MPOA Flow Classification Design and Analysis

Hao Che
Department of Electrical Engineering
The Pennsylvania State University
State College, PA 16802

San-qi Li
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78712

***Abstract*—In this paper, we propose a framework for the performance analysis and flow classification design of Multi-Protocol Over ATM (MPOA) network. The study based on the real Internet/intranet traces shows that even at high cost with long delay for each shortcut setup, MPOA can offer significant performance gain over the traditional routed network in an inter ELAN communication environment. In comparison, the MPOA performance gain in an Internet backbone environment is much less significant, mainly because of the dominant short-lived flows contributed by both *dns* and *http* applications. We also propose a flow classification algorithm, which substantially reduces the implementation complexity while achieves the same level of performance as compared to the default flow classification algorithm proposed by MPOA standard. A simple timeout mechanism is also introduced to the flow cache table management for significant performance improvement. We further develop a stable, adaptive flow classification algorithm, which achieves a near-optimal solution to minimize the constrained MPOA resource utilizations.**

1 Introduction

After more than two years of effort by the Multi-Protocol Over ATM (MPOA) Working Group at the ATM Forum, MPOA was ratified as an industry standard in July, 1997. MPOA is an inter emulated LAN (ELAN) communication technology, which supports the transport of connectionless traffic via both layer-3 hop-by-hop forwarding through intermediate routers and layer-2 shortcut switching through ATM network. The objective is to alleviate the burden of rapidly increasing inter subnet traffic on routers, which have increasingly become the bottleneck in traditional routed network [1]. To the best of our knowledge, the effectiveness of MPOA technology has not been well studied. Major concerns have been raised with regard to its performance, mainly because of the complex signalling structure and large processing/propagation delays involved in each shortcut setup. In other words, the network resources required by shortcut setup may substantially undermine its claimed throughput performance.

In this paper, we develop a framework to characterize individual MPOA resources and propose a unified measure to quantitatively capture the overall resource requirement. The MPOA performance can therefore be studied on the basis of the real Internet/intranet trace simulations. Note that MPOA is data-driven, i.e., a shortcut setup is triggered by the first few packets of a flow and thus traffic flow dynamics have substantial impact on MPOA performance. One major contribution of this work is to show that MPOA can indeed provide significant performance gain over the traditional routed network in an inter ELAN environment. Even at high cost with large round-trip delay for each shortcut setup, its performance gain can reach more than a ten fold provided that a sufficiently large cache table size is available to accommodate most of the long-lived flows. In contrast, the performance gain in an Internet backbone environment is found to be much less significant, mainly due to the short-lived flow nature of *dns* and *http* traffic.

Flow classification (FC) and cache table management (CTM) are the two critical MPOA design issues, which have not been carefully examined. In an MPOA network, each multiprotocol client (MPC) runs a FC algorithm to identify every long-lived flow for shortcut setup; the state information for each shortcut connection is maintained by a CTM scheme. Hence, the MPOA performance is largely dependent on the design of FC algorithm and CTM scheme. The default FC algorithm proposed by the MPOA standard [4] is found inefficient in our study. The CTM in [4] requires a specified holding time to be assigned to each cached flow entry with possible extension of an extra holding time upon request. Yet, no scheme is proposed to effectively manage such a flow cache table for the overall network performance improvement.

In this paper we propose a FC algorithm, which substantially reduces the implementation complexity while achieves the same level of performance as compared to the default one in [4]. A simple timeout mechanism is also introduced in the CTM for significant performance improvement. We further present an *adaptive* FC algorithm to achieve a near-optimal performance gain, which is simple to implement with a single control parameter.

A few works are available on the performance analysis and FC algorithm design for the flow-based IP/ATM hybrid switching systems, which are mainly based on the Ipsilon IP switching architecture [1, 2]. However, there are two basic distinctions between MPOA and Ipsilon IP switching, which makes the separate study on MPOA necessary. First, the cost and delay for shortcut setup in MPOA are generally much greater than those for cut-through connection setup in Ipsilon IP switching. This is because the MPOA's complex signalling procedure for shortcut setup and its associated two end-to-end round-trip delays. Second, per connection maintenance cost in MPOA is generally much less than that in Ipsilon IP switching. This is because MPOA maintains a "hard state" for each shortcut connection, whereas the Ipsilon IP switching maintains a "soft state" for each cut-through connection. The paper [1] placed the empha-

This work was supported by the National Science Foundation under grant ANI-9714586 and by Southwestern Bell.

[1]The other trend to resolve the bottleneck is to build faster routers based on advanced hardware design and fast algorithm development for IP address table lookups. The performance analyses of such fast router designs are beyond the scope of this paper.

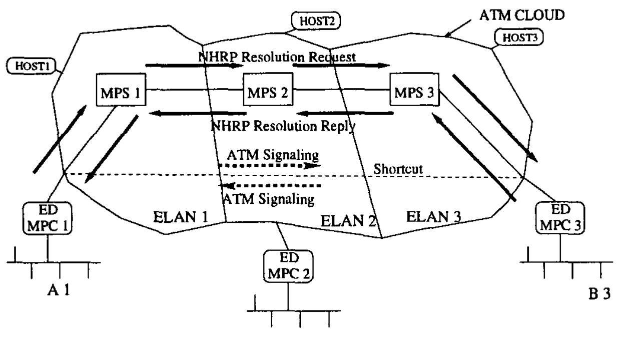

Figure 1: MPOA network and its shortcut setup procedures

sis on per connection maintenance cost under the assumptions of small processing cost and zero delay on cut-through setup. The study in [2] examined the delay effect while neglected both setup and maintenance costs for cut-through connections. In contrast, the MPOA analysis requires consideration of both long delay and high cost involved in shortcut setup while neglecting per connection maintenance cost.

