Performance Modeling of a Service Provisioning Design

Mohamed El-Darieby and Jerome Rolia

Systems and Computer Engineering Department
Carleton University, Ottawa, ON, K1S 5B6, Canada
{mdarieby, jar}@sce.carleton.ca

Abstract. The convergence of the telecommunications and computer industries is nearing a reality leading to service-based business models. The ability to provision services efficiently and quickly will be a competitive differentiator between service providers. This paper describes recent efforts to evaluate performance and predict the scalability of software architectures for provisioning services to end-users. We focus on network services that require the establishment of a Virtual Private Network. An analytical performance model for the service creation process is described. It expresses the high-level performance characteristics of the different subsystems involved in service provisioning. We study the effects of increasing numbers of network elements, end users, and autonomous systems on service provisioning infrastructure.

1. Introduction

The convergence of the telecommunications and computer industries [13] is nearing a reality. Service providers are able to use, rent or trade each other's resources and services. In such an environment, the competitive differentiator between service providers will be determined by the ability to provision well-performing services efficiently and quickly [11]. Service level management requires technical (e.g. performance) models [6] of the infrastructure of computer networks. This paper presents a modeling approach for such systems.

Service provisioning is of growing importance. For example, in 1998, the CMI corporation projected that the revenues from the provisioning of Virtual Private Network (VPN) services will be US$3 billion in 1999, US$10 billion in 2004, and US$47 billion in 2010. Corresponding provisioning systems must be designed to support increasing user loads and the growth in networks in general.

Service provisioning has two main functions:

1. **Service creation:** allocating network resources, establishing network connections, adding QoS features, and linking the service logic to control data flows.
2. **Service management:** operational support for accounting, billing, and managing Service Level Agreements (SLA).

This paper focuses on service creation. A model is developed and used to study how a service provisioning system performs as its managed network evolves. Network evolution is characterized by the number of autonomous systems, the

C. Linnhoff-Popien and H.-G. Hegering (Eds.): USM 2000, LNCS 1890, pp. 81-92, 2000.

number of network elements per each autonomous system and the rate of requests for services.

We consider the following questions: a) How will the increase in demand for the standardized broadband network services affect the performance of the service provisioning system? b) What is the impact of increasing the number of autonomous systems and the network elements they manage on performance?

We are not aware of similar work to study the scalability of service provisioning systems. However references are made to related scalability studies in the distributed system and software performance engineering literature.

Section 2 describes network services in more detail and the ITU effort to standardize their performance measures. A performance modeling approach is introduced that is used to model and predict the service provisioning system's performance measures. Its relevant features are described. Section 3 describes the service-provisioning environment considered in this paper and an example network topology. A scenario for service provisioning is introduced. Section 4 describes the corresponding predictive model. Section 5 gives results from the study along with concluding remarks.

2. Network Services and Their Performance Metrics and Models

Standardization bodies, for example the IEEE and ITU, have set standards for measuring and evaluating the performance of network services. Sections 2.1 and 2.2 explain the relationship between these efforts and our work. This is followed by a brief description of the method and tool we used to model the performance of creating a service.

2.1 Network Services

The word *service* is often used as a synonym for the word function and consequently is used differently in different contexts [9]. The following taxonomy for services is provided by the IEEE Project 1520 for standardizing programming interfaces for network resources.

1. **Application services** that are *provisioned* in response to a request from application services by Service Providers (SP). These services include services that are defined in ITU recommendation I.211 [1] and are categorized into two main categories: a) *Interactive services* such as retrieval, messaging and conversational services; and b) *Distribution services* that include distribution services with or without user individual presentation control. Other types of these services include value-added VPN services such Intranet, Extranet, and remote dial-in VPNs.
2. **Value-added communication** *services* that are used by a service provider to enable the provisioning of broadband services. Examples of such services include policy, Service Level Agreement (SLA) management, security, location and mobility management, and Virtual Private Networks (VPN) services. They also include operational support services such as billing and reporting.

3. **Basic communication services** that are provided by the network provider (i.e. the carrier). These services span:
   - Access networks: such as dial-in, cable, wireless, and TDM access services.
   - Backbone network: such as optical networks, Frame Relay, ATM, and VPN services.

