Transport-Aware IP Routers: A Built-In Protection Mechanism to Counter DDoS Attacks

Haining Wang, *Student Member*, *IEEE*, and Kang G. Shin, *Fellow*, *IEEE*

Abstract—The lack of service differentiation and resource isolation by current IP routers exposes their vulnerability to Distributed Denial of Service (DDoS) attacks [12], causing a serious threat to the availability of Internet services. Based on the concept of layer-4 service differentiation and resource isolation, where the transport-layer information is inferred from the IP headers and used for packet classification and resource management, we present a transport-aware IP (tIP) router architecture that provides fine-grained service differentiation and resource isolation among different classes of traffic aggregates. The tIP router architecture consists of a fine-grained Quality-of-Service (QoS) classifier and an adaptive weight-based resource manager. A two-stage packet-classification mechanism is devised to decouple the fine-grained QoS lookup from the usual routing lookup at core routers. The fine-grained service differentiation and resource isolation provided inside the tIP router is a powerful built-in protection mechanism to counter DDoS attacks, reducing the vulnerability of Internet to DDoS attacks. Moreover, the tIP architecture is stateless and compatible with the Differentiated Service (DiffServ) infrastructure. Thanks to its scalable QoS support for TCP control segments, the tIP router supports bidirectional differentiated services for TCP sessions.

Index Terms—Distributed Denial of Service (DDoS) attacks, layer-4 differentiation, resource isolation, packet classification.

1 Introduction

Denial of Service (DoS) attacks in the Internet are well-known to be difficult to defend against, and the recent occurrence of Distributed DoS (DDoS) attacks make it even more difficult for victims to block and trace back the attacking sources. To thwart DDoS attacks and provide service differentiation at victim servers, sophisticated resource management schemes at end-servers have been proposed, such as Resource Containers, WebQoS, and Escort [1], [2], [6], [23], [37]. Like the computing resources of an end-server, there is only a limited amount of network resources—such as bandwidth, buffer, and processing power of routers—available to Internet users. The vulnerability of the Internet to DDoS attacks roots in its best-effort model that provides no resource isolation among different IP flows, making it easy for attacking traffic to hog network resources. Having sufficient service differentiation and resource isolation at IP routers is essential not only to provide network Quality of Service (QoS) to end-users, but also to counter DDoS attacks as a powerful built-in protection mechanism inside the Internet.

To support network QoS, the Differentiated Service (DiffServ) infrastructure [5] has been proposed as a promising solution due mainly to its scalability and robustness. Based on the DS field in the IP header, IP flows are classified into different Behavior Aggregates (BAs). Services are provided for aggregates, not for individual flows, and defined by a small set of Per-Hop Behaviors (PHBs), which are the forwarding behaviors applied to different aggregates at IP routers. According to the three different services provided by DiffServ, three types of PHBs are specified: *Expedited Forwarding* (EF), *Assured Forwarding* (AF), and *Best-Effort* (BE). Although EF traffic is strictly policed and conditioned at edge routers, which protects the rest of the network resources from the flooding EF traffic, the violated AF traffic is only remarked without strict policing at network edges. More importantly, no conditioning is applied to BE traffic, which is the main component of the Internet traffic. Compared with the best-effort service model, DiffServ is more resilient against DDoS attacks, but it is still susceptible to DDoS attacks, especially the BE flooding traffic.

In the current DiffServ architecture, the QoS classification at core routers depends solely upon the DS field in the IP header,[1] yielding only coarse-grained service differentiation and resource isolation. No further service differentiation and resource isolation are provided among different transport-layer protocols within a BA. On the other hand, UDP and TCP are two dominant transport-layer protocols in the current Internet, but their services and traffic behaviors are quite different. Furthermore, UDP and ICMP flooding attacks have been widely used for stealing network bandwidth and disabling a victim server. It is necessary to provide resource isolation among TCP, UDP, and ICMP traffic and the resource consumption of UDP and ICMP traffic should be bounded. Besides meeting the requirement of the bidirectional service differentiation to TCP sessions, which the current DiffServ fails to achieve [28], [45], there

• *The authors are with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122.*
E-mail: {hxw, kgshin}@eecs.umich.edu.

Manuscript received 20 Aug. 2002; accepted 15 May 2003.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 118641.

1. Multifield packet classification is limited to edge routers in the DiffServ architecture.

are three reasons for differentiating TCP control segments—SYNs, FINs, ACKs, and RSTs—from data segments, especially in the best-effort service model.

R1. Usually, TCP control segments have much smaller packet size than data segments, so they consume much less network bandwidth than data segments.

R2. The loss of a TCP control segment, especially SYNs, incurs more serious performance degradation than the loss of a data segment.

R3. DDoS attack tools usually utilize TCP control segments for generation of DoS attacks, such as TCP SYN and ACK flooding attacks.

In other words, the coarse-grained service differentiation and the lack of resource isolation on metadata packets not only degrade the assured service of TCP sessions, but also expose the vulnerability of Internet to DDoS attacks [12].

In this paper, we propose a transport-aware IP (tIP) router architecture to provide fine-grained service differentiation and resource isolation among thinner aggregates without compromising scalability. The basic concept employed is layer-4 service differentiation and resource isolation in which the transport-layer information is inferred from the IP headers and used for packet classification and resource management at IP routers. To support layer-4 service differentiation and resource isolation, we present a fine-grained QoS classifier and an adaptive weight-based resource manager with which the tIP router infrastructure is built. The fine-grained QoS classifier divides each BA into thinner aggregates and the adaptive weight-based resource manager provides service differentiation and resource isolation among these thinner aggregates. At core routers, we employ a two-stage packet classification mechanism to decouple the routing lookup from QoS lookup. The first stage performs the routing lookup at the input port and the second stage performs the fine-grained QoS lookup at the output port after the packet is routed through the switching fabric.

The performance of the tIP router architecture is evaluated by simulation. The simulation results show that 1) the resource isolation provided by the tIP router significantly throttles the flooding traffic received by the victim server, 2) most of the flooding traffic is dropped close to the attacking sources, thus confining flooding damage and saving network resources, and 3) the flooding traffic has little impact on the normal traffic that belongs to a different transport protocol, e.g., the UDP flooding or ICMP flooding traffic cannot interfere with the transmission of normal TCP traffic. Therefore, it can be utilized as a built-in protection mechanism of IP routers to counter DDoS attacks. Moreover, the tIP router provides better end-to-end TCP performance to applications: A high-tiered TCP session is guaranteed to have lower ACK loss rate and higher effective throughput than a low-tiered one.

Note that tIP routers do not require the support of DiffServ infrastructure and they also work under the best-effort model. Actually, the best-effort model can be viewed as a simplified case of the DiffServ model in which only single BA (best-effort) exists. To make our work more general and applicable in the future, we investigate tIP routers with a more sophisticated DiffServ model. The subset of our results, which is based on the study of best-effort aggregates, can be directly applied to the current Internet.

While some network attackers attempt to subvert the service of a victim using a few specially-crafted packets, most of DDoS attacks are conducted by flooding a large number of bogus packets to the victim. In this paper, only the flooding DDoS attacks are considered. The remainder of the paper is organized as follows: Section 2 presents the background of our work, including our initial motivation, the brief description of DDoS attacks, and the related work. Section 3 describes a fine-grained QoS classifier and an adaptive weight-based resource manager, the key components of the tIP router architecture. Section 4 evaluates the performance of the tIP router architecture. Finally, the paper concludes with Section 5.

2 Background

There are three parts of background behind our work. First, we present our initial motivation. Second, we briefly describe the working mechanism of DDoS attacks and some available attack tools. Third, we give a short survey of previous router-based DDoS defense mechanisms.

2.1 Initial Motivation

Our work was initially motivated by the desire of providing preferential treatments to TCP ACKs and achieving bidirectional differentiated service for each TCP session. By default, the current DiffServ architecture treats TCP ACKs from different user classes as BE traffic, sharing the same FIFO queue with BE data packets. The congestion caused by BE data packets results in buffer overflow at routers and, hence, bursty ACK losses. Especially, under the condition that the TCP sender's congestion window size is small, the TCP sender is more vulnerable to ACK losses in the backward path and its data transmission is interrupted by retransmission timeouts. Therefore, providing service differentiation for ACK flows is essential to TCP-based applications.

