AN ADAPTIVE FLOW CLASSIFICATION ALGORITHM FOR IP SWITCHING

Jun Zheng, Victor O. K. Li, Fellow, IEEE, and Xiaochun Yuan

Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: {jzheng, vli, xcyuan}@eee.hku.hk

Abstract

Flow classification is one of the key issues in IP Switching. To achieve better performance, flow classification should be matched to the varying IP traffic and an IP switch should make use of its hardware switching resources as fully as possible. Based on these arguments, this paper proposes an adaptive flow classification algorithm for IP Switching. By dynamically adjusting the values of its control parameters in response to the present usage of the hardware switching resources, this adaptive algorithm can efficiently match the varying IP traffic and thus improve the performance of an IP switch. Simulation study based on real IP traces is performed to demonstrate the efficiency of this adaptive algorithm and to show the improvement of the system performance.

I. Introduction

IP Switching [1] has emerged as an efficient routing technology for improving the performance of conventional IP routers. An IP switch must perform flow classification to decide whether a flow should be switched directly in the ATM hardware or forwarded hop-by-hop by the routing software. This is implemented by inspecting the values of the packet header fields and making a decision based upon a local policy. Generally, long-lived flows with a large number of packets should be selected for ATM switching while short-lived flows with a small number of packets should be handled by normal hop-by-hop forwarding [1]. For this purpose, flow classification should be able to effectively classify incoming flows into short-lived flows and long-lived flows. To achieve better performance, a variety of flow classification policies/algorithms have been proposed [1][2][3]. Most of them are static in that their criteria or control parameters are set statically. However, IP traffic is highly varying, which makes it difficult to predict the traffic characteristics exactly and thus set optimal criteria or control parameters. For this reason, a static flow classification policy/algorithm may be unable to match the varying IP traffic and achieve optimal performance. To further improve the performance, flow classification should take into account the varying characteristics of IP traffic. In addition, an IP switch should make use of its hardware switching resources as fully as possible for better performance. Based on these arguments, this paper proposes an adaptive flow classification algorithm for IP switching. By dynamically adjusting the values of its control parameters in response to the present usage of the hardware switching resources, this adaptive algorithm can efficiently match the varying IP traffic and thus improve the performance of an IP switch.

The remainder of this paper is organized as follows. In section II, we give a brief review of static flow classification policies/algorithms currently available. In section III, we present the proposed adaptive flow classification algorithm. In section IV, we describe our simulation study and give some numerical results to show the improvement of the system performance with this adaptive algorithm. In section V, we conclude our study.

II. Static Flow Classification Policy/Algorithm

One flow classification policy, the protocol-based policy [2], is to simply classify flows by protocols. With this policy, all TCP flows are selected for ATM switching while all UDP flows are forwarded hop-by-hop by the routing software. The argument is that connection-oriented services generally last longer and have more packets to be sent over a short time than connectionless services. Similarly, flows can also be classified by applications, such as ftp, smtp and http [2]. This application-based policy is dependent on the statistical measurement of the average duration and the average number of packets of each flow. Only those applications that tend to generate long-lived flows and contain a large number of packets are selected for ATM switching. However, an analysis of the FIXWEST backbone traces in [3] showed a great diversity of flow duration and packet numbers for each flow in some common applications such as http and DNS. Both policies are not effective enough in separating short-lived flows from long-lived flows.

Another flow classification algorithm [1] is to count the number of packets received on each flow so that the first X packets of each flow are always forwarded by the routing software while further packets are switched in the ATM

0-7803-5796-5/99/$10.00 © 1999 IEEE

hardware, independent of protocols and applications. A modified version is the X/Y algorithm [2]. This algorithm also counts the number of packets received on each flow. If the number of packets of a flow received within Y seconds exceeds X, further packets of the flow will be switched in the ATM hardware. The basis behind these two algorithms is that if there have already been X packets received on a flow (within Y seconds), it is reasonable to expect that there will be more packets on this flow. The simulation studies in [1] and [2] showed that both algorithms could more effectively separate short-lived flows from long-lived flows and give superior performance over the protocol-based and the application-based policies.