Our focus is on provisioning application services to end-users. Of these services we will study services that require the establishment of a Virtual Private Network (VPN) between end users. We call such services VPN-based Application Services. Such services include: media on demand services, video/audio conferencing, tele-education, and Intranet and Extranet services.

2.2 Performance Metrics

The levels of Quality of Service (QoS) are defined in the ITU recommendations E.800 [2]. Basically, they are categorized into service support, operability, serviceability, and security performance metrics. Serviceability is our focus in this paper. We are concerned with the mean response time of the service creation process as a performance measure. The response time of the process belongs to the service accessibility performance metrics that is a subset of the service serviceability metrics. The response time includes the response time of the network and of the software processes that participate in the service.

2.3 Performance Models

There are two widely applied approaches to predict a system's behavior: a) use performance measurements from a controlled testbed, or b) use predictive models. Measurements techniques usually consume more time and costs, and presume that the system or a substantial prototype exists. Estimates from performance models can often reveal performance blunders [15] and provide feedback for making system design decisions even before prototypes exist. We follow the latter approach.

To study the system we develop a Layered Queuing Model (LQM) [4], [8], [7], [10]. LQMs describe the request scenarios submitted to a system and the resources they consume. Requests pass through layers of software processes, consuming network, processor, and disk resources. The layered model is decomposed into a series of queuing network models that are solved iteratively, using Mean Value Analysis (MVA), to provide performance measures for the system as a whole [10].

An LQM is characterized by the:

- The mix and rate of request scenarios by end users
- The resource demands of requests (CPU, disk, network) as they pass through processes
- The nature of interactions between processes: for example whether they are synchronous or asynchronous
- The number of instances of each process and their maximum permitted level of concurrency (they are queuing centers)

- The allocation of software processes to server nodes and the resulting loads on the networks connecting them

The MoL tool implements the Method of Layers algorithm [7], [10]. It has a system description language that defines server nodes and their CPU and I/O rates, the underlying network infrastructure, software processes, their objects, methods, and concurrency characteristics, sequences for the flow of control of scenarios, and workload intensity and mixes of the sequences. It also supports the design of full-factorial experiments through the definition of factors and levels for different parameters in the model. The tool provides fast analytic performance estimates including: estimates for the utilization, mean response time, and average queue length of software processes and for the mean response time of requests. These predictions include contention for both device and logical software process resources.

3. Service Provisioning

Typical service provisioning environments consists of a carrier's core network, an access network and service provider systems. Such an environment is shown in Figure 1.

Users use an access network to dial in and get connected to the service provider. Usually, users dial in to a Point of Presence (PoP) of the service provider. Access networks deploy technologies such as Subscriber Digital Lines (SDL). The service provider systems consist of policy and service management servers. These servers are responsible for enforcing the policies of the service provider, creating service instances in response to user requests, maintaining service levels as agreed upon with users in accordance to the Service Level Agreements (SLA), and billing the users.

The carrier's network consists of many Autonomous Systems (AS). Each of them embraces a number of network elements (i.e. routers or switches). An AS usually uses a specific technology to transport packets for example IP, ATM, or Frame Relay.

The number of AS and the number of network elements each embraces characterize the number of network elements that participate in service provisioning.

3.1 Service Provisioning Scenario

This paper describes an example scenario for creating a broadband service for a hypothetical service provisioning system. The scenario considers the fundamental features of such a scenario and system and is used to guide the development of the corresponding performance model. The scenario is a best-case scenario that assumes no faults or exceptions. The scenario, shown in Figure 2, is as follows:

1. A user logs in to the Access Server, using his login name and password.
2. The Access server consults the Remote Access Dial In User Service (RADIUS) server for Authorization & Authentication.
3. The user ID is returned to the Access Server.

