An Extension of Integrated Services with Active Networking for Providing Quality of Service in Networks with Long-range Dependent Traffic

Tibor Gyires

Applied Computer Science Department
Illinois State University, Normal, Illinois 61790-5150
tbgyires@ilstu.edu

Abstract

Although today's network capacity is exponentially increasing, new applications are demanding higher and higher bandwidth. The available bandwidth seems always less than what the new applications require. This tendency results in congested networks and packet loss that we can expect to continue in the foreseeable future. Congestion can be caused by several factors. The most dangerous cause of congestion is the burstiness of the network traffic. Recent results make evident that high-speed network traffic is more bursty and its variability cannot be predicted as assumed previously. It has been shown that network traffic has similar statistical properties on many time scales. Traffic that is bursty on many or all time scales can be described statistically using the notion of long-range dependency. Long-range dependent traffic has observable bursts on all time scales. Reasons, such as traffic burstiness, make providing Quality of Service (QoS) in high-speed networks increasingly important. QoS implies mechanisms to avoid congestion by allocating network resources optimally, rather than continually increasing network capacities. The objective of our paper is to present an extension of a QoS mechanism called Integrated Services (IntServ) with active networking in networks with long-range dependent traffic.

Keywords

Telecommunications, simulation modeling, active networks, traffic burstiness.

1 Introduction

Today's digital communications networks are store-and-forward networks. In a store-and-forward network, a message from a sending host computer to a destination host computer is divided into basic transmission units called packets. Packets are transmitted via a sequence of communications devices called routers or switches, which are interconnected by communications links. The purpose of the network is to support the sharing of the communications links. When too many packets are present in (a part of) the subnet, performance degrades. This situation is called congestion. When the number of packets sent into the subnet by the hosts is within the network's carrying capacity, the packets are all delivered (except for a few that are damaged or lost due to transmission errors), and the number delivered is proportional to the number sent. However, as traffic increases too far, the routers are no longer able to forward the packets, and they begin losing them. This tends to make matters worse. At very high traffic, performance collapses completely, and almost no packets are delivered. Congestion can be caused by several factors. The most dangerous cause of congestion is the burstiness of the network traffic. Recent results make evident that high-speed network traffic is more bursty and its variability cannot be predicted as assumed previously. It has been shown that network traffic has similar statistical properties on many time scales. Traffic that is bursty on many or all time scales can be described statistically using the notion of long-range dependency [14,15,17,18,19]. Long-range dependent traffic has observable bursts on all time scales. For reasons, such as traffic burstiness providing Quality of Service (QoS) in high-speed networks becomes increasingly important. QoS implies mechanisms to avoid congestion by allocating network resources optimally, rather than continually increasing network capacities. The optimal allocation of the forwarding capacity of routers and switches are the goal of QoS. Resource Reservation

0-7803-7087-2/01/$10.00 © 2001 IEEE

Protocol (RSVP) is one of several other protocols that intend to provide QoS. The RSVP is part of the framework called Integrated Services (IntServ) that provides end-to-end QoS for network applications.

A new research direction, programmable or active networking, offers a promising solution for the implementation of congestion control protocols. Current packet-switched networks enable the sharing of transmission facilities so that packets may be efficiently moved among connected systems. Traditional packet networks perform only the processing necessary to forward packets toward their destination. As computing power becomes cheaper more and more functionality can be deployed inside the network. Programmable networks represent a significant step in this evolution by providing a programmable interface in network routers and mechanisms for constructing or refining new services from existing network services. Programmable networks support dynamic modification of the network software and hardware to manipulate the network's behavior. Routers receive packets from users and other routers, and then perform a computation based on the control information carried in the packet. As a result of that computation, the routers may forward one or more packets toward other routers or to users.

A special way of network programmability is when special programs called software or software agents are carried in the packets to the routers. Software agents are loaded, executed, migrated, and suspended in order to implement some network functions. Using system-independent execution environment like Java, a piece of code can be injected into the network that can traverse from router to router. The execution environment of the routers can perform the code required for some network functions. Software agents are immediate solutions for rapid-prototyping of network protocols and new software systems as long as routers' programmability has not been implemented in a standardized manner in switches and router. Due to the ever-increasing demand for new network services and slow standardization processes, software agents are anticipated to implement new services before they become standards. Software agents [3,4,5,8, 9] provide the highest possible degree of flexibility in congestion control as well. They can carry congestion-specific knowledge into the network at locations where it is needed, rather than transferring information to the sending hosts as it happens in traditional solutions.