III. Adaptive Flow Classification Algorithm

All the flow classification policies/algorithms discussed in Section II are static. To match the varying characteristics of IP traffic, Hao Che *et al.* introduced a new concept of adaptive flow classification and proposed an adaptive flow classification algorithm in [3]. However, this algorithm aims at adaptive resource management and balanced utilization of the software forwarding resources and the hardware switching resources, which may not be the best policy for the performance improvement of an IP switch, especially under light traffic load. Imagine the case in which the traffic load is light and most of the incoming flows are long-lived flows. It is obvious that balanced utilization of the software forwarding resources and the hardware switching resources can not achieve optimal performance. The goal of IP Switching is to improve the forwarding performance of conventional IP routers by exploiting the ATM hardware. From this viewpoint, we argue that the hardware switching resources of the ATM switch should be made use of as fully as possible. On the other hand, IP Switching itself provides a mechanism that could automatically adjust the utilization of the two different kinds of resources. For example, a flow that is selected for ATM switching but unable to get a virtual circuit identifier (VCI) will still be forwarded by the routing software. No flow or packet losses will be caused by the unavailability of VCIs as long as the software forwarding resources are not overloaded. Therefore, there is no need to balance the utilization of the software and hardware resources in an IP switch. Rather, it is important to make full use of the hardware switching resources to offer better performance for IP traffic.

As shown in [2], the virtual circuit (VC) space of an IP switch is an important hardware switching resource that has a great impact on the system performance. When a flow is selected for ATM switching, the IP switch must request a VCI from its VC space so that the flow can be switched directly in the ATM hardware. Obviously, a larger VC space will allow a larger number of flows to be switched in the ATM hardware and thus produce better performance. However, since the VC space of an IP switch is constrained, a flow that is selected for ATM switching might be blocked at the VC space and thus could not be switched in the ATM hardware. To achieve better performance, it is desirable to select as many flows as possible for ATM switching. On the other hand, if too many flows are selected for ATM switching, those flows that last comparatively long might be blocked at the VC space by short-lived flows, which is not desirable. For these reasons, a flow classification algorithm should be able to dynamically adjust the values of its control parameters so as to match the varying IP traffic. For this purpose, we now propose another adaptive flow classification algorithm.

The proposed algorithm is based on the static X/Y algorithm, where Y stands for the time interval in seconds and X is the number of packets received in Y seconds. The objective is to make use of the VC space as much as possible and meanwhile to offer better performance for long-lived flows. Here we only focus on the VC space and assume that other system resources are relatively less constrained. To match the varying IP traffic, it is desirable to dynamically adjust the control parameters (X and Y) of the static X/Y algorithm in response to the present VC usage. The simulation study in [2] has shown the impacts of X and Y on the VC usage. The VC usage exhibits a monotone property with respect to X and Y, as shown in Table 1. By increasing X and/or decreasing Y, the VC usage may be decreased. Accordingly, we can use this monotone property to dynamically control the VC usage. The principle is that, when the VC usage is small, X (or Y) may be properly decreased (or increased) so that more flows can be switched in the ATM hardware. On the contrary, when the VC usage is overloaded, X (or Y) may be properly increased (or decreased) so that those flows that last comparatively longer and contain a larger number of packets will not be blocked at the VC space and thus be switched in the ATM hardware. To implement this, the algorithm should periodically measure the VC usage and then decide how to make the adjustments based on a target function.