One simple way to achieve this is that end-hosts mark each ACK as a premium, assured, or best-effort packet, corresponding to the class of the data packet being acknowledged, but no enhanced mechanisms implemented at IP routers to distinguish ACKs from data segments. If the network resources are overprovisioned, the validity of this simple marking scheme has been confirmed by the authors of [28]. However, without proper resource provisioning and traffic conditioning for ACK aggregates, the ACKs and data segments that share the same queue could interfere with each other. Our simulation results in Section 4.3 show that the simple ACK marking scheme provides insufficient service differentiation and isolation to ACK flows when network resources are under-provisioned.

Furthermore, there are two serious drawbacks with this simple marking scheme: 1) The best-effort TCP traffic, which will continue to be the dominant load in the Internet, will not receive any performance improvement with the simple marking scheme; and 2) DDoS attacks in the Internet make the simple marking scheme much less attractive since it is more vulnerable to various TCP flooding attacks. The

simulation results in Section 4.2 confirm this claim. Therefore, to achieve better network QoS and counter DDoS attacks, we need to differentiate the TCP control segments from data segments and provide resource isolation between them at IP routers.

In the previous work [3], in order to improve the TCP performance in the context of network asymmetry, the *acks-first* scheduling scheme has been proposed, giving TCP ACKs priority over TCP data segments. So, the router always forwards ACKs before data segments. However, this *acks-first* scheme could cause starvation of data packets and violation of traffic profiles, especially under ACK flooding attacks. Also, no ACK identification scheme at routers was provided there.

This initial motivation makes our very different from the previous work for countering DDoS attacks because it is not a pure security mechanism. It can significantly improve the end-to-end TCP performance for clients, justifying the need for the wide deployment of *t*IP router architecture.

2.2 Description of DDoS Attacks

A DDoS attack system can usually be described as a hierarchical model in which an attacker controls a handler (master) that, in turn, dictates the hordes of agents (slaves) to flood the bogus packets to the victim. The communication between the attacker and the handler and between the handler and the agents is called the *control traffic*, while the communication between the agents and the victim is called the *flooding traffic*. To hide the attacker from its detection, the control traffic is encrypted and its channel is covert and password protected. The recruitment of agents is achieved by employing automatic scanning and propagation techniques, which search for security holes and inject the attacking instructions into the subverted machines.

The mechanism of DDoS attacks works as follows: The master sends control packets to the previously compromised slaves, instructing them to target a given victim. The slaves then generate and send high-volume streams of flooding messages to the victim, but with fake or randomized source addresses so that the victim cannot locate the attackers. DDoS attacks are difficult to defend against, as they do not target specific vulnerabilities of a computer system, but rather, the very fact that it is connected to the network. With the appearance of Trinoo, which only implements UDP flooding attacks, many tools have been developed to create DDoS attacks. These readily available attacking tools, such as Tribe Flood Network (TFN), TFN2K, Trinity, Plague, and Shaft, generate various flooding attacks [9].

While the conventional flooding attack is a system resource consumption attack, the recent Distributed Reflection DoS (DRDoS) attacks [14], [30] virtually "disconnect" a victim server from the Internet by hogging the link bandwidth between the victim and its ISP with an excessive number of response packets (also known as. bandwidth consumption attack). The DRDoS attacks masquerade the source IP address of each spoofed request with the victim's IP address and spray the spoofed requests to a large number of Internet servers, which are exploited as reflectors. The reflectors will then send combined replies to the victim, redirecting and amplifying the flooding traffic.

2.3 Related Work

As the observed prevalence of DDoS attacks in the Internet [24], many router-based defense mechanisms have been proposed, including router filtering [11], [29], router throttling [13], [46], Pushback [19], [17], traceback [4], [33], [35], [36], [40], and detection mechanisms [15], [29], [43]. However, most of them are used solely for security purpose. The Internet Service Providers (ISPs) usually do not have strong incentive to embed these security mechanisms into their routers since no direct benefit is brought to their own clients.

Ingress filtering [11] configures the internal router interface to block spoofed packets whose source IP addresses do not belong to the stub network. This limits the ability to flood spoofed packets from that stub network since the attacker would only be able to generate bogus traffic with internal addresses. Given the reachability constraints imposed by the routing and network topology, the route-based distributed packet filtering (DPF) [29] exploits routing information to determine if a packet arriving at the router is valid with respect to its inscribed source/destination addresses. The experimental results reported in [29] show that a significant fraction of spoofed packets are filtered out and the spoofed packets that escaped the filtering can be localized into five candidate sites which are easy to trace back.

Yau et al. proposed a *router throttle* mechanism [46] which is installed at the routers that are close to the victim. These routers proactively regulate the incoming packets to a moderate level, thus reducing the amount of the flooding traffic toward the victim. The key idea of pushback is close to that of router throttle, and it identifies and controls high bandwidth aggregates in network [17], [19]. The router could ask adjacent upstream routers to limit the amount of traffic from the identified aggregate. This upstream rate-limiting is called pushback and can be propagated recursively to routers further upstream.

Since the source addresses of flooding packets are faked, various traceback techniques [4], [33], [35], [36], [40] have been proposed to find out the origin of a flooding source. Probabilistic packet marking [33] was proposed to reduce the tracing overhead at IP routers, which was refined by Song and Perrig in the reconstruction of paths and the authentication of encodings. Snoeren et al. presented a hash-based IP traceback, which can track the origin of a single packet delivered by the network in an efficient and scalable way. Stone [40] built an IP overlay network for tracking DoS floods, which consists of IP tunnels connecting all edge routers. The topology of this overlay network is deliberately simple and suspicious flows can be dynamically rerouted across the tracking overlay network for analysis. Then, the origin of the floods can be revealed.

A data-structure called MULTOPS [15] is a tree of nodes that keeps packet-rate statistics for subnets at different aggregation levels. Based on the observation of a significant disproportional difference between the traffic flowing into and out of the victim, routers use MULTOPS to detect ongoing bandwidth attacks. To detect TCP SYN flooding attacks, a simple and robust detection mechanism has been proposed [43]. Based on the distinct protocol behavior of

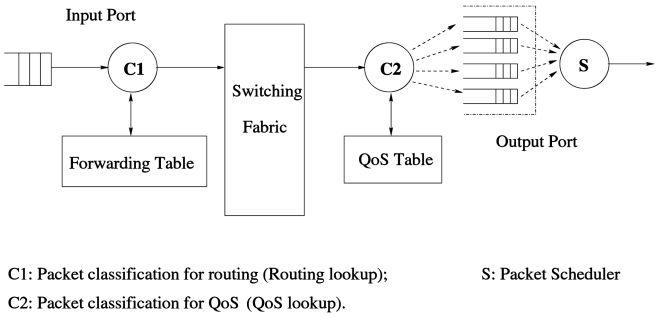

Fig. 1. The architecture of the two-stage packet classifier at core routers.

TCP connection establishment and teardown, the SYN flooding detection is treated as an instance of the Sequential Change Point Detection. A nonparametric Cumulative Sum method is applied, making the detection mechanism insensitive to sites and access pattern.

3 Transport-Aware IP Router Architecture

To provide layer-4 service differentiation and resource isolation, we propose a fine-grained QoS classifier and an adaptive weight-based resource manager, both of which are essential to the *t*IP router architecture. The granularity of the classifier is still based on aggregates, not individual flows, and the resource manager is stateless, thus preserving the scalability and robustness of the original DiffServ or current Internet infrastructure. Moreover, at core routers, the QoS lookup is decoupled from the routing lookup by employing a two-stage packet classification mechanism. The set of QoS filtering rules is small and has the same/similar size at both edge and core routers. In contrast to routing lookup, QoS lookup is independent of network size and does not cause any scalability problem in packet classification. The *t*IP router architecture is detailed in the remainder of this section.

