QoS-Centric Stateful Resource Management in Information Systems
Ye, Nong
Information Systems Frontiers; Jul 2002; 4, 2; ProQuest
pg. 149

Information Systems Frontiers 4:2, 149–160, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

QoS-Centric Stateful Resource Management in Information Systems

Nong Ye
Arizona State University, Information and Systems Assurance Laboratory, P.O. Box 875906, Tempe, AZ 85287-5906, USA
E-mail: nongye@asu.edu

Abstract. *There is little resource management and Quality of Service (QoS) guarantee in the "best-effort" model underlying existing information systems, leaving information systems vulnerable to exploits and denial-of-service attacks. To overcome these problems, an engineering approach to QoS-centric stateful resource management in information systems is presented in this paper. System engineering principles are first used to identify various scales and levels of resources in information systems. QoS attributes of resources are then discussed. We also introduce the topological and algebraic structures of propagating QoS attributes across levels and scales of various resources for service contracting and admission control. We use a control-theoretic structure to specify QoS-guarantee functions for managing individual resources, including functions of admission control, scheduling & control, QoS conformance monitoring, and state probing and testing, along with scheduling techniques and statistical process control (SPC) techniques to support these functions.*

Key Words. *information systems, quality of service, resource management, internet, cyber attacks*

Introduction

In this paper, we consider an information system as a network of computational resources within an administrative domain of an organization. Computational resources may include computation resources (e.g., CPU), data resources (e.g., files and databases), communication resources (e.g., routers and communication links), visualization resources (e.g., printers and screens), and so on. The information system is used to support information operations within an organization and across the organizational boundary. Information infrastructures such as Internet and Computational Grids (Foster and Kesselman, 1999) have enabled the sharing of computational resources across many information systems on the network of an information infrastructure for distributed information storage/retrieval, web service, parallel computation, fault tolerance through geographically distributed redundant resources, and so on.

However, the "best-effort" model underlying existing information systems leaves information systems vulnerable to malicious exploits and denial-of-service attacks. In the "best-effort" model, a computational resource does its best to satisfy service requests from users as long as the resource is not fully utilized. That is, the resource is available to use regardless of the *state* of resources, leading to a danger of completely depleted resources by malicious attackers. For example, a denial-of-service attack can be accomplished by simply sending a large number of service requests to a computational resource (e.g., telnet server) within a very short period of time to deplete the resource and thus deny the service of the resource. Intrusions into information systems present a significant threat to our society (Barnes, 1998; Boulanger, 1998; Garfinkel and Spafford, 1996; Godwin, 1999; Jajodia, Ammann, and McCollum, 1999; Kaufman, Perlman, and Speciner, 1995; Mann, 1999; Neumann, 1999; Simons, 2000; Stallings, 1995). Moreover, under the "best-effort" model, quality of service (QoS) for a user's application is not guaranteed because other users may emerge at any time to compete for and share computational resources. Therefore, there is little resource management and QoS guarantee in existing information systems. The "best-effort" model can be satisfactory if information systems are used in isolation without the sharing of resources with other information systems and users on the network of an information infrastructure. However, once the sharing of resources occurs in the environment

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

of networked computing and communication, the "best-effort" model creates the environment for denial-of-service attacks and many other forms of exploits.

As we increasingly rely on information systems to support critical operations in defense, banking, telecommunication, transportation, electric power and many other domains, information systems must be protected from malicious exploits and intrusions to provide QoS guarantees through stateful resource management. That is, a resource is *stateful* rather than *stateless*. The use of a resource must be controlled with respect to the *state* of the resource such as the available capacity. For example, if the level of a requested service from a resource exceeds the available capacity of the resource, the requested service must be rejected. If a requested service from a resource is admitted to receive the service from the resource, the requested level of service must be satisfied and maintained for QoS guarantee. QoS guarantee for information systems will increase the dependability of information systems, especially robustness to intrusions, and enhance the quality of service from information systems.

Although work on the QoS-centric management of communication resources exists (Aurrecoechea and Hauw, 1996; Giroux and Ganti, 1999), the QoS-centric management of other computational resources in information system must also be established. This paper outlines an engineering approach to the QoS-centric stateful resource management in information systems. System engineering principles are first used to identify various scales and levels of resources in information systems. QoS attributes of resources are then discussed. We also introduce the topological and algebraic structures of propagating QoS attributes across levels and scales of various resources for service contracting and admission control. We use a control-theoretic structure to specify QoS-guarantee functions for managing individual resources, including functions of admission control, scheduling & control, QoS conformance monitoring, and state probing and testing, along with scheduling techniques and statistical process control (SPC) techniques to support these functions. We hope that the theoretical discussions of this engineering approach to the QoS-centric stateful resource management will stimulate vast interests and research efforts in this approach towards highly dependable information systems.

A Hierarchy of Resources in Information Systems

In an information system, resources depend on each other to deliver services. For example, software resources such as a web server request services from hardware resources such as CPU time and memory. The following are typical resources in an information system:

- Software resources, such as the operating system (e.g., Window NT), servers (e.g., web server) and application programs (e.g., Microsoft WORD);
- Information resources, such as data files and databases;
- Hardware resources to implement software resources and information resources, such as processing devices such as CPU, storage/retrieval devices such as RAM and hard drive, communication devices such as routers and network links, visualization devices such as computer screen and printer, and so on.

