Adaptive Resource Management for Flow-Based IP/ATM Hybrid Switching Systems

Hao Che, *Member, IEEE*, San-qi Li, *Member, IEEE*, and Arthur Lin

***Abstract*— This paper addresses a fundamental problem in resource management for flow-based hybrid switching systems. Such systems aim at efficient transport of layer-3 connectionless IP traffic over layer-2 connection-oriented ATM switching fabrics. One idea behind flow-based hybrid switching is to decompose individual IP packet streams into flows and then to classify them into short-lived and long-lived flows. While the short-lived flows are good for forwarding by the embedded software through permanent virtual connections (PVC's), the long-lived flows are more effectively transmitted by hardware through switched virtual connections (SVC's). Clearly the flow identification/classification mechanism will have great impact on the utilization of the system's resources. Unlike the traditional emphasis on resources such as link bandwidth and cell buffer size, our paper focuses on the resources which are directly associated with packet processing power, signaling capacity, and flow cache table size. Our study indicates that the presently available *static flow classification* methods have a vital shortcoming in balancing the utilization of the system's resources. We propose a novel approach for *adaptive flow classification* based on the min–max objective for the system resource utilizations to match with the time-varying traffic/resource characteristics. Based on the monotone properties and sensitivity analysis of the resource utilizations as functions of the control parameters, we first prove that the optimal solution of the static min–max problem is achieved at a unique balance point for the resource utilizations. With the intuition gained from the static results, we then design an adaptive controller formulated as a hierarchical stochastic automata control system with local search. The optimality of the proposed adaptive controller is tested against the static optimal control based on real trace simulations. The simulation studies in highly nonstationary environments show the viability of the proposed flow adaptation for dynamic resource management in hybrid switching system design. The algorithm is simple to implement and only requires the adaptation of two global variables at time intervals of every few seconds based on the present usage of resources.**

***Index Terms*— Adaptive resource management, cut-through switching, flow-based IP/ATM hybrid switching, flow cache management, flow classification.**

Manuscript received June 6, 1997; revised September 30, 1997 and May 19, 1998; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Rom. This work was supported by the National Science Foundation under Grant ANI-9714586 and by Southwestern Bell.

H. Che was with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA. He is now with the Department of Electrical Engineering, the Pennsylvania State University, State College, PA 16802-2705 USA (e-mail: hxc30@psu.edu).

S. Li is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: sanqi@globe.ece.utexas.edu).

A. Lin is with Shasta Networks, Inc., Sunnyvale, CA 94089-1300 (e-mail: alin@shastanets.com).

Publisher Item Identifier S 1063-6692(98)08139-4.

I. INTRODUCTION

IP TRAFFIC is becoming the killer application for asynchronous transfer mode (ATM) networks. One challenge in current network research is how to effectively transport internetworking protocol (IP) traffic over ATM networks. IP was independently developed on the basis of a connectionless model, while ATM was originally designed for connection-oriented services. IP traffic is usually switched using packet software-forwarding technology, which is expensive and has substantially limited forwarding capacity. In contrast, ATM switches are designed with high transmission bandwidth, but having limited connection setup processing capacity due to its complex signalling structure developed for connection-oriented services.

The RFC 1932 [1] at IETF indicates the trend of IP over ATM toward a hybrid approach to support both layer-3 IP software forwarding and layer-2 ATM hardware switching. It is understood that short-lived flows composed of a few packets are well suited for hop-by-hop layer-3 software-forwarding while long-lived flows containing a large number of packets are good for layer-2 hardware switching. Multiprotocol over ATM (MPOA) is an example of hybrid switching which supports both the default forwarding across subnetwork borders via MPOA servers (MPS's) and cut-through switching to bypass MPS's. In the recent release of MPOA document at the ATM forum [2], flow identification/classification is considered to be the key component for cut-through switching at an ingress MPOA client (MPC). Other hybrid switching approaches have been proposed under such names as IP switching, cell switched router (CSR), tag switching, and aggregate route-based IP switching (ARIS) [3]–[6], where layer-3 routing and label binding/swapping are used as a substitute for layer-2 ATM routing and signalling for ATM hardware switching connection setup. The technology is generally called *multiprotocol label switching* (MPLS) [7], which can be further classified into the data-driven approach [3], [5] and the topology driven approach [4], [6]. The data-driven approach is flow-based, which is to set up a layer-2 connection on the first few packet arrivals of each flow. A flow identification/classification algorithm needs to be implemented on-line to decide whether a cut-through connection is to be set up for each incoming flow. Hence, one key issue in the flow-based hybrid switch design is the on-line identification/classification of each flow into long-lived or short-lived flow upon its first few arriving packets.

Most ATM system analyses so far have focused on cell transmission bandwidth allocation and cell buffer dimension-

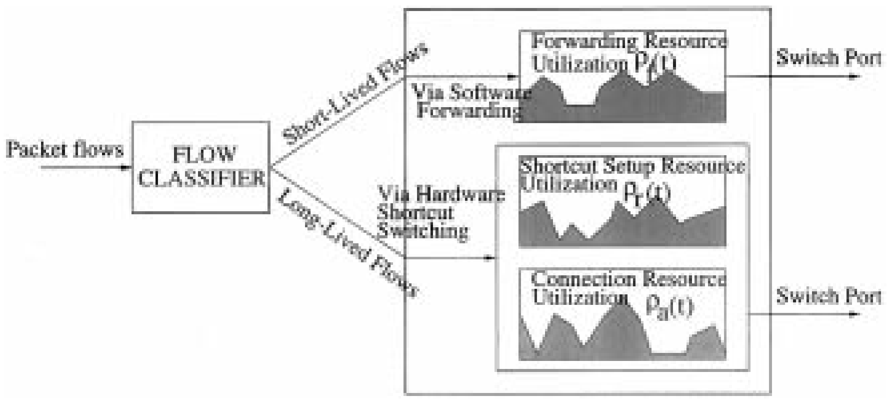
Packet flows
FLOW CLASSIFIER
Short-Lived Flows
Long-Lived Flows
Via Software Forwarding
Via Hardware Shortcut Switching
Forwarding Resource Utilization $\rho_f(t)$
Shortcut Setup Resource Utilization $\rho_r(t)$
Connection Resource Utilization $\rho_a(t)$
Switch Port
Switch Port

Fig. 1. Schematic flow diagram for a flow-based hybrid switching system.

ing/scheduling for user information transfer. Breaking down the cost of any ATM switching system, it is not difficult to recognize that a significant portion of the system cost will be for other resources directly associated with packet processing power, signalling capacity, and flow cache table size. Let us define these major resources: 1) *software-forwarding capacity*, i.e., the available rate to forward packets by embedded software; 2) *connection-setup capacity*, i.e., the available rate to set up local hardware switched connections for transport of long-lived flows; and 3) *active-connection capacity*, i.e., the number of cache table entries available for maintaining the states of all active flows.

Fig. 1 shows a logical diagram for a flow-based hybrid switching system. Incoming packets are identified and classified into short-lived and long-lived flows. The short-lived flows are sent to layer 3 for software-forwarding, consuming the software-forwarding resource. The long-lived flows are cut-through switched via a layer-2 connection, taking both connection-setup and active-connection resources. Obviously, an inadequate allocation of any of these resources can cause substantial damage to network performance.

Note that the three resources may not reside in the same physical system. For example, MPOA flow identification/classification takes place at ingress MPC's. Packets of a short-lived flow are forwarded via a default forwarding path (PVC) to its default MPOA server (MPS). It is then the MPS's responsibility to find the next hop for further forwarding. Hence, its software-forwarding resource actually refers to the processing capacity at the default MPS, which is shared by multiple peer MPC's. Also, for an IP switching system such as an Ipsilon IP switch or CSR, a cut-through switched connection is to be set up between two adjacent switches and the state information for the connection need to be cached in the table at both switches. So the flow cache table resource is nonlocal and dependent on the cache table occupancy at both switches.

One can further break down the resource allocation within a switch. A typical mid-sized switch is configured by a central control card and multiple trunk cards. If the switch is designed with distributed processing structure, all three resources can be located at each individual trunk card. Otherwise, they will be located at the central control card and shared by all the trunk cards. Moreover, these resources are not dedicated to nonreservation-based IP services, but more likely to be shared with other reservation-based services. The reservation-based services especially have higher priority to use the resources over the nonreservation-based IP services. Hence, not only the IP traffic characteristics but also the resource availabilities themselves are unknown *a priori* and change with time.

In addition, the overall resource availability at each individual switching system is based on its own design strategies, which vary depending on time, vendors, technologies, and applications. Especially in the product design stage, it is difficult to predict future applications and their traffic characteristics, of which variation can drastically change the demand for individual resources. In consequence, the utilization of the major resources at a switching system is likely to be highly unbalanced and time-varying. A system which may have sufficient resources for today's demand may soon become partially overloaded for tomorrow's demand due to its highly unbalanced operation and changing traffic/resource characteristics.

To gain a better understanding of the resource demands, let us first study the flow statistics for today's Internet/intranet traffic. Three traces are used for this study, each of which is 20 min long and collected at a different site. One is from a 100-Mb/s fast Ethernet on campus at the Cisco Systems Inc., collected in June 1997. One is from the 100-Mb/s fast Ethernet backbone at Lawrence Berkeley Laboratory in July 1997, and the other is from a 100-Mb/s Internet backbone FDDI ring at the FIXWEST on Sept. 26, 1996, downloaded from the anonymous FTP site at *www.nlanr.net/Traces/FR+/960926/*. For the convenience of later reference, we call them *cisco-trace*, *lbl-trace*, and *fixwest-trace*, respectively. The utilizations for *cisco-trace* and *lbl-trace* at the time of collection were about 5%. The utilization for *fixwest-trace* at the time of collection was about 27%. For consideration of the corresponding resource demand on a high-speed link, we linearly upscale the average resource demand required for each above trace as if running at 100% utilization on a 100-Mb/s link. Our statistical analysis then shows that the average flow arrival rate is on the order of 1–10 K per second and the average number of active flows on the order of 10–100 K, given that each individual flow is identified by a distinct pair of $\{$*source-address*, *source-port*$\}$ and $\{$*destination-address*, *destination-port*$\}$. For flows to be identified with a coarser granularity such as by each pair of *source-address* and *destination-address*, our statistics based on the same traces with the upscaling show that the flow arrival rate is on the order of 1 K per second and the number of active flows on the order of 10 K.