3.1 Two-Stage Packet Classification

For service differentiation, layer-4 switching [20], [38] has been proposed, in which routing decisions are made based on the destination address as well as on the header fields at the transport or higher layer. Routing and QoS lookups are integrated into a single framework to fulfill layer-4 switching and, therefore, the forwarding database of a router consists of a large number of filters to be applied on multiple header fields. The deployment of a large-scale packet filtering mechanism [16], [18], [38] makes it feasible to implement layer-4 or higher switching at edge routers or at the front-end of server farms. However, layer-4 switching has primarily been used for load balancing by connection routers in server farms. It is very difficult to implement layer-4 switching at core routers due mainly to security and scalability difficulties. Even with fast and scalable packet classification, the problems with layer-4 switching at core routers are: 1) addition of higher-layer information—such as port numbers—and more routing entries enlarges the routing table at core routers, causing the routing lookup to require more memory and time, and 2) when IP payload is encrypted, higher-layer headers become inaccessible.

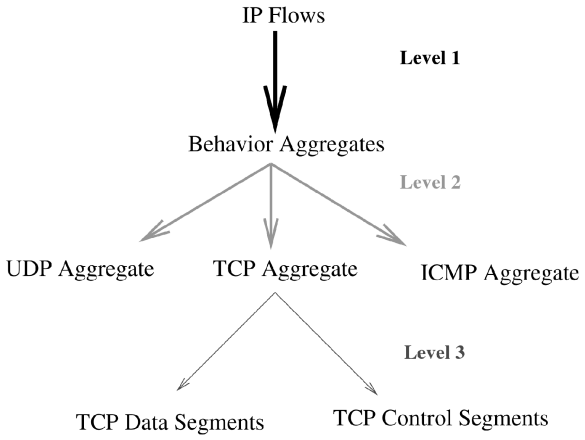

Fig. 2. The three-level QoS classification.

To support layer-4 service differentiation and resource isolation, the fine-grained QoS classification has to work well at both edge and core routers. Therefore, we decouple the fine-grained QoS lookup from the routing lookup at core routers by employing a two-stage packet classification mechanism. In addition to overcoming the scalability problem at core routers, there are several other reasons for this decoupling as follows:

- Routing decisions must be made at the input port, but most of service differentiation—buffer management and packet scheduling—is performed at the output port.
- There is a large difference between the search spaces of routing lookup and QoS lookup. The size of the routing table is very large and ever-increasing with the growth of the Internet, but the filtering rule set of QoS classification is small and remains stable.
- Conventional routing lookup is based solely on destination addresses, which is a one-dimensional search, but QoS lookup is based on multiple fields, which is a multidimensional search.

Note that, at core routers, our QoS lookup is restricted to the IP header fields and the transport-layer information is accessed only at edge routers if necessary. The proposed QoS lookup should not become the performance bottleneck inside the router. Moreover, the decoupling greatly simplifies the implementation of packet classification. The architecture of the two-stage packet classification is illustrated in Fig. 1, where the forwarding table is the local version of routing table in the line card. With the forwarding table, the routing/switching decision can be made locally at each input port.

3.2 Fine-Grained QoS Classifier

As the key component of the *t*IP router architecture, the proposed QoS classifier at routers uses several fields in the IP header for QoS classification, in addition to the DS field. Transport-layer information is extracted to further divide a BA into a UDP aggregate, a TCP aggregate, and an ICMP aggregate and, then, to distinguish TCP control segments that mainly consist of ACKs from TCP data segments in the TCP aggregate. QoS classification can be modeled at three different hierarchical levels, as shown in Fig. 2.

At the first two levels, it is straightforward to set the filtering rules. By checking DS and *protocol type* fields in the IP header against the filtering rules, the QoS classification is simple and does not cause any ambiguity. However, the accurate TCP control segment identification at level 3 could be much more complex. To implement the identification of TCP control segments at IP routers, the easiest way is to utilize one of the unused bits of DS field in the IP header. However, the problem with this solution is that it requires the modification of the IP header and cooperation from end-hosts. Moreover, the IETF has proposed the use of the two unused bits of DS field for the deployment of the Explicit Congestion Notification (ECN) mechanism [31] at routers.

So, we propose size and port-based identification of TCP control segments without requiring any new bit. Note that we present below a detailed description of a general TCP control segment identification which can be applied to the identification of each individual TCP control segment like SYN, FIN, ACK, and RST. The only difference at the port-based identification is to check different bits in the 6-bit flag field.

3.2.1 TCP Control Segment Identification

Initially, each TCP segment is encapsulated in a single IP packet, but IP fragmentation could occur at intermediate routers. Once the IP packet is fragmented, only the first fragment contains the TCP header. So, the IP packet that contains the TCP header must have a zero fragmentation offset. By checking the *fragmentation offset* field in the IP header, the ambiguity caused by IP fragmentation is eliminated. Theoretically, if an IP packet encapsulates a TCP control segment, it should meet the following requirements: 1) Its protocol type is TCP, 2) its fragmentation offset is zero, and 3) the corresponding flags in the TCP header are ON. The first two conditions can be verified by checking the IP header, but the validation of the third condition needs to access the TCP header. Then, based on whether the TCP header is accessed or not, we have two versions of TCP control segment identification: *lightweight* and *heavyweight*.

Recent Internet traffic measurements [8], [41] have shown that about 40 percent of all IP packets are 40 bytes long, most of which are TCP control segments, implying that an overwhelming majority of TCP control segments are 40 bytes long. Since IP options are included primarily for network testing or debugging and the fraction of packets with IP header options is typically less than 0.003 percent [21], it is reasonable to assume that no IP option fields are attached to TCP control segments.

3.2.2 ACK Identification

The lightweight version is a size-based classifier. It takes advantage of the above observations. By checking the *total length* field in the IP header, it can tell TCP control segments from data segments without accessing the TCP header. The base filtering rule of TCP control segment identification is that the TCP segments whose total length are 40 bytes are classified as TCP control segments. The rationale behind this is that, since the IP header without options is 20 bytes and the TCP header without options is 20 bytes, a total of 40 bytes is the minimum size of an IP packet that encapsulated a TCP segment (without any payload). Considering TCP options—MSS option (4 bytes), Window scale factor option (3 bytes), Timestamp option (10 bytes), and Selective Acknowledgment option (10/18/26 bytes)—that can appear in the TCP control segments like SYNs or ACKs and the requirement of a 4-byte boundary by padding NOOP options, the complete filtering rule of TCP control segment identification is set as:

$$Rule : \{\ X \mid X = 4 \cdot n;\ 10 \leq n \leq 20\},$$

where X is the total length of an IP packet and n is an integer. Since the maximum space that TCP options can use is 40 bytes, the maximum packet length of a TCP control segment is 80 bytes.

The main advantage of the lightweight version is that there is no need to access the TCP header, thus reducing overhead significantly. Its chief disadvantage is inaccuracy. The tiny TCP data segments that meet the filtering rules will be misclassified as TCP control segments. However, because the tiny TCP data segments are most likely to belong to interactive TCP sessions, they have similar features of TCP control segments, i.e., small size and loss-sensitive TCP performance. For end-to-end TCP performance, it is beneficial to separate them from, and give priority over, other TCP data segments. Moreover, the proposed adaptive weight-based resource manager has the ability to cope with this inaccuracy.

Since the lightweight version does not access the TCP header, it cannot further differentiate TCP control segments into SYNs, FINs, ACKs, and RSTs. To achieve accurate and fine-grained TCP control segment identification, the TCP header needs to be accessed. The heavyweight version of TCP control segment identification is a port-based classifier. The matching scope is outside the IP header and, hence, the TCP flags of the TCP header are checked. Besides the additional overhead in accessing the TCP header, IPSec makes the port-based classification difficult. However, a multilayer IPSec protocol [48] has been proposed, which allows trusted routers to access the transport-layer information.

Even in the heavyweight version of TCP control segment identification, especially for ACK identification, we still need to rely on the *total length* field in the IP header to eliminate the ambiguity caused by the following two facts:

- Some TCP implementations always set the ACK-flag bit ON once the TCP connection is established [39]. Furthermore, some malicious TCP senders could intentionally set the TCP flag bit ON in its TCP data segments.
- The piggyback mechanism in which ACKs are sent along with a reverse-direction data flow results in a packet that could be interpreted as a TCP data segment or TCP ACK.

