PERGAMON Mathematical and Computer Modelling 38 (2003) 839–848

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Software Agents Architecture for Controlling Long-Range Dependent Network Traffic

T. GYIRES
School of Information Technology
Illinois State University
Campus Box 5150, Bloomington-Normal, IL 61790-5150, U.S.A.
tbgyires@ilstu.edu

Abstract—Recent measurements of local and wide-area network traffic have proven that the widely used Markovian process models cannot be applied for today's network traffic. If the traffic were Markovian process, the traffic's burst length would be smoothed by averaging over a long time scale contradicting with the observations of today's traffic characteristics. Measurements of real traffic also prove that traffic burstiness is present on a wide range of time scales. Traffic that is bursty on many or all time scales can be characterized statistically using the concept of long-range dependency. Long-range dependent traffic has noticeable bursts, long periods with extremely high values on all time creating traffic congestions. Several conventional methods have been implemented to avoid congestion, but they are not responsive enough to the varying transmission capacity and network delay in high-speed networks.

A new research direction, programmable networking, offers a more promising solution for congestion control. 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 support dynamic modification of the network software and hardware to manipulate the network's behavior. A special way of network programmability is when special programs called mobile 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. Software agents provide the highest possible degree of flexibility in congestion control. 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 flow control solutions.

In our paper, we propose a new programmable network architecture using software agents that can reduce the harmful consequences of congestions due to aggregated bursty traffic, such as packet losses, extremely long response times, etc.

Keywords—Computer networks, Long-range dependent traffic, Active networking, Simulation modeling, Network performance.

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, 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

0895-7177/03/$ - see front matter Typeset by AMS-TEX
doi: 10.1016/S0895-7177(03)00285-1

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 (see the definition below). Long-range dependent traffic has observable bursts on all time scales.

One of the consequences is that combining the various flows of data, as it happens for example in the internet, does not result in the smoothing of traffic. Measurements of local area network traffic and wide-area network traffic have proved that the widely used Markovian process models cannot be applied for today's network traffic. If the traffic were Markovian process, the traffic's burst length would be smoothed by averaging over a long time scale contradicting with the observations of today's traffic characteristics. Combining bursty data streams will also produce bursty combined data flow. Various papers discuss the impact of the burstiness on network congestion [1–3]. Their conclusions are that congested periods can be quite long with losses that are heavily concentrated.

Several conventional methods have been implemented to avoid congestion, but they are not responsive enough to the varying transmission capacity and network delay in high-speed networks. A new research direction, programmable networking, offers a more promising solution for congestion control. A special way of network programmability is when particular programs called mobile or software agents are carried in the packets to the routers. The objective of our paper is to demonstrate a new programmable network architecture using software agents to avoid congestions without the negative effects of the traditional methods.

In Section 2, we provide the background information and the definitions of long-range dependent traffic and related terms. Section 3 discusses some traditional solutions for congestion control. Section 4 gives an overview of the literature of programmable networks with software agents. In Section 5, we present our software agents approach that implements a solution, called traffic distribution for congestion control. Section 6 gives the results of the simulation model implemented in the COMNET discrete event simulation tool. Section 7 summarizes the results of the paper.

2. BACKGROUND

For more details of long-range dependency in time series and the associated statistical tests, see [1–3]. Our paper follows the definitions of these papers.

2.1. Definitions

Let $X = (X_t : t = 0, 1, 2, \dots)$ be a covariance stationary stochastic process. Such a process has a constant mean $\mu = E[X_t]$, finite variance $\sigma^2 = E[(X_t - \mu)^2]$, and an autocorrelation function $r(k) = E[(X_t - \mu)(X_t + k - \mu)]/E[(X_t - \mu)^2]$ $(k = 0, 1, 2, \dots)$ that depends only on k. It is assumed that X has an autocorrelation function of the form

$$r(k) \sim \alpha k^{-\beta}, \qquad k \to \infty, \tag{1}$$

where $0 < \beta < 1$ and α is a positive constant. Let $X^{(m)} = (X_{(k)}^{(m)} : k = 1, 2, 3, \dots, m = 1, 2, 3, \dots)$ represent a new time series obtained by averaging the original series X over nonoverlapping blocks

of size m. For each $m = 1, 2, 3, \ldots$, $X^{(m)}$ is specified by $X_k^{(m)} = l/m\,(X_{km-m+1} + \cdots + X_{km})$ $(k \geq 1)$. Let $r^{(m)}$ denote the autocorrelation function of the aggregated time series $X^{(m)}$.