For some existing ATM products, the switched connection setup capacity is in the range of several hundred per second for the entire switch, which is far from sufficient for IP applications. Their active-connection capacity, for example, measured by the maximum number of VPC's and VCC's supportable on an OC-3 trunk port, is in the range of 4 K, which is also inadequate to support active flows. Since a large portion of the IP traffic flows are short-lived, flow-based hybrid switching technologies hold the promise of effective transport of IP traffic over ATM cloud. On the one hand, sending short-lived flows via layer-3 software-forwarding can greatly reduce the number of cut-through switched flows, saving the limited resources on switched connection setup and VPC/VCC space. On the other hand, sending long-lived flows through layer-2 hardware cut-through switching can substantially alleviate the burden on layer-3 software-forwarding. The effectiveness of a flow-based hybrid switching system largely relies on the design of a flow identification/classification algorithm to balance the short-lived and long-lived flows.

So far, the proposed flow identification/classification algorithms are static and the selection of the control parameters is solely based on some empirical real trace simulations. A chosen static flow identification/classification may work well in one system and/or for one application at one time, but is likely to fail to apply at other times, to other systems, and/or for different applications. Here we propose a novel approach, i.e., *adaptive* flow identification/classification, based on a min–max objective, which is to ensure any given switching system to operate in a mode such that the maximum of its three major resource utilizations is minimized in a highly variable traffic environment with time-varying resource availabilities. There are two major contributions in this paper. First, we show that the static version of the min–max problem can be reduced to a static balance problem with the existence of a unique balance point, at which the global optimal solution is achieved. This result has important implication in the sense that, to find the static optimal operation point, it suffices to design a controller with local search of a unique balance point. This result also provides tremendous intuitions on how to design an adaptive controller in terms of the min–max objective. Second, we design an adaptive controller which is formulated as a hierarchical automata control system with local search. The optimality of the proposed controller is tested against the static optimal control based on the real Internet/intranet trace simulations. The simulation study also shows the viability of the proposed algorithm for dynamic resource management in hybrid switching systems, as well as its robustness to the time variation of traffic characteristics and system resources. The algorithm only requires the adaptation of two global variables at time intervals of every few seconds based on the present usage of the three resources.

The remaining part of this paper is organized as follows. Section II reveals the critical restrictions of static flow identification/classification on resource management. Section III proposes the concept of adaptive flow identification/classification and formulates the problem in terms of a min–max objective. In Section IV we prove that the optimal solution of the static min–max problem is achieved at a unique balance point among the three resource utilizations. Section V designs a hierarchical stochastic automata control system based on the general min–max design objective. The simulation study of the proposed algorithm is given in Section VI, while Section VII provides the conclusion and the directions of future work.

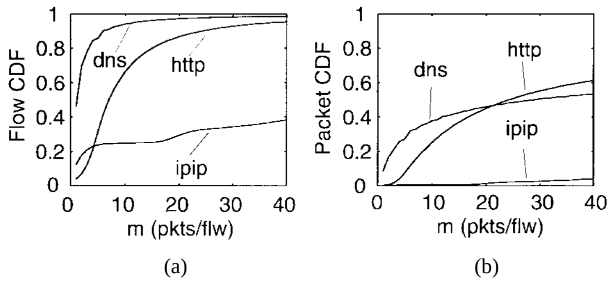

Fig. 2. Flow statistics at $T = 64$ s. (a) CDF of flows with less than m packets and (b) CDF of packets for flows less than m packets.

II. Static Flow Identification/Classification

A flow is identified as a sequence of packets that are treated identically by the routing function [3]. In this paper, every packet in a flow is identified by its flow identifier with timeout T. Adjacent packets with an identical flow identifier belong to the same flow as long as their interarrival time is less than T. In other words, a flow terminates when time T elapses without receiving a new packet with the same identifier. The flow identifier can be defined at various granularities. In this paper, we only consider the granularity of the flow identifier defined by a pair of $\{$*source-address*, *source-port*$\}$, and $\{$*destination-address*, *destination-port*$\}\}$. Namely, all the packets in a flow must have this same flow identifier which are found in their IP headers. For simplicity, we call flows at this granularity the *host+port* flows. Classifying flows at this granularity allows for QoS differentiations of various applications.

One classification approach proposed in [3], [5] is to classify *host+port* flows into long-lived and short-lived flows by their applications, such as *ftp*, *http*, or *ipip*, which are directly identifiable from each packet's port ID's. It is purely dependent on the longterm statistical measurement of the average flow duration and the average number of packets per flow, with respect to each application. An application flow with its average flow duration and average number of packets per flow smaller than the respective predefined thresholds is classified as short-lived, otherwise, it is classified as long-lived. So, by looking at the port ID of the first packet of a flow, a decision can be made for whether to set up a cut-through connection for the flow.

Our analysis of the backbone trace, *fixwest-trace*, however, shows a great variation of flow duration and packet number per flow from one flow to another for some major application such as *http* or *dns*. This renders application-based flow classification inefficient. Fig. 2(a) shows the cumulative distribution of the number of packets per flow identified at a given timeout $T = 64$ s, with respect to each of the three dominant applications *http, dns,* and *ipip*. (The other applications represent less than 28% of the total traffic in packets). It is clear that most *http* and *dns* flows contain a small number of packets. For instance, there are 87% *http* flows and 97% *dns* flows with less than 20 packets. Yet these flows only generate 45% of the total *http* packets and 46% of the total *dns* packets, as can be seen in Fig. 2(b) for the packet percentage of *http* and *dns* flows with less than or equal to m packets per flow. In other words, the remaining 13% of *http* flows and 3% of *dns* flows will generate 55% of the total *http* packets and 54% of the total *dns* packets. *Http* and *dns* flows are short-lived on average compared with *ipip* flow, as can be seen from Fig. 2. If both *http* and *dns* are classified as short-lived flows while all the other applications are assumed to be hardware switched, about 50% of the overall packets in the trace will have to be software-forwarded, which can easily exceed the software-forwarding capacity at a switch. In contrast, when both are classified as long-lived flows, they alone will require the capacity of holding 50 000 active connections and handling 600 connection setups per second on average, which is excessive for an average of 27-Mb/s traffic load. It is obvious that the application-based flow classification will result in highly unbalanced usage of the system resources.

Another classification approach recently proposed in [8] is to introduce a counter to every flow such that the first X packet arrivals in each flow are always software-forwarded and the remaining packets will be hardware-switched, independent of applications. Since both timeout T and counter value X are statically assigned, we call it the static X/T algorithm. The preliminary study in [8] examined the effect of X on system resources at the host to host granularity for a given $T = 60$ s. They found that the static X/T algorithm effectively separates short-lived flows from long-lived flows as compared to the application-based algorithm. Note that the active-connection resource taken by each individual long-lived flow is released upon its termination (i.e., once the packet interarrival time exceeds T). Our investigation on the joint effect of X and T using *fixwest-trace* indicates that the average demand on each individual resource can be significantly affected by the selection of X and T.

In the MPOA document [2], a default algorithm is proposed. A flow is classified to be long-lived if there are more than X packet arrivals within a given time interval Y. Each active flow entry in the cache table is assigned a holding time T and it is deleted when the holding time expires. A similar application-independent algorithm, where X, Y are used for flow identification and T for flow cache entry management, was recently examined in [9] for the Ipsilon IP switching system. Again, the algorithm is found to give superior performance over the application-based algorithm.

Up until now, all of the algorithms have been static, i.e., their control parameters such as (X, T) or (X, Y, T) are statically assigned. The studies in [8], [9] focused on the selection of such static values based on some available IP traffic statistics, without considering the constraint on individual resource availabilities. In this paper, we propose an adaptive flow identification/classification. That is, the control variables are dynamically adjusted according to both the time-varying IP traffic characteristics and the present constraint on individual resource availabilities, such that the maximum utilization of the three constrained resources will be minimized. Our analysis will be built upon a so-called $X/Z/T$ algorithm. Here, Z is used to timeout a flow before it is cut-through switched, T is to timeout a cached flow entry when the flow is cut-through switched, and X is the number of arriving packets detected before cut-through switching. Hence, the X/T algorithm is a special case of the $X/Z/T$ algorithm with $Z = T$. The design principle can be generally applied to other algorithms such as the above (X, Y, T) model. Note that since our study focuses on the fundamental issues in flow adaptation algorithm development, we are not concerned with the implementation details introduced due to the nonlocal nature of the resources, as mentioned in the introductory section.

III. Adaptive Flow Identification/Classification

The proposed adaptive flow identification/classification algorithm is based on the static $X/Z/T$ algorithm, subject to a periodic adaptation of the control parameters (X, T) to the traffic/resource change. For control simplicity, we fix Z at 15 s, a value which we found to be statistically large enough to relay most of the packets in a flow.

The resource demand at discrete time n are denoted by $S(n)$, $C(n)$, and $A(n)$ for software-forwarding, connection-setup, and active-connection, respectively. Without loss of generality, we assume fixed resource capacities for a given system, denoted by $S_{\max}$, $C_{\max}$, and $A_{\max}$, respectively. Several possible cost functions can be selected for the adaptive control optimization. Here we choose to minimize the maximum of the three resource utilizations. In other words, our objective is to avoid the overloading of individual resource(s). Hence, a natural choice of the metric to be measured is the instantaneous utilization of the three resources, expressed by

$$\rho_1(n) = \frac{S(n)}{S_{\max}}, \quad \rho_2(n) = \frac{C(n)}{C_{\max}}, \quad \rho_3(n) = \frac{A(n)}{A_{\max}}. \qquad (1)$$

As in the design of any feedback control systems, it is highly undesirable to have the controll system adapt to high-frequency disturbance, which otherwise may drive the control system unstable. To avoid such overreactions to small demand variations, we use a first-order low-pass filter operation to damp the variation in $\rho_i(n)$, i.e.,

$$\hat{\rho}_i(n) = (1 - \omega_i)\hat{\rho}_i(n-1) + \omega_i \rho_i(n), \quad \text{for } i = 1, 2, 3 \qquad (2)$$

where ω_i is the weighting factor taken between 0 and 1. One can strengthen the damping by choosing a small ω_i. It is equivalent to taking the moving average operation.

