Editor: Harrick Vin
University of Texas at Austin

Multimedia-Friendly Server and Communication Design

Kang Shin, Tarek Abdelzaher, Seungjae Han, John Reumann, and Emmanuel Abram-Profeta
University of Michigan

This article reports on a set of projects to

- design, implement, and evaluate software frameworks for providing performance assurances in emerging Internet services and real-time applications with a focus on multimedia, and
- design and evaluate video-on-demand (VOD) servers as an application.

First we describe the Adaptware project, which investigates adaptive software for server platforms. A complementary project addresses reliable transmission of QoS-sensitive data over packet-switched networks. Together, these two prongs compose an end-to-end approach to achieving flexible QoS guarantees for future Internet applications. We discuss VOD servers as a potential application of this two-pronged approach.

Adaptive software for servers

The Adaptware project addresses the server-side communication subsystem design to exploit application semantics for more efficient use of premium resources. Since the server communication subsystem manages an increasing number of connection end points, a QoS-sensitive implementation may significantly affect the amount of server traffic.

Communication subsystem extensions for servers

We augmented the server-side communication subsystem with the notion of *QoS contracts*, which express QoS requirements on the client's behalf.[1] A QoS contract expresses the nominal QoS level the client requests and a minimum acceptable QoS level for the client. It may also specify the utility (or price) of respective QoS levels and the penalty, if any, for violating the contracted QoS. For example, QoS levels may represent video frame rates and average frame sizes (derived from image resolution), audio bandwidth, and so on. The server will attempt to deliver the highest contracted quality but can degrade quality in case of overload. Our architectural extensions fall in three categories:

- *Contract creation:* Creates and activates a contract for a new client, at which point the server invokes admission control, chooses an optimal QoS level, and allocates a utilization budget for the contract.
- *Request classification:* Sorts client requests by the QoS contract in order to “charge” request execution to the corresponding contract's resource budget. Classification generally occurs according to client Internet protocol (IP) addresses, although other schemes are possible.
- *QoS control*: Enforces that no contract takes away the resources allocated to others and provides the “signed” QoS guarantees.

Figure 1 shows the general architecture of the communication subsystem. Upon creation of a new contract, a QoS mapping module translates contract QoS specifications into resource requirements. The system then performs capacity planning and admission control to determine if enough resources exist to honor the contract. In addition, the system selects a QoS level for each admitted contract according to load conditions, which monitors it dynamically.

We assume a multithreaded communication subsystem model in which each thread handles the flow of a particular contract. We call such a thread a contract handler thread (CHT). Flow control and policing thus convert into a CHT-sched-