The dependency of resources in the information system results in a hierarchy of resources in the information system. Several studies on system engineering report a four-level abstraction hierarchy that exists for understanding and managing many large-scale engineering systems such as nuclear power plants, manufacturing systems, and even software system (Rasmussen, 1986; Ye, 1996a, 1996b). The four levels of abstraction are the objective level, the conceptual level, the functional level and the physical level.

At the objective level, goals of the system are stated. At the conceptual level, goals of the system are transformed into operational definitions of system behavior in terms of time phased and sequenced operations. At the functional level, each operation at the conceptual level is broken down into activities that are performed by functional subsystems and functional components in these subsystems. At the physical level, each activity at the functional level is implemented by tasks that are executed by physical units implementing functional subsystems and components. From the objective level to the physical level, the focus shifts from a global, abstract and general understanding of the system itself to a local, specific and concrete understanding of system components.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

Hence, a system can be decomposed into subsystems, each of which can be decomposed into components. That is, a large-scale system has two dimensions: abstraction levels including the objective, conceptual, function and physical levels, and scales including the system itself, subsystems, and components. Obviously, a system can have more than three scales, if there is a need to decompose a subsystem into sub-subsystems.

Take an example of an armature-winding machine—a discrete-part manufacturing system (Ye, 1996a, 1996b). At the objective level, the goal of the armature-winding machine is specified as winding coils around an armature at a given rate, e.g., 10 armatures per minute. The inputs to the machine are the coils and the armature. The output from the machine is the armature with coils. At the conceptual level, a number of time-phased operational steps are defined to accomplish the goal of winding coils around an armature at a given rate. For example, the first step is to load the armature on the machine, the second step is to close the winding heads around the armature, and so on. At this level, operations of winding an armature are defined without referring to subsystems and components of the machine. So operations are conceptual. At the functional level, several mechanical and electrical functions are designed to carry out the operation of loading an armature, including the mechanical function of a fixture to hold the armature, the mechanical function of an arm to pick up the armature and place it on the fixture, and the electrical functions of a motor, a signal processing unit, a signal transmission path, sensors and actuators to control the movement of the mechanical components. At this level, mechanical and electrical functions of subsystems and components for other conceptual operations are also designed. At the physical level, specific components such as the fixture, the arm, the motor, sensors, actuators, and cables are selected, made or purchased to provide the functional features of those components. The physical configuration and layout are also determined.

Take another example of a computer software system. At the objective level, system requirement analysis is carried out to specify the software system in terms of information inputs to the system, information outputs from the system and other requirements. At the conceptual level, system specification is carried out to define information processing operations of transforming inputs into outputs in terms of information flows and information structures. At the functional level, software design is carried out to construct software modules of performing information processing operations, resulting in the modular structure of the software system and the function of each module. At the physical level, each software module is implemented by software code using a specific programming language. Hence, at the physical level, all physical configuration, appearance and details of the software system are shown.

The four-level abstraction hierarchy of a system can be used to identify the levels and scales of resources in an information system, as shown in Fig. 1. If the information system is a network of host machines, at the system scale the resource is the information system as a whole—the network of host machines. User's applications are processes requesting services from this resource. The subsystem scale consists of the host machines as the subsystems. Tasks of an application, which are allocated to a host machine, are processes requesting services from the host machine as a subsystem. The component scale consists of individual resources, e.g., hardware resources, software resources, and information resources on each host machine. Processes from application tasks on each host machine request services from various software resources, such as application programs (e.g., Microsoft WORD), servers (e.g., a web server), databases and files, and the operating system (e.g., WindowsNT), and various hardware resources such as CPU, RAM, hard drive, computer screen, printer, etc. If an information system contains a network of local networks, the information system can have more than three scales.

There are four levels of abstraction. At the objective level, the network of host machine is the abstract resource that serves a user's application as a whole. Multiple applications may be presented at a given time. At the conceptual layer, a user's application is decomposed into conceptual tasks that are allocated to individual host machines. Each conceptual task is further decomposed into functional processes—software processes—requesting services from software resources and/or information resources at the functional level. A functional process requesting services from one software resource such as an application program may generate another functional process requesting services from another software resource such as the operating system. Each software process is implemented by physical processes requesting services from hardware

