

THE PITFALLS OF DISTRIBUTED PROTOCOL DEVELOPMENT: Unintentional Interactions between Network Operations and Applications Protocols.

Hans Kruse
J. Warren McClure School of Communication Systems Management
Ohio University
Athens, OH 45701
{kruse@ohiou.edu}

ABSTRACT

The internet today is not as transparent as was envisioned at its inception. While most of the responsibility for handling advanced communications functions remains concentrated in the end systems, we are deploying more sophisticated processing inside the network. Applications and application-level protocols such as end-to-end security and congestion control can interact in unexpected ways with network based functions such as proxies, address translators, packet filters, and nodes that perform traffic shaping and differentiated services functions. In this paper we review several recent examples of these types of interactions and extract from these examples a sense of the basic protocol functions that cause potentially harmful interactions. Currently the mitigation of these interactions has to be addressed on a case by case basis in each development effort. We suggest possible formal remedies that, if accepted as part of the protocol development process, can improve the probability that protocol interactions are handled correctly. We also propose the creation of a diagnostic capability that can assist an application user or administrator in locating the cause of an applications failure without exposing the network to denial of service attacks inherent in the use of automatically generated network management response messages.

INTRODUCTION

The rapid growth of the Internet and the applications being supported on it put new stresses not only on the technical requirements for the hardware and software needed to operate the network, but also on the quality of the standards development process. The Internet has historically used a very rigorous peer review and "peer pressure" system to insure that the network was based on a predictable infrastructure. On top this infrastructure, applications could be developed with a very reasonable expectation that they would universally function within the network. Conflicts between standards were discovered and resolved during the peer review process (although some might argue that the resolution process was sometimes more political than technical).

Today the size of the community of Internet application developers, protocol developers, and network operators, has grown to a size where we begin to see the breakdown of this review process. There is still a very solid peer review process aimed at insuring protocol quality. However, there are now a number of increasingly disjointed groups involved in the evolution of the Internet. Commercial applications are developed mainly with a focus on the end-user host and client systems and their requirements. Another group of developers focuses on existing and new end-to-end transport protocols, as well as end-to-end security requirements. Yet another group is working on protocols and procedures to improve the efficiency and service delivery inside the network. While some researchers/developers are active in all groups, recent examples seem to indicate that this overlap is diminishing, or at least that the ability of one group to influence developments in another one is eroding.

In this paper we examine a number of recent examples of protocol interactions in an effort to understand the origins of the design decisions, and the effect of the observed interactions on the end-user's ability to deploy the new protocol. We are particularly interested in examining the protocol structures involved, and to determine why the traditional protection against protocol interactions inherent in layered protocols could not prevent the observed problems. The examples are drawn from e-mail list discussions and work-in-progress documents, and include: (1) the difficulty in deploying network address translators and performance enhancing proxies together with IP level security, (2)



the problems encountered in an attempt to deploy explicit congestion notification while certain types of tunneling protocols are in use, and (3) the apparent lack of backward compatibility found in a recently proposed differentiated services protocol.

It is the intent of this paper to begin a more formal analysis of the protocol development process in a network as large and complex as the Internet, and to point out the need for additional technical and procedural safeguards to allow an efficient and safe deployment of new capabilities at all protocol levels.

END-NODE AND NETWORK FUNCTIONS

Carpenter [2] recently defined the so-called "end-to-end problem" as a description of the "growing pains" that are experienced in the internet today. He points to the intent of the original internet design as one that places all intelligence in the end nodes. In this design the network simply acts as a transparent transport mechanism for individual packets. Each packet is labeled by globally unique source and destination addresses. There are several specific reasons why this simple model no longer applies:

- the use of firewalls and other packet filters at the boundary between the internet and corporate intranets;
- the use of network address translators (NAT) and performance enhancing proxies;
- the use of IP tunnels for a variety of purposes, including mobility, security, virtual private networks, and the deployment of new mechanisms like multicast or IPv6;
- the deployment of quality of service mechanisms that alter the original best-effort delivery strategy designed into the internet protocols.

These deployments create three distinct sets of issues affecting transparency:

1. Some network devices either attempt to read or read and modify portions of the transmitted packet which the sending system assumed to be fixed [3].
2. The use of IP tunnels creates the design issue of how to construct the second, "outer" IP header upon tunnel ingress, and the more complicated issue of whether the original, "inner" IP header needs to be modified upon tunnel egress based on changes that intermediate nodes made to the outer header.

3. Some devices on purpose, or due to limits in their design, prevent some packets from traversing that device.

The sections below describe several examples of these interactions. We will then present a series of recommendations for future protocol designs.