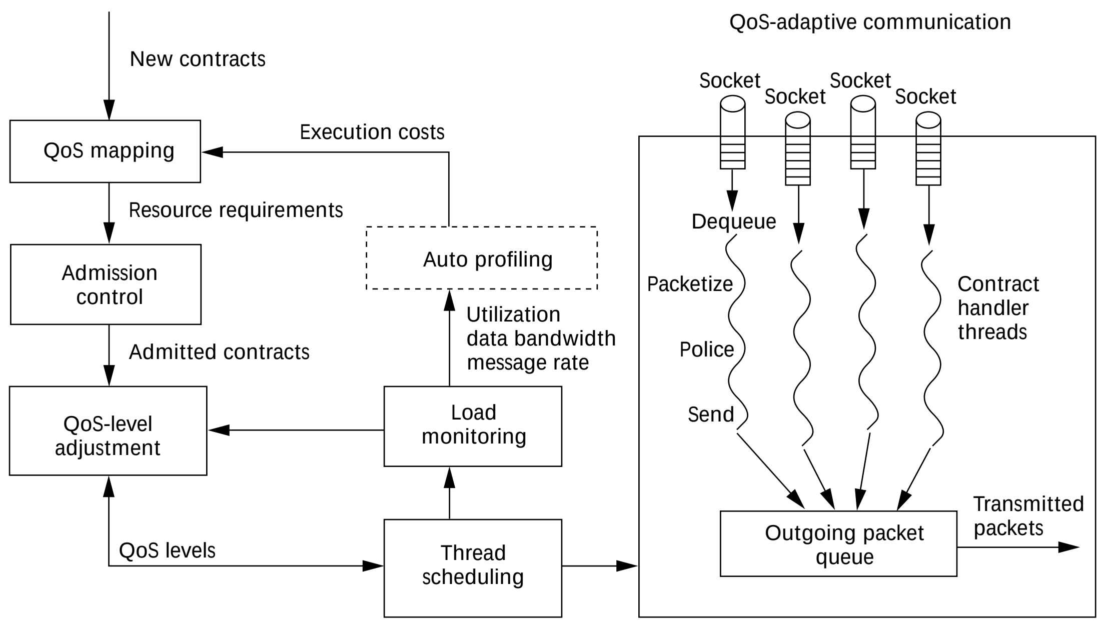

Figure 1. The communication subsystem.

uling problem. To decouple thread scheduling from packet transmission on the network, CHTs deposit their outgoing packets into a data buffer that the network device will later drain. An autoprofiling module computes processing overheads that the system feeds back into the QoS mapping module to better map contract requirements into resource constraints.

Server sharing support

Hosting performance-sensitive services on multipurpose servers becomes more important as multimedia elements become part of best-effort services. Furthermore, the explosive growth of Internet services creates the need for sharing the same execution platform among multiple services. Intranets and proposals for active and smart sensor networks, also, require server sharing for multiple purposes.

To permit simultaneous use of one server or group of servers for several services, we need to insulate services against each other. We provide this insulation via an appropriate extension of the operating system. Once insulated, the system must manage the services' resources and support applications to use these new facilities (see Figure 2).

Addressing operating system issues. Providing interprocess insulation means improving performance predictability of individual processes in a concurrent environment. We propose operating system resource-reservation mechanisms to meet individual processes' resource requirements. We found that a stripped-down version of capacity reserves—resource sharing over a specified base window—without high-level resource-management semantics proves a good approach to take. We can

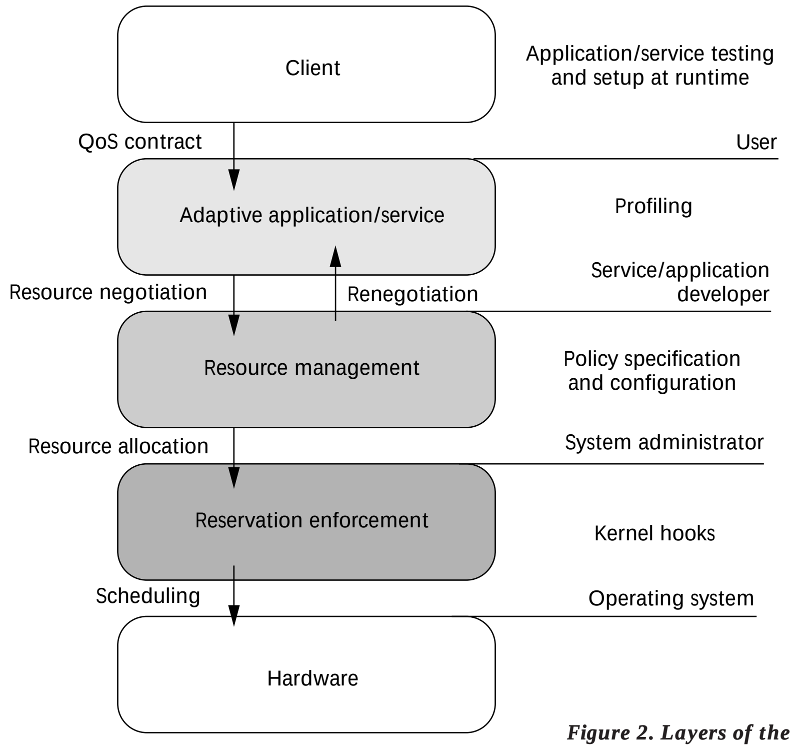

Figure 2. Layers of the resource-management architecture.

guarantee system integrity by requiring superuser privileges when using the new scheduling class for capacity reserves. Processes can be insulated from each other because they can only consume reserved shares of different resources. While identifying CPU capacity reserves as the key to system control, we also found it quite useful to employ additional controls, like I/O capacity reserves. Nevertheless, the additional predictability achieved for Internet-type services remains orders-of-magnitude lower than what we can achieve by accurately controlling processor arbitration.