2.2. Definition of Long-Range Dependency

The process X is called exactly self-similar with self-similarity parameter $H = 1 - \beta/2$ if the corresponding aggregated processes $X^{(m)}$ have the same correlation structure as X; i.e., $r^{(m)}(k) = r(k)$, for all $m = 1, 2, \ldots$ $(k = 1, 2, 3, \ldots)$.

A covariance stationary process X is called asymptotically self-similar with self-similarity parameter $H = 1 - \beta/2$ if for all k large enough $r^{(m)}(k) \to r(k)$ as $m \to \infty$, $0.5 \leq H \leq 1$.

A stationary process is called long-range dependent if the sum of the autocorrelation values approaches infinity: $\sum_k r(k) \to \infty$. Otherwise, it is called short-range dependent. It can be derived from the definitions that while short-range dependent processes have exponentially decaying autocorrelations, the autocorrelations of long-range dependent processes decay hyperbolically; i.e., the related distribution is heavy-tailed. In practical terms, a random variable with heavy-tail distribution generates extremely large values with high probability [4,5].

The degree of self-similarity is expressed by the parameter H or Hurst parameter. The parameter represents the speed of decay of a process' autocorrelation function. As $H \to 1$ the extent of both self-similarity and long-range dependence increases. It can also be shown that for self-similar processes with long-range dependency $H > 0.5$ [6].

Many methods for measuring traffic burstiness are based on the estimate of the Hurst parameter in addition to the bursts' mean and variance. The Hurst parameter H is a measurement to characterize long-range dependency of the network traffic or the traffic burstiness; the higher the value of H, the higher the burstiness, and consequently, the worse the queuing performance of switches and routers along the traffic path.

2.3. The M/Pareto Traffic Model and the Hurst Parameter

Recent results in [7,8] have proven that the M/Pareto model is appropriate for modeling long-range dependent traffic flow characterized by long bursts. Originally, the model was introduced in [8] and applied in the analysis of ATM buffer levels. The M/Pareto model was also used to predict the queuing performance of Ethernet, VBR video, and IP packet streams in a single server queue [9,10]. We apply the M/Pareto model not just for a single queue, but also for predicting the performance of an interconnected system of links, switches, and routers affecting the individual network elements' performance. We make use of some of the calculations presented in [7–10]. The M/Pareto model is a Poisson process of overlapping bursts with arrival rate λ. A burst generates packets with arrival rate r. Each burst, from the time of its interval, will continue for a Pareto-distributed time period. The use of Pareto distribution results in generating extremely long bursts that characterize long-range dependent traffic. The probability that a Pareto-distributed random variable X exceeds threshold x is

$$P(X > x) = \begin{cases} \left(\dfrac{x}{\delta}\right)^{-\gamma}, & x \geq \delta, \\ 1, & \text{otherwise}, \end{cases} \qquad 1 < \gamma < 2, \quad \delta > 0. \tag{2}$$

The mean of X, the mean duration of a burst $\mu = \delta\gamma/(\gamma - 1)$ and its variance is infinite [9]. Assuming a t time interval, the mean number of packets M in the time interval t is

$$M = \frac{\lambda \operatorname{tr} \delta\gamma}{(\gamma - 1)}, \qquad \text{and} \tag{3}$$

$$\lambda = \frac{M(\gamma - 1)}{\operatorname{tr} \delta\gamma}. \tag{4}$$

The M/Pareto model is described in [9,10] as asymptotically self-similar and it is shown that for the Hurst parameter the following equation holds:

$$H = \frac{(3 - \gamma)}{2}. \tag{5}$$

3. TRADITIONAL SOLUTIONS FOR CONGESTION CONTROL

Solutions for congestion control can be divided into two groups [6]: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure congestion does not occur in the first place. Once the system is up and running, corrections are not made. Tools for doing open-loop control make decisions without regard to the current state of the network. These tools are not suitable for a dynamic, high-speed network environment. Closed loop solutions are based on the concept of a feedback loop and are more appropriate for dynamic, high-speed networks than closed loop solutions. This approach has three parts when applied to congestion control.

(1) Monitor the system to detect when and where congestion occurs.
(2) Pass this information to places where action can be taken.
(3) Adjust system operation to correct the problem.

Various metrics can be used to monitor the subnet for congestion. The most relevant metrics are the percentage of all packets discarded for lack of routers' buffer space, the average queue lengths, the average packet delay, and the standard deviation of packet delay. In all cases, rising numbers indicate growing congestion.