Based on the stochastic control framework, the min–max control problem can be expressed as

$$\text{minimize}_{\{\mathcal{T}_\infty, \mathcal{X}_\infty\}} J \qquad (3)$$

with

$$J(\mathcal{T}_\infty, \mathcal{X}_\infty) = \lim_{N \to \infty} E_{\mathcal{T}_N, \mathcal{X}_N} \left[\frac{1}{N} \sum_{n=1}^{N} \max_i \{\hat{\rho}_i(n)\} \right]$$

TABLE I
THE UTILIZATION MONOTONE PROPERTY WITH RESPECT TO $\mathcal{T}_n$ AND $\mathcal{X}_n$

	$\rho_1(\mathcal{T}_n, \mathcal{X}_n)$	$\rho_2(\mathcal{T}_n, \mathcal{X}_n)$	$\rho_3(\mathcal{T}_n, \mathcal{X}_n)$
$\mathcal{T}_n \nearrow$	$\searrow$	$\searrow$	$\nearrow$
$\mathcal{X}_n \nearrow$	$\nearrow$	$\searrow$	$\searrow$

where J is the cost function, and $\mathcal{T}_n = \{T_m\}|_0^n$ and $\mathcal{X}_n = \{X_m\}|_0^n$ are the time sequences of the two control variables. Note that $\frac{1}{N}\sum_{n=1}^N[\cdot]$ takes the time average operation of the maximum of the three utilizations, whose expectation, represented by $E_{\mathcal{T}_N, \mathcal{X}_N}[\cdot]$, is to be minimized as time goes to infinity.

The following monotone property can be identified.

Property 1: All three resource utilizations, $\rho_1(\mathcal{T}_n, \mathcal{X}_n)$, $\rho_2(\mathcal{T}_n, \mathcal{X}_n)$, and $\rho_3(\mathcal{T}_n, \mathcal{X}_n)$, are monotonic functions of $\mathcal{T}_n$ and $\mathcal{X}_n$, described in Table I.

Consider two control sequences $\mathcal{T}_n = \{T_m\}|_0^n$ and $\mathcal{T}'_n = \{T'_m\}|_0^n$. Denote $\mathcal{T}'_n > \mathcal{T}_n$ if $T'_m \geq T_m$, $\forall m(\leq n)$, with at least one inequality. We say that $\rho_i(\mathcal{T}_n, \mathcal{X}_n)$ is a monotonically increasing function of $\mathcal{T}_n$ if and only if $\rho_i(\mathcal{T}'_n, \mathcal{X}_n) > \rho_i(\mathcal{T}_n, \mathcal{X}_n)$, for $\mathcal{T}'_n > \mathcal{T}_n$, $\forall n$ and vice versa. A similar definition applies to $\mathcal{X}_n$.

Property 1 can be understood easily from the physical meaning of each individual resource. For instance, let us examine the first row in Table I. Increasing $\mathcal{T}_n$ has the effect of merging multiple flows with the same flow identifier into a single flow, as more adjacent flow interarrival times become less than $\mathcal{T}_n$ when $\mathcal{T}_n$ increases. In consequence, the flow arrival rate will be reduced, which results in the decrease of ρ_2, assuming that no other system conditions are changed. By the same token, the software-forwarding capacity requirement, measured by ρ_1, will also be reduced. On the other hand, since the merged flows tend to be longer and require more connection holding time, the active-connection capacity requirement, ρ_3, is expected to increase with $\mathcal{T}_n$. Further consider the second row in Table I. Increasing $\mathcal{X}_n$ means more flows become short-lived without requiring connection resources, causing the reduction of both ρ_2 and ρ_3. The increase of ρ_1 is obvious due to the increased software forwarding of short-lived flows.

It is worth mentioning that the cost function J in (3) can be redesigned to reflect the performance difference caused by the individual resource overflows. The actual monetary cost of different resources may also be built into the cost function to help system engineers to optimize the resource allocation in system design stage.

IV. Theoretical Development

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. There is even no general approach available to test the optimality of such control system design unless the problem can be formulated within the Markov control framework. While the design of a controller based on rigorous theoretical test of its optimality under certain traffic assumptions will be pursued in the future, the emphasis of this paper is placed on the design of a heuristic adaptive flow classification algorithm for near-optimal control given a wide range of real traffic characteristics.

For simplicity, here we consider the static design of $\mathcal{T}_\infty$ and $\mathcal{X}_\infty$, i.e., $T_m = T$ and $X_m = X$, $\forall m$. The following theoretical developments and hence the engineering insights explored for such a static design will help us greatly to develop and evaluate the heuristic adaptive control algorithm in the next section. Similar to (3), we define the static control by

$$\text{minimize}_{\{T,X\}}\left\{\max_i\{\bar{\rho}_i(T,X)\}\right\} \tag{4}$$

with

$$\bar{\rho}_i(T,X) = \lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^N \rho_i(n,X,T), \quad \text{for } i = 1,2,3. \tag{5}$$

Hence, the static control is to find a fixed pair, denoted by $\{T^o, X^o\}$, which minimizes the maximum of the three time-averaged utilizations. The theoretical development below is to show that the global optimal solution of (4) corresponds to a unique balance point of the three time-averaged utilizations, i.e., $\bar{\rho}_1(T^o, X^o) = \bar{\rho}_2(T^o, X^o) = \bar{\rho}_3(T^o, X^o)$.

The following property is identified by empirical sensitivity analysis and Property 1.

Property 2: $\bar{\rho}_1(T,X), \bar{\rho}_2(T,X)$, and $\bar{\rho}_3(T,X)$ cannot increase or decrease simultaneously with respect to T and/or X.

It is obvious from Property 1 that Property 2 holds either when one of T and X changes, or when both T and X change simultaneously but in different directions. Yet Property 1 cannot be used to exclude the possibility of simultaneous increase/decrease of all three utilizations when both T and X change in one direction. To show this is also impossible, let $\Delta X = k\Delta T$. We have

$$\Delta\bar{\rho}_i = \left(\frac{\partial\bar{\rho}_i}{\partial T} + \frac{\partial\bar{\rho}_i}{\partial X}k\right)\Delta T, \quad i = 1,2,3. \tag{6}$$

Without loss of generality, assume that both T and X are reduced by ΔT and ΔX. From Table I, $\Delta\bar{\rho}_2 > 0$. The following condition then must be satisfied for $\Delta\bar{\rho}_1 > 0$ and $\Delta\bar{\rho}_3 > 0$:

$$k_3 < k < k_1 \tag{7}$$

with

$$k_1 = -\frac{\partial\bar{\rho}_1}{\partial T}\bigg/\frac{\partial\bar{\rho}_1}{\partial X}, \quad k_3 = -\frac{\partial\bar{\rho}_3}{\partial T}\bigg/\frac{\partial\bar{\rho}_3}{\partial X}. \tag{8}$$

Let us examine if such a condition holds by empirical sensitivity analysis using real trace simulations. Fig. 3 shows an example of the three average utilizations as a function of X and T based on *lbl-trace* simulation. As we can see, the software-forwarding demand $\bar{\rho}_1$ is sensitive to X but insensitive to T, except at small T's. That is, when T is already reasonably large, further increasing T will no longer effectively relay more packets into a flow. In contrast, changing X always has a direct impact on the volume of software-forwarding packets. In consequence, k_1 is likely to be small. On the other hand, the flow cache table demand $\bar{\rho}_3$ is

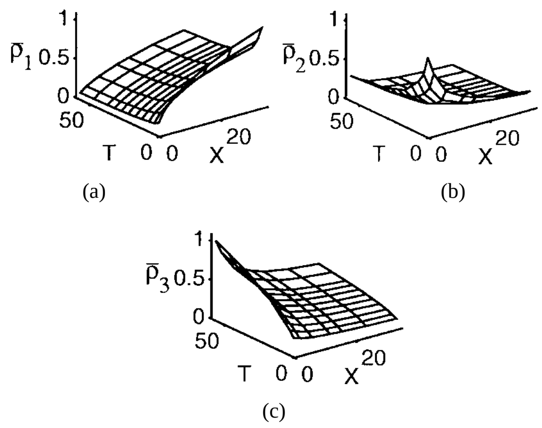

Fig. 3. The average resource demands at each given (T, X).

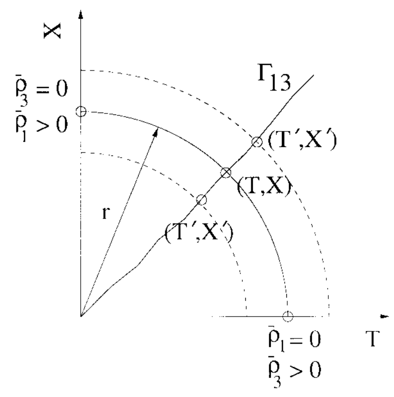

Fig. 4. Schematic diagram of Γ_{13}.

TABLE II
SENSITIVITY ANALYSIS

(T,X)	fixwest-trace	cisco-trace	lbl-trace
	k_3/k_1	k_3/k_1	k_3/k_1
(3,1)	220	3.4	18
(3,35)	250	2.3	3.3
(60,1)	13	8.4	20
(60,35)	9.2	8.3	3.3

sensitive to T but insensitive to X, unless X is small. This is because the time duration of a flow in the flow cache table is directly related to the timeout value T. Changing X may not significantly affect the flow cache table demand unless it can substantially change the number of long-lived flows requesting cut-through setup. This is likely to occur only when X is small since a large portion of the flows in current Internet/intranet are found to consist of a small number of packets, which are unlikely to be classified into long-lived flows unless X is small. Thus, k_3 is expected to be relatively large. In other words, the condition $k_3 < k_1$ is unlikely to hold in practice, such that the possibility of simultaneous increase (decrease) of the three average utilizations with the decrease (increase) of T and X can be largely neglected.

To further verify the above analysis, we performed the simulation study for the three Internet/intranet traces, where the control parameters $\{T, X\}$ are statically assigned in a wide range $T \in [3, 60]$ and $X \in [1, 35]$. Listed in Table II are the collected measurement of k_3/k_1 at the four $\{T, X\}$ boundary pairs. For all the three traces, k_3 is found significantly greater than k_1. From the monotone property, these k_3/k_1 values at the boundaries cover the worst case measurement. Hence, Property 2 is preserved.

Theorem 1: Assume that $\bar{\rho}_i(T, X), \forall i$, are continuous functions of T and X, whose boundary conditions satisfy

$$\bar{\rho}_1(T, 0) = 0, \quad \bar{\rho}_1(0, 0) = 0, \quad \bar{\rho}_2(\infty, \infty) = 0$$
$$\bar{\rho}_3(0, X) = 0, \quad \bar{\rho}_3(0, 0) = 0.$$

Then, there exists one and only one balance point, i.e.,

$$\bar{\rho}_1(T^o, X^o) = \bar{\rho}_2(T^o, X^o) = \bar{\rho}_3(T^o, X^o) \tag{9}$$

with $T^o, X^o \in [0, \infty]$.

Note that the boundary conditions hold in general and can be derived from the physical meaning of the three utilizations.