4. The Access Server issues a login event that is load balanced across a pool of servers capable of servicing the event. The event is then directed to the appropriate Policy Server (manager). In the example system presented in this paper, a layer-4 router acts as a load balancer dividing requests equally across pools of identical policy and service processes.
5. The Policy Manager consults the policy directory for policies assigned to the user.
6. The SLA corresponding to the user-application combination is returned to the Policy Manager that invokes the Service Manager.
7. The Service Manager gets the requested service template from a service model database (DB), and determines the network resource requirements of the service.
8. The Service Manager allocates the required network resources by invoking Bandwidth Brokers (BB).
9. The BB discovers resource availability via Topology and resource DB's.

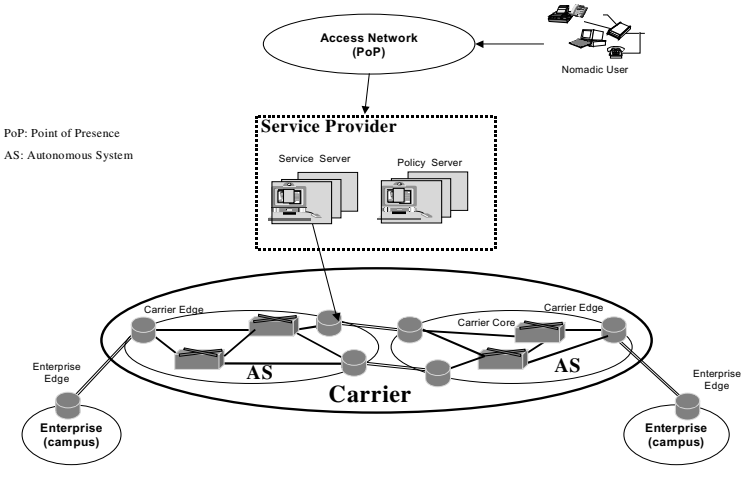

Fig. 1. A typical service provisioning environment.

10. If the service requires resources that span many Autonomous Systems (AS), Intra-BB negotiation takes place.
11. The service Manager is notified as the resource allocation process is completed to admit the service request and start administrative processing.
12. The service Manager downloads any required software components (a.k.a. filters) from a repository.
13. The controlling software components are uploaded to the different Edge Routers; enforcing the policies of the service provider.
14. Data packets flow and billing starts.

4. System Modeling for Performance

The MoL tool was used to document that scenario and obtain performance estimates for the scenario/service provisioning system documented in Figure 2. The tool has two distinctive characteristics. First, it is *representationaly efficient*. It permits the organization of system components (hardware and software) into hierarchical packages. These packages can be replicated. For example a policy server and its underlying hardware can be a package. As the workload for a system increases we can increase the number of replicates of this server's package. The visits to the server are divided equally across the replicates. This reflects the behavior of a load balancing

