Some Notes on Threat Modelling Congestion Management

Eric Rescorla
RTFM, Inc.
ekr@rtfm.com

Abstract

When designing any piece of Internet technology, one must consider the question of how it behaves in the presence of malicious behavior; congestion control is no exception here.

1 Introduction

IETF congestion protocols are generally designed under a nonadversarial model: If Alice and Bob want to exchange data at the full rate of their access links, without regard to the impact on other flows sharing the same intermediate network links, nobody from the IETF Transport Area will come to their houses and arrest them for violating RFC 5405 [EF08]. Rather, the IETF attempts to design protocols that provide acceptable congestion behavior *when used* and assume that network providers will appropriately traffic shape non-compliant traffic to prevent it from consuming all available bandwidth, thus addressing the adversarial case to some extent. This paper addresses a number of aspects of adversarial congestion control that are not well handled within this paradigm.

2 Attacker Types And Objectives

In general, an attacker will have one of two objectives with respect to a given flow:

- Cause more traffic to be sent than otherwise would
- Cause less traffic to be sent than otherwise would

Note that we are deliberately sidestepping the question of what division of traffic is appropriate. The IETF congestion algorithms are designed to produce a given division of traffic in the equilibrium state. Our sole concern is that it be difficult for the attacker to alter that equilibrium, not that that equilibrium is actually equitable. However, for convenience, we refer to the fraction of traffic that a flow should have as its *fair share* and a flow which uses more than its fair share as *overconsuming* while a flow which uses less than its fair share as *underconsuming*.

We must also consider three different attacker types:

- Unilateral attack by a sender or receiver
- Attack by a cooperating sender/receiver pair
- Attack by a third party who is neither a sender or receiver for a given flow

In general, any combination of attacker and objectives is possible. For instance, a sender and receiver might wish to cooperate to exchange more than their fair share of traffic on a given path. More exotically, a third party might want to act to cause a sender and receiver to underconsume (to leave more room for his own traffic) or to overconsume (as a form of a denial of service attack.) A properly designed congestion management system should prevent all of these attack modes.

3 Some Specific Cases

3.1 Two-party Overconsumption

The paradigmatic case is one in which a sender and receiver cooperate to exchange more than their fair share. As mentioned above, because both sides are explicitly noncompliant, this is not strictly a protocol issue. For that reason, this case has been extensively studied (see, for instance, [FF99]) and is probably the easiest to deal with via traffic shaping by intermediate network elements. The major challenge for those elements is to determine what sending rate to clamp a given flow/sender to (or more generally, how to prioritize traffic between flows). Two major approaches are available:

- Attempt to classify flows by sender and divide available bandwidth between senders regardless of which protocol they are using.

- Attempt to identify noncompliant flows (e.g., those not properly responding to congestion signals) and penalize them.

There are difficulties with both of these approaches, especially as traffic shaping must often be done at high speed in the network core. The first approach requires the ability to determine which flows correspond to a given sender/receiver pair (note that if one only attempts to provide fairness between individual flows, then senders and receivers will just use multiple flows). This can be problematic in face of an attacker which can forge addressing information since they can make a given logical flow appear to be between multiple hosts.[1] The second approach requires the ability to classify traffic and is therefore subject to techniques which make one form of traffic appear to be another. Consider, for instance, a flow which appears to be TCP but actually uses non-TCP-compliant rate control: An intermediate network element would need to do extensive analysis of such a flow in order to determine that it was misbehaving.

3.2 Receiver-Initiated Overtransmission

Moving into the field of protocol design, it is possible to have cases where the sender wishes to be compliant but the receiver does not (the dual of this case, where the sender does not wish to comply, is a straight traffic flooding attack and cannot really be addressed by protocol mechanisms). The receiver might wish to avoid congestion control for a number of reasons, ranging from simply getting more than its fair share of traffic to mounting what's effectively a DoS attack on the sender.

Savage et al.[SCWA99] describe a number of attacks and defenses for the specific case of TCP. For instance, they describe an attack in which the receiver splits up an acknowledgment into multiple acknowledgements and (because TCP's algorithms are phrased in terms of segments) forces the senders's congestion window to be much larger than it should be. [Note: I do not know if this has been fixed in more modern TCP implementations.] Another example is that the receiver can send acknowledgements for packets it has not yet received, thus increasing the sender's transmission rate; this mechanism does not preserve reliability and so in the context of reliable protocols is primarily useful for mounting an amplification DoS attack on the sender. Thus, for instance, a receiver with a very slow network could cause the sender to transmit far more data than he can actually receive. Moreover, in the context of *unreliable* protocols such as those used for voice and video can be used to increase sending rate (and hence quality) at the cost of some packet loss (though potentially the packet loss will be small since since other senders will get out of the way.)