The second step in the feedback loop is to transfer the information about the congestion from the point where it is detected to the point where something can be done about it. The traditional way is for the router detecting the congestion to send a packet to the traffic source or sources, announcing the problem. Obviously, these extra packets increase the load when the subnet is already congested. Feedback based congestion control systems are not responsive enough to the varying capacities of transmission links and network delay. The larger the end-to-end delay, the longer it takes to inform the sending nodes that the network has become congested. It was shown in [11] that the duration of the congestion at the congested routers is directly related to the bandwidth-delay product.

In all traditional feedback schemes, the expectation is that knowledge of congestion will cause the hosts to take appropriate action to reduce the congestion. But due to the inherent network delay, the hosts will react too sluggishly to be of any real use. End-to-end feedback mechanisms introduce an unacceptable delay. The situation is getting even worse when the traffic is generated at high-speed in long bursts and the traffic characteristics cannot be predicted by traditional statistical methods.

4. PROGRAMMABLE NETWORKS WITH SOFTWARE AGENTS

A new research direction, programmable or active networking, offers a more promising solution for congestion control. 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 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 a 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 [12–14] 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.

4.1. Some Research Initiatives

The smart packets project [15] is applying programmable network technology to assist with the growing problem of managing networks. The design goal of the project is to be able to encode meaningful and useful network management programs in less than 1 Kbytes; thus, it provides very compact representations that fit into single packets.

The ANTS toolkit [16], developed at MIT, features the notion of a capsule, which is a packet containing a small Java program. Capsules are decoded and executed by a Java virtual machine in the routers. Because the standard Java virtual machine does not support access to transmission resources at a sufficiently low level, implementations of ANTS are limited to the basic network capabilities provided by Java.

The ACC project recommends a programmable network solution to reduce the delay when congestion is signaled to the sending node [17]. When congestion occurs at a router, it determines the congestion window size (the number of packets the router can store), drops packets that would not be sent with the new window size, and notifies the sender about the new window size. Although the ACC solution is more responsive than traditional methods, packet drops are still inevitable and it has a built-in delay of notifying the sender about the new window size.

5. SOFTWARE AGENT ARCHITECTURE

Our programmable network consists of IP routers that connect users via virtual channels (VCs) that in turn are bundled into virtual paths (VPs) similarly to an ATM network configuration. Sending and receiving users communicate through a flow of packets carrying IP datagrams similarly to the model of the IETF's resource reservation protocol [18]. Each router has two main components: a forwarding part and an execution part. The forwarding part provides the traditional IP datagram routing services for the network. The execution part executes the code encapsulated in the software agents and carries out the functions requested by them. The execution part is further divided into two segments: the resource scheduler (RS) that manages the required resources of VCs, VPs, and packet flows, and the cost estimator (CE) that can estimate the cost of traffic management activities. The CE has knowledge of local conditions, resources, and capacities. We assume that each CE 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 SETUP COST. Assume 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 us 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 setup at a router R for a distributed flow T_{ik}. It includes the costs for diverting 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 network administrator estimates this cost.

NUMBER OF DROPPED PACKETS. Each router also estimates 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 $E_i(R)$, 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 local CE 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. (As an alternative, in case of rejecting the hypothesis H_0, the router's E-factor can be increased by a large constant in order for the router to compete in further traffic distribution processes with low chances to win.) 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 [19]. 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 a flow T_i is the sum of local setup 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

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

where R and Q are adjacent routers.

5.1. 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.

We are going to apply the statistical method developed in [20] to estimate the Hurst parameter. In our algorithm, the first packet 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. The Hurst parameter is merely a characterization of bursty traffic. Different routers can accommodate the same bursty traffic differently. That is, 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 traffic burstiness at a router 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 of the original traffic. 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 T_i exceeds a certain threshold. A estimates the volume requirement of the flow based on the Hurst parameter.

A sends a control packet and the Hurst parameter to the destination router X requesting resources for the flow.

(2) Router X creates software agents (SAs), packets with code and data that are sent back to A on all paths towards A. SAs are reproduced and executed at each subsequent router along the paths to A.
   (a) Each SA stores all visited routers along the path in the packet's data field.
   (b) Each SA requests resources from the resource scheduler (RS) for traffic distribution at each node based on the Hurst parameter. An RS estimates the resource requirement of the flow and makes a tentative reservation for a portion or for all of the resources required reserving buffer space and CPU capacities. (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.
   (c) Each SA collects estimates of the local setup cost, dropped packets, the E-factor, and the reserved volume from the CE at each visited router.