Focusing on resource management. By resource management, we mean a combination of admission control, resource allocation, and graceful degradation. It becomes necessary once resources can be allocated for quasiprivate use. Admission control avoids generating unanticipated overload situations. Graceful degradation improves system response to detected overload. Instead of requiring a specific resource management behavior to develop a framework, we identified the key components of resource management: admission control, measurement of degradation, fairness, and prioritization. This simplifies the design of resource-management strategies.

In our model, resource management resides on top of a basic reservation mechanism and uses a measure—called *adaptability*—to evaluate each process' ability to adapt. Generally, the system designer can tune the adaptability to certain system characteristics without having to reimplement the adaptation strategy, which balances the degradations experienced by all processes in the system. In fact, we measure degradation in terms of adaptability.

To obtain a concrete instance of adaptability, we took an interval-based resource reservation scheme as an example. Adaptability in this example defines resource management characteristics as follows.

1. *Admission control*: Minimal requirements must not exceed system resource capacity.

2. *Fairness*: All processes get an amount of resources resulting in the same amount of adaptability loss.

3. *Graceful degradation*: Processes degrade in accordance with their own preference for degradation and are informed of changes in their resource allocation.

Our general framework describes common and uncommon resource-management strategies quite well. It can thus be seen either as some sort of a template library or as a conceptual framework for developing resource-management strategies.

Providing support for adaptive/adaptable applications. Graceful degradation in our resource management—necessary to accommodate a growing number of services—may cause an individual service's resource availability to change. Therefore, individual services must be adaptive. At present, it's difficult to write an application/service that exhibits a "reasonable" responsive behavior when the system reallocates resources at the resource-management level.

In the previous section, we noted that QoS levels let applications react to changes brought on by flexible resource management.[2] This scheme also solves the problem of degrading the service when its internal load rises beyond a manageable level. To derive an objective function, which guides the suballocation of a service's resources to individual sessions, we used environmental information. We showed that measures given to evaluate the performance of the server in terms of consumer-perceived usefulness can create an adequate performance model, which in turn defines an adaptation strategy under overload or degradation scenarios.

Our abstraction builds on the concept of client-server sessions, which execute in a specific mode or *service class*. This mode can change dynamically subject to resource availability. Furthermore, the effect on the entire system of operation in a specific service class is captured by objective functions stored in a *subscription class*. Once the system identifies these two sides, service classes can be coded as different modes in our moded-thread library, and subscription classes can be declared in configuration files read at service startup time.

Our runtime system maximizes the value of the objective function on the fly. We've shown that our model trades off the number of adaptation operations necessary for optimal evaluation of the objective function. While the value of the objective function decreases linearly through the setting of certain configuration parameters, the number of adaptation operations followed an exponential decline. The work on the application side of adaptation showed that it's possible to implement efficient adaptive services, thus justifying our insulation/adaptation approach to resource management.

The future

Regarding operating system-level interservice insulation, we're currently working to achieve insulation without requiring resource reservations as explicit as capacity reserves. Instead of specifying resource requirements upfront, we want to support the initial specification of a coarse resource need estimate, which is refined during a service's execution. This makes capacity reservation portable between different execution platforms, thus greatly facilitating distributed service management.

At the resource-management layer, we're planning to incorporate monitoring to let the resource manager deal with many different types of workloads effectively.

The lack of good abstractions for adaptation at the application level remains the biggest hurdle to overcome. Instead of coding explicit modes into software, which forces the programmer to make adaptation choices at compile time, we would rather defer adaptation decisions as much as possible. Therefore, we're seeking to separate resource, QoS, and functional requirements so that programmers don't have to worry about them at the same time. Instead, they would implement options, which the resource manager would use at runtime depending on how adaptation works best on the service's platform. The methods found in the framework of Aspect-Orientation[3] appear the key to achieving our goal.