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

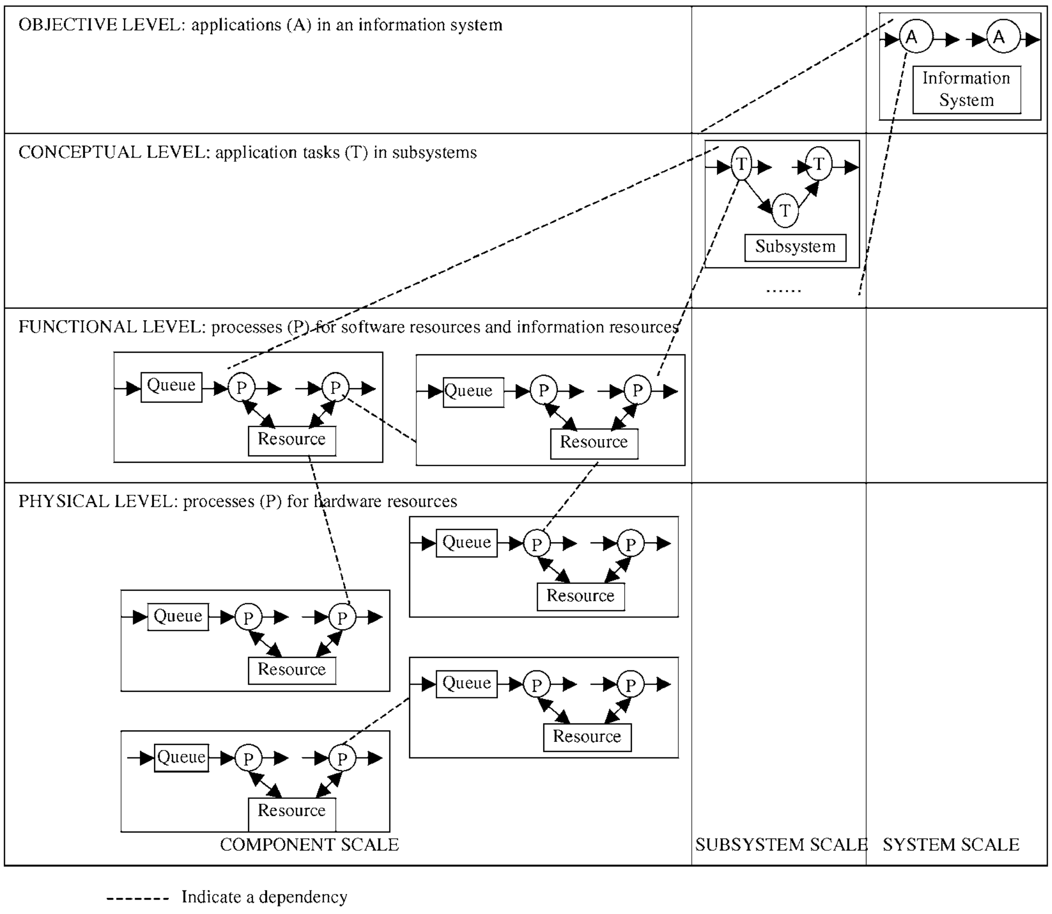
OBJECTIVE LEVEL: applications (A) in an information system
A
A
Information System
CONCEPTUAL LEVEL: application tasks (T) in subsystems
T
T
T
Subsystem
......
FUNCTIONAL LEVEL: processes (P) for software resources and information resources
Queue
P
P
Resource
PHYSICAL LEVEL: processes (P) for hardware resources
COMPONENT SCALE
SUBSYSTEM SCALE
SYSTEM SCALE
------- Indicate a dependency

Fig. 1. *A hierarchy of resources in an information system.*

resources such as CPU, RAM, hard drive, computer screen, printer, and so on.

For example, an organization has an information system that contains a network of host machines and the router connecting the network to Internet. The information system provides various services to users insides and outsides the network, including web services, at the objective level and the system scale. Web services are provided by one host machine in this network—a subsystem of the information system. The router is also a subsystem of the information system. When a user's web application through a Microsoft Outlook client program sends a request for web services from the information system, the request is decomposed into three application tasks: routing the request to the host machine with the web server by the router, executing web services by the host machine, and routing the outcome of web services to the source by the router, at the conceptual level and the subsystem scale. At the functional level and the component scale, the Microsoft Outlook web server program on the host machine breaks the web-service-executing task into a series of software processes, such as compiling the request, authenticating the user, retrieving the information from local files on the host machine, delivering the requested information, and so on. Each software process is decomposed into a series of processes or system calls for kernel functions in the operating system that is also a software resource at the functional level and the component scale. Each system call to the kernel of the operating system as a process for a software resource is implemented through a series of physical processes acting on hardware resources such as CPU, RAM, and so on.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

A resource (R), processes (P) requesting services and supply-demand relationships between the resource and processes form a *dynamic system* as shown in Fig. 1. Queues (Q) may be used to keep processes waiting for the resource. For example, an information system may consist of a router, several host machines, communication links connecting the router and the host machines, and so on. A user's ftp application enters this information system to request some information that resides on one host machine in this system. This application yields a series of processes (P): the process requesting the service of the router to send the request to the host machine, the process requesting the service of the host machine to retrieve and send out the information to the router, and the process requesting the service of the router to send out the information to the source of the request. The router may have two queues (Q): an in-bound queue for data packets going into the domain and an out-bound queue for data packets going out of the domain.

State of Resources

There are a variety of resources in an information system. In the information system, the resource pool changes over time with addition, deletion and modification, and processes requesting services from resources come and go dynamically. Hence, the state of the information system changes over time. The state of the information system depends on the state of resources with respect to the number and variety of all the resources and the dependability attributes of each resource such as availability, integrity and confidentiality. That is, each dynamic system in the information system is a stateful system.

The availability attribute of the state of a resource is related to the responsiveness of the resource to meet service requests. The responsiveness of the resource depends on the available or used capacity of the resource in relative to the maximal capacity of the resource. Take an example of a router. The available or used capacity of the router can be measured in terms of the in-bound and out-bound queue lengths (the numbers of data packets in the queues) and the actual flow rates (the numbers of data packets transmitted per time unit) of in-bound and out-bound traffic in relative to the maximal lengths of the in-bound and out-bound queues and the maximal flow rates of in-bound traffic and out-bound traffic. Hence, the availability attribute of the router state can be measured by queue length and traffic flow rate. That is, probing the queue lengths and the traffic flow rates will produce measures on the availability attribute of the router state.