The rest of the paper is organized as follows. Section 5.2 proposes an MPOA framework for performance analysis. Section 5.3 develops an adaptive FC algorithm and examines its performance. Section 5.4 is the conclusion.

2 Framework

This section is composed of three subsections. Section 2.1 formulates the process of each individual flow by a finite state machine. Section 2.2 identifies the requirement of MPOA resources and their associated costs for the process of aggregate flow dynamics. Section 2.3 then proposes a design objective to minimize the overall cost in FC algorithm design and resource management.

2.1 A Finite State Machine per Flow

As shown in Fig. 1, an MPOA network is logically divided into ELANs [5]. Each ELAN is composed of multiprotocol servers (MPSs), edge devices (EDs) and directly attached hosts. Each ED is further attached by one or more legacy shared-media LANs. Unlike the traditional inter-networking where EDs and MPSs are regular routers, MPOA employs the client-server model where EDs are clients and MPSs are servers. An MPS performs internetworking route calculation for all the associated EDs and directly attached hosts, and thus the number of devices participating in route calculation is substantially reduced, which increases the routing scalability. An important component in each ED is the so called multiprotocol client (MPC) that performs the MPOA client functions including FC, CTM and shortcut setup.

To explain how a flow is shortcut switched in Fig. 1, let us assume that an end-user A1 on an Ethernet behind MPC1 of ELAN1 wants to send a data flow to another end-user B3 on another Ethernet attached to MPC3 of ELAN3. In this case MPC1 is the ingress MPC and MPC3 is the egress MPC. Such a flow is first to be identified by the FC algorithm in MPC1. During the identification period, all its input packets will be hop-by-hop forwarded through intermediate MPSs to B3. Once it is identified as a long-lived flow, MPC1 will generate a shortcut setup request to MPC3. There are two steps for a successful shortcut setup. The first step is to resolve the MPC3's ATM address through the next hop resolution protocol (NHRP) via MPSs, i.e., the hop-by-hop forwarding path. The binding information from the layer-3 address of B3 to the MPC3's ATM address will then be cached in the MPC1's flow cache table. The second step is to establish a virtual channel connection (VCC) through the standard ATM signalling protocol via ATM network. This step may be bypassed if there already exists a VCC from MPC1 to MPC3. Notice that the management of all the VCCs at MPC1 requires a VPI/VCI table, which is separate from the flow cache table. In the worst case, a shortcut setup process requires two round trip delays, during which all the input packets of the flow are hop-by-hop forwarded.

One can describe the process of each flow by a finite state machine in Fig. 2. State *ID* represents the idle period before the first packet arrival of the flow. State *S* is the flow identification period before making a request for shortcut setup. State *SN* is the time period required to resolve the ATM address of the egress MPC. State *SCS* is the time period for ATM signalling to establish a VCC. While in states *S, SN* and *SCS*, all input packets of the flow are software forwarded. State *CT* is the only state during which all input packets are shortcut switched. Note that a flow can terminate at any time as described by the transition from any other state to state *ID* in Fig. 2. Further, the direct transition from state *SN* to state *CT* represents the situation in which the ATM signalling is bypassed. A transition probability $1 - \beta$ is assigned to describe the possibility for a flow to find an existing VCC to its egress MPC.

The actual value of β depends on VPI/VCI table management. The VPI/VCI table management again largely depends on the spatial distribution of the traffic flows. Multiple flows between each pair of ingress MPC and egress MPC may share a single VCC. Since the spatial distribution of the traffic flows cannot be extracted from the collected traces due to the address filtering for security consideration, our design will be solely based on a single node simulation without taking into account of the VPI/VCI management. Instead, we shall examine the sensitivity of MPOA performance to β by simply considering two extreme values, namely, $\beta = 0$ and $\beta = 1$.

2.2 MPOA Resource Classification

For each flow, one can use the finite state machine to characterize its present demand on individual MPOA resources according to its current state or transition. For aggregate flows, the demand on individual resources can therefore be measured by the number of flows staying in each individual state or taking each individual transition at a given time interval Δt. Defined below are the demand of aggregate flows on five major MPOA resources at the nth time interval:

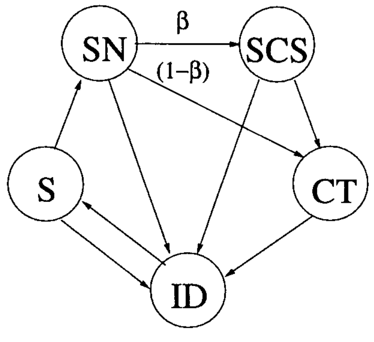

Figure 2: Finite state machine for flow state in a MPC

- $f(n)$: software-forwarding demand, measured by the number of packets generated per second by flows in states *S*, *SN* and *SCS*;
- $r(n)$: address-resolution demand, measured by the number of flows taking transition from state *S* to state *SN*;
- $a(n)$: active-address-binding demand, measured by the number of flows in states *SCS* and *CT*;
- $s(n)$: ATM-signaling demand, measured by the number of flows taking transition from state *SN* to state *SCS*;
- $c(n)$: active-shortcut demand, measured by the number of VCCs in state *CT*.