Reliable multimedia communication

The second project deals with the reliable transmission of multimedia data over packet-switched networks (such as the Internet). Distributed multimedia applications typically need a certain QoS guarantee (for example, bandwidth, delay, and delay jitter) from the network, which requires special communication services other than conventional best-effort services. We've extensively researched various issues of QoS guarantees, such as priority-based packet scheduling, hardware support for multimedia data transmission, end-host operating system issues, resource reservation, QoS routing, and so on. But here we'll focus on dependability support for QoS-guaranteed communication.

The packet-loss rate typically represents a communication service's dependability. In modern optical networks, most packet losses occur because the buffer overflows. This results from traffic congestion at network switches/routers, since the use of optical fibers reduces the transmission-error rate to a negligible level. Congestion-induced packet losses can be avoided or minimized by resource reservation in QoS-guaranteed communication. Plus, the packet loss rate can serve as a performance parameter of admission-control schemes with statistical guarantees rather than a measure of dependability. Moreover, multimedia applications can tolerate transient packet losses.

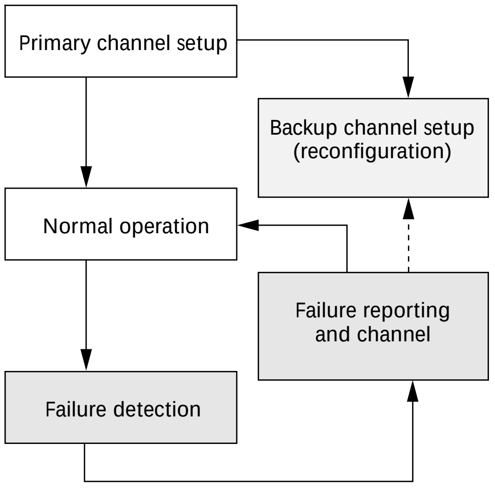
Primary channel setup
Backup channel setup (reconfiguration)
Normal operation
Failure reporting and channel
Failure detection

Figure 3. Overview of our failure-recovery scenario.

Thus, in multimedia communication, the dependability in a larger time scale (that is, session-management level as opposed to packet level) proves more meaningful. We use *connection availability*—the probability of a connection being available at any given time—for this purpose. For high availability, the network has to quickly restore the QoS connections affected by component failures (that is, failures of links or routers). Since a QoS guarantee can be realized by reserving resources on a static path and transferring packets only via that path—essentially establishing a virtual circuit—restoring a failed QoS connection requires establishing a new virtual circuit. Similar to the approach used in telephone networks that bound the service disruption delay to less than a second, we set up a backup channel for each QoS connection a priori to guarantee quick, successful failure recovery. However, our scheme[4] targets QoS communication over the Internet and substantially differs from that of telephone networks in many aspects.

Fast failure recovery occurs in three steps: backup channel establishment, failure detection and channel switching, and postrecovery resource reconfiguration. Figure 3 gives an overview of the failure recovery scenario.

A backup channel doesn't carry any data until activated, so it doesn't consume resources in a normal situation. However, a backup channel isn't free, since it requires the same amount of reserved resources as its primary channel in order to provide the same QoS as its primary upon activation. We call these resources *spare resources*. In a normal situation best-effort traffic can use spare resources, but they can't be used to accommodate other QoS connections. As a result, equipping each QoS connection with a single backup, which will be routed disjointly with its primary, reduces the network capacity of accommodating QoS connections by 50 percent or more. To alleviate this problem, we developed *backup multiplexing*, a resource-sharing technique that at each hop reserves only a small fraction of resources needed by all the backup channels taking the hop.

