Router-Transparent Packet Annotations

George Porter, Rodrigo Fonseca, Scott Shenker, Ion Stoica, Randy H. Katz*

August 1, 2005

Abstract

Annotations are a mechanism for adding state to packets that is orthogonal to the data payload. Historically, two ways of adding annotations to packets have been used: overloading reserved bits in the header, and using IP options. In the first case, the amount of space available is very limited and difficult to share. In the second case, packets with IP options are dropped on many Internet paths. We propose a more general and flexible mechanism for annotating packets that appends annotations to the end of packets, in the IP payload. We show that many existing proposals for adding state to packets can exploit this common mechanism for annotations. Our key result is that packets annotated in this way have a much higher probability of successful transport across the network than those for which IP options have been used.

1 Introduction

As the Internet has evolved, researchers have proposed a plethora of extensions to the original Internet Protocol (IP). Many of these extensions, such as IP traceback[9], XCP[5], Quick-start[4], SCORE[11], CETEN[1], and SIFF[13], require recording of some protocol-specific state in the packet header. We refer to this state as an *annotation*.

One of the two main ways of adding annotations to packets is to use IP options. IP options were designed to be incrementally deployable: a router or an end-host that did not understand a particular option was supposed to ignore that option and simply forward the packet. Unfortunately, this is not what happens in today's Internet. Because checking the IP options is an expensive operation, many routers chose to drop the packets with options they do not understand, rather than forwarding them. Our observations of 37,145 paths between PlanetLab nodes show that almost half of the paths dropped packets with options.

This lead the designers of IP extensions to avoid using IP options and choose the second way to annotate packets. They overloaded the semantics of fields[1] in the IP header. This is incrementally deployable as long as the unaware routers do not interpret the overloaded fields. On the downside, the number of bits in the packet header that can be overloaded is very small, and cannot be easily shared by different extensions.

In this paper, we present a simple, alternative solution that is incrementally deployable, yet flexible. Conceptually, we add annotation data to the packet by extending the IP payload to include that annotation. We have organized this extended area in the payload so that devices that understand our annotation scheme can easily detect them and access the data they contain. Legacy devices simply see a standard IP packet, and forward it accordingly, in its "fast-path". This annotation structure enables a large variety of annotations of different types, and allows multiple annotations to coexist. To demonstrate its generality, we show how it can be used to implement many of the previously proposed IP extensions including IP traceback, XCP, and SIFF.

The rest of the paper is organized as follows. Section 2 discusses previous mechanisms to annotate packets. Section 3 describes how to extend the payload to support annoations that are transparent to annotation unaware routers. Section 4 describes some IP extensions and shows how they can be deployed using our annotations. Section 5 evaluates the feasibility of our implementation using the PlanetLab experiments, and Section 6 describes the challenges and opportunities of annotations. Finally, Section 7 concludes the paper.

2 Related Work

Space for packet annotations was included in the Internet Protocol version 4[8]. IPv4 includes support for up to 40 bytes of options. This option space is divided into one or more variable length entries each consisting of a type, and possibly length and data fields. IPv6[2] generalizes the use of options to include two distinct sets of headers: hop-by-hop head-

*All authors: Univ of California, Berkeley, CA 94720-1776

[1]Some examples are using the Type of Service byte, and the fragment offset[11].

ers and end-to-end headers. Hop-by-hop headers are intended for intermediate routers, and thus are the only ones that affect the core of the network.

Another way to bind annotations to a packet without modifying its contents is to tunnel the packet along with the annotation in a larger packet. The tunnel endpoint must decapsulate the packet, recover the annotation, and send the packet to the original destination. A common tunneling protocol is Generic Routing Encapsulation (GRE)[3]. A major drawback to tunneling is that the semantics of the protocol are violated and the destination address in the packet is changed. Often, TCP packets are encapsulated into UDP packets, and congestion control is lost. Additionally, the communicating parties now must include a tunnel endpoint (and possibly entry point) into their fate sharing.

3 An Annotation Area for IP Packets

This section describes a mechanism for adding annotations to packets transiting the network. This data contains one or multiple annotations, which are simply a variable-length set of bits that can be used by applications residing in routers, endhosts, or middleboxes. Our design for annotations provides applications with a "scratch-pad" of bits that they can use for their particular application. We associate with each annotation application a type. Different annotation-aware applications need not understand each other's annotation formats, only their own type.