Note that $f(n)$, $r(n)$ and $s(n)$ are the rate measurement defined by the number of packets or flows per second, which are more related to the demands on CPU processing powers. $a(n)$ and $c(n)$ are the cumulative measurement defined by the number of flows, which are related to the memory requirement for table management. The capacities of the individual resources available in a given system are further represented by $F_{\max}$, $R_{\max}$, $A_{\max}$, $S_{\max}$ and $C_{\max}$, respectively. For simplicity, we assume these capacities are fixed, although our work generally applies to variable capacities.

Accordingly, $F_{\max}$ and $R_{\max}$ refer to the packet forwarding and NHRP capacities of the MPCs and MPSs along the hop-by-hop software forwarding path. $S_{\max}$ represents the signalling capacity of the MPCs and ATM switches along the shortcut switched path. These are nonlocal resources. Both $A_{\max}$ and $C_{\max}$ are referred to the flow cache table size and the VPI/VCI table size at the local ingress MPC. With these definitions, we implicitly assumed that the demands $a(n)$ and $c(n)$ at the ingress MPC are solely constrained by the corresponding local resources $A_{\max}$ and $C_{\max}$, respectively. In practice, however, the state information of each flow needs to be maintained at both ingress and egress MPCs, and a shortcut connection consumes the VPI /VCI table resource in each intermediate ATM switch along the path. Based on our single node simulation, such nonlocal resource constraints will not be considered.

Since each VCC between a given pair of ingress/egress MPCs can be shared by multiple flows, the demand on $C_{\max}$ is expected to be significantly less than that on $A_{\max}$. We then assume that resource $C_{\max}$ is unconstrained. We further assume that all the three nonlocal resources, $F_{\max}$, $R_{\max}$ and $S_{\max}$, are unconstrained. Hence, $A_{\max}$ becomes the only constrained resource. That is, a long-lived flow will be blocked from shortcut switching if and only if $A_{\max}$ is fully occupied by the existing flows, and a blocked flow due to $A_{\max}$ overflow can still be software forwarded without loss. Note that the unconstrained nonlocal resources are not for free. On the contrary, the objective of our adaptive FC algorithm design is to minimize the nonlocal resource utilizations subject to the local resource constraint.

2.3 Cost Function and Design Objective

Let us first consider the cost for maintaining and updating the two local resources at the ingress MPC. Define the cost per unit time per VPI/VCI entry in the active-shortcut demand by C_c and the cost per unit time per cached flow entry in the active-address-binding demand by C_a. The aggregate cost of the two demands at the nth Δt interval can then be represented by $a(n) \times C_a \times \Delta t + c(n) \times C_c \times \Delta t$. For the three nonlocal resources, we define C_f as the cost per packet in the software-forwarding demand, C_r as the cost per flow in the address-resolution demand, and C_s as the cost per flow in the ATM-signalling demand. The total cost on all the resources at the nth Δt interval can therefore be described by

$$C(n) = a(n)C_a\Delta t + c(n)C_c\Delta t + f(n)C_f + r(n)C_r + s(n)C_s. \tag{1}$$

Also, we need to develop a unified cost measure among the five different cost factors. For simplicity, we take $C_f = 1$, which is the cost per software-forwarding packet. That is, all the other cost factors will be measured in the units of software-forwarding packet. For instance, taking $C_r = 20$ ($C_s = 20$) means that the cost of address resolution (ATM signalling) to set up a shortcut connection per flow is equivalent to the cost of software-forwarding 20 packets. Since MPOA maintains a "hard state" for each shortcut connection, the cost of maintaining and updating both VCC and flow cache tables are comparatively negligible, i.e., $C_a \approx 0$ and $C_c \approx 0$. We then have

$$C(n) \approx f(n) + r(n)C_r + s(n)C_s \tag{2}$$

where $f(n)$ is the number of software-forwarding packets and $r(n)C_r + s(n)C_s$ is the equivalent switching overhead.

Given this cost structure, we can introduce a performance measure, called *switching gain*, G, which is defined as the ratio of the total number of software-forwarded and hardware-switched data packets to the total number of software-forwarded data packets plus the switching overhead. Accordingly,

$$G = \frac{\sum_n \{d(n) + f(n)\}}{\sum_n \{f(n) + r(n)C_r + s(n)C_s\}} \tag{3}$$

where $d(n)$ represents the number of hardware-switched packets at the nth Δt interval. Notice that for pure layer-3 forwarding, we get $d(n) = r(n) = s(n) = 0$ and so $G = 1$. If $G < 1$, a negative performance gain is achieved by the hybrid switching as compared to the pure software forwarding. Otherwise, a positive gain is achieved and the value of G provides us a quantitative measure on how many times the processing power is saved by the hybrid switching over the pure software forwarding. Our objective is to maximize G, constrained by the local flow cache table size, through the design of an adaptive FC algorithm. For the adaptive FC, the instantaneous demand on each individual resource is controllable through the adaptive assignment of a control vector $\vec{x}_n$ based on the present condition of flow dynamics and resource utilizations at time n. Denote the control sequence up to time n by $\mathcal{X}_n = \{\vec{x}_m\}|_0^n$. We have

$$\max_{\{\mathcal{X}_\infty\}} G \ . \tag{4}$$

Indeed, when such a control objective is achieved at every ingress MPC, the overall usage on the MPOA network processing resources is expected to be minimized. Note that the adaptive control is important to achieve near-optimal performance, especially given the time varying behavior of the traffic and the dynamic sharing nature of the resources.