Proof: Consider a set $\Gamma_{13} = \{(T, X) : \bar{\rho}_1(T, X) = \bar{\rho}_3(T, X)\}$. We first show that $(T, X) \in \Gamma_{13}$ must increase or decrease simultaneously. In Fig. 4, we draw a circle of radius r in the first quadrant of the $\{T, X\}$ plane. Notice that $\bar{\rho}_1 > 0$ and $\bar{\rho}_3 = 0$ at $(T, X) = (0, r)$ and $\bar{\rho}_1 = 0$ and $\bar{\rho}_3 > 0$ at $(T, X) = (r, 0)$. As (T, X) changes from $(0, r)$ to $(r, 0)$ clockwise along the circle, we know from Property 1 that $\bar{\rho}_1$ monotonically decreases to zero while $\bar{\rho}_3$ monotonically increases from zero. Hence, there must be a unique point on the circle in the first quadrant where $\bar{\rho}_1 = \bar{\rho}_3$ at each given r. Hence, Γ_{13} must represent a unique curve in Fig. 4.

Then, consider two distinct points on the curve, i.e, (T, X) and $(T', X') \in \Gamma_{13}$. By contradiction, let us show that we must have $T < T'$ and $X < X'$, or $T > T'$ and $X > X'$. Assuming $T \leq T'$ and $X > X'$, from Property 1, we have $\bar{\rho}_1(T, X) > \bar{\rho}_1(T', X')$ and $\bar{\rho}_3(T, X) < \bar{\rho}_3(T', X')$. It implies that we cannot have both (T, X) and $(T', X') \in \Gamma_{13}$, contradictory to the assumption. Similarly, it is easy to show that $T > T'$ and $X \leq X'$ cannot occur. Hence, $(T, X) \in \Gamma_{13}$ must increase or decrease simultaneously, as shown in Fig. 4. From the above result and Property 1, $\bar{\rho}_2(T, X)$ then must be a decreasing function of $(T, X) \in \Gamma_{13}$ with the limit value $\bar{\rho}_2(\infty, \infty) = 0$. Define $\bar{\rho}_{13}(T, X) = \bar{\rho}_1(T, X) = \bar{\rho}_3(T, X)$ for $(T, X) \in \Gamma_{13}$ with the limiting value $\bar{\rho}_{13}(0, 0) = 0$. From Property 2, $\bar{\rho}_{13}(T, X)$ must be a nondecreasing function of (T, X). Hence, there must be a balance point $(T^o, X^o) \in \Gamma_{13}$ where $\bar{\rho}_{13}(T^o, X^o) = \bar{\rho}_2(T^o, X^o)$ as in (9), and this balance point must be unique.

We can further show that $\bar{\rho}_{13}(T, X)$ is an increasing function of (T, X). For $(T, X) \in \Gamma_{13}$, we must have $\Delta\bar{\rho}_1 = \Delta\bar{\rho}_3$ in (6), which is equivalent to

$$\frac{\partial \bar{\rho}_1}{\partial X}(-k_1 + k) = \frac{\partial \bar{\rho}_3}{\partial X}(-k_3 + k).$$

From Property 1, both $\frac{\partial \bar{\rho}_1}{\partial X}$ and $\frac{\partial \bar{\rho}_3}{\partial X}$ must be nonzero. Also from $k_3 \neq k_1$, we get $\Delta\bar{\rho}_1 = \Delta\bar{\rho}_3 \neq 0$. Therefore, $\rho_{13}(T, X)$ must be an increasing function of $(T, X) \in \Gamma_{13}$. □

Theorem 2: The optimal solution of the min–max objective (4) corresponds to the unique balance point.

Proof: Let (T^o, X^o) be a global optimal solution of (4). We first claim

$$\bar{\rho}_3(T^o, X^o) = \max\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2(T^o, X^o)\} \tag{10}$$

which can be proved by contradiction. Assume $\bar{\rho}_3\,(T^o, X^o) < \max\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2(T^o, X^o)\}$. Then $\max\,\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2\,(T^o, X^o)\}$ must be the optimal value in (4). From Property 1, $\exists T' > T^o$ such that $\bar{\rho}_3\,(T', X^o) = \max\{\bar{\rho}_1(T', X^o), \bar{\rho}_2(T', X^o)\}$. Also from Property 1, $\max\{\bar{\rho}_1\,(T', X^o), \bar{\rho}_2(T', X^o)\} < \max\{\bar{\rho}_1(T^o,\ X^o), \bar{\rho}_2(T^o, X^o)\}$, contradicting the fact that $\max\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2(T^o, X^o)\}$ is the optimal value. Similarly, assume $\bar{\rho}_3(T^o, X^o) > \max\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2(T^o, X^o)\}$. Then $\bar{\rho}_3(T^o, X^o)$ must be the optimal value in (4). From Property 1, $\exists T' < T^o$ such that $\bar{\rho}_3(T', X^o) = \max\{\bar{\rho}_1(T',\ X^o), \bar{\rho}_2(T', X^o)\}$. Also from Property 1, $\bar{\rho}_3(T', X^o) < \bar{\rho}_3(T^o, X^o)$, again, contradicting the fact that $\bar{\rho}_3(T^o, X^o)$ is the optimal value. Hence, (10) must hold. Similarly, one can change (T^o, X^o) to (T^o, X') and show

$$\bar{\rho}_1(T^o, X^o) = \max\{\bar{\rho}_2(T^o, X^o), \bar{\rho}_3(T^o, X^o)\}. \tag{11}$$

There are two distinct cases in (10):

Case 1: $\max\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2(T^o, X^o)\} = \bar{\rho}_2(T^o, X^o)$;

Case 2: $\max\{\bar{\rho}_1(T^o, X^o), \bar{\rho}_2(T^o, X^o)\} = \bar{\rho}_1(T^o, X^o)$.

For case 1, $\bar{\rho}_1 = \bar{\rho}_2 = \bar{\rho}_3$ at (T^o, X^o) from (10) and (11). For case 2, one can only get $\bar{\rho}_{13} = \bar{\rho}_1 = \bar{\rho}_3 \geq \bar{\rho}_2$ at (T^o, X^o) from (10) and (11). Assume $\bar{\rho}_{13} > \bar{\rho}_2$ at (T^o, X^o). Then $\bar{\rho}_{13}(T^o, X^o)$ must be the optimal value in (4). Recall that $\bar{\rho}_2$ is a decreasing function of (T, X) and $\bar{\rho}_{13}$ is an increasing function of (T, X) in Γ_{13}. We must be able to find a pair $(T', X') \in \Gamma_{13}$ with $T' < T^o$ and $X' < X^o$ such that $\bar{\rho}_{13} = \bar{\rho}_2$ at (T', X'). But we also have $\bar{\rho}_{13}(T', X') < \bar{\rho}_{13}(T^o, X^o)$, contradicting the fact that $\bar{\rho}_{13}(T^o, X^o)$ is the optimal value. In consequence, $\bar{\rho}_1 = \bar{\rho}_2 = \bar{\rho}_3$ at (T^o, X^o) in case 2 as well.

Since the balance point is unique by Theorem 1, the optimal solution of (4) must also be unique. □

Hence, our static optimization problem is reduced to a balance problem for finding a unique balance point. Although the analysis cannot be generally extended to solving the adaptive optimization problem defined in (3), the understanding of such static optimal control helps us greatly to develop an effective adaptive control algorithm for near optimal control. Especially when the traffic characteristics change slowly, the quasi-static approximation can be applied to the adaptive control design. In other words, one can define the unique balance point in moving-average sense instead of longterm average, which is equivalent to replacing $\bar{\rho}_i(T, X)$ in (5) by $\hat{\rho}_i(n)$ in (2). Hence, the optimal adaptive control can be designed through the adaptive tracking of such a unique balance point in moving-average. The adaptive tracking is achieved through the adaptive assignment of (T_n, X_n) at the nth adaptation interval, $\forall n$. Since the moving-average balance point is expected to be unique at any given time, the quasi-static control only requires the local adaptation of (T_{n+1}, X_{n+1}) on the basis of (T_n, X_n) and $\hat{\rho}_i(n), \forall i$, at the nth adaptation interval.

In the next section, we design an effective adaptive algorithm based on the quasi-static control principle, where the problem is formulated as a stochastic learning automata system for the local search of a unique balance point in moving average. The optimality of the proposed controller will be examined against the static optimal solution by real trace simulation study.

V. Adaptive Control Design

A. Control Architecture

Depending on the relative differences among $\hat{\rho}_1(n)$, $\hat{\rho}_2(n)$, and $\hat{\rho}_3(n)$, seven operation regions can be identified:

R_0: $\hat{\rho}_1(n) \approx \hat{\rho}_2(n)$, $\hat{\rho}_1(n) \approx \hat{\rho}_3(n)$, $\hat{\rho}_2(n) \approx \hat{\rho}_3(n)$ which is the balanced region;

R_1: $\hat{\rho}_1(n) > \hat{\rho}_2(n) \approx \hat{\rho}_3(n)$ where $\hat{\rho}_1(n)$ is relatively overloaded;

R_2: $\hat{\rho}_2(n) > \hat{\rho}_1(n) \approx \hat{\rho}_3(n)$ where $\hat{\rho}_2(n)$ is relatively overloaded;

R_3: $\hat{\rho}_3(n) > \hat{\rho}_1(n) \approx \hat{\rho}_2(n)$ where $\hat{\rho}_3(n)$ is relatively overloaded;

R_4: $\min(\hat{\rho}_1(n), \hat{\rho}_2(n)) > \hat{\rho}_3(n)$ where both $\hat{\rho}_1(n)$ and $\hat{\rho}_2(n)$ are relatively overloaded;

R_5: $\min(\hat{\rho}_1(n), \hat{\rho}_3(n)) > \hat{\rho}_2(n)$ where both $\hat{\rho}_1(n)$ and $\hat{\rho}_3(n)$ are relatively overloaded;

R_6: $\min(\hat{\rho}_2(n), \hat{\rho}_3(n)) > \hat{\rho}_1(n)$ where both $\hat{\rho}_2(n)$ and $\hat{\rho}_3(n)$ are relatively overloaded.

Choosing the criterion $\hat{\rho}_i(n) \approx \hat{\rho}_j(n)$, instead of $\hat{\rho}_i(n) = \hat{\rho}_j(n)$ prevents overreaction to small disturbances, thus ensuring the stability of the control system. We define $\hat{\rho}_i(n) \approx \hat{\rho}_j(n)$ if and only if $|\hat{\rho}_i(n) - \hat{\rho}_j(n)| \leq \epsilon$, for some small ϵ.

