Multimedia-Friendly Server and Communication Design

Kang Shin, Tarek Abdelzaher, Seungjae Han, John Reumann, and Emmanuel Abram-Profeta
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
{kgshin, zaher, sjhan, reumann}@eecs.umich.edu

1 Introduction

This article reports on a set of projects to (i) design, implement, and evaluate software frameworks for providing performance assurances in emerging Internet services and real-time applications with a focus on multimedia; and (ii) design and evaluate video-on-demand (VoD) servers as an application. In the first project we have taken a two-pronged approach to providing performance assurances. In the first thrust, particular attention is given to new resource-management mechanisms at both OS and middleware layers for next-generation servers The second thrust deals with the problem of achieving reliable communication QoS guarantees. Together, the proposed techniques alleviate bottleneck resource congestion, exploit application-level semantics to adapt to load conditions, and reduce the cost and time of real-time software development by providing *reusable* multi-purpose real-time software components for achieving QoS guarantees. Section 2 describes the *Adaptware* project which investigates adaptive software for server platforms Section 3 presents a complementary project that 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. Finally, Section 4 presents VoD servers as a potential application of the above two-pronged approach.

2 *Adaptware*: Adaptive Software for Server Platforms

This project addresses the server-side communication subsystem design to exploit application semantics for better utilizing 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. As described in Section 2.1, we propose, implement, and evaluate an incrementally-deployable server-side communication subsystem architecture that maximizes the aggregate utility of QoS-sensitive connections over a community of clients even in the case of temporary overload. In Section 2.2 we develop mechanisms for platform sharing among multiple server processes, and explore the implications on OS design and resource management.

2.1 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 [5]. A QoS contract expresses the nominal QoS level the server is requested to deliver and a minimum acceptable QoS level for the client. It may also specify the utility (or

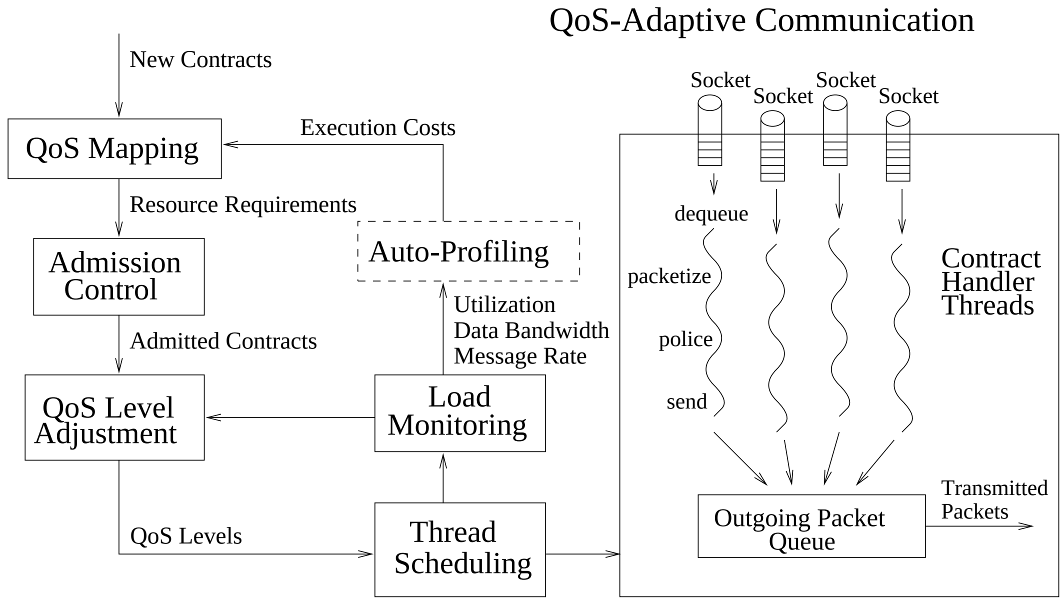

Figure 1: The communication subsystem

price) of respective QoS levels and the penalty, if any, for violating the contracted QoS. QoS levels may represent, for example, video frame rates and average frame sizes (e.g., derived from image resolution), audio bandwidth, etc. 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* allows a contract to be created and activated for a new client, at which point admission control is invoked, an optimal QoS level is chosen, and a utilization budget is allocated 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 is generally done according to client IP addresses, although other schemes are possible. *QoS Control* enforces that no contract takes away the resources allocated to others and makes sure to provide the "signed" QoS guarantees.