Three Internet/intranet traces are used in the simulation study, which are refered to *cisco-trace*, *lbl-trace* and *fixwest-trace*, respectively. The *cisco-trace* is a 20-minute trace collected from a 100-BT campus network at Cisco Systems Inc. on March 4, 1997. The *lbl-trace* is a 16-minute trace collected from a 100-BT at Lawrence Burkeley Laboratory (LBL) on July 14, 1997. The *fixwest-trace* is a 20-minute trace collected from the FDDI Internet backbone at FIXWEST on Oct. 21, 1996. The utilizations at the time of data collections are 5.5%, 4.0%, and 27.3%, respectively.

3 Flow Classification Design and Performance Evaluation

The following three key MPOA design issues will be examined in this section: (a) adaptive FC algorithm design for optimal performance gain, (b) effective flow cache table management, and (c) evaluation of G under the conditions of high cost and long delay for shortcut setup.

3.1 Static FC Algorithms

A flow is identified as a sequence of packets that are treated identically by routing function. The flow identifier can be defined at various granularity levels. Examples are *host+port* granularity and *host* granularity. The former is described by a pair of {*source address, source port*} and {*destination address, destination port*}. The latter is by a pair of {*source address*} and {*destination address*}. In this work, we only consider flows at *host* granularity.

Three static FC algorithms were proposed, which are called the application -based, $X/Y/T$, and $X/Z/T$ algorithms, respectively [1, 4, 2]. The application -based algorithm is to classify application flows based on the measured average flow duration and average number of packets per flow for each application. The algorithm was found ineffective as compared to the other two algorithms [2, 3] and thus will not be considered. Both the $X/Y/T$ and $X/Z/T$ algorithms use a timeout, T, for the flow cache entry management. The X/Y/T algorithm is to request a shortcut setup if and only if X packets with the same

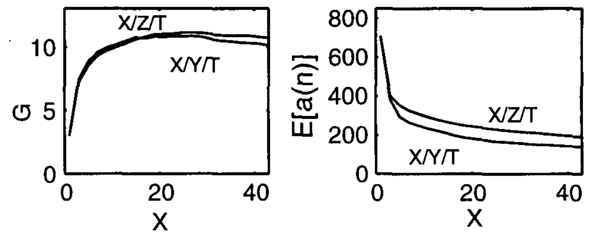

Figure 3: Performance comparison of the X/Y/T algorithm with the X/Z/T algorithm

flow identifier have been forwarded within Y seconds, which is the same as the default FC algorithm proposed in the MPOA specification [4]. The $X/Z/T$ algorithm is to request a shortcut setup if and only if X packets with the same flow identifier are software forwarded, where the interarrival time between any two successive packets are smaller than Z. The $X/Y/T$ algorithm keeps a cumulative counter of the forwarded packets and the timestamps in a fixed time window size Y per flow. Upon each packet arrival, the timestamps are updated by deleting the old ones while adding the new one, in addition to update the counter. In contrast, the $X/Z/T$ algorithm maintains a cumulative counter of the forwarded packets and a single timestamp for the last forwarded packet. Upon each packet arrival, both timestamp and counter are updated. Obviously, the $X/Z/T$ algorithm has the time/space complexity much less than that of the $X/Y/T$ algorithm, which is $O(1)$ versus $O(X)$ per flow.

Let us compare the performance of the two static algorithms. For simplicity, our simulation study assumes $C_r = C_s$. Further, the time required to resolve a destination ATM address is assumed identical to the time for ATM signalling to establish a VCC, which is denoted by T_d. Taking $C_r = C_s = 20$, $\beta = 1$, and $T_d = 200\ ms$ while fixing $Y = Z$ at 15 *sec* and T at 30 *sec*, depicted in Fig. 3 are the results of switching gain (G) and the average number of flows ($E[a(n)]$) as a function of X for *lbl-trace*, with respect to the two static algorithms. As one can see, the two algorithms achieve similar performance. In conclusion, the $X/Z/T$ algorithm should be adopted since it achieves similar performance as the $X/Y/T$ algorithm with much less complexity.

The following adaptive algorithm will be built upon the $X/Z/T$ algorithm through the adaptation of the control vector $\vec{x}_n = \{X_n, Z_n, T_n\}$ at each time interval n.

3.2 Design Complexity Reduction

The proposed adaptive control problem falls into the category of nonlinear stochastic feedback control, which is generally difficult to tackle in terms of optimal control design. To explore the possible design simplification, we first examine the sensitivity of G to individual control parameters Z, X, and T in the static sense. Setting $C_r = C_s = 20$ and $T_d = 600\ ms$, we performed multiple simulations on the basis of *lbl-trace* and *cisco-trace*, where one control parameter is selected to change at a time in a wide range. The results are summerized in Fig. 4.