(3) 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 traffic along the k new T_{ik} flows.

(4) After traffic distribution, each CE compares the actual cost and the estimated cost and recalculates its own E-factor for subsequent traffic distribution [19].

It can be proven by a slight modification of the methods in [19,20] that the newly selected paths are optimal in terms of the distribution cost and overall packet losses. It follows from the calculation of the distribution cost and the E-factor that it is a higher possibility for a router to receive distributed traffic if it can estimate the distribution cost close to the actual cost. Note that the ability to give a close cost estimate is not independent of carrying out traffic distribution. The distribution cost formula includes the E-factor that is based on the router's performance in previous traffic distributions. The E-factor is small, close to zero value if the router has statistically demonstrated that it is able to distribute the traffic with small costs. Therefore, a router's ability to estimate distribution cost depends on carrying out traffic distribution.

When the traffic burstiness falls below the threshold value, the distributed paths are collapsed into the original single flow. The simulation results are going to show that our algorithm can reduce the harmful consequences of bursty traffic, such as packet losses, high link utilization, and long response time.

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.

6. SIMULATION RESULTS

We implemented the Hurst parameter and a modified version of the M/Pareto model in the discrete event simulation system COMNET [21,22]. The parameters of the M/Pareto model correspond to a bursty traffic stream collected from high-speed ATM research networks. We assume that the selection algorithm above has already identified the paths with the least cost estimates. In order to demonstrate our distribution algorithm we constructed the following simplified model in COMNET's distributed software module shown in Figure 1.

In Figure 1, A, B, C, D, E, F, G, and X denote the routers of the ATM network connected by OC-3 links: Link AB, Link BE, etc. A is the origin; X is the destination of an ATM connection. An application source called "bursty source" attached to router A represents the connection. The "bursty source" application sends messages to the destination X with varying Hurst parameter. If the traffic burstiness is less than a threshold, A will choose a path via B. Otherwise, it will disperse

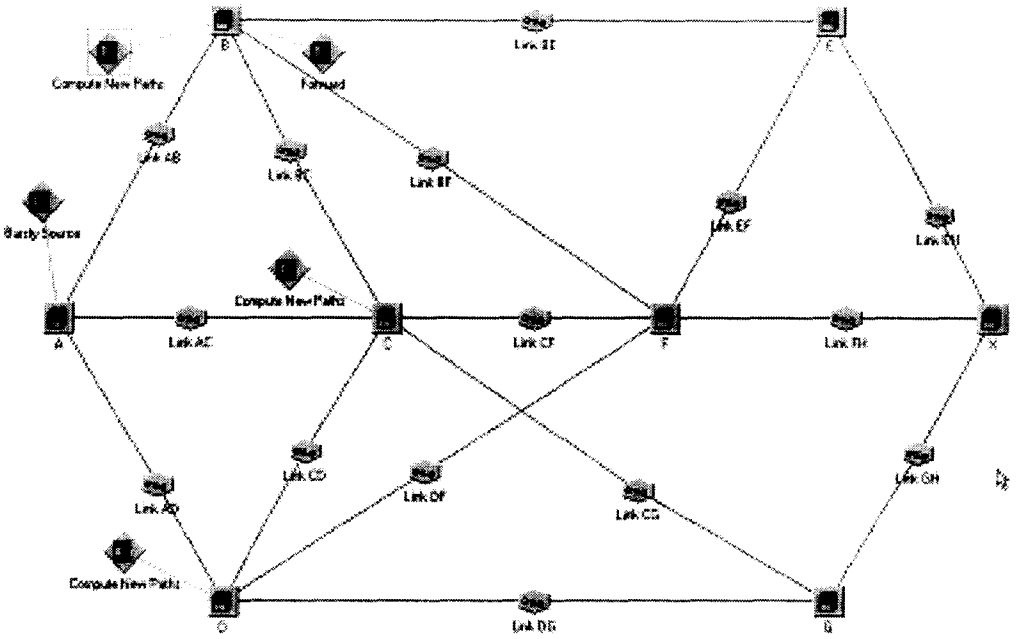


Figure 1. The COMNET model of the network.

the traffic among the adjacent routers identified by the selection algorithm implemented in the "compute new paths" applications. 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 (Table 1) 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.

Table 1. Number of packets dropped and input buffer usage.