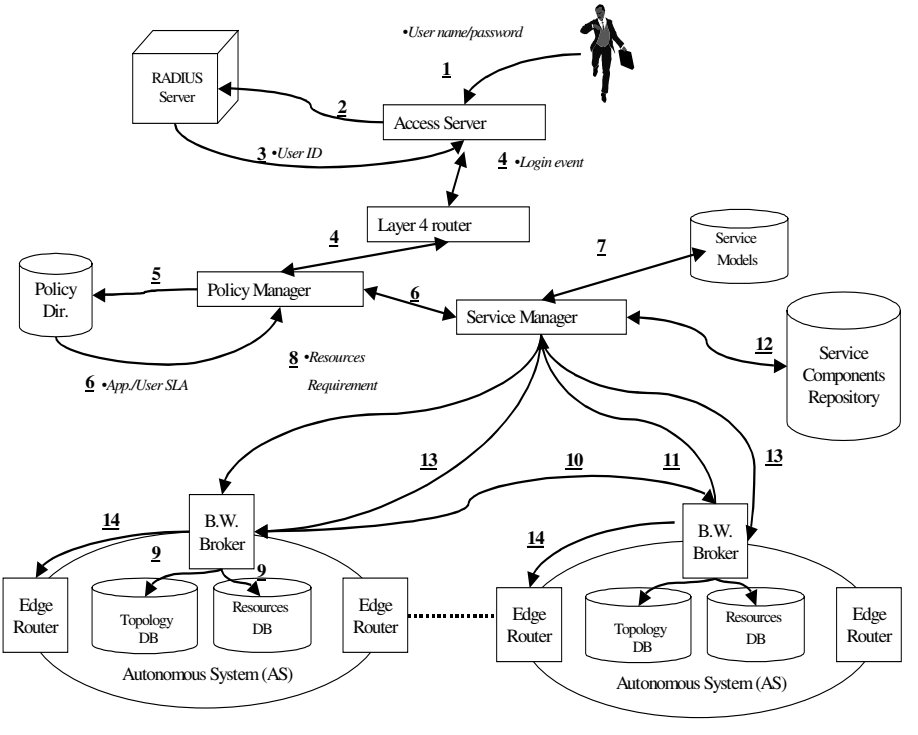

Fig. 2. Service provisioning scenario.

router in the model. The tool is also *computationally efficient*. Increasing the number of replicates does not increase the size of the model to be solved [7].

Figure 3 shows how the service provisioning system is modeled as a set of packages for the LQM. The figure shows also the number of requests for services each process issues for services offered by other processes. These are the processes involved in the scenario described in Section 3.

The model consists a parent package, *service_package* that groups all other packages. It contains the *service_server_package* for modeling the service provider environment, the *carrier_network_package* for modeling the network of autonomous systems, *and user_node* for modeling the behavior of the users and the rate they request services.

The *service_server_package* contains the *policy_servers* package that manages service policies and the *service_mgmt_node* package that creates a service instance and *vpn_mgmt_node* that instantiates a VPN. The policy and service servers communicate through a network that is modeled by the *net1* network component. The Replication Factor (RF) represents the number of each of the nodes.

The *carrier_network_package* consists of many *autonomous_system_package* communicating with each other via the *WAN2* network component. Within an autonomous system, there is the *b_w_broker_node, resource_db* and *the net_element_nodes* packages that represent the different components of physical autonomous systems. The interaction among the *user, service_server_package,* and *carrier_network_package* is carried via Wide Area Network represented by the *WAN1* component of the model.

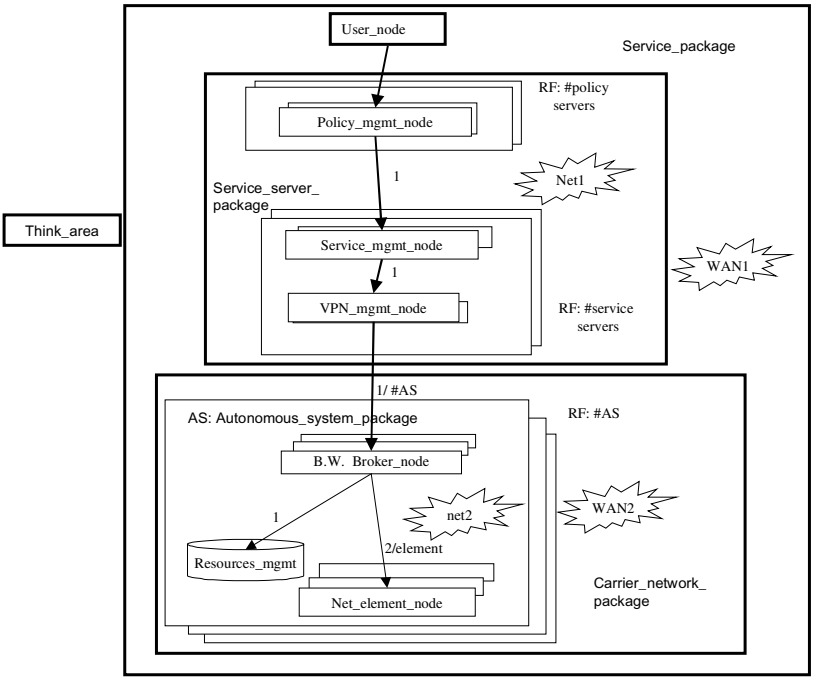

Fig. 3. A layered-queuing performance model.

Each of nodes contains processing entities that are characterized in terms of the CPU processing speed and the speed of Input/Output operations. Also, each one contains a running process that carries out the function of the package. The process has an average service demand for the scenario. The CPU and IO demands are estimates.