Table 1: The monotone property of the VC usage with respect to X and Y

	$X \uparrow$	$X \downarrow$	$Y \uparrow$	$Y \downarrow$
C_n	$\downarrow$	$\uparrow$	$\uparrow$	$\downarrow$

Assume that the adjustments are periodically performed at discrete time n. Without loss of generality, n may take integer values here (i.e. $n = 0, 1, 2, \cdots$) while in reality the adjustments may take place at a fixed time interval. To

make full use of the VC space, the VC usage at time n should be as large as possible. However, it should not be equal to the size of the VC space as in this case blocking will generally occur at the VC space and those flows that last comparatively longer and contain a larger number of packets may be unable to be switched in the ATM hardware. For this reason, we define the target function or the target region as follows

$$C_t < C_n < C_m$$

where C_n is the VC usage at time n, C_m is the size of the VC space, which we assume to be fixed here, and C_t is a threshold that is close to C_m. Now our adjustment objective is to drive the VC usage to converge to the target region. Let X_n and Y_n be the values of X and Y at time n, respectively. Obviously, both of them must be positive values. Then we have

$$X_{n+1} = X_n + \Delta X_n \quad \text{and} \quad Y_{n+1} = Y_n + \Delta Y_n$$

where ΔX_n and ΔY_n are the adjustment step sizes at time n. The adjustment strategy can be summarized in Table 2, where "0" is for no change, "↑" for increment and "↓" for decrement. Theoretically, the adjustments may be made not only by changing X_n or Y_n but also by changing both of them. However, as is shown in [2], by restricting the adjustments to X_n only, we can produce the effect of allowing adjustments to both X_n and Y_n. Now the remaining problem is how to determine the adjustment step size at each adjustment time.

Table 2 Adjustment strategy of X and Y

	$(0 \le C_n \le C_t)$	$(C_t < C_n < C_m)$	$C_n = C_m$
(X_n, Y_n)	(↓, ↑)	(0, 0)	(↑, ↓)

One simple method is to assign small fixed values to ΔX_n and ΔY_n, independent of the present usage of the VC space [3]. For example, ΔX_n may take values 1 for "↑" and −1 for "↓" as long as $X_n \ge 0$. With this approach, the convergence may be quite slow when the VC usage is small. To achieve better performance, it is desirable that the VC usage converges to the target region as quickly as possible. For this reason, a larger value of ΔX_n or/and ΔY_n is desired in the case of smaller VC usage. When the VC usage gets closer to the target region, a smaller value of ΔX_n or/and ΔY_n is preferred so that C_n will not be over-adjusted. Therefore, we may adjust ΔX_n and ΔY_n based on the difference (L_n) between the present VC usage (C_n) and the desired VC usage (C_d).

For example, we may define the desired VC usage as the middle point of the target region, i.e.

$$C_d = \frac{(C_t + C_m)}{2}.$$

Accordingly, we have

$$L_n = (C_n - C_d)$$

Then the adjustment step size can be determined as follows

$$\Delta X_n = \begin{cases} 0 & |C_n - C_d| < C_m - C_d \\ -I[\frac{|L_n|}{L_0} * \Delta X_0] & C_d - C_n \ge C_m - C_d \\ I[\frac{|L_n|}{L_0} * \Delta X_0] & C_n - C_d \ge C_m - C_d \end{cases}$$

$$\Delta Y_n = \begin{cases} 0 & |C_n - C_d| < C_m - C_d \\ -I[\frac{|L_n|}{L_0} * \Delta Y_0] & C_d - C_n \ge C_m - C_d \\ I[\frac{|L_n|}{L_0} * \Delta Y_0] & C_n - C_d \ge C_m - C_d \end{cases}$$

where ΔX_0 and ΔY_0 are the initial values of the adjustment step sizes and $I[\omega]$ is the integer portion of ω. The relationship of the relevant parameters is shown in Fig. 3.1. Since this method adjusts (X,Y) linearly in proportion to the difference between the present VC usage and the desired VC usage, we call it the linear method.