NAT AND PERFORMANCE ENHANCING PROXIES

Network Address Translators [5], [6] are in wide-spread use at the edge of the internet. They allow the use of private IP address space in a private intranet, while maintaining connectivity to the internet through one or more global IP addresses. Since many applications assume that the end-system address is globally unique, NAT usually requires Application Level Gateways which modify application-specific sections of the packet where the end-system address has been embedded. These gateways cause changes in the packet that are unanticipated by the end systems.

Performance Enhancing Proxies (PEP) [1] are used in networks with unusual link characteristics. These proxies may attempt to read transport-level information in the packet, or they may add and delete packets from the flow. Many of these proxies can be bypassed by flows that do not permit such interactions, at the risk of suffering from poor performance.

Both NAT and PEP devices vastly complicate the deployment of IP-level security between end systems [3], and they may cause other failures that can be quite difficult to diagnose [2]. For the purposes of this paper it is important to note that NAT and PEP devices usually do not report the fact that they either failed to correctly handle a packet or were bypassed.

IP TUNNEL ISSUES

IP tunnels are defined as a section of the network in which IP packets are encapsulated inside a second IP header (often called the "outer" header). The tunnel is designed to transport packets between two intermediate points in the network, without making reference to the actual IP packets during that section of the packet's path. Tunnels can serve a number of purposes, including:

- Transport of service types that are not supported by intermediate nodes, such as multicast, or IP version 6.

- Creation of virtual private networks. In this scheme, packets between two sites are carried over IP tunnels to isolate the original packet completely from the addressing and routing requirements of the internet. A similar type of tunnel can be used to connect an off-site user to the corporate network in such a way that traffic from that user appears to be local to the corporate network.
- Security tunnels provide safe passage between two nodes at the edges of trusted domains. Inside the tunnel the original IP packets are encrypted and therefore completely inaccessible.

The use of tunnels for any purpose creates specific types of protocol interaction problems. Specifically, how should the outer IP header be constructed at the tunnel ingress point? In general it seems reasonable to copy fields from the original IP header. However, that is not always the correct approach. In networks that provide quality of service control through resource reservation [4] or differentiated services [8], the tunnel may be used to traverse a portion of the network that cannot provide these services, and therefore requires that some of the original IP settings not be copied. In other cases [7], the ability of the tunnel egress point to provide certain types of processing will determine how to construct the outer IP header.

By far the more complicated issue arises upon tunnel egress. Some portions of the outer IP header may have been modified during tunnel traversal. Examples include the update of header fields that mark the packet as being in a particular differentiated services group, or the update of fields designed to provide explicit congestion notification to end-points. The tunnel egress node must merge the original IP header with the - possibly modified - outer IP header. The rules for doing this are often not unambiguous, and different rules may emerge from different goals of the network and protocol designer. For example, from a performance and application perspective one would wish to propagate congestion notification information across security tunnels. From a security perspective, one may wish to discard the entire outer IP header regardless of its content to prevent attacks based on the ability of hostile systems to modify the unprotected outer IP header inside the tunnel.

PACKET FILTERS

Several devices can cause packets to be discarded before they reach the end system they were addressed to. Most prominently, firewalls are

designed to do just that for all packets that have not been entered in a permission list. Typically the sending system is not notified of the fact that the packet was dropped (although auditing of dropped packets can be performed at the site of the firewall). It is also significant in the context of this paper that most firewalls will not permit network management packets (such as ICMP) to pass through, in order to prevent possible attacks using the management protocol.

Other devices, specifically NATs and some proxies, are forced to drop packets for which they have insufficient information to allow them to be processed. A NAT (Network Address and Port Translator) device, for example, cannot forward a connection request from the internet to the private network side unless an administrative mapping has been provided for the port requested in the incoming packet. Other packets may be dropped or misrouted because the NAT did not have a correct application level gateway and failed to make corrections in the packet to allow the application's peer to respond. Finally, authenticated or secured packets will be examined by the security software at the receiving end, where the modifications made by a NAT or PEP device will be interpreted as illegal tampering; the packet will be discarded by the security software. While the packet drop is an auditable event, the sender of the packet is not usually notified of the packet drop.

RECOMMENDATIONS

Until now, the protocol interactions outlined in this paper have been addressed in an ad hoc fashion. During design or testing potential interactions are discovered and possible solutions are offered. [7] and [4] are examples of these design steps. It is often at this stage that different engineering groups must reconcile conflicting goals and view points.

Worse yet, in many cases the end user is left to discover just what will and will not work for their network and applications. As pointed out in [2], the failure of certain application can be intermittent and extremely difficult to diagnose in a network that is subject to the protocol interaction described here.

Our analysis of the examples mentioned here and the underlying reasons for the difficulties caused by these protocol interactions suggest a number of possible remedies. None of these are easy or simple to

implement. Especially in the short run they are not generally practical in the context of protocols being deployed now. However, it is crucial that we begin to put protocol development on a track that makes it easier to deal with a network where functionality is no longer housed only in the end systems as originally envisioned.