The integrity attribute of the state of a resource is related to the correctness of the resource to meet service requests. A resource may produce a correct output for a process but taking a very long time and thus not meeting the responsiveness if too many processes are in the dynamic system to share the resource. The correctness of the resource depends on the internal functioning of the resource. For example, if the internal functioning of a router is pre-configured to allow only web and email accesses to an administrative domain, the correct output of a ftp process should be a message indicating the rejection of the service request. If the ftp process receives the requested data file as the output, this indicates that the integrity attribute of the router state has been compromised. Since the correctness of the router depends on the internal functioning of the router that is difficult to probe, two methods can be used to measure the integrity attribute of the router state. One method is to specify the correct functioning of the router (e.g., in form of security policies) and passively monitor traffic flows through the router against the specification. Another method is to design efficient benchmark processes with low overhead that can be launched to test whether the outputs of those processes are correct. For instance, a tiny ftp process can be designed to quickly check if a message of rejection rather than a data file is received as the output. Unlike the constantly passive monitoring of traffic flows with large overhead on the router, the method of benchmark tests is more efficient because low-overhead benchmark processes are selected and tested only when QoS problems are detected.

The confidentiality attribute of the state of a resource is related to the precision of the resource to meet service requests, that is, whether the resource produces the precise amount of the output for a given input. The precision of the resource also depends on the internal functioning of the resource. For example, an intruder may use a Trojan horse program to alter the internal functioning of a router in an administrative domain. As a result, each stream of data packets from the administration domain is sent to not only the correct destination of the data stream but also another destination where the intruder uses to collect data. For each process requesting the transmission of a data stream, the output

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

of the process includes not only the data stream to the correct destination but also the same data stream to another destination. The actual amount of the output is greater than the desired output of the process for the input requesting the data stream to be sent to the correct destination only. Since the confidentiality attribute of the router state depends on the internal functioning of the router that is difficult to probe, benchmark tests can be designed to launch benchmark processes and check their outputs only when QoS problems are detected.

Therefore, the state of resources can be measured using probes and tests that are designed for a variety of resources. A probe is to get a specific piece of state information at a specific location on a resource, e.g., the length of the in-bound queue in a router. A test is to launch a benchmark process with a known input and a known output on the resource and check the actual output of the process against the expected output of the process.

QoS Attributes of Processes

The state of resources in an information system impacts the output performance of processes entering, executing or waiting in the information system. The output performance of processes determines QoS received by processes. QoS metrics have been investigated extensively in many fields (Chatterjee et al., 1997; Giroux and Ganti, 1999; Lawrence, 1997; Sabata et al., 1997). Lawrence (1997) generalizes three QoS attributes: timeliness, precision and accuracy. Timeliness measures how fast an output for a given input is produced. Precision measures how much output is produced for a given input, relating to the quantity of the output. Accuracy measures how good an output is, relating to the content quality of the output. Different metrics for the three QoS attributes are required for processes on different resources with different output characteristics.

Take an example of a process requesting the service of the router to send a stream of data packets to a destination outside an information system. The timeliness attribute of QoS can be measured in term of the traffic flow rate computed by dividing the total number of bytes in this data stream by the total amount of time that the router actually takes to complete this transmission of this data stream. The precision attribute of QoS can be measured in terms of what percentage of data packets are actually sent out, e.g., 100% when the router is functioning correctly to send all data packets of this data stream to the correct destination, or maybe 200% when the router sends the data packets to the correct destination and another destination in a situation of a Trojan horse program running on the router. The accuracy attribute of QoS can be measured through a distance value between the computed checksum of the data stream and the original checksum of the data stream.

In an information system, the state of resources impacts the QoS of processes. For example, the availability attribute of the resource state has impact on how many processes are served at a given time, how many processes wait in the queue, and consequently how long it takes to produce the output of an process—the timeliness attribute of QoS. The integrity attribute of the resource state has impact on how correctly the output is—the accuracy attribute of QoS. The confidentiality attribute of the resource state has impact on how much output is produced—the precision attribute of QoS. Intrusions may compromise the availability of a resource, leaving a process waiting in the queue and no output produced for the process over a long period of time. Intrusions may compromise the integrity of a resource, resulting in an incorrect output of a process. Intrusions may compromise the confidentiality of a resource, leaking information to unauthorized users and producing more than a precise amount of authorized output for the process.

For example, a user's process requests the driving direction from one location to another location from a web site such as Yahoo.com. The input of the process is the query for the driving direction. The output of the process is the driving direction. The user may require receiving the driving direction, which should include a map and a text description, within 5 minutes. The QoS level may be measured in terms of what output information the user receives, how much output information the user receives, and how long it takes to get the output information. If the web site is compromised, the following may happen: (1) the user receives a message "no driving direction is available" rather than the driving direction; (2) the user receives only the text description but no map; (3) the user waits forever to get any response from the web site; or other forms of degraded QoS.