The ambiguity caused by always-on ACK-flag and piggybacking can be solved by simply checking the total IP packet length. The large packet with ACK-flag ON is classified as a TCP data segment, but the small packet with ACK-flag ON is classified as a TCP ACK. Considering the addition of TCP option fields like the Timestamp in the TCP header, we set the threshold to 80 bytes, as the available bytes for TCP options is 40. If the total packet length is

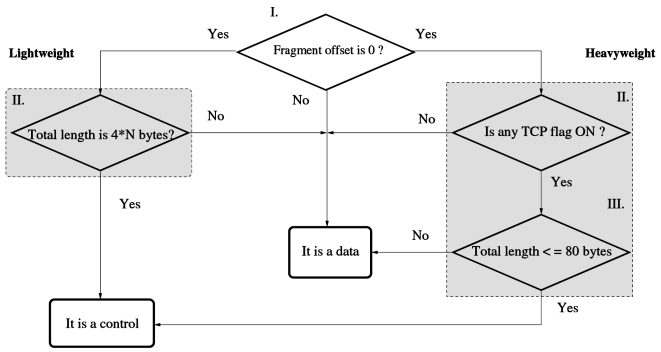

Fig. 3. The flowchart of the TCP control segment identification algorithm.

larger than 80 bytes, the packet is classified as a TCP data segment. Otherwise, it is classified as a TCP ACK.

In summary, we use a lightweight version of flow identification at core routers. All transport-layer information is derived from the IP header only and there is no need to access the TCP header. So, it does not violate the layering concept. At edge routers or firewalls, layer-4 or 7 switching and packet filtering have been implemented and deployed by many vendors. We provide a heavy-weight identification scheme which accesses the TCP header only as an option at edge routers. Therefore, even if the format of TCP header changes in future, only the heavy-weight identification scheme, if employed at edge routers, needs to be modified. We believe, however, that the format of TCP header is unlikely to change, due mainly to the compatibility problems with the already widely deployed TCP.

A detailed description of the complete TCP control segment identification algorithm is given in Fig. 3, where N is an integer such that $10 \leq N \leq 20$. There are three major steps in the heavyweight version, but only two steps in the lightweight version are given in Fig. 3.

3.2.3 Validating Lightweight Identification

To validate the lightweight version of TCP control segment identification algorithm, six Internet traces taken at three different sites [27] are used. All the traces were collected between February 2002 and March 2002. The three chosen sites are located at high-bandwidth interconnection points. ADV represents the site where the OC3c (155Mbps) PoS (Packet over Sonet) link connects the Advanced Network and Services premises in Armonk, New York, to their ISP and the Internet2/Abilene network. ANL refers to the OC3c link between the Argonne National Laboratory and the Ameritech Network Access Point (NAP) in Chicago. BUF is the site where the OC3 PoS link connects NYSERnet's router and the State University of New York at Buffalo's router.

From these traces, we found that no TCP control segment is misclassified as a data segment and only an insignificant number of tiny TCP data segments are misclassified as control segments. The filtering rule of 4-byte boundary significantly reduces the inclusion of tiny data segments (less than 80 bytes) in control segments. For example, in BUF-1 trace, without this rule, there would be 13,318 tiny data segments that are misclassified. However, after applying this rule, the number of misclassified data segments is reduced to 2,218. Table 1 gives the percentage of misclassification of TCP data segments versus the total data segments in each trace. Moreover, over 97 percent of the misclassified tiny TCP data segments are with "PSH" flag ON in its TCP header, indicating the quick delivery requirement of these segments.

TABLE 1
Effect of Misclassification

Traces	Misclassified	Total Number	Error ratio
ADV-1	665	97053	0.68%
ADV-2	354	45463	0.78%
ANL-1	4351	560223	0.77%
ANL-2	2665	437085	0.61%
BUF-1	2218	208301	1.06%
BUF-2	2390	605127	0.39%

3.2.4 Discussion

Currently, only a negligible part of the IP traffic is reported to belong to IPv6 [8], [41], so the proposed QoS classifier is based only on IPv4. Compared to IPv4, the most important changes in IPv6 lie in the packet format. The header format of IPv6 is very different from that of IPv4. However, the following two features of IPv6 will make the TCP control segment identification even easier and faster: 1) IPv4's variable-length options field is replaced by a series of fixed-format headers and each IP packet has a base header followed by zero or more extension headers, and 2) no fragmentation occurs at intermediate routers and all the fragmentation and reassembly are restricted to end-hosts. So, it is easy to adjust the proposed QoS classifier to work properly in the context of IPv6.

The emergence of Internet Telephony—also called voice over Internet Protocol (VoIP)—does not pose any difficulty on the identification of lightweight version TCP control segments since VoIP data streams are carried by Real-Time Transport Protocol (RTP) that is running on top of UDP, instead of TCP [32]. Moreover, the real audio streams show a significant regularity on packet lengths—concentrating on 244/254, 290/300, and 490/502 bytes [22], which are much larger than 80 bytes.

3.3 Adaptive Weight-Based Resource Manager

To enable better service differentiation and resource isolation between thinner aggregates, we propose an adaptive weight-based resource manager for IP routers. A hierarchical link-sharing structure is built, which is similar to the hierarchy of QoS classification and the class-based queuing management [10]. As shown in Fig. 4, the root of the resource tree is the total link capacity. As the level of the resource tree gets lower, the IP flows that share the link are split into thinner aggregates. Each leaf node has its own queue and every classified incoming packet is then inserted into the appropriate queue. Subsequently, the weighted round-robin scheduler will take care of these queued packets and select the next packet for transmission.

At level 1 of the resource tree, each node represents the allocated bandwidth to each BA. The bandwidth allocation and priority assignment at the BA level are done by the Bandwidth Broker (BB) [26] via Service Level Agreements

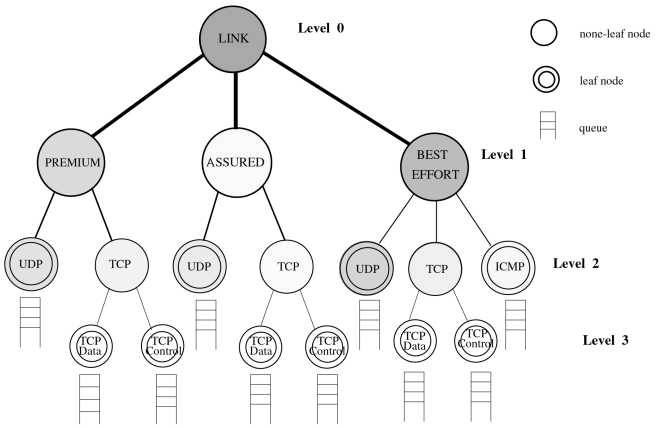

Fig. 4. The link-sharing framework.

(SLAs). At level 2 of the resource tree, the BA is divided further into a UDP aggregate and a TCP aggregate. Each node at level 2 corresponds to the bandwidth allocated to the thinner UDP/TCP aggregate. Within each BA, the UDP aggregate and the TCP aggregate have the same priority and a weighted round-robin scheduling scheme is used between them. The composition of each BA is different. Most of an EF (Expedited Forwarding) aggregate is UDP real-time audio/video. However, in the AF (Assured Forwarding) and BE (Best-Effort) aggregates, the majority of packets belong to TCP. The weights assigned to the thinner aggregates are based on the recent empirical studies reported in [8], [21], [22], [41], which are listed in Table 2. Here, we assume that the total weight of each BA is 1. The total weight assigned to UDP and TCP aggregates in BE is 0.99 since ICMP packets account for less than 1 percent of all IP packets and, by default, these ICMP packets are treated as BE traffic. Note that the weights at level 2 are tunable parameters that can be adjusted by network administrators to meet their local requirements. The weight assignment is strictly enforced only when there is no empty queue. Once a queue is empty, its assigned weight can be temporarily shared by other nonempty queues until the next packet enters the queue.