Figure 1 depicts its general architecture of the communication subsystem. Upon creation of a new contract, a QoS mapping module translates contract QoS specification into resource requirements. Capacity planning and admission control are then performed to determine if there are enough resources to honor the contract. A QoS level is selected for each admitted contract according to load conditions, which are monitored dynamically. We assume a multithreaded communication subsystem model, in which each thread handles the flow of a particular contract, and is therefore called a *Contract Handler Thread* (CHT). Flow control and policing are thus converted into a CHT-scheduling problem. To decouple thread scheduling from packet transmission on the network, CHTs deposit their outgoing packets into a data buffer which will later be drained by the network device. An auto-profiling module computes processing overheads that are fed back into the QoS mapping module to better map contract requirements into resource constraints.

2.2 Server Sharing Support

Hosting performance-sensitive services on multi-purpose servers becomes more important as multimedia elements become part of traditionally pure best-effort services. Furthermore, an explosively growing number of Internet services creates the need for sharing the same execution platform among multiple services. Intranets and proposals for active and smart sensor networks, too, require a server to be shared

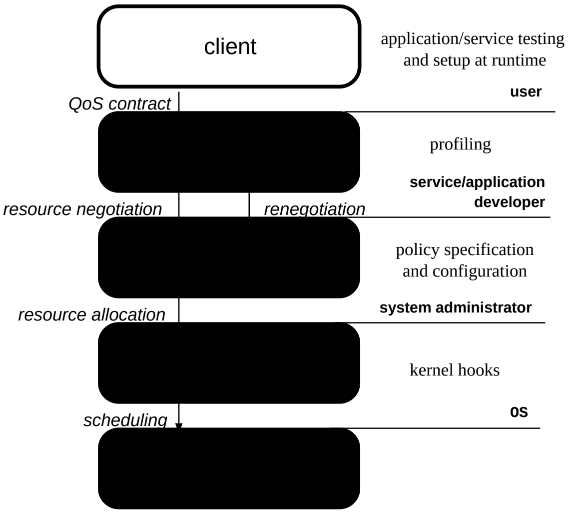

Figure 2: Layers of the resource-management architecture

for multiple purposes. The driving force behind this trend is economic efficiency.

To permit simultaneous use of one server or group of servers for several services, one needs to insulate services against each other. We provide this insulation via an appropriate extension of the OS. Once services' access to resources is insulated, it is necessary to manage the resources and provide good support for applications to use these new facilities (see Figure 2).

Addressing OS issues: Providing inter-process insulation means to improve performance predictability of individual processes in a concurrent environment. We propose OS resource-reservation mechanisms in order to meet individual processes' resource requirements. We found that a stripped-down version of capacity reserves — resource shares over a specified base window — without any high-level resource management semantics, is a good approach to take. It is possible to guarantee system integrity by requiring superuser privileges in using the new scheduling class for capacity reserves. Processes can be insulated from each other as 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 is orders-of-magnitude lower than what we can achieve by controlling processor arbitration accurately.

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

In our model resource management resides on top of a basic reservation mechanism and uses a measure, which we call *adaptability*, to evaluate each process' ability to adapt. Generally, the system designer can tune the adaptability to certain system characteristics without having to re-implement the adaptation strategy, which balances the degradations experienced by all processes in the system. Degradation itself is measured in terms of adaptability. We took an interval-based resource reservation scheme, i.e., [min, max], as an example to obtain a concrete instance of adaptability, which in turn defines (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; and (3) *graceful degradation* — processes are degraded in accordance with their own preference for degradation and are also informed of changes in their resource allocation. Our general framework is shown to be able to describe common and uncommon resource-management strategies quite well and can thus be seen either as some sort of a template library, or as a conceptual framework for the development of resource-management strategies.

Providing support for adaptive/adaptable applications: Graceful degradation in our resource management — that is necessary to accommodate a growing number of services — may cause the resource availability seen by an individual service to change. Therefore, individual services must be adaptive. At present, it is not easy to write an application/service that exhibits a "reasonable," responsive behavior when resources are re-allocated at the resource-management level.

We have used the notion of QoS levels that was presented in the previous section to allow applications to react to changes brought on by flexible resource management [2]. An added benefit of this scheme is that it 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 sub-allocation of a service's resources to individual sessions, we are using environmental information — economical information in our case study. In fact, we showed that measures given to evaluate the performance of the server in terms of consumer-perceived utility can be used to create an adequate performance model, which in turn defines an adaptation strategy under overload, or in degradation scenarios. Our abstraction is based on the concept of client-server sessions, which are executed in a specific mode, or *service class*. This mode can be changed dynamically subject to resource availability. Furthermore, the impact of operation in a specific service class on the entire system is captured by objective functions stored in a *subscription class*. Once these two sides are identified, service classes can be coded as different modes in our moded thread library and subscription classes can be declared in configuration files that are read at service startup time. Our runtime system maximizes the value of the objective function on-the-fly. It has been shown that our model allows to trade off the number of adaptation operations necessary for optimality of the objective function evaluation. While the value of the objective function only 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 is possible to implement efficient adaptive services, thus justifying our insulation/adaptation approach to resource management.