We propose an extension of the IntServ model with active networking methodology in order to make it more suitable for controlling congestion caused by bursty traffic. Several methods have been implemented to provide QoS and avoid congestion, but they are not responsive enough to the varying transmission capacity and network delay in high-speed networks. The larger the end-to-end delay, the longer it takes to inform the sending nodes that the network has become congested. Our approach offers a new solution for providing QoS. We describe an active networking architecture for minimizing the packet loss due the congestions caused by long-range dependent traffic. When the traffic burstiness of a packet flow increases at a router, the router distributes the flow over k new paths leading to the destination(s) of the original traffic. The selection of the new paths is based on cost estimates of the traffic distribution and routers' effectiveness in reducing the number of dropped packets in previous traffic distributions. We define an architecture for the components of our active network and the behavior of the software agents. Using a discrete event simulation model we demonstrate how our algorithm improves the IntServ model.

The second section gives a brief overview of the IntServ framework and RSVP. The third section presents the Software Agents Framework. The proposed Software Agents Algorithm is described in the forth section followed by the conclusion.

2 Overview of IntServ

The purpose of the IntServ framework is to provide QoS support in IP based networks [1,2, 23, 24]. The framework defines a "flow" as a stream of related datagrams that is created by a single application or user activity and requires the same QoS. A flow is a simplex mode of packet transmission. Quality of Service is implemented in a router and a host for a particular data flow by a module called traffic control. It includes a Packet Classifier, Admission Control, Policy Control, Packet Scheduler, and Estimator. First, the Packet Classifier determines the QoS class for each packet. It classifies various packet flows of different applications and gives higher priority for those packets that are eligible for higher priority handling. The result of traffic classification is that the flow is assigned to a

corresponding buffer queue for forwarding. Secondly, routers need to apply some queue servicing algorithm that determine the rate and the method of packet forwarding from each queue. The Packet Scheduler manages the queues for each output port and determines the order of packets and the packets to be discarded. The decision is based on the packet classification and the Traffic Control Database. During reservation setup, the Admission Control and Policy Control modules validate a QoS request. Admission Control determines whether the router has sufficient available resources to deliver the requested QoS. Policy Control determines whether the user has administrative authorization to request the reservation. The Estimator measures the characteristics of traffic flows, calculates statistics that control the packet scheduling and admission control. The components above are supported by the Reservation Setup Agent and the Management Agent background routines. The Reservation Setup Agent implements RSVP to set up resource reservations. The Management Agent modifies the Traffic Control Database and sets up controlled link-sharing and admission control policies.

RSVP reserves resources in routers along the traffic flow as requested by an application according to the traffic classification and queue servicing algorithms. RSVP can be considered the configuration protocol for IntServ. The sending host sends a PATH message to the destination(s) of the traffic flow. It carries the flow specification of the offered load: the identity of the application, the traffic characterization, (bandwidth, burstiness, average and peak transmission rate), and data for traffic classification. When the PATH message arrives at the destination(s), it responds by sending RESV messages identifying the requested flow specification: the receiving application, the type of IntServ services (latency bound, minimum bandwidth guarantee, etc.) and the amount of resources requested by the destination(s). RESV messages return to the source along the reverse path of the PATH messages. As RESV messages arrive at the routers, the devices decide whether there are sufficient resources available for the traffic flow. If there are enough resources and there is no conflict between the request and local policies established for the destination, the request is admitted. Otherwise, the request is rejected and an error message is sent to the network devices along the path notifying other RSVP-aware devices that the request was not admitted. (Although RSVP has many drawbacks, such as lack of scalability, recently it has come back stronger for reasons, such as changing the way RSVP handles per-flow traffic and the processing of the PATH and RESV signaling messages only at selected nodes.)