We then describe such a system by a finite state machine where each state is associated with one region. The system often drifts away from state R_0 to other unbalanced states due to the nonstationary traffic variation and/or the resource capacity change. Our control objective is to drive the system to converge to R_0 in finite steps with near-optimal performance. Based on the quasi-static approximation, we assume that the optimal control in terms of (3) corresponds to the successful tracking of such a unique balanced state R_0. As described below, the problem can be formulated as a hierarchical stochastic learning automata system [10].

A stochastic learning automata system is composed of N states. In our case, $N = 7$. Among them, there is a desired state, to which the system is driven to converge under nonstationary disturbances. In our case, this state is the balanced state R_0. The inputs from the environment are measured at discrete time n, which are $\rho_1(n)$, $\rho_2(n)$, and $\rho_3(n)$ in our case. They are used to identify the system state at time n. Assigned to each state are a set of *actions* and its associated probability set. In our case, each action is represented by a pre-assigned adaptation of (T_n, X_n) to (T_{n+1}, X_{n+1}) at time n. To drive the system from any undesired state to state R_0, an action is probabilistically selected from the action set based on the given probability set. The probability set in a particular state, for example, R_i, is self-adjusted by a *learning algorithm* at time $n+1$ given that the system was in state R_i at time n. The learning algorithm is designed using the information at both current and previous states (including the previous action selected).

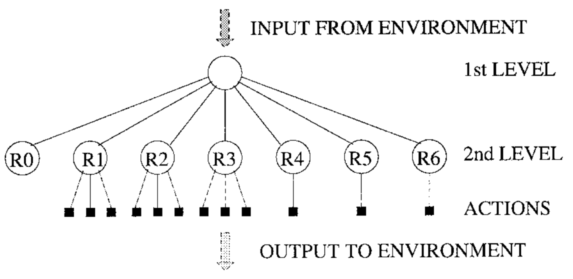

Fig. 5. Hierarchical structure of the control system.

We first define our stochastic learning automata system in mathematical terms. Use $R(n)$ to represent the system state at time n. For a state $R(n) = R_j$, assume there are r possible actions associated with it, denoted by $\vec{\alpha}^j = \{\alpha_1^j, \alpha_2^j, \cdots, \alpha_r^j\}$. One of them, denoted by $\alpha_m^j(n)$, is probabilistically selected from $\vec{\alpha}^j$ using the present probability vector $\mathbf{p}^j(n) = \{p_1^j(n), p_2^j(n), \cdots, p_r^j(n)\}$. Such a probability vector at time $n+1$ is adjusted by a learning algorithm $(\mathbf{LA})$

$$\mathbf{p}^j(n+1) = \mathrm{LA}\big[R(n+1) = R_i, \alpha_m^j(n), \mathbf{p}^j(n) \,\big|\, R(n) = R_j\big]. \quad (12)$$

Note that the next state $R(n+1) = R_i$ is the outcome of both the action $\alpha_m^j(n)$ in state $R(n) = R_j$ and the present random traffic arrivals. Hence, $\mathbf{p}^j(n+1)$ is updated based on $R(n+1) = R_i$, $\alpha_m^j(n)$, and $\mathbf{p}^j(n)$. This is a learning algorithm in the sense that the adjustment of the probability for taking a particular action in the future is determined by the outcome $R(n+1) = R_i$ induced by the same action at time n. If the outcome R_i is a favorable (unfavorable) state, the probability for taking the same action in the future is increased (decreased), whereas the probabilities for the other actions are relatively reduced (increased). Such probability update prevents the system from being trapped in an unbalanced state or jumping among several unbalanced states indefinitely.

Fig. 5 shows the structure of a two-level hierarchical learning automata system in our problem setting. The reason behind the specific choice of the number of actions for each state will soon be clarified. The input from the environment are the instantaneous utilizations $\rho_k(n+1)$, $\forall k$, measured at time $n+1$. The first level is to identify the current state $R(n+1) = R_i$ based on $\hat{\rho}_k(n+1)$, $\forall k$, obtained from (2). At the second level, the probability vector $\mathbf{p}^j(n)$ of the previous state $R(n) = R_j$ is updated to $\mathbf{p}^j(n+1)$ by (12); the new control action as output will then be selected from $\vec{\alpha}^i$ by the probability vector $\mathbf{p}^i(n+1)$ of the current state $R(n+1) = R_i$. The control output is described by $(\Delta T_{n+1}, \Delta X_{n+1})$, which represents the incremental change of the control variables (T_{n+1}, X_{n+1}) from the previous (T_n, X_n). Obviously, controlling the increment $(\Delta T_{n+1}, \Delta X_{n+1})$ has the advantage of requiring a much smaller control action set in each state as compared to the direct control of (T_{n+1}, X_{n+1}).

Finally, some design rules need to be specified. We require that both X_{n+1} and T_{n+1}, whenever updated, be immediately applied to the existing and forthcoming flows. More specifically, whenever a nonzero ΔX_{n+1} is identified, any forthcoming flow will require forwarding X_{n+1} packets before cut-through switching. Further, all the existing flows, which have forwarded more than X_{n+1} packets but have not been cut-through switched, will request the setup of cut-through connection immediately. Similar rules apply to the adaptation of T_{n+1}.

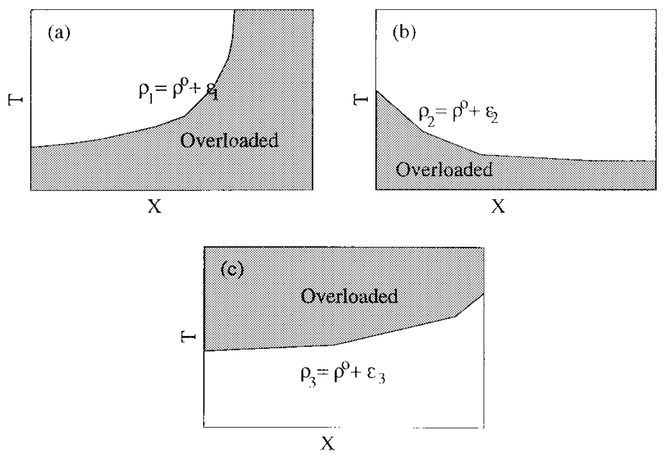

Fig. 6. Schematic plots of $\rho_i = \rho^o + \epsilon_i$ which divides the overload region from the underload region: (a) software-forwarding, (b) connection-setup, and (c) active-connection.

For simplicity, here we have assumed the instant setup of each cut-through connection. In practice, however, packets which arrive during the cut-through connection setup period will still be software forwarded. The actual cut-through connection setup time is system dependent. For the Ipsilon IP switching, the setup time is on the order of 10 μs [9]. The study in [9] indicates that such a short setup time has negligible impact on the overall switching performance. In this case, our assumption of zero setup time can be directly applied. For MPOA, the setup time can be much longer because of the end-to-end address resolution, table caching, and signaling delays. In our modeling, the extra packet forwarding required during the connection setup can be taken into account by setting a proper minimum value of X in the adaptation.

B. Algorithm Design

To identify a proper action set associated with each state and so to design an effective learning algorithm, let us first qualitatively characterize the properties of the seven states with respect to (T, X) in the static sense. Based on the monotone properties of each individual resource with respect to T and X in Table I, one can readily obtain the phase diagram of the three resource utilizations in Fig. 6 where an equi-utilization curve is drawn to divide the operation into overloaded and underloaded regions for each resource. Assume that the optimal solution is found at (T^o, X^o), where all three average utilizations are balanced at ρ^o. For each resource, we then draw the equi-utilization curve at $\rho_i = \rho^o$, with respect to (T, X) (note that the same ρ_i can be achieved with different (T, X)'s when only a single resource is considered). Of course, the cross point among the three curves, when they are all plotted together, represents the optimal solution (T^o, X^o). In our application, since we use a balanced region $|\rho_i - \rho_j| < \epsilon$ instead of a balanced point $\rho_1 = \rho_2 = \rho_3$, the single optimal point is

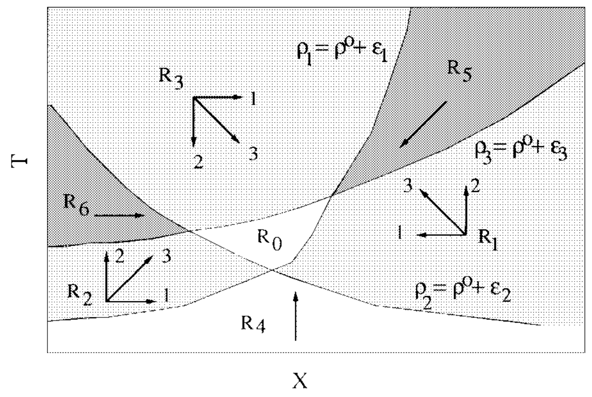
$\rho_1 = \rho^o + \varepsilon_1$
R_3
R_5
1
2
3
T
$\rho_3 = \rho^o + \varepsilon_3$
3
2
R_6
1
R_1
R_0
2
3
R_2
1
$\rho_2 = \rho^o + \varepsilon_2$
R_4
X

Fig. 7. Phase diagram.

TABLE III
CONTROL ACTIONS $(\Delta T_n, \Delta X_n)$ IN EACH STATE: 0 FOR NO ACTION, + FOR POSITIVE VALUE, AND − FOR NEGATIVE VALUE

	R_0	R_1	R_2	R_3	R_4	R_5	R_6
$(\Delta T_n, \Delta X_n)$	$(0,0)$	$(0,-)$ $(+,0)$ $(+,-)$	$(0,+)$ $(+,0)$ $(+,+)$	$(0,+)$ $(-,0)$ $(-,+)$	$(+,0)$	$(-,-)$	$(0,+)$

extended to an optimal operation region. This is why we have chosen $\rho_i = \rho^o + \epsilon_i$ for the equi-utilization curve in Fig. 6, where ϵ_i is properly adjusted to reflect $|\rho_i - \rho_j| = \epsilon$, which defines the boundaries.

Fig. 7 shows the three overlapped curves, which naturally divides the (T, X) assignment into seven regions as defined by the states $R_0 \sim R_6$. Associated with each unbalanced region in Fig. 7, we further use arrows to describe the possible actions to adjust T and X for the purpose of driving each unbalanced region into the balanced region R_0. For instance, in R_4 where both ρ_1 and ρ_2 are overloaded, from Table I one can identify that increasing T has the effect of simultaneously reducing ρ_1 and ρ_2. Similarly, in R_1 where ρ_1 is relatively overloaded, it is shown in Table I that both reducing X and increasing T can have the same effect of reducing ρ_1. This is why three possible actions are selected in R_1: reducing X (action 1), increasing T (action 2), or both (action 3), each of which is represented by one arrow within region R_1. The proper actions in the remaining unbalanced regions can then be similarly constructed from Table I.