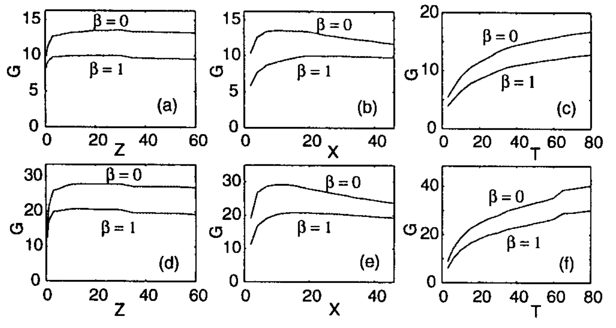

Figure 4: Sensitivity of G to Z, X, and T. Upper ones for *lbl-trace*, and lower ones for *cisco-trace*. (a),(d): $X = 20, T = 30$; (b),(e): $Z = 15, T = 30$; (c),(f): $Z = 15, X = 20$

From Figs. 4a,d, G is found insensitive to Z when Z is greater than 5*sec*. It means that most end-to-end traffic streams have their packet interarrival time less than 5*sec*, and hence further increasing Z will no longer significantly change the flow dynamics. Similarly from Figs. 4b,e, G is found insensitive to X once $X > 5$, especially when β is large. Notice that increasing X has the combination effect of reducing the number of shortcut setups and increasing the number of software forwarding packets. When X is large, the gain through the reduction of shortcut setups is virtually cancelled out by the increase of software forwarding packets. When X becomes small, the significant increase of G with X in Figs. 4b,e suggests that a large portion of flows are actually short-lived with only a few packets in each flow. From Figs. 4c,f, G is found sensitive to T and monotonically increases with T. From the given constraint in (4), the selection of T will be largely restricted by the flow cache table size,

The above static sensitivity analysis suggests that it suffices to fix Z and X at some reasonably large values, e.g., $Z_n = 15$ *sec* and $X_n = 20, \forall n$. Hence, T_n becomes the only adaptive control parameter in the adaptive FC design.

Now consider two control sequences $\mathcal{T}_n = \{T_m\}|_0^n$ and $\mathcal{T'}_n = \{T'_m\}|_0^n$. Denote $\mathcal{T'}_m > \mathcal{T}_m$ if $T'_m \geq T_m, \forall m$, with at least one inequality. We say that a function $f(\mathcal{T}_n)$ is a monotonically increasing function of $\mathcal{T}_n$ if and only if $f(\mathcal{T'}_n) > f(\mathcal{T}_n)$ for $\mathcal{T'}_n > \mathcal{T}_n$. The following monotone properties can then be identified:

□ Both $G(\mathcal{T}_n)$ and $\rho_a(\mathcal{T}_n)$ are monotonically increasing functions of $\mathcal{T}_n$.

The monotone properties are directly obtained by inspection. As $\mathcal{T}_n$ increases, a flow entry will stay longer in active and hence more packets will be relayed through the shortcut. In consequence, both shortcut setup rate and software-forwarding packet rate will be reduced, which leads to the increase of G but at the expense of increasing ρ_a.

Since ρ_a is a monotonically increasing function of $\mathcal{T}_n$, increasing ρ_a is equivalent to increase the control sequence. Further, because G is also a monotonically increasing function of $\mathcal{T}_n$, increasing control sequence is equivalent to increase G. Also since flows which are blocked due to flow cache table overflow can still be software forwarded without incuring any loss, occasional cache table overflow should not reduce G. Hence, G can be expected to be maximized at high $\rho_a(\mathcal{T}_n)$ values. In other words, the algorithm design for maximizing G can be transformed to the design of an algorithm which maximizes $\rho_a(\mathcal{T}_n)$ with zero or small flow cache table overflows. However, it is difficult to determine exactly how large the $\rho_a(\mathcal{T}_n)$ values should be such that G is maximized, which is largely determined by the flow arrival statistics which is unknown a priori. In the next section, we propose two adaptive algorithms which achieve near-optimal performance.

3.3 Adaptive FC Algorithm Design

A. LRU-based Algorithm

An algorithm which achieves near-optimal solution is the so called *least recently used* (LRU) algorithm, which ensures the maximization of $\mathcal{T}_n$ at any time n without cache table overflow. The LRU algorithm [7] was originally proposed for the page-replacement for computer virtual memory scheduling. Consider the cache table management by the LRU algorithm, Upon each shortcut setup request, a free flow entry will be allocated if it is available, otherwise the flow entry which has been idle for the longest period of time will be re-allocated to the new flow. Note that the LRU algorithm does not always give near-optimal solution because of its nonblocking nature, which may cause the so called *thrashing effect*. That is, when the cache table size is too small compared with the number of shortcut setup requests, the flow entries can be frequently added and deleted, leading to frequent shortcut setups and so the substantial reduction of G. However, since up to X packets in a flow will be software forwarded each time the flow is blocked for shortcut setup, the thrashing effect is much less significant for flow cache management than that for virtual memory management. Hence, the LRU algorithm can be expected to provide near-optimal solutions, except for very small cache table size.