In a normal IP packet, the header has a length field as well as a checksum field that is a function of the header. We choose to store annotations at the *end* of the IP packet, as an extension to the packet's IP payload. This necessitates making two changes to the header: the length must be increased to reflect the new annotation, and the packet's IP checksum must be recomputed. To non-annotation devices (such as standard core routers) this encoding will appear as a standard IP packet. Figure 1 shows an annotated IP packet with updated length and checksum fields.

For annotation-aware routers or endhosts to recognize that the packet is annotated, we store a special set of signaling bits at the end of the packet. These bits can be the same for every packet, or they can change on a packet-by-packet basis. We choose the signaling bits to be the value of a hash of the IP header (excluding fields that change hop-by-hop). We store this hash value at the very end of the annotation space (and thus, in the last few bytes of the packet). This choice allows annotation-aware routers to easily check whether packets transiting through them are annotated: they compute a hash of persistent fields in the IP header and compare to the last

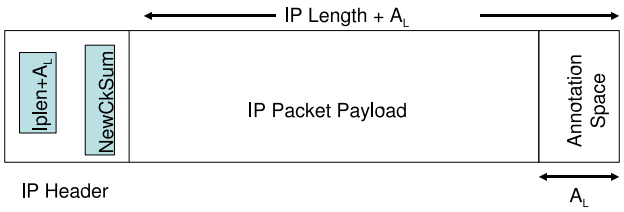

Figure 1: An annotated IP packet with updated IP length and checksum fields.

bytes of the packet. Given that the IP checksum is 16-bits, our design reserves 16-bits for the signaling bits. Based on a trace of data collected from the UC Berkeley EECS research network, we found that the false positive rate of unannotated traffic, defined as packets whose last two bytes accidentally match the computed hash value, is approximately $\frac{1}{2^{-15}}$ There is a possibility that the last two bytes of the packet happen to collide with our hash of the header (a false negative). Although rare, we could prevent this collision by fragmenting the packet. We expect this fragmentation to affect about 1 in 2^{16} packets.

Starting at the end of the packet and working to the front, the annotation space consists of two signalling bytes, an index data structure, and finally zero or more annotations. Since the annotations are of variable length, we use an index structure to facilitate locating them. The index is very simple–it consists of a length byte followed by a list of <*type,offset*> pairs. The length byte indicates how many pairs there are. Each <*type,offset*> pair allows annotation-aware routers to locate annotations by looking *offset* bytes past the index into the annotation area. Note that this structure implicitly identifies the length of each annotation. We do not specify the meaning or semantics of individual annotations–that is up to the application. Figure 2 shows this structure.

To remove annotations, an annotation-aware router or end host first determines the length of the annotation space (by examining the index). It then subtracts that value from the IP header length, and recalculates the checksum. The packet is now no longer annotated.

For some environments, we envision that endhosts will support annotations natively in their network stack. Some of the examples given in Section 4 require this. In other environments, it will be network appliances and intermediate devices that make use of packet annotations, and those annotations will be removed before the reach endhosts. In this case the endhosts are unaware of the annotations.

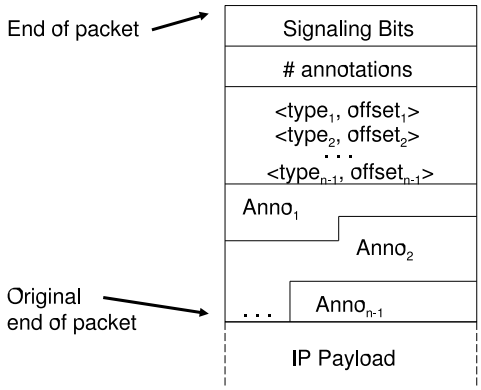

Figure 2: The annotation format: zero or more annotations, an index structure, and signaling bits.

4 Annotations Illustrated

We now consider a set of recent proposals for new extensions to IP, and consider how each could be implemented using packet annotations. Each of these proposals would be associated with an annotation type, and the annotation index stored at the end of the packet would map the type to the appropriate location in the annotation space. We first describe quality-of-service extensions (XCP, Quick-start, and SCORE). We then describe security extensions (IP traceback and SIFF). Lastly we describe a performance extension for lossy channels (CETEN).

4.1 XCP: Congestion control for high bandwidth-delay product networks

XCP[5] addresses TCP instability and inefficiency that arises in networks with high bandwidth-delay products. It does this by applying control theory to provide both efficiency and fairness by taking more information into account than TCP does. Indeed, TCP relies primarily on packet loss to detect congestion in the network. XCP extends the information that each packet conveys by adding the sender's TCP congestion window and round-trip time estimate, as well as a third field that intermediate routers modify to affect the sender's congestion window to an XCP header stored in each packet. This data allows routers to regulate the flow of multiple senders using them.