Our adaptive algorithm is designed on the basis of the phase diagram in Fig. 7. First, since the balance point is expected to be unique by the quasi-static approximation, the local search of (T_n, X_n) on the basis of the present (T_{n-1}, X_{n-1}) should be sufficient. In other words, only the incremental changes $(\Delta T_n, \Delta X_n)$ need to be assigned to the control actions in each region, with respect to $T_n = T_{n-1} + \Delta T_n$ and $X_n = X_{n-1} + \Delta X_n$. As summarized in Table III for the design of ΔT_n and ΔX_n, we use "0" for no change, "+" for some positive value, and "−" for some negative value. Both T_n and X_n must be positive. In practice, X_n is defined in packet units and T_n in second units.

The actual value of each action can be optimized by careful tunning of the action value based on real trace simulations. For all our simulation studies, we simply fix $\Delta T_n = \pm 1$ and assign ΔX_n, as listed in Table IV. Our real trace study indicates that $\hat{\rho}_k(n), \forall k$ is more sensitive to X_n than to T_n when X_n is small. In other words, when X_{n-1} is small, taking $\Delta X_n = 1$ or -1 can significantly change the balance among the three utilizations, which may lead to undesired control oscillation. This is because many flows in IP traffic consist of a few packets. When X_{n-1} is small, taking $\Delta X_n = 1$ or -1 may substantially change the balance between short-lived flows and long-lived flows. To prevent this from happening, we assign a noninteger value to ΔX_n when $X_n < 5$. A noninteger X_n can be implemented through a simple probabilistic assignment. For instance, taking the control variable $X_n = 1.25$ means that 75% of flow arrivals will be classified by $X_n = 1$ while the other 25% will be classified by $X_n = 2$.

TABLE IV
THE ASSIGNMENT OF $\Delta X_n \in \{0, -, +\}$ DEPENDING ON X_{n-1}

X_{n-1}	$\Delta X_n \in \{0, -, +\}$
$(0,1)$	$\{0, -0.20, +0.20\}$
1	$\{0, -0.20, +0.25\}$
$(1,2)$	$\{0, -0.25, +0.25\}$
2	$\{0, -0.25, +0.33\}$
$(2,3)$	$\{0, -0.33, +0.33\}$
3	$\{0, -0.33, +0.50\}$
$(3,4)$	$\{0, -0.50, +0.50\}$
4	$\{0, -0.50, +1.00\}$
≥ 5	$\{0, -1.00, +1.00\}$

For the three possible control actions associated with each of R_1, R_2, and R_3 in Table III, a probabilistic assignment is used to choose one of the three actions. We need to develop a learning algorithm as defined in (12) to update the corresponding probability vectors $\mathbf{p}^1(n)$, $\mathbf{p}^2(n)$, and $\mathbf{p}^3(n)$. Every vector has three probability elements, defined for three actions and denoted by $\mathbf{p}^j(n) = [p_1^j(n), p_2^j(n), p_3^j(n)]$ with $p_i^j(n) + p_2^j(n) + p_3^j(n) = 1$, $\forall n$, for $j = 1, 2, 3$.

The design of the learning algorithm is summerized in Table V. We can use Fig. 6 to understand the entries in the learning algorithm. Suppose that action 1 was taken at $R(n) = R_1$, moving the system to $R(n+1) = R_2$. According to the phase diagram in Fig. 7, it is very likely that such a move from R_1 to R_2 is partially due to the overreacting of action 1 at R_1. The learning algorithm should then be able to reduce the probability of action 1 at R_1, which is achieved by increasing the probability of the other action(s), such as action 2 in the phase diagram. In Table V, such a probability adjustment is represented by $\{1-, 2+\}$, which is to weaken action 1 while strengthen action 2 once the state R_1 is moved to R_2 through one adaptation. The purpose is to reduce the likelihood for the stochastic learning automata system to repeat the same mistake when it comes back to state R_1. If action 1 in state R_1 somehow leads to state R_5 instead of R_2, it is obvious from Fig. 7 that such a transition is unlikely to be caused by action 1 but by other traffic/resource variation factors. Hence, no probability adjustment is required as indicated by *null* in Table V for the transition from R_2 to R_5 upon action 1.

All the possible probability adjustments for different state transitions can be similarly constructed as listed in Table V. For some transitions, however, one may not be able to exactly identify if the transition is attributed to the

TABLE V
PROBABILITY ADJUSTMENT FOR ACTIONS IN STATE $R(n)$. "$i+$" MEANS TO INCREASE THE PROBABILITY OF ACTION i AND "$i-$" MEANS TO REDUCE THE PROBABILITY OF ACTION i

$(R(n), action)$	$R_1(n+1)$	$R_2(n+1)$	$R_3(n+1)$	$R_4(n+1)$	$R_5(n+1)$	$R_6(n+1)$
$(R_1, 1)$	*null*	{1-,2+}	*null*	{1-,2+}	*null*	{1-,2+}
$(R_1, 2)$	*null*	*null*	{1+,2-}	*null*	{1+,2- }	*null*
$(R_1, 3)$	*null*	{1-,2+,3-}	{1+,2-,3-}	{1-,2+,3-}	{1+,2-,3-}	{1-,2+,3-}
$(R_2, 1)$	{1-,2+}	*null*	*null*	{1-,2+}	*null*	*null*
$(R_2, 2)$	*null*	*null*	{1+,2-}	*null*	*null*	{1+,2-}
$(R_2, 3)$	{1-,2+,3-}	*null*	{1+,2-,3-}	{1-,2+,3+}	{1-,2+,3-}	{1+,2-,3-}
$(R_3, 1)$	{1-,2+}	*null*	*null*	*null*	{1-,2+}	*null*
$(R_3, 2)$	*null*	{1+,2- }	*null*	{1+,2-}	*null*	{1+,2-}
$(R_3, 3)$	{1+,2-,3- }	{1+,2-,3-}	*null*	{1+,2-,3-}	{1-,2+,3-}	{1+,2-,3-}

action taken in the previous state. For instance, consider the transition from R_3 to R_1 upon action 1. From the phase diagram, it is not clear if such a transition is caused by action 1 in R_3, which otherwise would require further knowledge of the operating point within regions R_3 and R_1. In this situation, we simply choose to weaken action 1 while strengthen action 2 in state R_3 for the purpose of reducing the likelihood of repetition of such transitions.

There are two types of probability adjustment in Table V. One only deals with the probability adjustment of action 1 and action 2, expressed by

$$\begin{aligned} p^j_{\alpha_1}(n+1) &= p^j_{\alpha_1}(n) - \lambda p^j_{\alpha_1}(n) \\ p^j_{\alpha_2}(n+1) &= p^j_{\alpha_2}(n) + \lambda p^j_{\alpha_1}(n) \\ p^j_3(n+1) &= p^j_3(n) \end{aligned} \tag{13}$$

with $\alpha_1, \alpha_2 \in \{\text{action 1}, \text{action 2}\}$ and λ is a properly selected tunning parameter in the range of $(0, 1)$. The other type requires the probability adjustment of all three actions, expressed by

$$\begin{aligned} p^j_{\alpha_1}(n+1) &= p^j_{\alpha_1}(n) \\ &\quad - \lambda p^j_{\alpha_2}(n) p^j_{\alpha_1}(n) / \left(p^j_{\alpha_3}(n) + p^j_{\alpha_1}(n)\right) \\ p^j_{\alpha_2}(n+1) &= p^j_{\alpha_2}(n) + \lambda p^j_{\alpha_2}(n) \\ p^j_{\alpha_3}(n+1) &= p^j_{\alpha_3}(n) \\ &\quad - \lambda p^j_{\alpha_2}(n) p^j_{\alpha_3}(n) / \left(p^j_{\alpha_3}(n) + p^j_{\alpha_1}(n)\right) \end{aligned} \tag{14}$$

with α_1, α_2, $\alpha_3 \in \{\text{action 1}, \text{action 2}, \text{action 3}\}$.

C. Design Complexity

There are three basic components in the design of the above adaptive algorithm. The first one is for the function of flow identification/classification given the current $\{X_n, T_n\}$. It requires a timer t_i and a counter x_i at each layer-3 flow cache table entry i, where t_i provides the arrival time of the previous packet and x_i gives the number of packet arrivals in the current flow. Assume there is a global time clock t which provides the current time. Upon each packet arrival at entry i, the component takes the following four steps.

- If $t - t_i > T_n$, set $x_i = 1$ to start a new flow identification.
- If $t - t_i \leq T_n$ and $x_i \leq X_n$, set $x_i = x_i + 1$ to continue flow classification.
- If $t - t_i \leq T_n$ and $x_i > X_n$, set up connection for a long-lived flow.
- Set $t_i = t$ to upgrade packet arrival time.

Note that $t - t_i$ is the current packet interarrival time. These steps are identical for both static and adaptive flow identification/classification schemes.

The second component provides the on-line measurement and filter operation for the present utilization of individual resources, i.e., to identify $\hat{\rho}_i(n), \forall i$.

The third component is designed for periodic update of the global parameters $\{X_n, T_n\}$. The following steps are required per adaptation.

- Identify the current state $R(n)$ (with no more than three subtractions and three comparisons).
- Decide whether a probability vector is to be updated (with no more than four comparisons).
- Update the probability vector by (13) or (14) (with no more than three additions, five multiplications and two divisions).
- Choose a control action (with no more than two comparisons, plus one random number generation).
- Update T_n and X_n (with no more than two additions/subtractions).

Since the adaptation time interval is in the range of a few seconds. the time complexity of this component is negligible as compared to the first component.

VI. SIMULATION STUDY

In this section, the performance of the proposed adaptive flow classification algorithm is tested based on the real Internet/intranet trace simulations. Two important performance aspects will be examined. First, the optimality of the proposed adaptive algorithm is tested against the optimal static solution in a relatively smooth traffic environment, given vastly different initial conditions. Second, we show the robustness of the adaptive algorithm to the abrupt changes of input traffic characteristics and available resource capacities, where the quasi-static approximation may no longer hold.

The following parameters are provided for the simulation study. The adaptation time interval is set to 2 s. The criterion for $\hat{\rho}_i(n) \approx \hat{\rho}_j(n)$ is measured by $\epsilon = 0.08$. The weighting factors $(\omega_1, \omega_2, \omega_3)$ for damped utilization are chosen as $(0.5, 0.6, 0.6)$. The parameter λ in the learning algorithm is fixed at 0.5.