The Future: On the level of OS-level inter-service insulation we are currently working to achieve insulation without requiring resource reservations as explicit as capacity reserves. Instead of specifying resource requirements up-front, we want to support 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 are planning to incorporate monitoring to enable the resource manager to deal with many different types of workloads effectively.

The lack of good abstractions for adaptation at the application level is probably 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 are seeking to separate resource, QoS, and functional requirements so that a programmer need not worry about all of the above at the same time. Instead, he would implement options, which would be used by the resource manager at runtime, depending on how adaptation works best on the

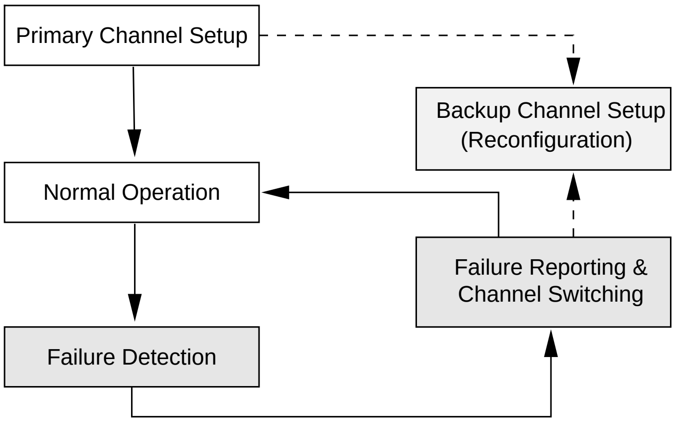

Figure 3: Overview of our failure-recovery scenario

service's platform. The methods found in the framework of Aspect-Orientation appear to be the key to achieving our goal.

3 Reliable Multimedia Communication

The second project deals with the reliable transmission of multimedia data over packet-switched networks (e.g., Internet). Distributed multimedia applications typically need a certain QoS guarantee (e.g., bandwidth, delay, and delay-jitter) from the network. It is widely believed that special communication services other than conventional best-effort services are necessary to provide the QoS guarantees required by these multimedia applications. We have been extensively researching 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 will focus on dependability support for QoS-guaranteed communication.

The dependability of a communication service is typically represented as a packet loss rate. In modern optical networks, most packet losses are attributed to the buffer overflow resulting from traffic congestion at network switches/routers, as the use of optical fibers reduces the transmission error rate to a negligible level. Congestion-induced packet losses are avoided or minimized by resource reservation in QoS-guaranteed communication, and the packet loss rate is used as a performance parameter of admission-control schemes with statistical guarantees rather than a measure of dependability. Moreover, multimedia applications are relatively tolerant of transient packet losses. Thus, in multimedia communication, the dependability in a larger time scale (i.e., session-management level as opposed to packet level) is more meaningful. We use *connection availability*[1] for this purpose. For high availability, the network has to quickly restore the QoS connections inflicted by component failures (i.e., failures of links or routers). Since a QoS guarantee is realized by reserving resources on a static path and transferring packets only via that path, which is essentially a procedure of establishing a virtual circuit, restoration of a failed QoS connection requires establishment of a new virtual circuit. Similar to the approach used in telephone networks which bound the service disruption delay to less than a second, we set up a backup channel for each QoS connection *a priori* to guarantee successful and quick failure recovery. Though its fundamental idea is similar, our scheme [1], which targets at QoS communication over the Internet, is substantially different from that of telephone networks in many aspects.

[1] Informally defined as the probability of a connection being available at any given time

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

A backup channel does not carry any data until it is activated, so that it does not consume resources in a normal situation. However, a backup channel is not free, as it requires the same amount of resources to be reserved as its primary channel, in order to provide the same QoS as its primary upon its activation. The resources reserved for a backup connection is called *spare resources*. In a normal situation spare resources can be used by best-effort traffic, but they cannot 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 % or more. To alleviate this problem, we have developed a resource-sharing technique, called *backup multiplexing*, which, at each hop, reserves only a small fraction of resources needed by all of 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 does not require the global knowledge on the traffic demand over the entire network. Thus, it is scalable and can be used in an environment where connections are dynamically set up and torn down. Third, it is applicable to any QoS communication scheme, because it uses a meta-admission test, so that 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 failure occurs to a QoS connection, it should be quickly detected and reported to the connection's end nodes, so that the inflicted connection can be switched to its healthy backup. We use a simple heartbeat protocol (which is similar to the Hello protocol) as a primary means of failure detection. As a secondary means, an end-to-end protocol, which detects connection failures using negative acknowledgments, is used. 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, resource reconfiguration is conducted to preserve the dependability of the QoS connections which are directly or indirectly affected by failures. New backups should be established for the QoS connections which lack backups. We are currently extending the unicast solution to dependable multicast services.