A widely used technique for implementing the LRU algorithm is to keep a stack of doubly linked list of data structures, each of which is further doubly linked to the corresponding flow entry in the flow cache table. The data structure which corresponds to the least recently used flow entry is placed at the bottom of the stack whereas the data structure corresponding to the most recently used flow entry is placed at the top of the stack. Upon each packet arrival at a given flow entry, the corresponding data structure in the stack will be moved to the top of the stack with the exchange of six pointers. For *cisco-trace* and *lbl-trace*, each of them has an average bit rate around 5 *Mbps* and the average packet size of 200 bytes, equivalent to about 3,000 updates per second.

B. T-based Algorithm

An alternative approach is to periodically update T_n at n, which ensures $\bar{\rho}_a(\mathcal{T}_\infty) < 1$ but close to 1. It is directly achieved from the monotone property. First, in order to avoid over-reaction to small demand variations, we introduce a first-order low-pass filter operation to damp the variation in $\rho_a(\mathcal{T}_n)$, i.e.,

$$\hat{\rho}_a(\mathcal{T}_n) = (1-\omega)\hat{\rho}_a(\mathcal{T}_{n-1}) + \omega\rho_a(\mathcal{T}_n) \tag{5}$$

where ω is the weighting factor taken values between 0 and 1. One can strengthen the damping by choosing a small ω. Since $\rho_a(\mathcal{T}_n)$ is a monotonically increasing function of $\mathcal{T}_n$, the following simple control scheme can be applied,

$$\mathcal{T}_n = \begin{cases} \mathcal{T}_{n-1} + \Delta T_1, & \text{if } \hat{\rho}_a(\mathcal{T}_{n-1}) \le \rho_{min}; \\ \max\{\mathcal{T}_{n-1} - \Delta T_2, T_{min}\} & \text{if } \hat{\rho}_a(\mathcal{T}_{n-1}) \ge \rho_{max}; \end{cases} \tag{6}$$

with $0 < \rho_{min} < \rho_{max} < 1$. Here, T_{min} serves as the lower bound to avoid the thrashing effect. We call this adaptive FC algorithm the *T-based algorithm.*

In our simulation study, the update interval Δt is fixed at 2 *sec*. After each update, every active flow entry is checked to see if it should be deleted by comparing its idle time with the present T_n. It requires one subtraction and one comparison operation per active flow entry. Unlike the LRU-based algorithm, no separate data structure needs to be built for the T-based algorithm. For both *cisco-trace* and *lbl-trace*, the maximum number of concurrently active flows is about 600. So up to 600 flow entries need to be checked at every Δt interval, i.e., up to 300 flow entry checks per second. Thus, the complexity of the T-based algorithm is at least one order of magnitude less than that of the LRU-based algorithm.

Since ρ_a is a monotonic function of $\mathcal{T}_n$, the proposed T-based algorithm can guarantee the stability of the adaptive control given the proper selection of the control parameters $\{\Delta T_1, \Delta T_2, \rho_{max}, \rho_{min}\}$. The explicit control of T_n also provides us a direct prediction of the desired holding time for each flow entry in the cache table, which can be used to request extra holding time for the long-lasting active flow entries as defined in the MPOA standard. Since the LRU-based algorithm is expected to offer near-optimal performance with reasonably large cache table size, it can be used as a benchmark for the performance analysis of the T-based algorithm.

3.4 Performance analysis

We first examine the MPOA performance by running both the T-based and LRU-based algorithms with *lbl-trace* and *cisco - trace*. Choosing $\omega = 0.5$, $\Delta T_1 = 2$, $\Delta T_2 = 5$, $\Delta t = 2$ *sec*, $C_r = C_s = 20$, $\beta = 0$, and $T_d = 200$ *ms*, depicted in Fig. 5 are the performance of G as a function of the cache table size for both algorithms. For the T-based algorithm, two different parameter settings are selected as indicated in Fig. 5.

When the cache table size is sufficiently large (about 600 entries), both algorithms can achieve $G = 22$ and 60 with respect to *lbl-trace* and *cisco-trace*, respectively. Although the LRU-based algorithm out-performs the T-based algorithm for most of the cache table sizes, the difference between the two is small. It implies that the T-based algorithm also offers near-optimal performance. We note that G is not very sensitive to the changes of ρ_{max} from 0.95 to 0.98 and ρ_{min} from 0.8 to 0.95, which implies that the selection of the thresholds is not critical for the T-based algorithm to achieve near-optimal control. When the

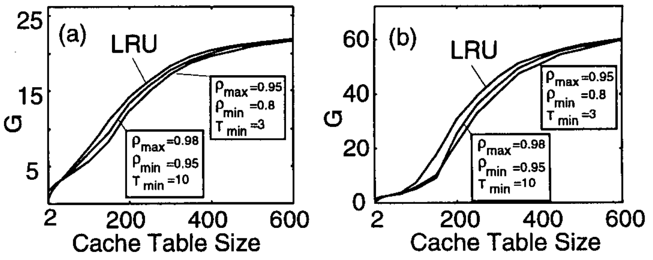

Figure 5: Switching gain G for the two adaptive FC schemes: (a) *lbl-trace* and (b) *cisco-trace*.