3 Software Agents Framework

Our Software Agent Framework extends IntServ by adding new components and new functionalities to the framework. The new module, the Executive executes the code encapsulated in the software agents and carries out the functions requested by them. The new functionalities further augment the functions of the Reservation Setup Agent and the Estimator. We assume that the Traffic Control database has the following knowledge of flows terminating at or crossing the router: flow identifier, required resources, priority, and the maximum number of flows that can be established on outgoing links.

Local set up cost: Assuming a packet flow T_i. In the subsequent paragraphs we will explain that T_i is going to be distributed over some number of new paths if the traffic burstiness exceeds a threshold value. Let's denote the distributed portion of the packet flow T_i on path k by T_{ik}. Each router that is in a distribution path may have to make some resource and routing reassignments. $L_{ik}(R)$ denotes the cost of such a set-up at a router R for a distributed flow T_{ik}. It includes the costs for distribution activities, such as assigning buffers to the flow being distributed, rerouting lower priority flows, rearranging the router to accept the new traffic flow, etc. The local Estimator estimates this cost.

Number of dropped packets: Each Estimator can estimate the packet loss in the new, distributed flow. $D_{ik}(R)$ denotes the estimated number of packets dropped at router R in the distributed flow T_{ik}.

The E-factor (Effectiveness): The E-factor for a router R, denoted by $E_{ik}(R)$, is a measure of its inefficiency in rerouting a distributed portion T_{ik} of the packet flow T_i on path k. Small (close to zero) values of the E-factor indicate high quality and efficiency, and high values indicate that router R is unreliable and unstable in rerouting of distributed traffic. The Estimator recalculates a router's E factor after each traffic distribution using the E values from previous traffic

distributions. After each traffic distribution, it is determined how realistic a router's cost estimate has been with respect to the actual cost. By testing a statistical hypothesis it can be determined if it has been unrealistic. If the corresponding hypothesis H_0 is rejected no request messages will be sent to this router in the future for diverting distributed traffic. If a router's cost estimate has been realistic, i.e., the corresponding hypothesis H_0 is accepted, the E factor is recalculated. The closer the estimated cost is to the actual cost, the smaller is the E factor. A local network administrator provides initial values of the E-factor. Subsequent values of the E-factor are calculated using the statistical sampling method given in [10]. The motivation for using the E-factor is to enable the traffic distribution process to learn from past performances and use this knowledge to select the most efficient distribution paths.

Distribution cost: The cost associated with a distributed flow T_{ik} of flow T_i is the sum of local set up costs, the estimated dropped packets, and the E-factors along a path k. The cost of a distributed flow $T_{ik}(R,P)$ between two remote routers R and P is denoted by $C(T_{ik}(R,P))$ and defined recursively by:

$C(T_{ik}(R,P)) = [C(T_{ik}(Q,P) + E_i(Q) + L_{ik}(Q) + D_{ik}(Q)]$,

where R and Q are adjacent routers.

4 Software Agents Algorithm

We propose a new network algorithm using software agents that can reduce the harmful consequences of congestions due to aggregated bursty traffic, such as packet losses, extremely long response times, overrunning communications link capacities, etc. Our algorithm also eliminates the undesirable delay inherent in traditional congestion control methods.

IntServ recommends the use of the Token Bucket scheme to characterize the degree of burstiness. A Token Bucket is defined by two parameters: a token rate R that specifies the sustainable data rate of the traffic flow, and the bucket size B that specifies the amount by which the data rate can exceed R for a short periods of time. But measurements of real traffic indicate that burstiness is present on a wide range of time scales that can be described statistically using the notion of long-range dependency. Long-range dependent traffic has observable bursts on all time scales. Many methods for measuring traffic burstiness are based on the estimate of the Hurst parameter H in addition to the bursts' mean and variance. Therefore, measuring burstiness based on the Hurst parameter is more appropriate than the Token Bucket scheme that is based on measurements of only short periods of time. We are going to apply the statistical method developed in [11] to estimate the Hurst parameter of bursty traffic flows.