They express the relative costs of different processes and requests and permit us to present the proposed modeling approach.

Visits between processes are synchronous, except for calls from the *vpn_mgmt_node* to different bandwidth *brokers* and calls from a bandwidth *broker* to the different *elements_mgmt* processes within the same autonomous system. These are represented in the model using asynchronous interactions followed by a join operation. For example, the *broker* issues asynchronous resource allocation calls to all network elements that will be traversed in VPN connections within that autonomous system. Consequently, as the number of network elements increases, the number of methods the broker waits for increases.

The performance modeling experiments estimate performance and predict the scalability [14] of the system as the network evolves. As mentioned above the two parameters characterizing that evolution are: the number of network elements and the number and type of user requests for services. The average total number of network elements is the product of the average number of autonomous systems traversed by a VPN and the number of network elements per autonomous system. For the sake of simplicity we assume that all autonomous systems have the same number of network elements and statistically identical behavior.

Each experiment provides an estimate for the mean time taken by the system as a whole to create a service instance. This is known as the service creation response time and it is the sum of the (blocking) response times of each of the different subsystems; namely, the policy servers, the service management servers, the VPN management servers, and the processing that takes place within an autonomous system. The results also help to identify system bottlenecks. A bottleneck, software or hardware, is the resource that limits throughput and contributes most to increases in service creation response time. The effect of the delays caused by network latencies for communication among the subsystems is also taken into consideration.

The following factors and their levels are considered:

- The number of autonomous systems (AS): 3, 6, and 9
- The number of elements per AS: 20, 30, and 40
- The average number of end user requests per unit time: 10, 50, and 100

The MoL tool conducts a full-factorial experiment to calculate the effect of the experiment factors on different parts of the model. These changing factors reflect the evolution of the network. In our experiment, we assumed an autonomous system to be a domain for Open Shortest Path First (OSPF) protocol. The OSPF specification does not recommend an OSPF domain to embrace more than 50 network elements.

The resource demands of the various subsystems depend on the above factor levels. We assumed and reflected the following relationships in the model:

- The response time of the *policy_mgmt_node* depends on number of users, amount of data per user representing the profile of the user, service level agreements with a user, and number of service provider's policies. Only the arrival rate of requests affects response times at this node in our experimental design.
- The response time of the *service_mgmt_node* depends on the response time of the service model database that is a function of the product of the number of services and the number of templates for each service. Another component that contributes to this time is the service instance database that maintains a list of the instances in

operation. Only the arrival rate of requests affects response times at this node in our experimental design.

- The response time of the *vpn_mgmt_node* is the solution time needed to discover tentative paths across all autonomous systems to establish a VPN. This depends on Dijkstra's shortest path algorithm. Its solution time is of order $O(N^2)$ where N is the number of AS in the system. A multiplier of N^2 is used to inflate service demands on this node as the number of AS increases. The solution time also depends on the topology of the network and the IP addresses of source/destination nodes that will be connected via the VPN to use the service. However we do not yet reflect these aspects in this high level model and consequently we've assumed that the average number of autonomous systems spanned for each requests is equal 3.
- The response time of the *broker_node* is the summation of the time taken to find the shortest path within an AS, the time taken to allocate and reserve resources of different elements, and the time taken to enforce the policies of the server provider along that path. The time to find a shortest path is of $O(N^2)$ where N is the number of network elements per autonomous system. A multiplier of N^2 is used to inflate service demands on this node as the network topology changes. The time taken to allocate resources and enforce policies depend on the number of network elements within an autonomous system and the performance of the network connecting the broker and the elements within an autonomous system

5. Results and Conclusions

The mean response time estimates for the four main sub-systems are shown in Figures 4, 5 and 6.

Figures 4, and 5 show the response time for 20, and 100 requests/ unit time for services issued by end user, respectively. The mean response time for the service creation process as a whole is dominated by the response time of the VPN manager process. As we increase the request rate to 200 requests/ unit time, the number of elements to 40, and the number of AS from 3 to 9, the VPN process gets fully utilized (utilization of 0.98, 0.96, and 0.97 as number of AS changes from 3, to 6, and 9). This is shown in Figure 6 and illustrates the high response times for the service creation process. Consequently, the VPN manager process is the main performance bottleneck in the system.