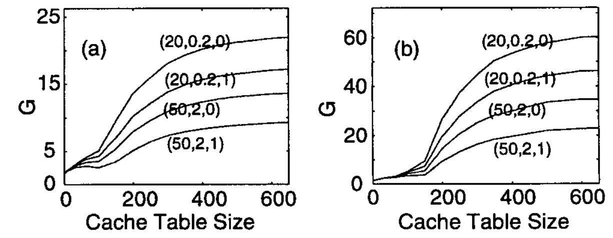

Figure 6: Switching gain G for the T-based algorithm with different assignments of (C_r, T_d, β): (a) *lbl-trace* and (b) *cisco-trace*.

cache table size becomes extremely small, however, the LRU-based algorithm led to the negative performance gain ($G < 1$) because of the thrashing effect, whereas the T-based algorithm out-performs the LRU-based algorithm. Further, the two algorithms converge to identical performance as the cache table size increases. This is because T_n can be arbitrarily large for both algorithms when the cache table size becomes unconstrained.

Next, we examine the performance in both best and worst case scenarios using the T-based algorithm. Consider both C_r and C_s in the range of [20, 50]. That is, the resource requirement per address resolution and also per ATM singalling is equivalent to that of 20 to 50 software-forwarding packets, where 50 is expected to be the worst case scenario. Further take T_d in the range of $[0.2, 2]$ *sec*, where 2 *sec* is expected to be the worst case scenario. For instance, in an inter ELAN communication environment with 3 to 6 hops, the typical delay for an SVC setup is about 0.1 $\sim$ 0.6 *sec* [8]. Hence, the best case scenario is represented by $(C_r, T_d, \beta) = (20, 0.2, 0)$ while the worst case scenario is by $(C_r, T_d, \beta) = (50, 2, 1)$ given $C_s = C_r$. The following parameters are set for the simulation: $\omega = 0.5$, $\Delta Z_1 = 2$ *sec*, $\Delta T_2 = 5$ *sec*, $T_{min} = 3$ *sec*, $\rho_{max} = 0.98$, $\rho_{min} = 0.95$, and $\Delta t = 2$ *sec*.

The simulation results are presented in Fig. 6 with different assignments of (C_r, T_d, β). When the cache table size becomes sufficiently large, the switching gain G can reach 22 and 60 for the best case scenario, and 9 and 22 for the worst case scenario, with respect to *lbl-trace* and *cisco-trace*, respectively. That is, MPOA can achieve significant switching gain in an inter ELAN environment. Notice that the performance degradation by changing β from 0 to 1 is less than 40% at each given

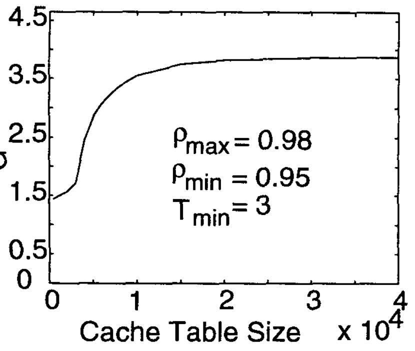

Figure 7: Switching gain G v.s. flow cache table size for *fixwest-trace*

(C_r, T_d). Similarly, the performance degradation by changing (C_r, T_d) from the best $(20, 0.2)$ to the worst $(50, 2)$ is less than 60% at each given β. When the cache table size is significantly small, we observe some reduction of G with the increase of the table size. This is due to the selection of $T_{min} = 3$ *sec*, which is too small for the worst case scenario to guard against the thrashing effect. Such behavior is expected to disappear when T_{min} is set at a relatively larger value.

In the above T-based algorithm, we set X at 20. To study the sensitivity of the performance to X, we further used the LRU-based algorithm to find an optimal X for the maximum switching gain using *lbl-trace*, at each given table size of 200, 300 and 400 respectively. Our comparison study indicates that such maximum performance achieved by the LRU-based algorithm using the optimal X is within the 5% difference from the performance achieved by the T-based algorithm using the fixed $X = 20$ at each given table size. In other words, not only the performance is insensitive to X given the range of X is properly selected, but also the T-based algorithm indeed provides the near-optimal performance. The same T-based algorithm has also been tested against the LRU-based algorithm on the basis of other 5 traces collected at two different 100-BT campus networks at Cisco Systems Inc. and all the results are consistent.

So far the studies are based on the intranet traces. We now examine the performance of the T-based algorithm using a backbone Internet trace, *fixwest-trace*. Fix (C_r, T_d, β) at $(20, 0.2, 0.5)$ with $C_s = C_r$ while the rest parameters remain unchanged. As one can see in Fig. 7, G saturates at about 3.8 even as the cache table size approaches 40,000. Obviously, MPOA achieves much less switching gain in the backbone Internet environment as compared to that in the campus intranet environment. Given the 27.3% utilization of *fixwest-trace* on a 100 *Mbps* FDDI ring, its cache table demand is also significant. For instance, the corresponding cache table demand on an OC-3 link in backbone networks may well exceed 100,000.