In our algorithm, the flow specification of a new packet flow and subsequent control packets carry the anticipated traffic's Hurst parameter. It is calculated from previous connections and applications' traffic patterns at the sending host. It can also be estimated and updated by intermediate routers. Different routers can accommodate the same bursty traffic differently. E.g., a router can reserve more input buffers for the incoming packet flow; another router can assign more processing power for handling the flow, etc. In order to establish a common metrics in terms of network performance parameters, we assume that each router can convert the Hurst parameter to the traffic volume V in bits per second that the router can handle based on empirical observations. (Note, that this measurement is not necessarily the forwarding capability of the router.)

When the Hurst parameter of the flow specification of a new request or the traffic burstiness of an existing flow exceeds a certain threshold, the router divides the traffic into k new paths. Instead of using a single path, the router will divide the traffic flow over k number of new paths, leading to the destination(s) of the original traffic. (For simplicity, we assume unicast instead of multicast sessions. The algorithm below can easily be expanded for multicast as well.) The k new paths will reunite at the destination using some existing techniques, such as packet sequence numbering. The traffic distribution is implemented in the following steps:

1. Assume that router *A* detected that the traffic burstiness of a flow or the Hurst parameter of a new request exceeds a certain threshold. *A* estimates the volume requirement of the flow based on the Hurst parameter. *A* sends a PATH message and the flow specification to the destination router *X* requesting resources for the flow.

2. Router *X* creates software agents RESV; packets with code and data that are sent back to *A* on all paths towards *A*. The RESV packets are

reproduced and executed at each subsequent router along the paths to *A*.

3. Each RESV stores all visited routers along the path in the packet's data field.

4. Each RESV carries the flow specification requesting resources from the Reservation Setup Agent for traffic distribution at each router. The Estimator estimates the resource requirement of the flow. If there are enough resources, the request is tentatively admitted and reservation is made for a portion or for all of the requested resources. When the request is rejected reject messages are sent along the data path, notifying RSVP–aware routers of the failure. (Note that the reservation does not guarantee that packets will not be lost because it is based on an estimate only.) The reservation is valid only for a limited t time interval. If there is no confirmation arriving for the reservation in time t the resources can be reallocated for other purposes.

5. Each RESV collects estimates of the local set up cost, dropped packets, the E-factor, and the reserved traffic volume from the Estimator at each visited router.

6. Upon receiving all cost estimates from all paths, router *A* selects the k paths with the least distribution costs that collectively can carry the volume of the original flow, and distributes the new or existing traffic along the k new T_{ik} flows.

7. After traffic distribution, each Estimator compares the actual cost and the estimated cost and recalculates its own E-factor for subsequent traffic distribution [10]. It can be proven that the newly selected paths are optimal in terms of the distribution cost and overall packet losses. When the traffic burstiness falls below the threshold value, the distributed paths are collapsed into the original single flow. If the traffic's resource requirement exceeds the router's resources before the selection for the "least-cost" paths is completed, then *A* makes the decision immediately based on data from previous traffic distributions.

We implemented the Software Agent Algorithm in COMNET [16]. We ran the simulation for 1500 seconds with and without traffic distribution and measured the number of packets dropped at the routers. The simulation statistics show that 1340 packets have been dropped without traffic distribution and only 49 packets have been dropped with traffic dispersion due to the lack of buffer space. The details of the simulation cannot be included in this paper due to space limitations. The model can be obtained from the author upon request.

5 Conclusion

We described an extension to the IntServ framework for minimizing the packet loss due to congestion caused by long-range dependent traffic. When the traffic burstiness of a packet flow increases at a router, the router, with the help of software agents, distributes the flow over k new paths that lead to the destination of the original traffic. The selection of the new k paths is based on cost estimates of the traffic distribution and the effectiveness of the routers in reducing the number of dropped packets in previous distributions. We defined an architecture for the components of our active network that can provide QoS and minimize the packet loss due to congestion. Although RSVP reserves the resources along the path in advance, it may happen that a new flow cannot be admitted due to the lack of enough capacities. Our approach admits flows that otherwise would not be admitted in the IntServ. Using a discrete event simulation model we demonstrated that our approach can improve the performance of the IntServ framework.

References

[1] Yoram Bernet, "The Complementary Roles of RSVP and Differentiated Services in the Full-Service QoS Network," IEEE Communications Magazine, February 2000, pp. 154-162.