The initial deployment strategy for XCP does not require each packet to carry an XCP header. Instead, border routers at the edge of AS clouds would keep state on flows leaving the AS. They would periodically exchange reports through separate control packet containing XCP data in them with border routers in destination networks. While this does not require packets modification, it does require support in the border router, which must now keep track of state for many flows. Using annotations, senders could record the actual XCP header in the form of an XCP annotation. Routers that are XCP-aware would be able to manipulate the appropriate field in order to support XCP. Routers that are annotation- or XCP-unaware would see a standard IP packet and would forward it along. The use of annotations makes XCP and end-to-end, rather than an AS-to-AS protocol suitable for incremental deployment.

4.2 Quickstart for TCP and IP

In very high speed networks, nodes must send a large number of packets before their TCP connections reach their steady-state speeds. This is due to TCP's conservative slow-start policy. Quickstart[4] proposes that the sender annotate SYN packets it sends out with its desired sending rate. As routers along the network path see this SYN packet, they might adjust this rate downward if they are unable to support a new connection at the desired speed. When the packet reaches the destination, the initial rate field represents the instantaneous bottleneck bandwidth of the connection. This rate then determines the TCP parameters for the connection.

In addition to the initial sending rate, there is also a Quick-start TTL field that Quick-start aware network routers decrement. This can be compared to the TTL in the packet to determine whether to use the mechanism (only paths with all Quick-start aware routers can use the feature). Deployment with annotations would require both of these fields, since now all routers on the path would have to be both Quick-start and annotation-aware. Note that SYN packets are easy to annotate, since they are often much smaller than the network MTU.

4.3 The Stateless Core Network Architecture (SCORE)

SCORE is a scalable network architecture that allows one to provide per-flow services such isolation and QoS without requiring routers to maintain per-flow state [11]. The main idea is to have packets carry this state, instead of routers maintaining it. By using this technique, called dynamic packet state (DPS), it is possible to provide a variety of services such as approximating per-flow fair queueing [12], guaranteed services [11], and per-flow load balancing [10].

SCORE is deployed on a domain-by-domain basis. The ingress routers maintain per-flow sate and insert the DPS related state in the packets' headers, while core routers implement DPS and maintain no per-flow state. The egress routers remove the state form the packets.

Our annotation scheme naturally supports the

SCORE architecture. Instead of using IP options, or overloading the IP ToS and the fragment offset fields to store the DPS related state (as proposed in the original SCORE design), we can simply use annotation to store this sate into packets.

4.4 Network support for IP traceback

IP traceback is an enhancement of the standard network forwarding model that allows autonomous systems (ASes) to locate the source of packet floods during denial-of-service (DoS) attacks. Conceptually, routers along an Internet path keep enough state to reconstruct the set of links to a DoS source. To facilitate this, support at the network-level has been proposed in [9] to mark packets with identifiers that represent sampled network path edge pairs. These edge pairs, which consist of two router IP addresses and a distance measurement, make their way to the victim network. With sufficient such edge pairs, the victim can reconstruct the correct ordering of links to the source of the attack.

The samples stored in packets are 72-bits long (two 32-bit IP addresses and an 8-bit distance field). The authors chose to segment these samples and spread them out over multiple packets. This, combined with a very compact encoding, allows the samples to fit into the 16-bit IP identification field (used for supporting fragmentation). Using annotations, IP traceback would be supported by an annotation 72-bits long. No additional encoding would be needed, and entire edge pairs could be stored in a single packet, rather than spread over multiple packets in a flow.

4.5 SIFF: Mitigation of DDoS Attacks

Distributed denial of service (DDoS) attacks occur when a large number of (usually compromised) machines on the Internet send traffic floods to a victim network. Because of IP's best-effort service model, core routers unwittingly end up acting as conduits for the attack. The authors of SIFF[13] propose to add capabilities to packets which are granted by recipients. These capabilities are carried in each packet, and can be easily validated by core and edge routers. The advantage of the SIFF approach is that DDoS attacks can be prevented throughout the network–not just in the recipient's edge network.

SIFF relies on three new fields added to the IP header: a set of three flag bits, a capability field, and possibly a capability reply field. Both sender and receiver have to have modified network stacks to support the use of packet capabilities. To support SIFF with annotations, these three fields would be stored in the annotation space of the packet, and both the sender and receiver would be SIFF and annotation aware. Incremental deployment would be possible since annotation-unaware legacy routers would simply not detect either field.