Node	# of Packets Blocked		Max. Buffer Use in Bytes	
	Without Traffic Distribution	With Traffic Distribution	Without Traffic Distribution	With Traffic Distribution
B	1340	49	989274	971099
C	0	49		971099
D	0	40		960250

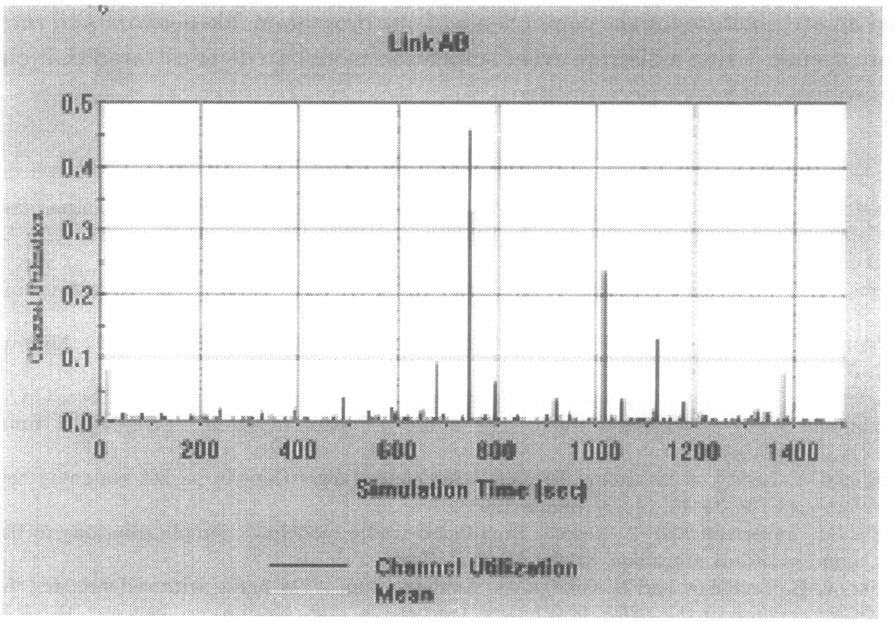
Table 2. Average and maximum utilization of Link AB.

Link AB	Average Utilization	Maximum Utilization
Without traffic distribution	0.26%	46.7%
With traffic distribution	0.0848%	12.8%

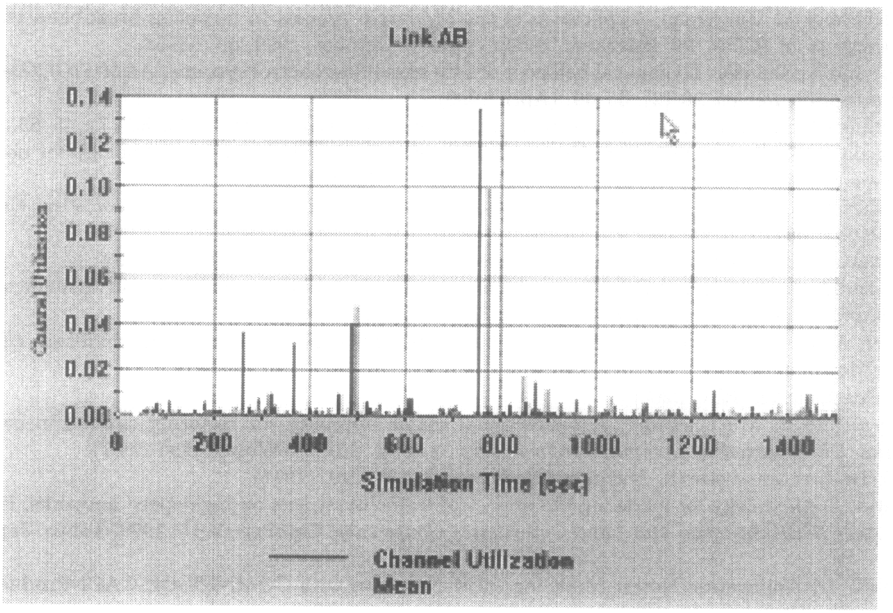
For Link AB we received the utilization with and without traffic distribution shown in Figure 2. It is shown that the link capacity is not a bottleneck in this particular simulation. The bottleneck is the routers' buffer capacity. We received the figures shown in Table 3 for the corresponding message delay between A and X.

Table 3. Average and maximum message delay between A and X.

Message Delay between A and X	Average Delay (msec)	Maximum Delay (msec)
Without traffic distribution	52.843	1183.193
With traffic distribution	44.556	443.675



(a)



(b)

Figure 2. Utilization of Link AB (a) without and (b) with traffic distribution.

