpose of later analysis.

- Reconfiguration and Adaptation
  - Layers of reconfiguration sustain selective QoS commitments under new operating conditions, or adapt to a new level of QoS compatible with current conditions;
  - Reconfiguration and adaptation are appropriate at the mechanism and resource layers; within middleware; and at the application and user levels.

Security and system management requirements change throughout system lifecycles. Coordinating these changes is a part of providing the required system QoS. Specifically, mediating the need for varying degrees and levels of security, in different situations, is analogous to providing, negotiating, and adapting to system QoS.

**Emerging Concepts**

Because QoS is an area of current investigation and experimentation, it does not yet have commonly understood vocabulary and abstractions. The following are emerging as useful concepts in introducing, achieving, integrating, and composing QoS, based on a sampling of activities in the area.

- **contract**: a summary of expectations, current measured conditions, and rules for adaptation
- **resource reservation**: a resource management strategy that preallocates and dedicates a quantity of a resource
- **adaptation**: the ability to change implementation choices based on changes in the resources available or expected to be available
- **operating region**: a range of measured quality of service, often accompanied by an indication of which behaviors are appropriate for indicated levels of service
- **aspect**: one of a number of QoS dimensions which need to be weaved together to form an integrated QoS implementation
- **composition**: the grouping of individual QoS into a larger granularity, end to end QoS
negotiation: agreements by providers and consumers on the targeted level of QoS to be provided at a specified time

QoS specification: a means for describing the desired or supplied QoS, as well as information associated with managing QoS; sometimes described using a quality description language (QDL) that describes aspects of an application’s QoS and possible regions of desired and actual QoS

monitoring QoS: a function for keeping watch on the level of QoS actually being provided over some time period

QoS Mechanisms: methods for affecting, controlling and changing the level of QoS for a user or class of users; the mechanisms themselves have associated QoS attributes.

QoS Policies: the strategies for effecting a system-wide resource allocation outcome

Time Epochs: the various times at which QoS decisions are made, ranging from design time, to implementation time, to configuration or compile time, to startup time, to on the fly runtime.

Frequently Asked Questions and Open Issues

- How is QoS introduced into an environment?

  There are a variety of ways in which aspects of QoS management have been inserted into applications. These different methods require varying levels of software (re)development. Among the ways that reuse legacy code bases, one technique has been to embed QoS into the application, i.e., the application provides interfaces for specifying QoS parameters while the mechanisms and enforcement are provided by the OS. Another way is to embed QoS management into the resource infrastructure (e.g. communication network), effectively hiding it from the application. Wrappers represent a middle ground where existing objects are made to exhibit QoS properties by sandwiching them between layers of QoS aware software, using predetermined strategies. Combinations of these mixed with new development of QoS abstractions is also being experimented with to support insertion of QoS at many different levels.

  There is general agreement that QoS and functional concerns should be kept as separate as possible. For introducing QoS, some people believe that it can be provided by wrappers and middleware, while others contend that QoS cannot be incorporated through middleware alone, and must be managed at a level where it can be enforced, such as the OS or network. Still others believe that, in many cases, enforcement is not as important as notification and adaptation. That is, rather than trying to guarantee QoS, the system does its best to provide it and tries to adapt when expectations are not met. There are situations in which enforcing QoS requirements are more important than other situations (hard vs. soft QoS requirements). Most likely, QoS will become prevalent in all of these places since they serve different purposes, customized to the current prevailing orientation of their major constituencies.

- How aware should client applications be of QoS: unaware, awareness without pain, immersion?

  There are a wide variety of opinions on the desirability of application-level awareness of QoS. In some situations, applications and users require complete awareness of QoS. Many practitioners believe that some techniques, such as wrappers and embedding QoS into the application itself, provide this level of awareness. In other situations, applications and users want QoS issues to be transparent. However, complete transparency is seldom, if ever, wanted.
As an analogy, airline passengers typically don’t want to worry about QoS, but they want someone (e.g., the pilot, the mechanics) to worry about it. This reasoning motivates the need for another role in software development, that of the QoS engineer. In many cases, the lines between the roles will be blurred, and it’s possible that one person or set of persons will develop both the functional and QoS part of applications. However, in many cases, someone will require a particular quality of service, while someone else will decide what policies and mechanisms are needed to provide it.

- How do we specify or measure it? Is there a common framework?
  There is no single definition of QoS yet, but examples suggest that it can be addressed by a common framework. There is also no well-established notation for describing QoS yet.

- What issues are being addressed now and what are some of the hard problems to overcome?
  Providing QoS means building QoS contracts, striking a balance between conflicting non-functional requirements, and providing the tools to make tradeoffs. This creates a new engineering role, that of the quality engineer with the expertise to make these tradeoffs.

  Providing a system-level notion of QoS and developing the technology base and methodology needed to build applications aware of the quality they need and able to adapt to changes in it.

  Composition of quality contracts both across different granularities and across different attributes is a key new area that will need to be addressed.

  Bridging the gap between the high-level notions of QoS that are appropriate for application convenience, and low-level QoS that mechanisms and resources can provide effectively deeper within the system implementation. For example, OMG currently is developing a specification for real-time CORBA and is soliciting a real-time ORB, i.e., one that does not get in the way of real-time operating systems operating below it.

References

Cross Reference:
Distributed Computing Environments, see OMG, DCOM, DCE
Resource Management Strategies
Reservations, RSVP
Dependability, Fault Tolerance, Security, Synchronization
Multimedia Applications

**Dictionary Terms:**

**Quality of Service**

One of a number of dimensions of requirements and control of properties of a distributed application beyond the functionality provided.