Each node at level 3 of the resource tree represents the bandwidth allocated to TCP data or control segments. The TCP data and control segments within the same TCP aggregate are also scheduled according to the weighted round-robin policy. The guiding principle for the weight setting at level 3 is that preference is given to control segments, but there is a strict limit on the weight of control segments. The weight preference to control segments is the embodiment of resource overprovisioning as suggested in [28], and the strict limit prevents the misuse of preference. Because the overwhelming majority of the TCP control segments are TCP ACKs, we first present the rule of setting the weight of TCP ACKs, and then use the ACK weight as the baseline to derive the weights of all TCP control segments.

3.3.1 Setting the Weight of ACKs

The rule of thumb for setting the weight for TCP ACKs is to approximate the upper bound of their bandwidth consumption. If the weight is measured in number of packets, the ratio of the weight of TCP ACKs to that of TCP data segments is 1:1. This *one data segment versus one ACK* policy is based on the fact that the transmission of a TCP data segment will later trigger a TCP ACK. Considering wide deployment of the delayed ACK mechanism at TCP receivers, weights are assigned to give preference to TCP ACKs and this preference is also intended to provide a safety cushion to other TCP control segments and tiny TCP data segments in which the lightweight version of TCP control segment identification is used.

TABLE 2
Level-2 Weight Distribution in Different BAs

Type	Expedited	Assured	Best-Effort
UDP	0.7	0.05	0.04
TCP	0.3	0.95	0.95
ICMP	0	0	0.01
Total	1	1	1

Adhering to the policy of *one data segment versus one ACK*, if the weight is measured in number of bytes, then we should consider the average packet size. As mentioned earlier, a great majority of TCP ACKs are 40 bytes long, so the average size of TCP ACK is 40 bytes. Common MSSs of TCP implementations are 512, 536, and 1,460 bytes [8], [41]. Including the 40 bytes of both the IP and TCP headers, the total packet lengths for these MSSs are 552, 576, and 1,500 bytes, respectively. Their observed ratio is 1:1:2. Thus, the average size of a TCP data segment is 1K bytes and the ratio of the weight of ACKs to that of data segments is 1:25.

The traffic load distribution inside the DS domain will be balanced by routing algorithms and traffic engineering. At an interface of a core router, the volume of outgoing TCP data traffic equals that of incoming TCP data traffic and each outgoing TCP data segment[2] implies an incoming TCP ACK to the interface. Therefore, this *one data segment versus one ACK* policy is valid for core routers inside a DS domain.

However, at a leaf router or a boundary router between two different DS domains, this policy is often invalid due to the asymmetry of traffic load. This traffic load asymmetry has been observed in traffic measurements of the trans-Atlantic link [41]. The weight of TCP ACKs should depend on the weight allocated to the TCP data segments in the reverse direction. The weight of the ACK aggregate at leaf or boundary routers can be set based on traffic measurements or with the help of the Bandwidth Broker. Like the weight setting at level 2, the weight assigned to ACKs, which is the baseline of TCP control segments, is also a tunable parameter and can be adjusted locally.

Once the weight of the ACK aggregate is set, we use a simple adaptive calibration scheme to derive the weight of all TCP control segments for which the ACK weight is used as the baseline. The mechanism of the weight calibration works similarly to the adaptive-weighted packet scheduling of EF traffic [44]. Its goal is to increase the flexibility of the resource manager to absorb bursty control traffic and the tiny data segments that are misclassified as TCP control segments.

2. Or, two outgoing TCP data segments if the delayed ACK mechanism is ON at the TCP receiver.

3.3.2 Adaptive Weight Calibration of TCP Control Aggregate

As in [44], we use the estimated average queue size of the TCP control aggregate to adaptively adjust the weight. The average queue size of the TCP control aggregate is calculated by using a low-pass filter with an exponentially-weighted moving average. Let avg be the average queue size, q the instantaneous queue size, and f_l the parameter of the low-pass filter, then the average queue size of TCP control aggregate is estimated as:

$$avg \leftarrow (1 - f_l) \cdot avg + f_l \cdot q.$$

To reduce the instantaneous fluctuation of queue size, the parameter of the low-pass filter f_l is set to 0.01.

Assuming that the weight of TCP ACKs is w_a, we set the original weight of the TCP control aggregate w_c to $1.2\ w_a$. To adaptively calibrate the weight of the TCP control aggregate, two thresholds, min_{th} and max_{th}, are introduced. By keeping the average queue size of the TCP control aggregate below the maximum threshold, bursty losses of TCP control segments are prevented. To accomplish this, the weight of the control aggregate should be proportionally increased once the average queue size of the control aggregate exceeds the minimum threshold. The values of min_{th} and max_{th} are set to one-fourth and three-fourths of the buffer, respectively. The linear relationship between the weight and the average queue size of control aggregate is given by:

$$f(C) = \begin{cases} w_c, & C \in [0, min_{th}) \\ \frac{(U - w_c) \cdot (C - min_{th})}{max_{th} - min_{th}} + w_c, & C \in [min_{th}, max_{th}) \\ U, & C \in [max_{th}, full], \end{cases}$$

where $f(C)$ is the weight function of the control aggregate, U is the upper limit that the weight of the control aggregate can reach, and C is the average queue size of the control aggregate. Since the total weight for TCP aggregates is fixed, the increase of the control aggregate's weight must cause the same amount of decrease in the data aggregate's weight. However, once the average queue size of the control aggregate reduces below max_{th}, the weights taken from the data aggregate will be returned.

The weight calibration favors the control aggregate, but disfavors the data aggregate, which is consistent with the guiding principle of the weight settings. The rationale behind this is that the bandwidth taken by the data aggregate is usually much more than the bandwidth consumed by the control aggregate; the small amount of bandwidth shift from the data aggregate to the control aggregate can prevent bursty losses of the control segments, but only leads to a single isolated data packet loss or just a longer queuing delay. However, the weight of the control aggregate cannot exceed the upper limit, which prevents the abuse of preferential treatment of TCP control segments and protects the TCP data aggregate from starvation. U is set to $2w_c$ in our simulation.

4 Performance Evaluation

The proposed tIP router architecture is evaluated by simulation with *ns*-2 [25], [42]. According to the purpose of simulation, we categorize the simulation experiments into two different classes. One is used for evaluating the capability of resource isolation under flooding attacks, the other is used for evaluating the capability of service differentiation for end-to-end TCP performance. The network topologies of the two classes are different. In the presentation of simulation results, *Existing* refers to the current DiffServ (or Internet) architecture, *Marking* refers to the ACK marking scheme proposed in [28], and *Reserved* refers to the reserved bandwidth for EF and AF flows.

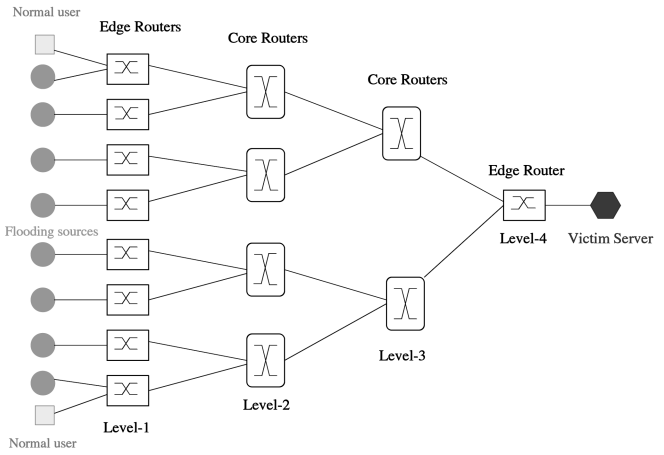

Fig. 5. The simulated network topology used for resource isolation.

4.1 The Simulation Setup

In our simulation experiments, each end-host is connected to its respective edge router and the edge routers are connected via core routers. The link capacity and one-way propagation delay between an end-host and an edge router are 10 Mbps and 1 ms, respectively. The one-way propagation delay between an edge router and a core router is 8 ms, but that between two core routers is 16 ms. The UDP/TCP data segment size is set to 1,000 bytes and the TCP control segment size is set to 40 bytes. The version of TCP used in the simulation is TCP New-Reno since it has been widely deployed in the Internet, and the delayed-ACK mechanism is ON.