4.6 Cumulative Explicit Transport Error Notification (CETEN)

TCP lowers its sending rate in response to observed packet drops in the network. In cases where packet drops are due to congestion losses, this is a desired behavior. However, in cases where packet drops are due to corruption losses (for example, from a lossy wireless link), this behavior can lead to sub-optimal TCP performance. CETEN[1] is a proposal to enhance TCP by discovering these two loss rates explicitly. It estimates corruption-based losses by encoding a "corruption survival probability" header into each packet. As routers in the network forward the packet, they update this value based on the probability of corruption they are seeing on their link. When the packet gets to the destination, the final value of that field is echoed back to the sender.

The CETEN annotation would consist of the corruption survival probability, a four-byte floating point value. To support CETEN with annotations, both the sender and receiver would be annotation aware. This is because the receiver has to echo the probability value stored in the packet's annotation back to the sender in a future packet. CETEN's incremental deployment strategy would then involve CETEN-aware network routers to adjust the appropriate packet annotation.

5 Feasibility Study

In this section we present experiments conducted with real packets in the wide area that support the use of annotations in lieu of IP options for signaling.

We found that support for packets with IP options was quite limited in the wide area, and it is not a viable solution for annotating packets on today's Internet. We ran an experiment to test how the paths among 195 PlanetLab [7] hosts would cope with packets containing IP options. These were selected as one per PlanetLab site, of a subset of the nodes we could reach on July 13th, 2005. We implemented a small application that was run on all nodes. A host would send an UDP packet containing either no IP options or one of No-op, Record Route, or Timestamp options. The receiving host would reply with an echo UDP packet with no options, much like the `ping` utility. We also ran a similar test using the standard ping utility, with and without the `-R` command line switch, which turns the Record Route IP option on. The results of these experiments are summarized in Table 1. We were able to collect measurements for

Packet Type	Success
ICMP	83.49%
ICMP-RR	49.53%
UDP	75.39%
UDP-NOOP	55.54%
UDP-TS	47.62 %
UDP-RR	48.51 %

Table 1: Results from sending packets with and without options between all pairs of 195 PlanetLab nodes

37,145 pairs of hosts,[2] and observed that about 40% of the paths on which ICMP packets were successfully delivered failed to deliver equivalent packets with options. For the UDP packets in our tests, this number is around 30% for the different options. These numbers are similar to results in [6], where the authors do experiments with TCP packets with and without options, suggesting that they were not artifacts of the particular set of paths we chose, but rather a more widespread state of affairs on the Internet.

The next question then was to verify whether packets with our Annotations do get delivered. When adding options to packets, we have to fix the IP header length field and the IP checksum to keep the packet a valid IP packet. However, transport protocols generally have checksums as well, and the extra data will make for invalid higher layer packets if received by end applications. Clearly, the packets have to be deannotated before being delivered to the transport layer in the end host, but an interesting question is how the packets traverse the network. Any network element along the path that checks transport layer semantics, such as firewalls, NATs, or intrusion detection systems, may drop packets with an invalid TCP checksum, which depends on the packet payload. To test this we modified the `tcptraceroute` tool to use annotated TCP packets. This tool works like `traceroute`, but uses TCP SYN packets as the probing packets, and allows one to verify where in the path a packet is dropped, if at all. We selected 157 PlanetLab nodes (a subset of the 195 nodes in the other experiment that were reachable from the source for this experiment at the time) and ran `tcptraceroute` from a single computer outside of any firewall, using both annotated and normal packets. In 4 of these paths the annotated packets were dropped before getting to the end host, and in all other cases they got to the end host. In the blocked cases, the packets were always dropped close to the destination (2 to 4 hops before the end host), and not in the core.

The conclusion is that although some routing elements may decide to drop packets with bytes added disregarding the integrity of transport protocol packets, this was very rare in our experiments, and always very close to the end points. This is the worst case for annotations. It is always possible to fix the payload in a transport protocol-dependent way, such as making the TCP checksum valid, to eliminate the dropping of the packets by transport-layer aware routing elements. In this case, care must be taken to ensure that all packets get deannotated. We envision in most cases that the TCP checksum will remain unchanged, ensuring that endhosts do not mistakenly receive annotations into their TCP bytestream.

6 Annotation Challenges and Opportunities

The introduction of annotations raises several issues. First, since we are appending data to the payload of IP packets, we have to ensure that those packets do not reach end hosts unaware of it. This requires that annotation-aware routers reside on each path from annotation point to destination. In general, even if that device does not understand any particular annotation, it must at least be able to strip off annotations so that the receiver is unaffected. As described in Section 3, removing annotations does not require state.