The other processes do not contribute much to the service creation response time. Their response times are almost constant with respect to our factors levels. In the model, they are only sensitive to the end user requests rate.

The response time of the VPN process depends a great deal on the number of network elements in the network. This is not a big surprise since its resource demands is a function of the square of the number of network resources considered (due to Dijkstra's algorithm). As the resource demands grow, increasing arrival rates of requests cause dramatic increases in queuing for physical and logical resources leading to very large response times (e.g. Figure 6)**.** The performance models help to illustrate how catastrophic such combinations can be for service performance.

Moreover, performance models allow us to study how we can solve the problems arising from software bottlenecks. An obvious solution is to increase the number of the software processes/nodes causing the bottleneck. We have increased the number of VPN and bandwidth brokers nodes from 1 to 10 and studied the impact on the response time of service creation. We have assumed these 10 processes operate independently and consequently we have not included overhead for synchronization; such overheads may be included in the future. Table 1 shows the results. There is little impact under low loads. For the case with 20 requests per unit time, 9 AS and 40 elements/AS, the mean response time decreases slightly as we increase the number of bandwidth broker and VPN management nodes. The response times decrease by half.

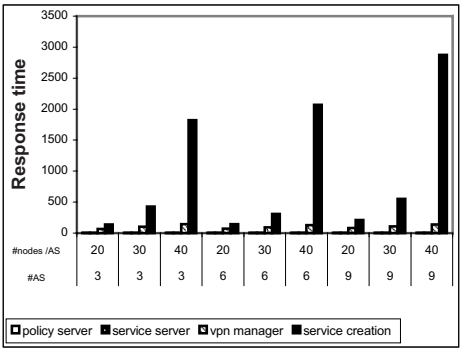

Fig. 4. Response time for 20 requests/unit time

However, the effect of replication is more significant as we increase the load on the system. For example, for 200 requests per unit time, 9 autonomous systems and 40 elements/AS, the response time decreases by a factor of 20. This motivates the need for the parallel processing of requests.

This study demonstrates that it is important for a corresponding implementation of this design to support replication of processes and their corresponding nodes. Such replication is more important for the bandwidth broker and VPN management nodes than for other nodes.

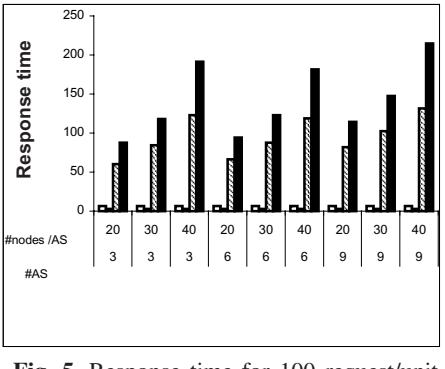

Fig. 5. Response time for 100 request/unit time

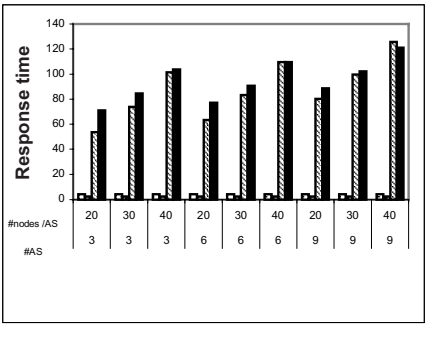

Fig. 6. Response time for 200 requests/unit time

Performance modeling is an important tool for those designing such systems. In our future work we plan to introduce further detail into the models and investigate the performance of the different techniques for creating and deploying VPNs. This will

help to determine which methods are best for various workload mixes and network configurations.

Furthermore, we believe this work complements programmable network research. It offers a framework that could be used to reflect and predict the impact of programmed network changes on a system as a whole.