4.2 Resource Isolation

In the flooding experiments, the link capacity between an edge router and a core router is 6 Mbps, but that between two core routers is 5 Mbps. As the network topology for the purpose of a DDoS attack, it is convenient to consider it in terms of a tree graph. The victim's machine is at the root of the tree, with network routers being intermediate nodes in the tree. The leaf nodes of the tree are the flooding sources and normal end-hosts. The simulated network topology for DDoS attacks is shown in Fig. 5. In our flooding experiments, there is a flooding source in each stub network except for the one that the victim belongs to. The flooding rate at each source is constant and set to 5,000 packets per second in SYN and ACK flooding attacks, but 500 packets per second in UDP and ICMP flooding attacks. At the same time, there are TCP connections running from normal end-hosts to the victim as the background traffic.

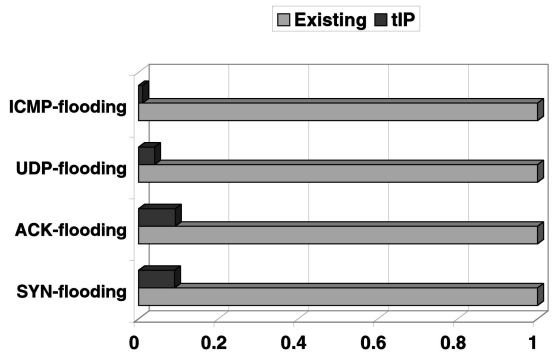

Fig. 6. The flooding traffic volume reached the victim.

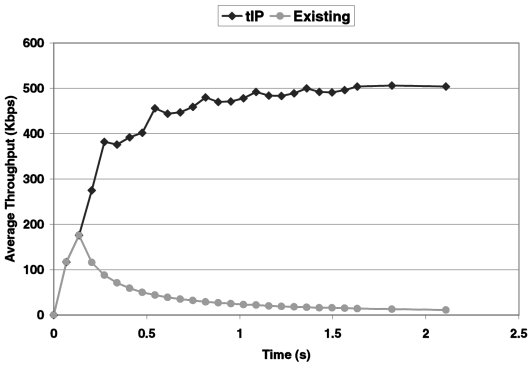

Fig. 8. Average throughput of the background TCP connection.

We first measure the volume of the flooding traffic that reaches the victim under different IP architectures. Four types of flooding attacks—SYN, ACK, UDP, and ICMP flooding—are simulated. All the flooding traffic is transported by the BE service. Since there is no difference in treating the BE traffic in the current Internet and DiffServ architectures, we normalize the various flooding traffic reached the victim in these architectures to 1 to make the presentation easier. Then, the flooding traffic received at the victim in the *t*IP router architecture is properly scaled based on the normalization. Fig. 6 shows that the *t*IP router throttles the flooding volume that reaches the victim and effectively protects the victim from flooding attacks. More importantly, most of the flooding traffic will be dropped by the first few routers before they reach the core of the network, thus confining the damage caused by the flooding source mainly to the local stub network where it originated. The cascaded throttling of flooding traffic at the first few routers shields the rest of the Internet unaffected, and saves the network bandwidth. The cascaded throttling of flooding traffic is depicted in Fig. 7.

Moreover, during flooding attacks, the effective TCP throughputs in *Existing* and *Marking* are reduced almost to zero, but the one in *t*IP router can still achieve 95 percent (in the cases of UDP and ICMP flooding) and 85 percent (in the cases of SYN and ACK flooding) of the bandwidth assigned to the entire BE traffic, thanks to the layer-4 resource isolation. Fig. 8 illustrates the dynamics of average throughput of a background TCP connection under the UDP or ICMP flooding attack that starts at 0.2s. Since the TCP connection is not interfered with by UDP or ICMP floods, its average throughput follows a typical sawtooth-like curve.

Our simulation results show that the *t*IP router provides a built-in protection mechanism to counter DDoS attacks. Splitting layer-4 traffic greatly reduces the performance degradation caused by such DDoS attacks as flooding of UDP, ICMP, TCP SYN, and ACK traffic. The resource isolation provided inside the BE traffic class is especially valuable since the edge routers in the DiffServ architecture perform traffic conditioning and policing on EF and AF traffic, but not on BE traffic.

For EF or AF traffic, both UDP and ICMP flooding do not cause much damage in all DiffServ architectures because the edge routers perform traffic conditioning and policing on EF and AF traffic. However, the simple *Marking* scheme exposes more vulnerability to the ACK flooding attacks. In this architecture, the ACK flows are accepted without strict policing based on the belief that the small bandwidth requirement by ACK flows can be absorbed by over-provisioning. The flooding ACK flows marked as EF or AF traffic can seriously violate the traffic profile between the stub network and the leaf router that connects the stub network to the Internet. Even worse, in large-scale DDoS attacks, even if only a small number of ACKs are flooded from each attacking source, once these ACKs are aggregated at core routers where no traffic conditioning is performed, the flooding ACK aggregates can "steal" the reserved bandwidth from the conformant aggregates. Fig. 9 charts the goodputs of conformant EF and AF flows from a normal end-host to the victim in *Marking* and *t*IP router, and clearly shows the vulnerability of *Marking* to the ACK

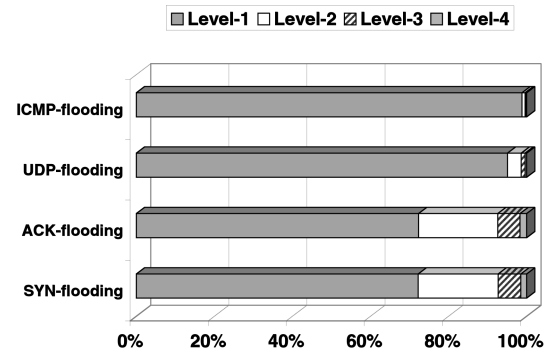

Fig. 7. The distribution of dropped packets at different level routers.

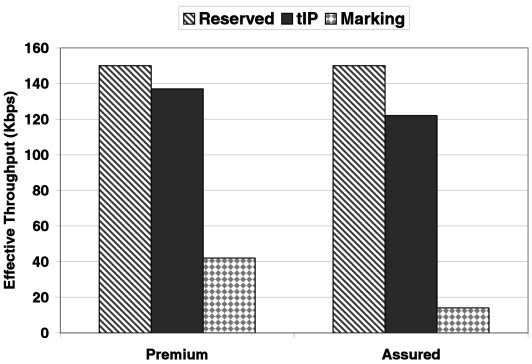

Fig. 9. The goodput of conformant EF and AF flows under the ACK flooding attack.

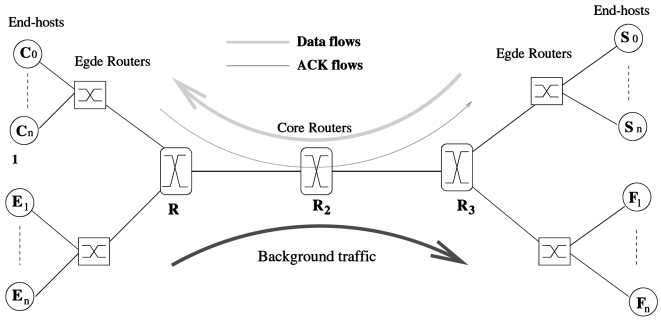

Fig. 10. The simulated network topology for service differentiation.

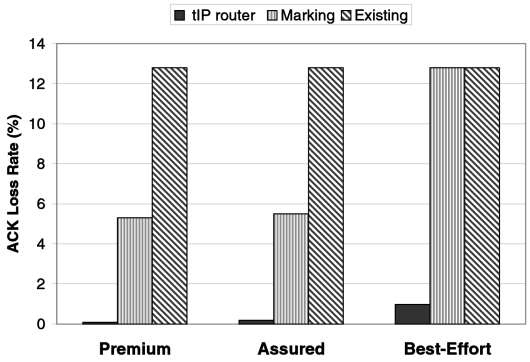

Fig. 11. The ACK loss rate in different DiffServ architectures.

flooding attack and the robustness of tIP router to the same attack. Note that the conformant EF flow is carried by UDP, but the AF one is carried by TCP.