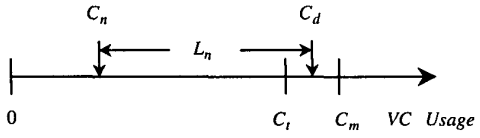

Fig. 3.1 Relationship of the relevant parameters

IV. Simulation Study

In this section, we give some simulation results to demonstrate the efficiency of our proposed adaptive algorithm, to compare the convergence of the simple adjustment method and the linear method, and to show the improvement of the system performance. The simulation is based on the real FIXWEST traces obtained from the National Laboratory for Applied Network Research (NLANR)[1]. Since the duration of each trace is limited, we

[1] FIXWEST/NLANR traces are available at http: //moat.nlanr.net/traces/traces

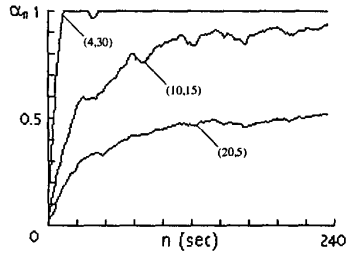

Fig. 4.1 Impact of the initial values of (X,Y) on the VC usage for the static X/Y algorithm

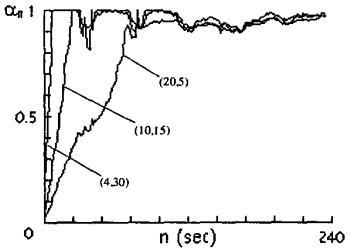

Fig. 4.2 Convergence of the VC usage for the unbalanced adaptive algorithm

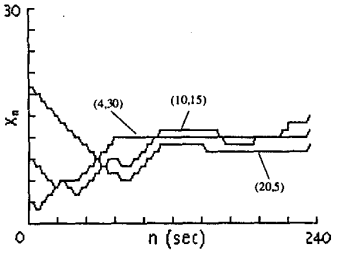

Fig. 4.3 Adjustment of X

only give the simulation results in 240 seconds, which is long enough for our purpose. Without loss of generality, we only consider host+port flow granularity [2] and focus on one single IP switch input port. The adjustment time interval is taken to be 5 seconds. The flow delete time is set at 30 seconds, which is the holding time of an active VC. If the interarrival time between two adjacent packets of a flow exceeds the flow delete time, the corresponding VC will be removed and thus the flow will be terminated. To simulate a highly non-stationary traffic load, we assume that the initial VC usage is 0. In addition, we define $\alpha_n = \frac{C_n}{C_m}$ as the utilization ratio of the VC space and assume that $C_m = 2000$ *flows* and $C_t = 90\% C_m$.

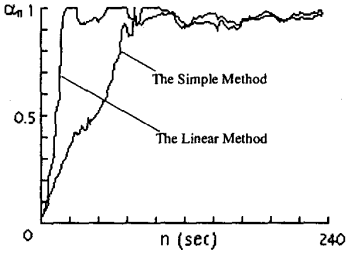

Fig. 4.4 Impact of the adjustment methods on the convergence of the VC usage

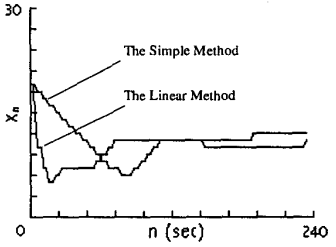

Fig. 4.5 Adjustment of X

First, we investigate the impacts of different initial values of (X,Y) on the VC usage for the static X/Y algorithm, as shown in Fig. 4.1. For the different sets of (X_0,Y_0), the utilization ratio of the VC space tends to stabilize at different values after a transient duration. For (4,30), since X_0 is very small and Y_0 is very large, a large number of flows are selected for ATM switching, which makes the VC usage increase quickly and the utilization ratio soon become 1. This means that, in this case, there are not enough VCIs available to the selected flows and blocking will occur at the VC space. For this reason, those flows that last comparatively longer and contain a larger number of packets may be blocked by shorter-lived flows and thus are not switched in the ATM hardware, which is not desirable. For (20,5), since X_0 is very large and Y_0 is very small, only a small number of flows are selected for ATM switching. Accordingly, the VC space is not fully used and the utilization ratio of the VC space is very small, which is also not desirable. However, (10,15) is quite suitable for the FIXWEST trace used in terms of both the VC utilization ratio and the performance for long-lived flows.