This raises the issue of how annotation-aware routers and endhosts can determine if any nodes on the path to the destination support annotations. This information is important since packets should only be annotated if a node down the path can receive them. A simple way of determining this information is to make use of a modified version of the *traceroute* tool. This new tool would send packets with ascending TTL fields and annotations. Annotation routers that receive those traceroute packets would respond with an annotation of their own. In this way a sender or network appliance can determine the capabilities of routers on the path. By periodically issuing these probes, a sender could cease annotation packets if the underlying path changes in such a way that annotation-aware routers are no longer on the path. By choosing an appropriate probe frequency, the sender can bound the amount of time that annotated packets are sent to non-annotation parts of the network.

Another issue raised is how annotations interact with network MTUs (maximum transmission units). It is clear that we cannot add new data to a packet that is already as large as the MTU. In practice, we

[2]Because of connectivity issues in PlanetLab, we couldn't collect the results for all 195×194 pairs. We gathered reports from 192 of the 195 nodes. Of these, 190 had measurements to all other 194 nodes, while one had measurements to 172 and another to only 113 nodes.

have found that packet sizes tend to fall into a trimodal distribution of small packet (about 60 bytes), medium sized packets (about 600 bytes), and large packets (close to 1500 bytes). For example, on a subset of the UC Berkeley Computer Science network (consisting of about 40 desktops and a dozen servers), we found that about 90% of packets are less than 1400 bytes. It is unclear that this distribution universally holds, however by restricting annotations to packets with sufficient room to hold them, we expect to be able to annotate a significant number of packets.

The need for packet annotations has also grown through the deployment of a variety of middleware boxes. They are devices in the network that execute an application on packet flows. Examples of such boxes include intrusion detection systems (IDSes), spam detection appliances, firewalls, network address translators, iSCSI storage switches, VPN endpoints, and others. Some applications like IDSes can benefit from inter-device communication. By taking advantage of multiple vantage points, their ability to detect attacks can be increased. The availability of the annotations will give middleware boxes a way of sharing network observations with each other by annotating information into packet streams. Our payload extension annotations enable multiple network appliances to interact with each other despite intermediate legacy switches and routers. As long as the other appliances are also annotation aware, they could be stripped from packets before they reach the destination.

7 Conclusions

We have proposed a mechanism to support packet annotations in the Internet. This mechanism stores annotation data at the end of packets, by extending the IP payload. Other than the length and IP checksum fields, no IP, TCP, or UDP header fields are modified. We tested our scheme on PlanetLab and found that in all but four paths our annotations arrived at the destination. This is significantly better than using IP options, which were dropped about half the time. We hope that packet annotations will provide a useful and lightweight alternative to accommodate new network extensions and appliances.

Acknowledgements

We would like to thank Mark Allman for his invaluable contributions to this work.

References

[1] Mark Allman. A(nother) technique for improving transport protocol performance in lossy networks, http://www.icir.org/mallman/talks-/ceten-icir-lunch.ps. *ICIR Wednesday Lunch*, April 2004.

[2] S. Deering and R. Hinden. RFC 2460: Internet Protocol, Version 6 (IPv6) specification, December 1998.

[3] S. Hanks, T. Li, D. Farinacci, and P. Traina. RFC 1702: Generic routing encapsulation over IPv4 networks, October 1994. Status: INFORMATIONAL.

[4] A Jain, S Floyd, M Allman, and P Sarolahti. Quick-Start for TCP and IP. IETF draft-tsvwg-quickstart-00.txt, May 2005.

[5] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 89–102, New York, NY, USA, 2002. ACM Press.

[6] Alberto Medina, Mark Allman, and Sally Floyd. Measuring the evolution of transport protocols in the internet. *ACM Computer Communication Review*, 2(35), April 2005.

[7] PlanetLab, http://www.planet-lab.org.

[8] J. Postel. RFC 791: Internet Protocol, September 1981.

[9] Stefan Savage, David Wetherall, Anna R. Karlin, and Tom Anderson. Practical network support for IP traceback. In *SIGCOMM*, pages 295–306, 2000.

[10] I. Stoica and H. Zhang. Lira: An approach for service differentiation in the internet. In *Proceedings of Nossdav*, Jun 1998.

[11] Ion Stoica. *Stateless Core: A Scalable Approach for Quality of Service in the Internet*. PhD thesis, Carnegie Mellon University, December 2000.

[12] Ion Stoica, Scott Shenker, and Hui Zhang. Core - stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *SIGCOMM*, pages 118–130, 1998.

[13] Avi Yaar, Adrian Perrig, and Dawn Song. An endhost capability mechanism to mitigate DDoS flooding attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004.