There is a distinction between the three QoS attributes of the real-time output performance of process and design attributes of the output. For example, the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

user-friendliness of the driving direction as the output of a process requesting services from a web site is associated with the interface design of the output. Design features of the output determine what output should be expected for a given input. After the expected output for a given input is designed and implemented, the QoS attributes measure the real-time performance of the actual output with respect to the expected output. Hence, user-friendliness is a design attribute of the output rather than a QoS attribute of the real-time output performance. Given what is expected for an input, the QoS attributes focus on how much the expected output is obtained, how fast the expected output is obtained, and how well the actual output matches the expected output. The QoS attributes measure the actual real-time output performance against the given specification of the output performance rather than the quality of the specification. A poor-designed web site may still achieve a high QoS level if the actual levels of the three QoS attributes meet the expected level of these QoS attributes.

Propagation of QoS Attributes Among Resources

To guarantee QoS of a user's application, the user must provide QoS requirements of the application. Without QoS requirements, any level of QoS can be considered satisfactory. Using QoS metrics of processes to be established for a variety of resources, a user can specify QoS requirements of an application in terms of QoS requirements of individual processes in this application. The specification of QoS requirements for an individual process provides the target QoS values for QoS metrics of this process.

In addition to the QoS requirements of these processes that specify the desired output performance of these processes, the user should also specify the characteristics of the inputs to these processes. Take an example of a process requesting the service of a router to transmit a data file. The inputs to the process include the data file itself, the destination address, and so on. The output from the process is the data file at the destination address. The user should specify the size of the data file along with the QoS requirements for the output performance of the process. If the actual size of the data file exceeds the specified size of the data file, the information system cannot guarantee to meet the QoS requirements of the process. Hence, a user's application specification should include both the input characteristics and QoS requirements of processes in the application.

Resources requested by processes in the user's application specification usually are software and information resources, such as application programs and data files. These user-level resources depend on machine-level resources such as the operating system and hardware resources to deliver services. To obtain QoS requirements of machine-level processes, QoS requirements of user-level processes need to be propagated to machine-level processes. In other words, target QoS values of the user's processes must be transformed into target QoS values of processes for machine-level resources.

The hierarchy of resources as shown in Fig. 1 establishes the topological structure of propagating QoS requirements from user-level processes to machine-level processes, based on the dependency of resources. For example, if resource A depends on resources B and C, the target QoS value of the process for resource A must be decomposed into target QoS values of processes executing on resources B and C. We also need to establish the algebraic structure of QoS attributes for transforming QoS requirements along the topological structure.

Suppose that a process depends on a series of component processes. Anderson and Mathews (1998) propose the following algebraic structures of QoS attributes:

- Accuracy is cumulative in that the accuracy of an process is the product from multiplying the accuracy of each component process;
- Timeliness is additive in the case of latency in that latency is the sum of each process's latency and the delay between the completion of one process and the start of the next process; and
- Precision of processes is determined by the minimum precision among these processes.

QoS Contracting and Admission Control

After obtaining QoS requirements of processes involved in an application at all levels and scales of the information system, the information system must decide whether QoS requirements of the process can be satisfied. If QoS requirements of all processes in an

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

application can be satisfied, the application is admitted into the information system, and the user's specification of output QoS requirements and input characteristics for an application becomes a QoS contract; otherwise, the application is rejected. Hence, to determine if an application can be admitted into an information system, we must determine if each process in the application can be admitted into a dynamic system containing the resource to execute the process.

An existing method of admission control for a communication process (Giroux and Ganti, 1999) is to take a small segment of the input (e.g., a small number of data packets), execute a small trial of the communication process with this small segment of the input, and use the performance feedback on this trial process to determine whether or not to admit the communication process with the whole set of the input. However, many processes request services from non-communication resources. The input to those processes, e.g., processes for application programs, may not be decomposable to make a small trial and get the performance feedback on the small trial. We describe a uniform set of admission control techniques regardless of the type of a resource below.

For the timeliness attribute of QoS requirements of a process requesting to enter a dynamic system for a resource, we propose to apply a scheduling technique to scheduling this process and other processes already in the dynamic system. If the best schedule of all the processes on the resource conforms to the timeliness attribute of QoS requirements of these processes, we say that the timeliness attribute of QoS requirements of the process is acceptable. Several factors may be taken in account during scheduling, including resource utilization, lead time (the sum of waiting time and execution time), process priority, and so on.

This admission control technique for evaluating the timeliness attribute of QoS requirements combines scheduling into admission control. Once a process is admitted to enter the dynamic system, the schedule of this process is available to indicate when the process will be served. A set of scheduling techniques has been established in many fields, including the scheduling of network traffic on ATM networks and the scheduling of production jobs for customers' orders on machines in manufacturing firms (Giroux and Ganti, 1999; Smith, 1989).

For example, we can use a scheduling technique based on the "first-come—first-serve" principle. Suppose that there are three processes—A, B and C—in the dynamic system requesting services from a resource. If process A enters first, process A will be served before processes B and C are served. If process B enters second while process A is still executing on the resource, process B will wait until process A is completed.

The precision and accuracy attributes of QoS requirements of the process can be satisfied if the internal functioning of the resource is correct. The correct functioning of the resource can be judged based on whether the resource meets the precision and accuracy attributes of QoS requirements of processes recently executing on the resource. This requires the monitoring of the actual precision and accuracy values of processes going through the resource against their target QoS values. If the precision and accuracy attributes of QoS requirements of recent processes are satisfied, we say that the precision and accuracy attributes of the QoS requirements of the process are acceptable.

If the timeliness, precision and accuracy attributes of QoS requirements of the process are all acceptable, the process is acceptable for admission into the dynamic system, and the time when the process receives service from the resource is determined by the schedule of processes.