Clearly, it must be the difference between the flow dynamics of campus intranet and backbone Internet that has the major impact on the MPOA performance. Here we further decompose the flow dynamics into different applications at the *host+port* granularity. Listed in Table 1 are the detailed average statistics of such flow dynamics for the three traces, given the flow timeout value $T = 128$ *sec*. For clarity, only the statistics of major applications are collected. For *cisco-trace*, the major applications are represented by those whose flow ratio is greater than 1% or packet ratio is greater than 8%. For the other two traces, they are represented by those whose flow ratio is greater than 5% or packet ratio is greater than 4%. For the two campus traces, let us first focus on the unknown applications. Although these applications comprise only a small fraction of the total flows, they actually represent the major portion of the total packets. In other words, the flows of these applications consist of a large number of packets, which are best suited for shortcut switching. They are the major contributors to the large switching gain achieved by MPOA for the campus traces. Another major contributor is the well-known *tcp* X-window application.

Table 1: Average flow statistics

trace	(prot,port)	flow%	pkt%	byte%	pktpf
cisco	(tcp, 4240=unknown)	0.0	36.1	10.4	556773.
	(tcp, 6000-6063=Xwin)	0.3	8.8	3.6	666.6
	(udp, 111=sunrpc)	2.3	0.1	0.0	1.2
	(udp, 1020=unknown)	0.4	8.7	9.6	542.7
	(udp, 8224=unknown)	2.1	0.4	1.9	4.5
lbl	(tcp, 80=http)	5.5	3.0	2.7	11.2
	(tcp, 514=cmd)	0.0	4.2	8.6	2392.7
	(tcp, 1023=rsv)	0.0	4.8	3.5	43673.0
	(tcp, 6000-6063=Xwin)	0.1	4.3	5.1	978.5
	(udp, 53=dns)	5.3	1.7	0.6	6.3
	(udp, 1021=unknown)	0.2	41.6	27.8	5270.5
fixwest	(tcp,20=ftp-data)	0.5	7.7	1.0	213.0
	(tcp, 80=http)	33.6	36.2	43.2	13.5
	(udp, 53=dns)	40.1	15.3	7.0	4.8
	(ipip or MBONE)	2.2	18.1	18.0	102.4

Notice that both *http* and *dns* applications appeared in *lbl-trace* and *fixwest-trace*, but not in *cisco-trace*. It is because the former two traces were collected from educational institutes while the latter one is from a vendor. In contrast, *http*, *dns* and *ipip* in *fixwest-trace* are the three applications which contribute to the major portion of the total packets. While *ipip* is suited for shortcut switching, most flows in *http* and *dns* consists of a small number of packets. Yet, both *http* and *dns* contribute 74% of the total flows. This is why only a marginal switching gain is achievable by MPOA for *fixwest-trace*. Similarly, the reason for *lbl-trace* to achieve significantly less switching gain than that of *cisco-trace* is the non-negligible impact of *http* and *dns* applications.

In summary, both *http* and *dns* applications in the backbone Internet environment are the major factor to the ineffectiveness of the data-driven MPOA technology. In contrast, since data file transfers are the major applications in the campus intranet environment, a significant performance gain can be achieved by the MPOA technology.

4 Conclusion

In this paper, we developed a framework for the performance analysis and flow classification design of MPOA network. Our study indicates that MPOA can offer significant performance gain over the traditional routed network in an inter ELAN communication environment. A timeout mechanism was introduced to flow cache table management for performance improvement with reduced complexity. An effective adaptive flow classification algorithm was proposed to achieve a stable and near-optimal control for the maximum performance gain on the constrained MPOA resources. As we have seen, application composition in traffic has great impact on MPOA performance. As more and more multimedia services, such as $MBONE$, being deployed and more features being added in *http* application, such as $JPEG$ pictures and $MPEG$ videos, one can expect that the MPOA performance will be improved in an Internet backbone environment.

Acknowledgement

The authors would like to acknowledge NLANR for the FIXWEST trace and Cisco Systems Inc. for the campus traces.

References

[1] P. Newman, G. Minshall, and Tom Lyon, "IP Switching: ATM Under IP," *IEEE/ACM Trans. Networking*, Vol. 6, No. 2, April 1998, pp. 117-129.

[2] S. Lin, and N. Mckeown, "A Simulation Study of IP Switching," *Proceedings ACM SIGCOMM'97, France, Sept. 1997.*

[3] H. Che, S. Q. Li, and A. Lin, "Adaptive Resource Management for Flow-Based Hybrid Switching Systems," *IEEE/ACM Trans. Networking*, Vol.6, No. 5, Oct. 1998, pp. 544-557.

[4] "Multi-Protocol Over ATM Version 1.0," The ATM Forum Technical Committee, July, 1997.

[5] "LAN Emulation over ATM Version 2 – LUNI Specification," *The ATM Forum*, 1997.

[6] D. Katz, D. Piscitello, B. Cole, J. Luciani, "NBMA Next Hop Resolution Protocol (NHRP)," *IETF Internet Draft*, draft-ietf-rolc-nhrp-12.txt, Oct., 1997.

[7] S. Galvin, "Operating System Concepts," *Addison-Wesley Publishing Company, Inc., 1994.*

[8] D. Niehaus, A. Battou, A. McFarland, B. Decina, H. Dardy, V. Sirkay, B. Edwards, "Performance Benchmarking of Signaling in ATM Networks," *IEEE Communications Magazine*, Vol. 35, No. 8, Aug., 1997.

[9] "Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)," *The ATM Forum*, March 1996.