To test the optimality of the proposed algorithm, we first consider the Internet backbone traffic, represented by *fixwest-trace*. The following *fixed* resource capacities are assumed to

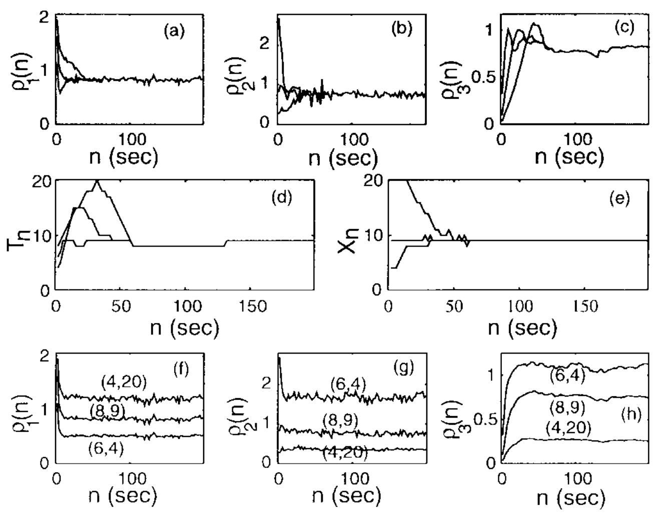

Fig. 8. Transient behavior of initial impact with $(T_0, X_0) = (4, 20), (6, 4), (8, 9)$. (a)–(c) Adaptive control performance, (d), (e) control sequence, and (f)–(h) static control performance.

be available:

$$S_{\max} = 5000 \text{ pktps}, \quad C_{\max} = 300 \text{ flwps} \\ A_{\max} = 5000 \text{ flw} \tag{15}$$

where $C_{\max}$ is considered to be relatively more constrained. To see the impact of different selections of the initial control parameters (T_0, X_0) on the transient convergence behavior of the adaptive controller, three sets of (T_0, X_0) are selected, i.e., $(4, 20)$, $(6, 4)$, and $(8, 9)$. In order to stimulate a highly nonstationary transient period, all our simulations further start from zero initial resource utilizations. Fig. 8 shows the first 200-s convergence behavior of the three instantaneous resource utilizations defined in (1). Focusing on the adaptive results in Fig. 8(a)–(c), we observe that the substantial overloading and underloading of the initial impact are soon extinguished despite the vastly different initial conditions. After the initial 70-s period, all $\rho_i(n)$'s are approaching 80%. Also plotted in Fig. 8(d) and (e) are the control sequences of (T_n, X_n), which are approaching $(8, 9)$, respectively. *Fixwest-trace* is found to be rather smooth during the 20-min collection. This is why all $\rho_i(n)$'s are soon stabilized and (T_n, X_n)'s stay almost unchanged after the initial impact.

The smoothness of the traffic lends us a natural means to examine the optimality of the proposed algorithm. For comparison purposes, we ran multiple static control simulations based on the same trace, each of which is conducted with a different pair (T, X). A complete static search gave us the same near-optimal solution $(T^o, X^o) = (8, 9)$ as found by the adaptive algorithm, where all the three utilizations approach to the balanced point 80%. Clearly, the performance of the adaptive control quickly converges to the optimal solution in spite of vastly different initial conditions.

Also plotted in Fig. 8(f)–(h) are the simulation results of the static flow classification for (T, X) simply assigned by the three different sets of the initial condition (T_0, X_0), where the three utilizations are highly unbalanced for the two cases which are not optimal in static sense. In other words, the optimal solution (T^o, X^o) for the static classification is strongly dependent on traffic characteristics and resource availability, which are *a priori* unknown.

TABLE VI
Resource Utilizations and Loss Rates Sampled at Different Times of Day with a Static Parameter Assignment $(T, X) = (5, 5)$.

time	$\bar{\rho}_1$	$\bar{\rho}_2$	$\bar{\rho}_3$
$10:00am$	0.4339	0.4737	0.4390
$12:50pm$	0.1338	0.2696	0.3403
$3:15pm$	0.9980	0.7701	0.5666

Note that the trace used for the simulation is only 20 min long, which is relatively stationary in the sense that the daily traffic level and pattern changes have not come into play. Hence, even if a fixed operation point is found to be optimal at some time of a day, it is likely for the same system to operate in a highly unbalanced mode as the daily traffic level and pattern change. For instance, let us study three 5-min traces from the same fast Ethernet hub at Cisco Systems, Inc., at different times on March 4, 1997. Given that the capacities of the three resources are fixed at $(S_{\max}, C_{\max}, A_{\max}) = (500, 20, 500)$, with the static control we found $(T^o, X^o) = (5, 5)$ for the trace collected at 10:00 am Pacific time. From Table VI, we see that all the average utilizations are well balanced with less than 4% relative difference. Also listed in Table VI are the statistics of the other two 5-min traces collected at 12:50 pm and 3:15 pm, using the same static control $(T^o, X^o) = (5, 5)$. As one can see, the utilizations now become highly unbalanced, especially for the heavy load trace collected at 3:15 pm. The low utilizations at 12:50 pm are the consequences of the low traffic volume during the lunch break time.

Let us now examine the optimality of the proposed adaptive algorithm in an intranet environment. We found that all the 100-Mb/s fast Ethernets, where the traces were collected, are highly underutilized (about 5% on average). A flow-based hybrid access switching system in a LAN environment is expected to support multiple fast Ethernets. For instance, consider a hybrid access switching system which has eight 100-Mb/s fast Ethernet ports. The aggregated traffic is therefore generated by the superposition of eight 5-min fast Ethernet traces, collected at Cisco Systems, Inc. The aggregated traffic volume is 34 Mb/s on average. We set the resource capacities at $(S_{\max}, C_{\max}, A_{\max}) = (3000, 400, 3000)$. Fig. 9(a)–(e) shows the convergence behavior of the adaptive control for (T_n, X_n) approaching the optimal balance point $(3, 3)$, starting at the initial condition $(T_0, X_0) = (15, 15)$. The same optimial balance point has been verified by the static control at fixed $(T^o, X^o) = (3, 3)$ displayed in Fig. 9(f)–(h). Again, we have seen the fast convergence of the adaptive control to the unique balance point regardless of the initial conditions.

Next, we examine the adaptivity and robustness of the proposed algorithm to potential abrupt traffic changes, where the quasi-static approximation may no longer hold. *Fixwest-trace* is adopted here and its major applications are abruptly turned off and on. The three dominant applications in the

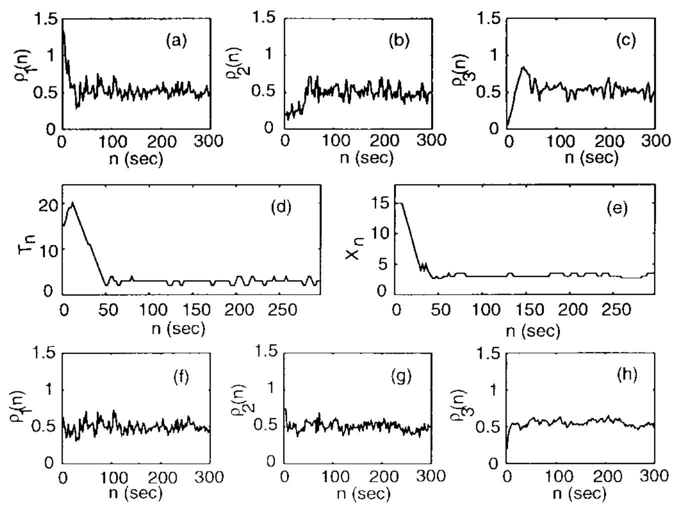

Fig. 9. Control performance on an intranet trace: (a)–(c) adaptive control performance with initial condition $(T_0, X_0) = (15, 15)$, (d)–(e) control sequence (T_n, X_n), and (f)–(h) static control performance at $(T^o, X^o) = (3, 3)$.

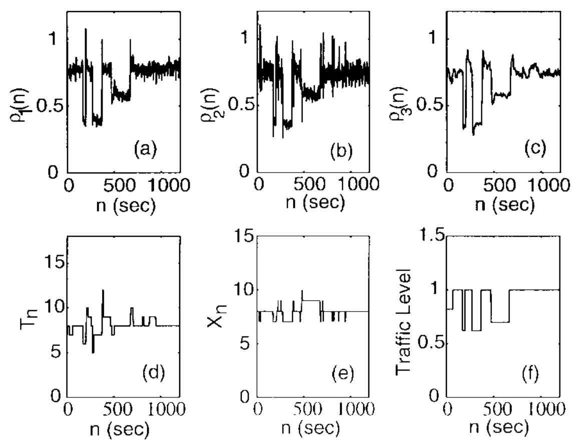

Fig. 10. Adaptive resource allocation to abrupt traffic change.

trace are *ipip*, *dns*, and *http*, each of which consists of 18%, 16%, and 38% of the total packets, respectively. Different applications may have different requirements on resources at the same given (T, X). Hence, turning these applications off and on within a 20-min time period would create a somewhat exaggerated worst-case scenario for the daily traffic change. Within the first 100 s, the *ipip* traffic is turned off. The *http* traffic is then turned off within the next two separate time intervals, i.e., $n = (200, 230]$ and $n = (300\,400]$ in seconds. The *dns* traffic is turned off last within $n = (500, 600]$. The same resource capacities defined in (15) are assumed to be available. Fig. 10 shows the time adaptation of the dynamic resource management to the abrupt traffic changes. The initial value (T_0, X_0) is set at $(2, 2)$, which is far from the optimal point. To concentrate on the effect of the abrupt traffic changes, the first 32-s transient period of the initial impact has been neglected in the plots. Also displayed in Fig. 10 are the control sequences of T_n and X_n, as well as the traffic level variations. Obviously, the proposed algorithm has quickly adapted to the abrupt traffic changes and converges quickly to the new balance point with only one or two adaptations.

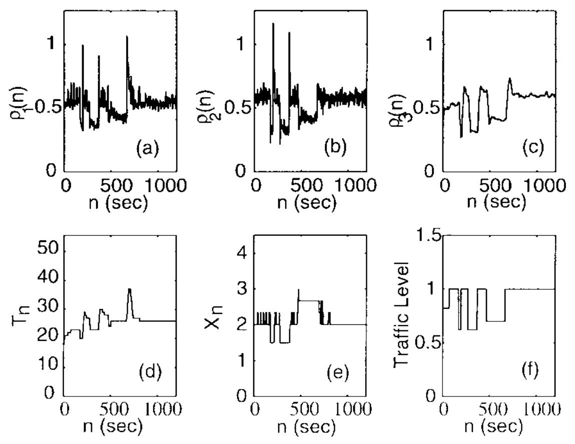

Fig. 11. Adaptive resource allocation to abrupt traffic change with less software-forwarding capacity.