We recommend remedies of two types: (1) Protocol design principles that improve everyone's ability to develop and deploy new protocols more reliably; and (2) diagnostic procedures that at least make it easy for an application to inform the user that a connection attempt has failed due to an intermediate device's actions, and why. In particular, we suggest the following principles:

1. Avoid overloading of header fields:

Any time an existing header field is re-used for a new purpose, potential problems of backward compatibility arise. This is for example the case in the Differentiated Services protocol which uses the old IP TOS field in IP version 4. Since IP version 6 provides better support for IP options, it may be advisable to create new option fields for new network functionality. Defining access rules as called for below is certainly much easier when a field only serves one purpose.

A less obvious version of field overloading is the use of global network addresses in application layer protocols. Any time an application can further abstract information it tries to convey to its peer, network management problems are eased; usually at the cost of network and/or application performance. For example, using service names that require a DNS lookup will slow an application down, but it allows the network to use a defined translation mechanism to deal with global addresses that are either not unique or not stable over time.

2. Define header access rules:

Any new network header field should be proposed only after a set of access rules have been defined and reviewed. These rules will state if the field can be read inside the network (especially outside the security perimeter), if the network is allowed to modify the field, and the security implications of hostile modification of the field. This set of rules would vastly simplify exception processing

within network devices (see below), and allow for a fairly straightforward definition of IP tunnel ingress and egress rules.

3. Create a diagnostic feedback capability:

As mentioned earlier, one of the most troublesome aspects of protocol interactions is the fact that they are often very hard to diagnose. In our opinion it will be necessary to deploy better diagnostic capabilities into devices that have the potential to disrupt packet flows. This is particularly true if the audit trail generated by the device is not accessible to the end user(s) generating the packets. This could happen for example if a PEP device fails as IP level security is turned on at an unrelated user site.

There are already examples, primarily in the area of routing and packet fragmentation failures, where ICMP messages are generated to advise upstream devices of the problem. As noted above, however, many of these messages are blocked by firewalls before they can return to an end user. The ICMP blocking is most often deployed to prevent denial of service attacks. Simply returning an ICMP message upon a failure, for example, in a PEP or NAT device, carries with it the danger that the message may be blocked, that the message type may be abused for a denial of service attack, or that the diagnostic messages create a load problem for the network.

We therefore suggest the creation of a "diagnostic packet" header option. Failure messages would propagate back to the sender only for packets marked as diagnostic. This would ensure that failure message traffic would be created only when a user or administrator is actually actively trying to figure out why a connection attempt is failing. Diagnostic traffic can be controlled and bandwidth limited to prevent it from being abused for denial of service attacks. Finally, firewalls can monitor outbound diagnostic traffic and allow the return of failure messages in response to those diagnostic requests.

CONCLUSIONS

We have presented an analysis of the challenges being faced in the development and deployment of new protocols and network

capabilities. The fact that "intelligent" processing can take place inside the network as well as in the end nodes leads to a network that is not as transparent as the internet designers envisioned. We have discussed a few recent examples of the difficulties and presented a number of possible remedies. Future work is obviously needed to create a consistent environment for protocol development that takes non-transparency into account in much the same way as security is (or should be) taken into account in today's protocol development efforts.

BIBLIOGRAPHY

- [1] M. Allman, D. Glover, L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", RFC 2488, 1999.
- [2] Brian Carpenter, "Internet Transparency", work in progress; available on-line as <http://www.ietf.org/internet-drafts/draft-carpenter-transparency-03.txt>
- [3] Hans Kruse, "Protocol Interactions and Their Effects on Internet-Based E-Commerce", Proceedings of the Second International Conference on Telecommunications and Electronic Commerce (ICTEC), Nashville, TN, October 6-8, 1999. (Online as <http://www.csm.ohiou.edu/kruse/publications/HK-ICTEC99.pdf>).
- [4] A. Terzis, J. Krawczyk, J. Wroclawski, L. Zhang " RSVP Operation Over IP Tunnels", RFC 2746, 2000.
- [5] T. Hain, "Architectural Implications of NAT", work in progress; ; available on-line as ; available on-line as <http://www.ietf.org/internet-drafts/draft-iab-nat-implications-04.txt>.
- [6] M. Holdrege, P. Srisuresh, "Protocol Complications with the IP Network Address Translator (NAT)" , work in progress; ; available on-line as ; available on-line as <http://www.ietf.org/internet-drafts/draft-ietf-nat-protocol-implications-01.txt>.
- [7] S. Floyd, D. Black, K. Ramakrishnan, "IPsec Interactions with ECN", work in progress; available on-line as <http://www.ietf.org/internet-drafts/draft-ipsec-ecn-00.txt>.
- [8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Service", RFC 2475, 1998.