The effect that our proposed adaptive algorithm produces on the VC usage is shown in Fig. 4.2. Since the simulation study in [3] has shown that the VC usage is much more sensitive to X than to Y, we only use X as the control parameter and keep Y constant in the simulation.

From Fig. 4.2, we can see that no matter what the initial values of (X,Y) are, the VC usage can converge to the target region efficiently. The VC utilization ratio keeps high within the target region and blocking rarely occurs. This means that our proposed algorithm can efficiently match the varying characteristics of IP traffic. Fig. 4.3 gives the adjustment of X in the simulation. No matter how large the initial value of X is, X_n approaches stability after a transient duration, which is due to the relatively smooth FIXWEST trace.

The comparison of the simple method and the linear method for (20,5) is shown in Fig. 4.4. Obviously, with the linear method, the VC usage can converge to the target region more quickly. Fig. 4.5 shows the adjustment of X during the simulation period.

Table 3 Statistics of simulation

(X,Y)	Total packets	Packets switched	Percentage of packets switched
(4, 30)	788587	528423 (S) 537991 (A)	67.00% (S) 68.22% (A)
(10,15)	788587	526224 (S) 532206 (A)	66.70% (S) 67.50% (A)
(20, 5)	788587	491647 (S) 534802 (A)	62.35% (S) 67.80% (A)

S: The static X/Y algorithm
A: The proposed adaptive algorithm

Table 3 gives some statistics from the simulation. The total number of packets delivered is 788587. Here we assumed that those flows that are not selected for ATM switching could all be forwarded by the routing software. It can be seen that, no matter what the initial values (X_0,Y_0) are, the number of packets switched or the percentage of packets switched can efficiently be increased. The larger the X_0, the higher the percentage of packets switched. For the case of (4,30), the result is particularly important. Even though the VC utilization ratio for the static X/Y algorithm is larger than that for the proposed adaptive algorithm most of the time, the number of packets switched for the static X/Y algorithm is less than that for the proposed adaptive algorithm. This is because, with the proposed algorithm, the flows that are selected for ATM switching contain more long-lived ones than those with the static X/Y algorithm. Even though the total number of flows switched is a little bit decreased, the number of packets switched is still increased. This important result shows that, with our proposed adaptive algorithm, both the performance for long-lived flows and the overall performance of an IP switch in terms of the number of packets switched can be improved efficiently.

V. Conclusion

In this paper, we proposed an adaptive flow classification algorithm for IP switching. Based on the static X/Y algorithm, this algorithm can dynamically adjust the values of the control parameters (X or Y) in response to the present VC usage. The simulation study based on the real FIXWEST traces showed that it allowed the VC usage to converge to a target region efficiently and could thus match the varying IP traffic. The simulation results also showed that both the performance for long-lived flows and the overall performance of an IP switch in terms of the number of packets switched could be improved. It should be noted that this algorithm only focused on the VC space and assumed that other system resources are relatively less constrained. In reality, however, there may be other resources that are also constrained and have a great impact on the system performance, such as the circuit setup rate [4]. For this reason, further study is under way to improve this adaptive flow classification algorithm.

Acknowledgement

This research is supported in part by the University of Hong Kong Area of Fundamentals in Information Technology.

References

[1] Peter Newman, Greg Minshall and Thomas L. Lyon, "IP Switching—ATM under IP," IEEE/ACM Transactions on Networking, Vol. 6, No. 2, April 1998, pp. 117-129.

[2] Steven Lin and Nick McKeown, "A Simulation Study of IP Switching," ACM SIGCOMM' 97, pp.15-24.

[3] Hao Che, San-qi Li and Arthur Lin, "Adaptive Resource Management for Flow-Based IP/ATM Hybrid Switching Systems," IEEE/ACM Transactions on Networking, Vol. 6, No. 5, October 1998, pp.544-557.

[4] Bruce Davie, Paul Doolan and Yakov Rekhter, "Switching in IP Networks," Morgan Kaufmann Publishers, 1998.