Table 1. **Response Time (based on units) for processes at different replication levels**

			1 *VPN_mgmt* and 1 broker processes		1 *VPN_mgmt* and 10 brokers processes		10 *VPN_mgmt* and 10 brokers processes	
#users	#AS	#elements per AS	*VPN mgmt.*	Service creation	*VPN mgmt.*	Service creation	*VPN mgmt*	Service creation
20	3	20	53	71	52	70	52	69
20	3	30	74	84	72	83	72	80
20	3	40	101	103	97	101	97	96
20	6	20	63	77	62	77	62	75
20	6	30	83	90	82	90	82	87
20	6	40	109	109	107	108	107	102
20	9	20	80	88	79	88	79	85
20	9	30	99	102	99	101	99	97
20	**9**	**40**	**125**	**121**	**124**	**120**	**124**	**112**
100	3	20	59	87	53	80	53	72.
100	3	30	84	118	73	102	73	84
100	3	40	123	191	99	142	99	101
100	6	20	66	94	63	91	63	78
100	6	30	87	123	82	115	82	90
100	6	40	118	181	108	159	108	107
100	9	20	81	114	79	111	79	89
100	9	30	102	147	99	141	99	101
100	9	40	131	215	125	197	125	118
200	3	20	69	140	54	100	54	74
200	3	30	101	426	74	159	74	88
200	3	40	145	3823	101	423	101	107
200	6	20	70	143	63	122	63	80
200	6	30	94	308	83	206	83	94
200	6	40	129	2072	109	654	109	113
200	9	20	84	213	80	188	80.	92
200	9	30	106	550	99	387	99	106
200	**9**	**40**	**137**	**2876**	**125**	**1717**	**125**	**125**

References

1. "B-ISDN Service Aspects." ITU-T recommendations I.211
2. "Terms and Definitions relate to the Quality of Telecommunications Services." ITU-T recommendation E.800
3. Umar, A.: "Distributed Computing: A Practical Synthesis." Englewood Cliffs, Prentice Hall, N.J. 1993
4. Woodside, C.M.: "Performability Modeling for Multi-Layered Service Systems", Third International Workshop on Performability of Computer and Communications Systems, Bloomingdale, Illinois, USA, Sept. 7-8, 1996
5. Caswell, D., Ramanathan, S.: "Using Service Models for Management of Internet Services." HP Labs, HPL-1999-43, CA, 1999.
6. Wetherall, D., Legedza, U., Guttag, J.: "Introducing New Internet Services: Why and How." IEEE Network, May/June, 1998
7. Sheihk, F., Rolia, J., Garg, P., Frolund, S., Shepherd, A.: "Layered Modeling of Large Scale Distributed Applications," Appears in the proceedings of the 1st World Congress on Systems Simulation, Quality of Service Modeling, Singapore, September 1-3, 1997, pages 247-254
8. Franks G., Woodside, C.M.: "Performance of Multi-level Client-Server Systems with Parallel Service Operations", Proc. First Int. Workshop on Software and Performance (WOSP98), pp. 120-130, Santa Fe, October 1998
9. Vicente, J., Miki, J.: "Designing IP Router L-Interfaces." IP Sub Working Group, IEEE P1520, http://www.ieee-pin.org/
10. Rolia, J.A., Sevcik, K.C.: "The Method of Layers," IEEE Transactions on Software Engineering, Vol. 21, No. 8, pp. 689-700, August 1995
11. Lazar, A.: "Programming Telecommunication Networks." IEEE Network, September/October 1997
12. Asawa, M.: "Measuring and Analyzing Service Levels: A Scalable Passive Approach." HP labs, HPL-97-138, Palo Alto, CA, 1997. http://fog.hpl.external.hp.com/techreports/
13. Decina, M., Trecordi, V.: "Convergence of Telecommunications and Computing to Networking Models for Integrated Services and Applications." Proceedings of the IEEE, vol. 85, No. 12, December 1997
14. Jogalekar, P., Woodside, C. M.: "Evaluating the Scalability of Distributed Systems", Proc. of Hawaii Int. Conference on Systems Sciences, January 1998
15. Smith, C.: "Performance Engineering of Software Systems", Addison-Wesley, 1990

Acknowledgments

The authors would like to thank Ferass El-Rayess, Anant Jalnapurkar, and Greg Franks for the help they provided during this research. This work was supported by grants from the Ontario Centers of Excellence CITO program and Nortel Networks.