QoS Conformance Monitoring

The performance monitoring of recent processes going through the resource is a part of QoS conformance monitoring. For each process in a dynamic system, QoS conformance monitoring checks if the actual QoS values of the process conforms to the target QoS values in the contract, and if the actual input characteristics of the process conforms to the specification in the contract.

Although the target QoS values of the process admitted into the dynamic system expect to be satisfied, it is possible that the actual QoS values are largely different from the target QoS values due to (1) the nonconforming input characteristics, or (2) the degraded state of the resource by intrusions or other causes. Hence, the actual QoS values of the process must be monitored to detect significant deviations from the target QoS values. Since we expect the actual QoS values of the process fluctuates slightly over the course of the process, only significant deviations warrant further investigation.

We can use Statistical Process Control (SPC) techniques (Montgomery, 1991) to monitor the actual QoS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

values of the process and signals when a significant deviation from the target QoS values is detected. We have developed several SPC techniques for intrusion detection (Ye, Chen, and Emran, 2000a, 2000b) to detect significant deviations of actual values from target values of activities in information systems.

Take an example of a process requesting the service of the router to send a stream of data packets to a destination outside an information system. The timeliness attribute of QoS is measured in term of the traffic flow rate computed by dividing the number of bytes transmitted by the amount of time that the router actually takes to transmit these bytes. The data stream contains 10 mega bytes of data. From the QoS requirements of this process, we obtain the target value of the flow rate (e.g., 1000 bytes/second). From the history of the router operation in the normal condition, we obtain the normal variance of the flow rate (e.g., 100 bytes/second). According to SPC techniques, we can use the 3-sigma rule to set the control limits to [target value $- 3 * $ standard deviation, target value $+ 3 *$ standard deviation], that is, $[1000 - 3 * 10, 1000 + 3 * 10]$ or [970, 1030], where 970 is the lower control limit and 1030 is the upper control limit. In the course of transmitting the data stream, we get measurements of the flow rate. If a measurement of the flow rate exceeds the control limit, a signal is generated to indicate a signal deviation of the actual value from the target value of 1000 bytes/second. This SPC technique for monitoring the flow rate of a process for a router is demonstrated in Fig. 2.

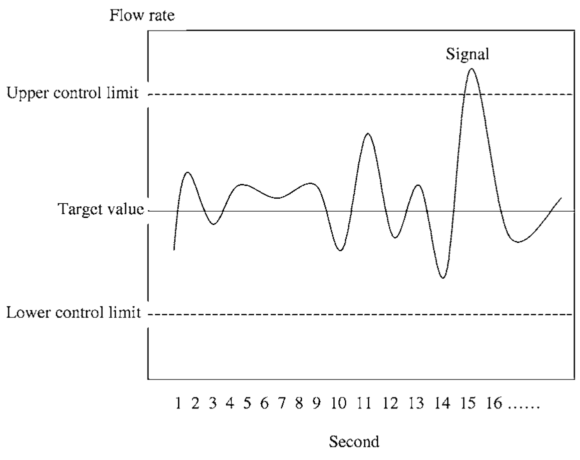

Fig. 2. *The illustration of a possible application of a SPC technique for monitoring the flow rate of a process executing on a router.*

We can also use SPC techniques to monitor the actual input characteristics of a process and signals when a significant deviation from the specified input characteristics is detected. In case of a signal, the process must be dropped for preventing it from consuming an extra amount of the resource capacity and thus depleting the resource capacity.

When a significant deviation of the actual QoS values of the process from the target QoS values is detected by the SPC techniques and the actual input characteristics of the process conform to the specification in the contract, state probing and testing is required to uncover the state of the resource. Appropriate probes or tests can be selected according to which QoS attribute demonstrates the significant deviation, and launch the selected probes or tests to verify whether certain attribute of the resource state is degraded. If the resource state is indeed degraded, processes in the dynamic system need to be rescheduled according to their priority. Those low-priority processes, whose QoS requirements cannot be satisfied according to the new schedule, must be temporarily halted. The system administrators are informed of the degraded resource state for their correction actions to bring the resource back to the normal state. If the system administrators cannot recover the resource in a short period of time, they can let the information system either redirect those temporarily halted processes to back-up resources or drop those temporarily halted processes. Dropped processes have to seek other resources in or outside the information system for services.

A Control-Theoretic Structure of QoS-Centric Stateful Resource Management

Fig. 3 illustrates the QoS-guarantee functions and their relationships for QoS-centric stateful resource management. These functions can be organized in a control-theoretic structure of closed-loop feedback control (Bollinger and Duffie, 1989; Ogunnaike and Ray, 1994) as shown in Fig. 4. This control-theoretic structure consists of three major components: dynamic system, controller, and monitor. The dynamic system has a resource, processes and queues.