4.3 Service Differentiation

In the experiments of service differentiation, the link capacity between an edge router and a core router is 3 Mbps, but that between two core routers is 1 Mbps. We study the TCP flows and measure their ACK loss rate and effective throughput, where we not only compare the tIP router with the existing DiffServ, but also with the marking scheme for TCP ACKs proposed in [28]. The simulation network topology for service differentiation is shown in Fig. 10, which is a relatively simple, yet sufficiently representative topology for validating end-to-end TCP performance. The experimental configuration is as follows: Three targeted TCP connections are established from S_1 to C_1, which receive premium, assured, and best-effort services, respectively. In addition to the three targeted TCP connections, two more TCP connections carry BE data from S_2 to C_2. All of them have infinite amounts of data to send, i.e., with commonly used "persistent" sources.

With respect to the direction of targeted TCP data flows, we name the path $R_3 \rightarrow R_1$ the *forward* path and the path $R_1 \rightarrow R_3$ the *backward* path. The resources along the forward path $R_3 \rightarrow R_1$ are properly provisioned for the premium and assured forwarding traffic, but the remaining network resources are periodically exhausted by the BE traffic, causing random data losses to occur in the forward path.

On the backward path $R_1 \rightarrow R_3$, similar background traffic is generated between E_i to F_i and shares the same path with the targeted ACK flows. The background traffic is a mixture of premium, assured, and BE traffic. Compared to the simulation configuration in the forward path, there are two key differences in the backward path:

- the network resources for the premium and assured services in the background traffic are underprovisioned, and
- the BE traffic consists of not only TCP flows but also UDP flows, which causes severe congestion in the backward path, thus resulting in bursty packet losses.

4.3.1 The Simulation Results

The ACK loss rates of targeted TCP connections are plotted in Fig. 11 and the effective throughputs of the targeted TCP connections are plotted in Fig. 12. The simulation results show us that:

- the tIP router provides better service isolation for ACK flows, significantly lowering the ACK loss rate and increasing effective throughput, and
- the ACK marking scheme cannot support service isolation for ACK flows when network resources are under-provisioned, thus resulting in bursty ACK losses and, hence, degrading TCP performance significantly.

For EF and AF traffic, the ACK loss rates of *Marking* are much lower than those of *Existing*, but are much higher than those of the tIP router. Moreover, most of ACK losses are bursty rather than random, lowering effective throughput. For EF traffic, the main reasons for bursty ACK losses are: 1) To support low delay for EF traffic, the buffer space for premium service is very small and can only accommodate one or two data packets; 2) the size of a data segment is much larger than that of an ACK. Once the buffer has been filled with one or two data segments, the subsequent incoming ACKs will be dropped.

In the *Existing* and *Marking* DiffServ architectures, AF traffic shares the same FIFO queue with BE traffic, but AF packets are much less likely to be dropped than BE ones. However, without proper resource provisioning for ACK flows, the ACKs are more likely to be marked as high drop-precedence packets at edge routers due to the corresponding traffic profile violation. Under severe congestion, all the packets marked with high drop-precedence will be dropped, causing bursty ACK losses. Since bursty ACK losses cause much severer degradation to TCP performance than random

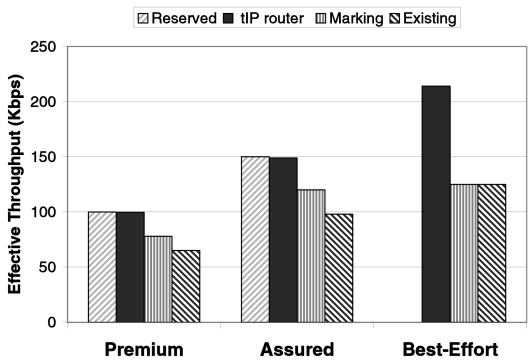

Fig. 12. The effective throughput in different DiffServ architectures.

ACK losses, even modest ACK loss rates for EF and AF can greatly reduce their effective throughput.

As the *Existing* DiffServ, the ACK marking scheme provides no improvement to BE traffic. BE ACKs experience a high loss rate in the backward path because of the congestion caused by the UDP flows in the background traffic. Furthermore, due to data losses in the forward path, an ACK loss for retransmission in the backward path leads to a timeout, reducing *cwnd* to 1, triggering a slow-start and, hence, degrading effective throughput significantly.

In contrast, the *t*IP router architecture significantly improves the performance of BE TCP traffic thanks to its resource isolation between UDP and TCP flows, as well as between ACKs and TCP data segments within the BE class. The *t*IP router not only provides better service quality to high-tiered services, but also significantly improves the performance of BE TCP sessions.

Note that, although the TCP version in our simulation experiment is New-Reno, most of the simulation results in this paper are applicable to all TCP variants for the following reasons. First, the TCP behaviors after a retransmission timeout for all of these schemes are similar. TCP variants differ only in the way of recovering from packet losses after a fast retransmit. Second, the ACK losses in the reverse path only lead to a timeout or slower congestion window growth, but cannot trigger a fast retransmit.

5 Conclusions

We presented a transport-aware IP router architecture to provide layer-4 service differentiation and resource isolation. The key components of the *t*IP router architecture are the fine-grained QoS classifier and the adaptive weight-based resource manager. A two-stage packet classification mechanism is devised to decouple the fine-grained QoS lookup from the routing lookup at core routers. BAs are further divided into thinner aggregates. By using separate queues and adaptive-weighted bandwidth allocation, better service differentiation and resource isolation are achieved for these thinner aggregates.

We evaluated the performance of the *t*IP router architecture by simulation. The simulation results show that:

- it provides a built-in protection mechanism to counter DDoS attacks: The flooding traffic is significantly throttled and most of them is dropped in a close proximity to their sources;
- the resource isolation of the *t*IP router protects the normal traffic from the flooding traffic that belongs to a different transport protocol;
- the *t*IP router guarantees that high-tiered TCP sessions receive better service and, hence, yield better performance in terms of loss rate, end-to-end delay, and effective throughput than low-tiered TCP sessions;
- it not only achieves better service quality for high-tiered services, but also significantly improves the performance of BE TCP sessions.

Furthermore, the simulation results demonstrate that a simple ACK marking scheme does not provide good service differentiation and resource isolation for ACK flows when network resources are underprovisioned. It exposes the vulnerability of EF and AF traffic to the ACK flooding attacks. The *t*IP router architecture is therefore necessary to provide better network QoS to TCP sessions and is a simple yet powerful built-in protection mechanism to counter DDoS attacks.

Acknowledgments

The work reported in this paper was supported in part by Samsung Electronics, Inc. and the US Office of Naval Research under Grant N00014-99-1-0465.