[2] G. Eichler, H. Hussmann, G. Mamais, I. Venieris, C. Prehofer, and S. Salsano, "Implementing Integrated and Differentiated Services for the Internet with ATM Networks: A Practical Approach," IEEE Communications Magazine, January 2000, pp. 132-141.

[3] H. De Meer, A. LaCorte, A. Puliafito, and O. Tomarchio, "Programmable Agents for Flexible QoS Management in IP Networks," IEEE Journal on Selected Areas in Communication,Vol. 18, No.2, February 2000, pp. 256-266.

[4] B. Schwartz et al., "Smart Packets for Active Networks," IEEE OPENARCH '99, 1999.

[5] R. Kawamura and R. Stadler, "Active Distributed Management for IP Networks," IEEE Communications Magazine, April 2000, pp. 114-120.

[6] M. Genesereth and S. Ketchpel, "Software agents," Communications of the ACM, vol. 37, no. 7, July 1994, pp. 48–53.

[7] T. Magedanz, K. Rothermel, and S. Krause, "Intelligent agents: An emerging technology for next generation telecommunications," INFOCOM'96, San Francisco, CA, Mar. 1996.

[8] K. Rothermel and R. Popescu-Zeletin, Eds., "Software agents," in Lecture Notes in Comp. Science LNCS1219, 1997.

[9] D. Alexander, W. Arbaugh, A. Keromytis, and J. Smith, "The SwitchWare active network architecture," IEEE Network, vol. 12, May/June 1998, pp. 29–36.

[10] Gyires, T., Muthuswamy, B., "A Bidirectional Search Approach for Restoring Circuits in Communication Networks", The Journal of Computer Information Systems, Winter 1996-1997, pp. 85-93.

[11] Gyires, T., "Methodology for Modeling the Impact of Traffic Burstiness on High-Speed Networks," in the Proceedings of the 1999 IEEE Systems, Man, and Cybernetics Conference, October 12-15, 1999, Tokyo, Japan, pp. 980-985.

[12] C. Yang and A. Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks," IEEE Network Magazine, vol. 9, July/August 1995, pp. 34-45.

[13] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic," Computer Communications Review, 23, Proc. of the ACM/SIGCOMM'93, San Francisco, September 1993, pp. 183-193.

[14] Neame et al., "Investigation of traffic models for high speed data networks," in the Proceedings of ATNAC '95, 1995.

[15] M. Taqqu, V. Teverovsky, and W. Willinger, "Estimators for long-range dependence: an empirical study," Fractals, Vol. 3, No. 4 1995, pp. 785-788.

[16] "COMNET III Application Notes: Modeling ATM Networks with COMNET III" CACI Products Company, 3333 North Torrey Pines Ct., La Jolla, California 92037, 1998.

[17] R. Addie, M. Zukerman, and T. Neame, "Broadband Traffic Modeling: Simple Solutions to Hard Problems," IEEE Communications Magazine, August 1998, pp.88-95.

[18] N. Likhanov, B. Tsybakov and N. Georganas, "Analysis of an ATM Buffer with Self-Similar ("Fractal") Input Traffic," in Proceedings of the IEEE INFOCOM Conference, 1995, Boston, pp. 985-992.

[19] T. Neame, M. Zukerman, and R. Addie, " A Practical Approach for Multi-Media Traffic Modeling," in Proceedings of Broadband Communications '99, Hong Kong, 10 – 12, November, 1999, pp. 73 - 82.

[20] T. Neame, M. Zukerman, "Application of the M/Pareto Process to Modeling Broadband Traffic Streams, " in the Proceedings of ICON '99, Brisbane, Queensland, Australia, 28 September – 1 October, 1999, pp. 53–58.

[21] B. Mandelbrot, "The Fractal Geometry of Nature," W.H. Freeman and Co., San Francisco, 1982.

[22] V. Paxson and S. Floyd, "Wide-area traffic: The failure of Poisson modeling," IEEE/ACM Trans. Networking, vol. 3, June 1995, pp. 226–244.

[23] Request for Comments: 1633, R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture:an Overview," June 1994.

[24] Request for Comments: 2205, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP)," September 1997.