4 Service Scalability in Video-on-Demand End-Systems

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 is of paramount importance to such deployment, given that economical viability of VoD depends on customers' appreciation of the service delivered. Based on the premise that customers' level of interactivity is affected by the architectural decisions in VoD end-systems — the local VoD server, and customers' premise equipment (CPEs) — our research aims at combining 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.

4.1 Current Status

Our research has focused on the following areas.

Storage organizations for VoD service: We showed that hybrid storage organizations, in which videos are first partitioned into several disk-array-based clusters and then striped across each disk array using coarse-grained striping (CGS), should be chosen for cost-effective video repository, low admission latency, and high availability. To provide further guidance in the choice of a storage organization, we examined at what cost a minimal "playback-only" service could be upgraded to the fully-interactive "true" VoD. As an alternative to disk striping, it is also important to identify practical situations in which the one-movie-per-disk (OMPD) storage scheme could be chosen, as this scheme is easier to deploy, upgrade, and experiment with when the service is expanding. We showed that even though it is often regarded as a naive approach to storage configuration, OMPD may actually be acceptable for highly reliable service of a small number of movies to a large user population.

Optimal scheduling in Near VoD systems: A cost-effective way to improve service scalability is to batch requests made by many different viewers for the same movie. In the commercially-available Near-VoD (NVoD) systems for instance, video programs for the same movie title are scheduled at regularly-staggered intervals, called *phase offsets*. We introduced an analytical, yet realistic approach to program scheduling, which 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 NVoD service, such scheduling channels based on a threshold of requests.

Scalable batching policies: After focusing on the commercially available NVoD, it is 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) non-static request arrival rates; (2) storage organization of the VoD server; and (3) customers' willingness to wait for service. When batching is done "on-demand", without any control in the channel allocation process, three factors are found to make the system prone to channel clumping, by which 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. These factors are (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 comparatively evaluated [3] various scalable batching policies and identified simple variations of NVoD which 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. We, therefore, introduced in [4] a scheme for fully-interactive, yet scalable multicast VoD. In the proposed framework, support for interactive operations is provided by (1) allocating a fraction of the VoD-server's channel capacity to handle interactive operations which 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, and therefore provide unrestricted VCR functionality. These results were confirmed for a wide range of CPE-buffer sizes, channel capacities, and VCR actions characteristics.

4.2 Future Directions

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 in order to support iterative refinements through implementation and experimentation.

Efficient Support for VCR-like Interactions: Several simplifying assumptions on video transport and support for interactions were made for illustrative purposes. For instance, all frames were assumed to be of the same size and VCR actions did not incur loss of frames. As most of the mechanisms presented can be adapted to variable bit-rate (VBR) video, future work will determine whether our framework can be used as efficiently as presented when the assumption of equal-size frames is relaxed.

Scalability in Other Internet Interactive Application: Various large scale Internet applications are being deployed before VoD. In the field of interactive networked games, for instance, there is a need to provide a service that is both (1) fun, and (2) fast, and scalability per se is more of a perpetual design constraint than a new "feature" that would be brought into a system once and for all. This is particularly true with client-server models of game architectures, as 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 are reminiscent of the problems observed with large-scale VoD. Even in the most "casual" of the Internet games (e.g., card games), where the tolerance to network latencies or bandwidth scarcity is quite high, the challenge is to devise a distributed architecture that can accommodate large numbers of long-lasting connections over wide-area networks, and scales as the demand fluctuates throughout the day. As a way to achieve this goal, and based on our results in the VoD field, we are currently investigating the synergistic combination of (1) message coalescence, that is, the delayed broadcast of agglomerated update messages; (2) asynchronous or rate-controlled transmission of those messages, and (3) hierarchical information caching.

References

[1] K. G. Shin and S. Han, "Fast low-cost failure recovery for reliable real-time multimedia communication," *IEEE Network*, vol 12, no 6, pp 56–63, 1998.

[2] John Reumann and Kang G. Shin. "Adaptive Quality-of-Service Session Management for Multimedia Servers," *NOSSDAV'98*, July 1998.

[3] E. L. Abram-Profeta and K. G. Shin, "Comparative study of scalable batching policies in disk-array-based deterministic video-on-demand servers," *IEEE International Conference on Computer Communications and Networks*, Lafayette, LA, October 1998.

[4] E. L. Abram-Profeta and K. G. Shin, "Providing unrestricted VCR functions in multicast video-on-demand servers," *Proc. IEEE International Conference on Multimedia Computing and Systems*, pp. 66-75, Austin, TX, June-July 1998.

[5] Tarek F. Abdelzaher and Kang G. Shin, "End-host Architecture for QoS-Adaptive Communication," *Proc. IEEE Real-Time Technology and Applications Symposium*, Denver, Colorado, June 1998.