References

[1] T. Abdelzaher and K.G. Shin, "End-Host Architecture for QoS-Adaptive Communication," *Proc. IEEE Real-Time Technology and Applications Symp.,* June 1998.
[2] G. Banga, P. Druschel, and J. Mogul, "Resource Containers: A New Facility for Resource Management in Server Systems," *Proc. Third Symp. Operating System Design and Implementation,* Feb. 1999.
[3] H. Balakrishnan, V. Padmanabhan, and R.H. Katz, "The Effects of Asymmetry on TCP Performance," *Proc. ACM/IEEE MOBICOM,* Sept. 1997.
[4] S.M. Bellovin, "ICMP Traceback Messages," Internet Draft: draft-bellovin-itrace-00.txt, Mar. 2000.
[5] Y. Bernet et al., "A Framework for Differentiated Services," IETF Internet Draft, Feb. 1999.
[6] N. Bhatti and R. Friedrich, "Web Server Support for Tiered Services," *IEEE Network,* vol. 13, no. 5, Sept./Oct. 1999.
[7] S. Blake et al., "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.
[8] CAIDA's Traffic Workload Overview, http://www.caida.org/outreach/resources/learn/trafficworkload/tcpudp.xml, 1999.
[9] D. Dittrich, "Distributed Denial of Service (DDoS) Attacks/Tools Page," http://staff.washington. edu/dittrich/misc/ddos/, 2002.
[10] S. Floyd and V. Jacobson, "Link-Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Trans. Networking,* vol. 3, no. 4, Aug. 1995.
[11] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," RFC 2267, Jan. 1998.
[12] L. Garber, "Denial-of-Service Attack Rip the Internet," *Computer,* Apr. 2000.
[13] A. Garg and A. Reddy, "Mitigation of DoS Attacks through QoS Regulation," *Proc. Int'l Workshop Quality of Service,* May 2002.
[14] S. Gibson, "Distributed Reflection Denial of Service," technical report, Gibson Research Corp., http://grc.com/dos/drdos.htm, Feb. 2002.
[15] T.M. Gil and M. Poletter, "MULTOPS: A Data-Structure for Bandwidth Attack Detection," *Proc. USENIX Security Symp.,* Aug. 2001.
[16] P. Gupta and N. McKeown, "Packet Classification on Multiple Fields," *Proc. ACM SIGCOMM,* Sept. 1999.
[17] J. Ioannidis and S.M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," *Proc. Network and Distributed System Security Symp.,* Feb. 2002.
[18] T.V. Lakshman and D. Stiliadis, "High Speed Policy-based Packet Forwarding Using Efficient Multi-Dimensional Range Matching," *Proc. ACM SIGCOMM,* Sept. 1998.
[19] R. Manajan et al., "Controlling High Bandwidth Aggregates in the Network," ICSI technical report, July 2001.
[20] J. McQuillan, "Layer 4 Switching," *Data Comm.,* Oct. 1997.
[21] S. McCreary and K. Claffy, "Trends in Wide Area IP Traffic Patterns—A View from Ames Internet Exchange," *Proc. Int'l Technical Conf.,* Sept. 2000.
[22] A. Mena and J. Heidemann, "An Empirical Study of Real Audio Traffic," *Proc. IEEE INFOCOM,* Mar. 2000.
[23] A. Miyoshi and R. Rajkumar, "Protecting Resources with Resource Control Lists," *Proc. IEEE Real-Time Technology and Applications Symp.,* May 2001.
[24] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial of Service Activity," *Proc. USENIX Security Symp.,* Aug. 2001.

[25] S. Murphy, "DiffServ Additions to ns-2," http://www.teltec.du c.ie/murphys/ns-work/diffserv/, May 2000.
[26] K. Nichols, V. Jacobson, and L. Zhang, "A Two-Bit Differentiated Services Architecture for the Internet," RFC 2638, July 1999.
[27] NLANR Network Traffic Packet Header Traces, http://pma.nlanr.net/Traces/, 2002.
[28] K. Papagiannaki, P. Thiran, J. Crowcroft, and C. Diot, "Preferential Treatment of Acknowledgment Packets in a Differentiated Services Network," *Proc. Int'l Workshop QoS,* June 2001.
[29] K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets," *Proc. ACM SIGCOMM,* Aug. 2001.
[30] V. Paxson, "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks," *ACM Computer Comm. Rev.,* vol. 31, no. 3, July 2001.
[31] K.K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," RFC 2481, Jan. 1999.
[32] D. Rizzetto and C. Catania, "A Voice over IP Service Architecture for Integrated Communications," *IEEE Network,* vol. 13, no. 3, June 1999.
[33] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," *Proc. ACM SIGCOMM,* Aug. 2000.
[34] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, Apr. 1998.
[35] A.C. Snoren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer, "Hash-Based IP Traceback," *Proc. ACM SIGCOMM,* Aug. 2001.
[36] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *Proc. IEEE INFOCOM,* Mar. 2001.
[37] O. Spatscheck and L. Peterson, "Defending against Denial of Service Attacks in Scout," *Proc. Third Symp. Operating System Design and Implementation,* Feb. 1999.
[38] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and Scalable Layer Four Switching," *Proc. ACM SIGCOMM,* Sept. 1998.
[39] W. Stevens, *TCP/IP Illustrated.* vol. 1, Addison-Wesley, 1994.
[40] R. Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," *Proc. Ninth USENIX Security Symp.,* Aug. 2000.
[41] K. Thompson, G.J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network,* vol. 11, no. 6, Nov./Dec. 1997.
[42] "Network Simulator," UCB/LBNL/VINT, ns-2, http://www.isi.edu/nsnam/ns/, 1999.
[43] H. Wang, D. Zhang, and K.G. Shin, "Detecting SYN Flooding Attacks," *Proc. IEEE INFOCOM,* June 2002.
[44] H. Wang, C. Shen, and K.G. Shin, "Adaptive-Weighted Packet Scheduling for Premium Service," *Proc. IEEE Int'l Conf. Comm.,* June 2001.
[45] P. Wang, Y. Yemini, D. Florissi, J. Zinky, and P. Florissi, "Experimental QoS Performances of Multimedia Applications," *Proc. IEEE INFOCOM,* Mar. 2000.
[46] D. Yau, J. Lui, and F. Liang, "Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles," *Proc. Int'l Workshop Quality of Service,* May 2002.
[47] L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two Way Traffic," *Proc. ACM SIGCOMM,* Sept. 1991.
[48] Y. Zhang and B. Singh, "A Multi-Layer IPsec Protocol," *Proc. Ninth USENIX Security Symp.,* Aug. 2000.

Haining Wang is a PhD candidate in the Electrical Engineering and Computer Science Department at the University of Michigan University and expects to complete his dissertation work in the Summer of 2003. His research interests lie in the area of networking, security, and distributed computing. He is particularly interested in network security and network QoS (Quality of Service) to support secure and service differentiated internetworking. He is a student member of the IEEE.

Kang G. Shin received the BS degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and both the MS and PhD degrees in electrical engineering from Cornell University, Ithaca, New York, in 1976 and 1978, respectively. He is the Kevin and Nancy O'Connor Professor of Computer Science and founding director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on QoS-sensitive networking and computing as well as on embedded real-time OS, middleware, and applications, all with emphasis on timeliness and dependability. He has supervised the completion of 42 PhD theses and authored/coauthored more than 500 technical papers and numerous book chapters in the areas of distributed real-time computing and control, computer networking, fault-tolerant computing, and intelligent manufacturing. He has coauthored (jointly with C.M. Krishna) a textbook, *Real-Time Systems* (McGraw Hill, 1997). He has received a number of best paper awards, including the IEEE Communications Society William R. Bennett Prize Paper Award in 2003, the Best Paper Award from the IWQoS'03, and an Outstanding *IEEE Transactions on Automatic Control* Paper Award in 1987. He has also coauthored papers with his students which received the Best Student Paper Awards from the 1996 IEEE Real-Time Technology and Application Symposium and the 2000 UNSENIX Technical Conference. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Service Excellence Award in 2000, and Distinguished Faculty Achievement Award in 2001 from the University of Michigan; and a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002. From 1978 to 1982, he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He has held visiting positions at the US Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at the Univeristy of California at Berkeley, and International Computer Science Institute, Berkeley, Calif., IBM T.J. Watson Research Center, Software Engineering Institute at Carnegie Mellon University, and HP Research Laboratories. He also chaired the Computer Science and Engineering Division, EECS Department, The University of Michigan for three years, beginning in January 1991. He is a fellow of the IEEE, IEEE Computer Society, and ACM, and a member of the Korean Academy of Engineering, was the general chair of the 2000 IEEE Real-Time Technology and Applications Symposium, the program chair of the 1986 IEEE Real-Time Systems Symposium (RTSS), the general chair of the 1987 RTSS, the guest editor of the 1987 August special issue of *IEEE Transactions on Computers* on real-rime Systems, a program cochair for the 1992 International Conference on Parallel Processing, and served on numerous technical program committees. He also chaired the IEEE Technical Committee on Real-Time Systems from 1991-1993, was a distinguished visitor of the IEEE Computer Society, an editor of the *IEEE Transactions on Parallel and Distributed Computing*, and an area editor of the *International Journal of Time-Critical Computing Systems*, *Computer Networks*, and *ACM Transactions on Embedded Systems*.