The controller manages the dynamic system by generating control inputs that

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

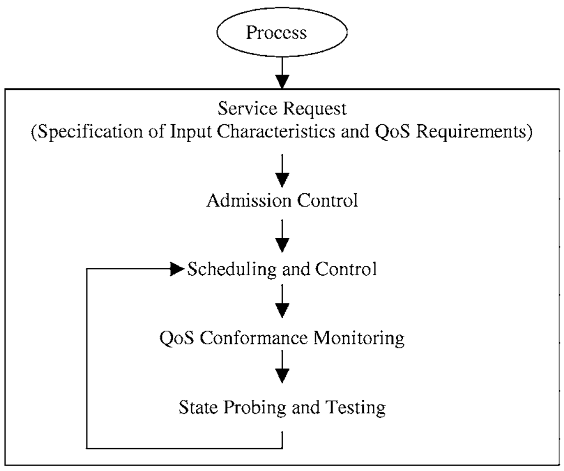
Process
Service Request
(Specification of Input Characteristics and QoS Requirements)
Admission Control
Scheduling and Control
QoS Conformance Monitoring
State Probing and Testing

Fig. 3. QoS-guarantee functions.

- Evaluate exogenous inputs—service requests of processes—from other dynamic systems in the same information system or outside the information system,
- Admit processes into the dynamic system,
- Schedule processes on the resource,
- Select and launch probes and tests to verify the state of the resource, and
- Issue control actions.

The controller also interacts with the system administrators to inform the degraded state of the resource and receives the instruction from the system administrators as to whether to redirect or drop temporarily halted processes.

The monitor evaluates the actual QoS values against the target QoS values, and signals significant deviations of the actual QoS values from the target QoS values. Uncertainties (see Fig. 4) may occur to the dynamic system due to intrusions or other causes. Those uncertainties have impact on the state of the resource and the output performance of processes. The monitor also evaluates the actual input characteristic against the specified input characteristics, and signals significant deviations.

Since an information system has a number of resources, there are a number of control loops. Each control loop manages a resource and processes in the dynamic system for the resource.

These QoS-guarantee functions in a control-theoretic structure place three layers of barrier against intrusions: intrusion prevention, fault masking, and error tolerance. The admission control function acts as the intrusion prevention mechanism to reject excessive service requests that cannot be satisfied by the capacity of the resource. This can prevent many denial-of-service attacks. The monitoring of input characteristics can prevent buffer overflow attacks. The scheduling and control function can mask the degraded state of the resource by rescheduling processes on the resource and temporarily halting some low-priority processes to assure high-priority processes not affected by the degraded state of the resource. The system administrators can provide the error tolerance function by correcting and recovering the resource to a normal state. The system administrators may designate some of the error tolerance function to the information system for contingency handling. The QoS conformance monitoring function can provide the detection of abnormal QoS values to trigger fault masking. The state probing and testing function can uncover the nature of the degraded resource state to assist the system administrators in error tolerance for system recovery.

We are currently developing the control-theoretic, QoS-centric resource management models for two common resources in a computer network system respectively: a router and a web server, to demonstrate the effectiveness of QoS-centric resource management. After completing the models, we plan to set up experiments for testing the effectiveness of these models for QoS assurance performance. For example, for a router we will set up two experimental conditions. In

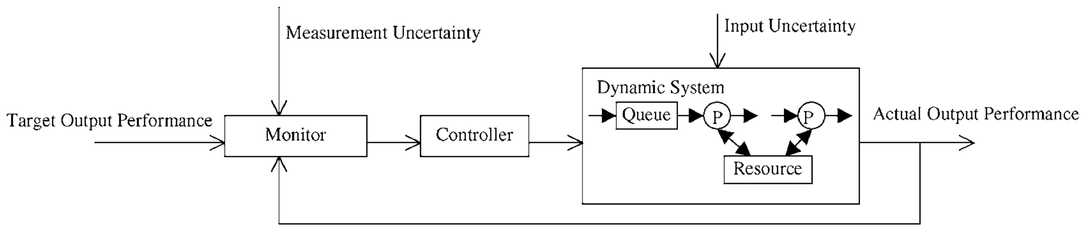

Fig. 4. A closed-loop feedback control structure of QoS-centric stateful resource management.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

the first experimental condition, the control-theoretic, QoS-centric resource management model will be employed to make decisions on the admission control, scheduling, QoS conformance monitoring, and state probing and testing. These QoS functions will need to be integrated with the routing functionality. Hence, we will use the Linux operating system to implement the QoS-centric resource management model for the router, because of the availability of the Linux source code. A stream of network packets will be presented to the router during the experimental testing. Network packets will include phases of normal network traffic and phases of intrusive network traffic for attacks such as denial-of-service attacks through flooding. In the second experimental condition, a regular router without the deployment of the control-theoretic, QoS-centric resource management model will be tested when the router is presented with the same stream of network traffic as in the first experimental condition. We will collect QoS measures of normal network traffic and intrusive network traffic, such as the percentage of normal network packets whose QoS is satisfied by the router. The effectiveness of the control-theoretic, QoS-centric resource management model will be demonstrated by a larger percentage of network packets whose QoS is satisfied under the first experimental condition than that in the second experimental condition. Details of these efforts on developing and testing the QoS-centric resource management approach for information assurance will be presented in future reports. The purpose of this paper is to introduce this innovative approach, and draw attention to this approach from the research community at an early stage so that more research efforts can be put into this direction.

Summary

This paper reviews the stateless resource management problem in the current design and operation of information systems, causing vulnerabilities of information systems to many malicious exploits and attacks on information systems. An engineering approach to QoS-centric stateful resource management is introduced to provide a conceptual overview. This approach is illustrated using a router example. We also briefly outline our ongoing research activities to develop and test this approach using some common resources in information systems, including a router and a web server, to illustrate how this approach can be implemented and tested. Currently, the discussion of this approach in this paper does not address application resources for decision support systems. As more researchers look into this approach in the future, we expect that research results for various resources in information systems will emerge.