Backup multiplexing has three unique characteristics. First, it allows per-connection dependability control. That is, each connection may request a different availability level depending on its application criticality. Second, it doesn't require global knowledge of the traffic demand over the entire network. Thus, it's scalable and works in an environment where connections are dynamically set up and torn down. Third, it applies to any QoS communication scheme because it uses a meta-admission test, so the admission test of any QoS-communication scheme can be plugged in. This way, backup channels can be run through multiple heterogeneous networks.

When a QoS connection fails, it should be quickly detected and reported to the connection's end nodes so that the damaged connection can be switched to its healthy backup. We use a simple heartbeat protocol (similar to the Hello protocol) as a primary means of failure detection. As a secondary means, we use an end-to-end protocol, which detects connection failures using negative acknowledgments. The measurement in our experimental testbed reveals that we can achieve high-detection coverage and low-detection latency with these protocols. We also developed a protocol for failure reporting and channel switching. After channel switching, the network reconfigures resources to preserve the dependability of the QoS connections directly or indirectly affected by failures. New backups should be established for the QoS connections that lack backups. We're currently extending the unicast solution to dependable multicast services.

Application

Now we'll discuss VOD servers as a potential application of our end-to-end approach. Seen today as one of the most promising applications emerging from advances in networking, large-scale interactive digital video service has the potential to provide millions of customers with convenient and unrestricted access to large collections of movie titles. Customers' QoS in interactions proves critical to such deployment, given that economical viability of VOD depends on customers' appreciation of the service delivered. Based on the premise that the architectural decisions in VOD end systems—the local VOD server and customers' premise equipment (CPEs)—affect customers' level of interactivity, our research aims to combine two orthogonal requirements:

1. service scalability, or the ability to accommodate possibly significant workload fluctuations and overloading situations without affecting customers' QoS in interactions, and

2. long-term, cost-effective resource allocation so that a large collection of movie titles and channels can be accessed by a maximum number of viewers.

Current status

Our research has focused on the following areas.

Storage organizations for VOD service. We showed that hybrid storage organizations, in which videos first get partitioned into several disk-array-based clusters, then striped across each disk array using coarse-grained striping (CGS), achieve a cost-effective video repository, low-admission latency, and high availability. To provide further guidance in choosing a storage organization, we examined at what cost a minimal "playback-only" service could be upgraded to achieve fully interactive "true" VOD.

As an alternative to disk striping, we felt it was also important to identify practical situations in which information systems designers could choose the one-movie-per-disk (OMPD) storage scheme, since this scheme proves easier to deploy, upgrade, and experiment with when the service expands. We showed that although many regard OMPD as a naive approach to storage configuration, it might actually be acceptable for highly reliable service of a small number of movies to a large user population.

Optimal scheduling in near-VOD systems. Batching requests made by many different viewers for the same movie proves a cost-effective way to improve service scalability. For instance, the commercially available near-VOD systems schedule video programs for the same movie title at regularly staggered intervals, called *phase offsets*. We introduced an analytical yet realistic approach to program scheduling that integrates customers' and service provider's points of view. We then derived the optimal schedule of movies of different popularities for various performance metrics such as throughput. We also analyzed variations on near-VOD service, such as scheduling channels based on a threshold of requests.

Scalable batching policies. After focusing on the commercially available near-VOD, it's equally important to systematically identify and compare other scalable batching schemes based on various factors for their possible impact on batching performance. These factors include

1. nonstatic request arrival rates,

2. storage organization of the VOD server, and

3. customers' willingness to wait for service.

When batching occurs "on demand," without any control in the channel allocation process, we found the following three factors make the system prone to channel clumping. This means a large fraction of the VOD server channel capacity tends to be allocated within a short period of time, thus triggering short- and long-term congestion cycles:

1. the inherent lack of variability in movie lengths,

2. the access locality to movie titles, and

3. the small number of different movie titles relative to the number of channels allocated to them.

Based on this observation, we evaluated[5] various scalable batching policies and identified simple variations of near-VOD that vastly improve uncontrolled on-demand channel allocation, while being easily implementable in both monolithic and clustered disk arrays.