Upon each turn off and turn on, as explicitly shown in the last plot of Fig. 10, we do see some momentary undershoot and overshoot behavior of the individual $\rho_i(n)$'s. This is a common phenomenon in control systems. With finer parameter tunning, which is beyond the scope of this paper, such momentary undershoots and overshoots can be further reduced. It is clear that after each abrupt traffic change the adaptive algorithm can soon resume the resource balancing, which is expected to be unique and optimal in moving average sense.

Let us now consider another set of resources, given by $(S_{\max}, C_{\max}, A_{\max}) = (2{,}000, 800, 30{,}000)$ using the same units as in (15). Compared to the previous set, this system has significantly less software-forwarding capacity but more connection-setup and active-connection resources. The results are presented in Fig. 11. Again, the proposed algorithm resumes the resource balancing shortly after each abrupt traffic changes.

So far, all our case studies have been focused on the effect of traffic characteristics on the performance of the adaptive algorithm, assuming fixed resource capacities. In practice, this assumption is unlikely to be true, as discussed in Section I. It is important to study the performance of the adaptive approach with time-varying resource capacities. Our simulation study is based on *fixwest-trace*. We first assume that the maximum resource capacities are $(S_{\max}, C_{\max}, A_{\max}) = (8000, 500, 8000)$. The static optimal control has been identified at $(T^o, X^o) = (10, 8)$, achieving the balanced average utilization of 50%. Consider then the variation of the three resource capacities. They are first reduced to 40% of their maxima, in turn with 100-s duration each, at time 68, 268, and 468 s, respectively. They are then changed to 80% of their maxima at time 668, 718, and 768 s, respectively. Such resource capacity time variations are depicted in Fig. 12(i). We first show the simulation result of the static control at the fixed $(T^o, X^o) = (10, 8)$ in Fig. 12(a)–(c). As one can see, the resource capacity variations easily drive the utilizations out of balance, resulting in constant overflow of individual resource(s) while the other resources are highly under-utilized. In contrast, Fig. 12(d)–(f) provides the corresponding simulation results of the adaptive control with the initial condition $(T_0, X_0) = (10, 8)$. It is obvious that the adaptive algorithm

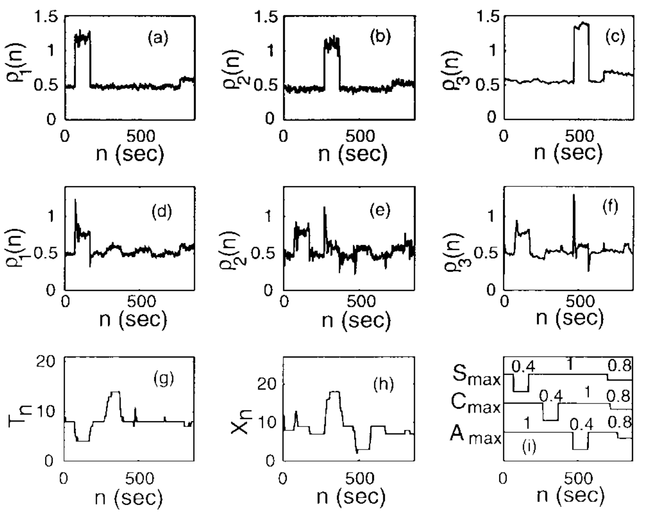

Fig. 12. Adaptive resource allocation to abrupt resource capacity changes: (a)–(c): static control performance with $(T, X) = (10, 8)$, (d)–(f) adaptive control performance at initial condition $(T_0, X_0) = (10, 8)$, (g), (h) control sequence (T_n, X_n), and (i) resource capacity change patterns.

effectively adjusts its control parameters to balance the whole resource utilizations despite the abrupt change of individual resource capacities. That is, the residual capacities of all resources are fully utilized to absorb the potential overflow of individual resources.

In summary, our simulation study exhibits the viability of the proposed flow adaptation to the dynamic allocation of constrained resources under time-varying traffic/resource conditions.

VII. Conclusions and Future Work

In the design of hybrid switching systems, we introduced a new concept of adaptive flow classification, which offers a unique way to dynamically minimize the maximum of the three major system resource utilizations under time varying traffic/resource environment. A static version of the min–max problem was shown to be equivalent to a balance problem with the existence of a unique balance point. Based on the quasi-static approximation, the adaptive control problem can be transformed into the tracking of the single balance point in moving average. An effective adaptive flow classification algorithm was then developed based on the stochastic learning automata system formulation. Our simulation study based on real Internet/intranet traces revealed the significant advantage of the adaptive flow classification approach over the static approach. The adaptive algorithm is found robust in the presence of abrupt changes of traffic characteristics and system resources with fast convergence. Since the time complexity of (T_n, X_n) adaptation is negligible as compared to other functions in a real system, more sophisticated algorithms can be developed for further performance improvement. For instance, one may build the monetary cost of individual resources into the control model such that the overall cost of the resources is to be minimized in the switch product design. One can also extend this paper to include other resource constraints. An important issue which we did not address in this paper is related to flow entry search upon each packet arrival. Since the flow cache table could be large, especially for flows defined at fine granularities such as *host+port* granularity, designing efficient algorithms for flow cache entry search is crucial for the success of the cut-through switching technologies. This issue is currently under investigation.

Acknowledgment

The authors would like to thank A. Arapostathis for helpful discussions on the theoretical aspects of the stochastic control problem formulations and constructive suggestions on the proof of the theorems in Section IV. The authors would also like to thank K. C. Claffy for her kind help with problems related to the FIXWEST trace formatting during the initial stage of this work. The authors would also like to acknowledge NLANR for the FIXWEST trace.

References

[1] R. Cole, D. Shur, and C. Villamizar, “IP over ATM: A framework document,” *IETF RFC 1932*, Apr. 1996.
[2] “Multi-protocol over ATM version 1.0,” *Letter Ballot. The ATM Forum Technical Committee*, May 29, 1997.
[3] P. Newman, T. Lyon, and G. Minshall, “Flow labeled IP: A connectionless approach to ATM,” in *Proc. Infocom’96*, 1996, p. 10b.2.1.
[4] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow, “Tag switching architecture overview,” *IETF Internet Draft*, draft-rekhter-ip-atm-architecture.txt, Sept. 1996.
[5] K. Nagami, Y. Katsube, Y. Shobatake, A. Mogi, S. Matsuzawa, T. Jinmei, and H. Esaki, “Flow attribute notification protocol (FANP) specification,” *IETF Internet-Draft*, draft-rfced-info-nagami-00.txt, Feb. 1997.
[6] R. Woundy, A. Viswanathan, N. Feldman, and R. Boivie, “ARIS: Aggregate route-based IP switching,” *IETF Internet Draft*, draft-woundy-aris-ipswitching-00.txt, Nov. 1996.
[7] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan, “A framework for multiprotocol label switching,” *Proposal at IETF MPLS Working Group*, Apr. 1997.
[8] P. Newman, T. Lyon, and G. Minshall, “Flow labeled IP: Connectionless ATM under IP,” *Networld+Interop*, Las Vegas, 1996. [Online]. Available WWW: http://www.ipsilon.com/staff/pn/papers.html
[9] S. Lin and N. Mckeown, “A simulation study of IP switching,” in *Proc. ACM SIGCOMM’97*, France, Sept. 1997.
[10] P. Mars, J. R. Chen, and R. Nambiar, *Learning Algorithms: Theory and Applications in Signal Processing, Control and Communications*. Boca Raton, FL: CRC Press, 1996.

Hao Che (M’98) received the B.S. degree from Nanjing University, Nanjing, China, in 1984, the M.S. degree in physics from the University of Texas at Arlington in 1994, and the Ph.D. degree in electrical engineering from the University of Texas at Austin in 1998.

In August 1998, he joined the faculty of the Department of Electrical Engineering, the Pennsylvania State University, State College, where he is presently an Assistant Professor. His research interests include network resource management, algorithm design, performance analysis, and traffic modeling.

San-qi Li (M'86) received the B.S. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 1976, and the M.A.Sc. and Ph.D. degrees from the University of Waterloo, Waterloo, ON, Canada, in 1982 and 1985, respectively, all in electrical engineering.

From 1985 to 1989, he was Associate Research Scientist and Principal Investigator at the Center for Telecommunication Research, Columbia University, NY. In September 1989, he joined the faculty of the Department of Electrical and Computer Engineering, University of Texas at Austin, where he is presently Professor. To date, he has published more than 120 papers in international archival journals and refereed international conference proceedings. The main focus of his research has been to develop new analytical methodologies and carry out performance analysis of multimedia service networks and, based on these methodologies, to understand system fundamentals and explore new design concepts. He is an Honorable Professor at Beijing University of Posts and Telecommunication, China.

Dr. Li has served as a member of the Technical Program Committee for the IEEE Infocom Conference from 1988 to 1997. Since 1995, he has served as an Editor of IEEE/ACM TRANSACTIONS ON NETWORKING.

Arthur Lin received the Ph.D. degree with honors in computer engineering from the University of Southern California, Los Angeles.

He is Founder, Vice President, and CTO of Shasta Networks, Inc., Sunnyvale, CA, a Silicon Valley startup focusing on building platforms for enabling value-added services for Internet services providers. Prior to Shasta, he was most recently Senior Engineering Manager and Technical Leader at Cisco Systems, Inc. He designed and led the implementation of Cisco's first home-grown ATM multiservice switching product—the LightStream-1010 series and the Catalyst-8500 Layer 3 switches. He was also a key individual contributor in the design and development of Cisco's Gigabit Switch Router (GSR) c12000 series, multi-layer Gigabit Ethernet switching products, xDSL multi-layer access platform, Tag switching, and c7X00 series routers. Before that, he held various engineering management and software development postitions at Kaleida Labs, Inc., SanCom Corporation, Magic Fusion Systems, Inc., and Pacific Bell Advanced Technology Group. He has published more than 50 technical papers and is the author of two book chapters published by IEEE Press and World Scientific. He has several patents awarded/pending on ATM switch designs, gigabit router designs, high performance multicast, traffic management, and QOS controls.

Dr. Lin was the recipient of two Best Paper awards.