This paper does not provide full details of the control-theoretic, QoS-centric resource management model for the router, because these details are under research and development. This paper intends to introduce this approach to the research community for promoting further research and development in this direction that will lead to a revolutionary way of designing and operating information systems for information security assurance. Therefore, this paper is to introduce this approach and raise relevant research issues and technical problems. As research in this direction becomes more mature after years of effort from the research community, practical applications of research results can be foreseen. Research results in this direction will mostly be implemented in software supporting information systems. Currently, the QoS-centric stateful resource management approach is still at the beginning of the research stage. We will describe our research results in this direction in future reports.

Acknowledgment

This work is sponsored by the Air Force Research Laboratory—Rome (AFRL-Rome) under agreement number F30602-01-1-0510. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFRL-Rome, or the U.S. Government.

References

Anderson DJ, Mathews R. Algebraic and topological structure of QoS within large scale distributed information systems. Final Report under AFRL contract number: F30602-98-C-0035, 1998.

Aurrecoechea CA, Hauw L. A review of QoS architectures. In: *Proceedings of the 4th IFIP International Workshop on Quality of Service*, 1996.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com

Barnes BH. Computer security research: A British perspective. *IEEE Software* 1998;15(5):30–33.

Bollinger JG, Duffie NA. *Computer Control of Machines and Processes*. Reading, Massachusetts: Addison-Wesley, 1989.

Boulanger A. Catapults and grappling hooks: The tools and techniques of information warfare. *IBM Systems Journal* 1998;37(1):106–114.

Chatterjee SJ, Sydir BS, Davis M, Lawrence TF. Modeling applications for adaptive QoS-based resource management. In: *Proceedings of the IEEE High Assurance Systems Engineering Workshop*, 1997.

Foster I, Kesselman C. *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco, California: Morgan Kaufmann Publishers, 1999.

Garfinkel S, Spafford G. *Practical UNIX & Internet Security*. Cambridge: O'Reilly & Associates, 1996.

Giroux N, Ganti S. *Quality of Service in ATM Networks*. Upper Saddle River NJ: Prentice Hall, 1999.

Godwin M. Net to worry. *Communications of the ACM* 1999;42(12): 15–17.

Jajodia S, Ammann P, McCollum CD. Surviving information warfare attacks. *Computer* 1999;32(4):57–63.

Kaufman C, Perlman R, Speciner M. *Network Security: Private Communication in a Public World*. Englewood Cliffs, New Jersey: Prentice Hall, 1995.

Lawrence TF. The quality of service model and high assurance. In: *Proceedings of the IEEE High Assurance Systems Engineering Workshop*, 1997.

Mann P. Pentagon confronts mounting cyber risks. *Aviation Week and Space Technology* 1999;150(12):82, 83.

Montgomery DC. *Introduction to Statistical Quality Control*. New York, NY: John Wiley & Sons, 1991.

Neumann PG. Risks of insiders. *Communications of the ACM* 1999;42(12):160.

Ogunnaike BA, Ray WH. *Process Dynamics, Modeling and Control*. New York: Oxford University Press, 1994.

Pandey R, Hashii B, Lal M. Secure execution of mobile programs. In: *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2000:362–376.

Rasmussen J. *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. New York, NY: North Holland, 1986.

Sabata B, Chatterjee S, Davis M, Sydir J, Lawrence TF. Taxonomy for QoS specification. In: *Proceedings of the IEEE Computer Society 3rd International Workshop on Object-Oriented Real-Time Dependable Systems*, 1997.

Simons B. Building big brother. *Communications of the ACM* 2000;43(1):31, 32.

Smith SB. *Computer Based Production and Inventory Control*. Englewood Cliffs, NJ: Prentice Hall, 1989.

Stallings W. *Network and Inter-network Security Principles and Practice*. Englewood Cliffs, NJ: Prentice Hall, 1995.

Ye N. The presentation of knowledge and state information for system fault diagnosis. *IEEE Transactions on Reliability* 1996a;45(4):638–645.

Ye N. A hierarchy of system-oriented knowledge for diagnosis of manufacturing system faults. *Information and System Engineering* 1996b;2(2):79–103.

Ye N, Chen Q, Emran SM. Hotelling's T^2 multivariate profiling for anomaly detection. In: *Proceedings of the IEEE SMC Information Assurance and Security Workshop*, West Point, New York, June 2000a.

Ye N, Chen Q, Emran SM. Chi-squared statistical profiling for anomaly detection. In: *Proceedings of the IEEE SMC Information Assurance and Security Workshop*, West Point, New York, 2000b.

Nong Ye received a B.S. degree in Computer Science from Peking University, Beijing, an M.S. degree in Computer Science from the Chinese Academy of Sciences, Beijing, and a Ph.D. degree in Industrial Engineering from Purdue University, West Lafayette, Indiana. Dr. Ye is currently a tenured associate professor with the Department of Industrial Engineering, Arizona State University, Tempe, Arizona. Her research interests are in assuring process quality and preventing faults and errors in information systems, human-machine systems, and manufacturing systems. Dr. Ye is a senior member of the Institute of Industrial Engineers, and a senior member of IEEE.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission. www.manaraa.com