Scalable fully interactive VOD service. Full support for interactive VCR functions can be achieved only by dedicating a channel for each customer. Unfortunately, channel dedication degrades scalability in a multicast VOD system. Therefore, we introduced a scheme[6] for fully interactive, yet scalable multicast VOD. In the proposed framework, support for interactive operations requires

1. allocating a fraction of the VOD server's channel capacity to handle interactive operations that would otherwise be blocked and

2. using partial caching of programs in CPEs to merge customers back in synchronization with broadcast channels.

Unlike shared buffers located closer to the server, we showed that small individual CPE buffers can be used in synergy with dedicated video channels to allocate and reclaim resources more efficiently than existing schemes, therefore providing unrestricted VCR functionality. We confirmed these results for a wide range of CPE buffer sizes, channel capacities, and VCR action characteristics.

Future directions

Our future research will cover the following areas.

Interactive behavior modeling. Modeling consumer use of continuous digital media, albeit inherently speculative, is inevitable to identify or anticipate the human factors and worst-case scenarios that will challenge system scalability. Nevertheless, throughout VOD deployment, the same emphasis should be put on actual monitoring of human-media interactions. This twofold effort requires appropriate tools located in end systems and delivery networks to support iterative refinements through implementation and experimentation.

Efficient support for VCR-like interactions. For illustrative purposes, we made several simplifying assumptions on video transport and support for interactions. For instance, we assumed all frames were of the same size and VCR actions didn't incur loss of frames. Since most of the mechanisms presented can be adapted to variable bit-rate (VBR) video, future work will determine whether our framework works as efficiently when we relax the assumption of equal-size frames.

Scalability in other interactive Internet applications. Various large-scale Internet applications are being deployed before VOD. In the field of interactive networked games, for instance, there exists a need to provide a service that is both fun and fast, and scalability per se remains more of a perpetual design constraint than a new feature that would be brought into a system. This holds true with client-server models of game architectures, since the requirement to handle a large number of users from a restricted set of game servers introduces a host of new constraints that game developers attempt to satisfy on a best-effort basis.

These design constraints mirror the problems observed with large-scale VOD. Even in the most casual Internet games (such as card games), where users have a high tolerance to network latencies or bandwidth scarcity, the challenge lies in devising a distributed architecture that can accommodate large numbers of long-lasting connections over wide-area networks and that scales as the demand fluctuates throughout the day. To achieve this goal, we're currently investigating the synergistic combination of message coalescence, that is, the delayed broadcast of agglomerated update messages; asynchronous or rate-controlled transmission of those messages; and hierarchical information caching. MM

References

1. T. Abdelzaher and K. Shin, "End-Host Architecture for QoS-Adaptive Communication," *Proc. IEEE Real-Time Technology and Applications Symp.*, IEEE Press, Piscataway, N.J., June 1998, pp. 121-130.
2. J. Reumann and K. Shin, "Adaptive Quality-of-Service Session Management for Multimedia Servers," *Proc. Int'l Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, IEEE Press, Piscataway, N.J., July 1998, pp. 303-316.
3. G. Kiczales et al., "Aspect-Oriented Programming," *Proc. European Conf. on Object-Oriented Programming*, Springer-Verlag, Berlin, June 1997, pp. 220-242.
4. K.G. Shin and S. Han, "Fast Low-Cost Failure Recovery for Reliable Real-Time Multimedia Communication,'' *IEEE Network*, Vol. 12, No. 6, 1998, pp. 56-63.
5. E.L. Abram-Profeta and K.G. Shin, "Comparative Study of Scalable Batching Policies in Disk-Array-based Deterministic Video-on-Demand Servers,'' *Proc. IEEE Int'l Conf. on Computer Communications and Networks*, IEEE Press, Piscataway, N.J., Oct. 1998, pp. 682-689.
6. E.L. Abram-Profeta and K.G. Shin, "Providing Unrestricted VCR Functions in Multicast Video-on-Demand Servers," *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems*, IEEE Press, Piscataway, N.J., June-July 1998, pp. 66-75.

Readers may contact the authors at the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122, e-mail {kgshin, zaher, sjhan, reumann}@eecs.umich.edu.