Therefore, the simulation results demonstrate that the traffic distribution improves the network performance.

7. CONCLUSION

We described a software agent architecture for minimizing the packet loss due the congestions caused by long-range dependent traffic. Software agents 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. 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 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 distributions.

We defined an architecture for the components of our programmable network and the behavior of the software agents. Using a discrete event simulation model we demonstrated that our algorithm could improve the network's performance.

REFERENCES

1. W. Leland, M. Taqqu, W. Willinger and D. Wilson, On the self-similar nature of Ethernet traffic, In *Computer Communications Review, Proc. of the ACM/SIGCOMM '93*, Volume 23, pp. 183–193, San Francisco, September 1993, (1993).
2. T. Neame *et al.*, Investigation of traffic models for high speed data networks, In *Proceedings of ATNAC '95*, (1995).
3. M. Taqqu, V. Teverovsky and W. Willinger, Estimators for long-range dependence: An empirical study, *Fractals* **3** (4), 785–788, (1995).
4. B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman and Co., San Francisco, CA, (1982).
5. V. Paxson and S. Floyd, Wide-area traffic: The failure of Poisson modeling, *IEEE/ACM Trans. Networking* **3**, 226–244, (June 1995).
6. C. Yang and A. Reddy, A taxonomy for congestion control algorithms in packet switching networks, *IEEE Network Magazine* **9**, 34–45, (July/August 1995).
7. R. Addie, M. Zukerman and T. Neame, Broadband traffic modeling: Simple solutions to hard problems, *IEEE Communications Magazine*, 88–95, (August 1998).
8. 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, IEEE Press, (1995).
9. 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.
10. T. Neame and M. Zukerman, Application of the M/Pareto process to modeling broadband traffic streams, In *Proceedings of ICON '99*, Brisbane, 28 September–1 October, 1999, pp. 53–58.
11. J. Bolot and A. Shankar, Dynamical behavior of rate based flow control systems, *ACM SIGCOMM Computer Communication Review* **20** (2), 35–49, (April 1990).
12. M. Genesereth and S. Ketchpel, Software agents, *Communications of the ACM* **37** (7), 48–53, (July 1994).
13. T. Magedanz, K. Rothermel and S. Krause, Intelligent agents: An emerging technology for next generation telecommunications, In *INFOCOM '96*, San Francisco, (March 1996).
14. K. Rothermel and R. Popescu-Zeletin, Editors, Software agents, In *Lecture Notes in Comp. Science LNCS1219*, (1997).
15. B. Schwartz *et al.*, Smart packets for active networks, *IEEE OPENARCH '99*, (1999).
16. D. Wetherall, J. Guttag and D.L. Tennenhouse, ANTS: A toolkit for building and dynamically deploying network protocols, In *Proc. of OpenArch '98*, San Francisco, (1998).
17. T. Faber, ACC: Using active networking to enhance feedback congestion control mechanisms, *IEEE Network*, 61–65, (May/June 1998).
18. RSVP Specification [RFC 2205].
19. T. Gyires and B. Muthuswamy, A bidirectional search approach for restoring circuits in communication networks, *The Journal of Computer Information Systems*, 85–93, (Winter 1996–1997).
20. I. Pohl, Bi-directional search, *Machine Intelligence* **6**, 127–140, (1971).
21. T. Gyires, Methodology for modeling the impact of traffic burstiness on high-speed networks, In *Proceedings of the 1999 IEEE Systems, Man, and Cybernetics Conference*, October 12–15, 1999, Tokyo, Japan, pp. 980–985.
22. COMNET III Application Notes: Modeling ATM Networks with COMNET III, CACI Products Company, La Jolla, CA, (1998).
23. J. Nagle, On packet switches with infinite storage, *IEEE Transactions on Communications* **35**, 435–438, (April 1987).
24. 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* **18** (2), 256–266, (February 2000).
25. R. Kawamura and R. Stadler, Active distributed management for IP networks, *IEEE Communications Magazine*, 114–120, (April 2000).
26. D. Raz and Y. Shavitt, An active network approach to efficient network management, In *1st International Working Conference on Active Networks*, Berlin, (July 1999).
27. D. Alexander, W. Arbaugh, A. Keromytis and J. Smith, The SwitchWare active network architecture, *IEEE Network* **12**, 29–36, (May/June 1998).
28. J. Pearl, *Heuristics*, pp. 63–64, Addison-Wesley, (1984).