The general strategy for preventing attacks of this type is, as Savage et al., indicate, to design the protocol in such a way that if the receiver provides bogus feedback it is either ignored or causes the sending rate to decrease rather than increase. For instance, false ACKing could be prevented by having the ACKs include proof that the receiver had seen the data being acknowledged. As far as I know, no widely deployed protocol is designed using these principles.

3.3 Third-party Traffic Manipulation

In an environment without overall network-level fairness mechanisms, an attacker who wishes to overconsume can simply send more than his fair share of traffic and expect other, compliant, flows to back off, thus making room for his traffic. However, if there are overall fairness mechanisms in place, then this strategy will be less effective. However, an attacker can potentially manipulate the congestion algorithms of compliant implementations to cause them to back off, thus reserving more capacity for itself. For instance, the Security Considerations section of RFC 5681 states:

> This document requires a TCP to diminish its sending rate in the presence of retransmission timeouts and the arrival of duplicate acknowledgments. An attacker can therefore impair the performance of a TCP connection by either causing data packets or their acknowledgments to be lost, or by forging excessive duplicate acknowledgments.

Similarly, it is possible for an attacker to mount attacks like those described by Savage in order to *increase* the traffic sending rate between two endpoints.

In general, it does not seem possible to stop an on-path attacker who can block packets from throttling other people's streams, since he can always simulate network congestion and we want flows to back off when they believe they have experienced congestion. However, it should be possible to prevent attack by off-path attackers and on-path attackers who can only inject packets. The general strategy is to cryptographically protect feedback packets so that an on-path attacker cannot

4 The RTCWEB Setting

The RTCWEB [Alv12] setting is an interesting combination of first and third-party attackers. In RTCWEB, we have two media endpoints which are exchanging large

[1]Obviously, this attack technique does not work when the unit of fairness is the access link.

traffic volumes but under partial control of a third party signaling server which may be malicious and which mediates call setup between the parties. Moreover, in the general case it is possible for a signaling server to recruit new endpoints without their consent (e.g., by using an advertising network to emplace malicious JS). The underlying technical assumption is that the *browser* acts as the trusted computing base which enforces correct behavior in the face of even malicious JS. Thus, it is not safe to send traffic just because the JS tells you to.

The solution RTCWEB has adopted has been to use ICE [Ros10] as a coarse-grained check of communications consent. The browser refuses to send traffic between two endpoints until ICE checks have completed [Res12], thus verifying that the media sink desires to receive the traffic. Periodic checks will then be required in order to continue sending media. Thus, a receiver which wishes to not receive traffic at all or which detects obvious noncompliance can stop responding to STUN checks, thus completely throttling the media.

However, this is a relatively coarse mechanism (partly due to the lack of specified congestion control algorithms for media traffic); the receiver can only force traffic to stop appearing, not adjust the sending rate. In order to do a better job we would need a congestion control system for RTCWEB based on the principles suggested in Section 3 but which was also suspicious of the signaling server. In particular, it needs to be designed so that:

- It is not possible to generate explicit congestion feedback without being on-path.
- On-path attackers cannot send explicit congestion feedback that has a bigger impact on sending rates than they could achieve just by blocking traffic.
- Feedback which tends to increase the sending rate must include some information known to the browser but not made available to the signaling server (thus preventing the signaling server from increasing the sending rate unilaterally).

While we can do an acceptable job of controlling misuse with the system as-is, if we are going to design a new set of mechanisms, adherence to these principles is an important part of making them as misuse resistant as possible.

References

[Alv12] H. Alvestrand. Overview: Real Time Protocols for Brower-based Applications. draft-ietf-rtcweb-overview-04, jun 2012.

[EF08] L. Eggert and G. Fairhurst. Unicast UDP Usage Guidelines for Application Designers. RFC 5405, November 2008.

[FF99] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Trans. Netw.*, 7(4):458–472, August 1999.

[Res12] E. Rescorla. RTCWEB Security Architecture. draft-ietf-rtcweb-security-arch-02, jun 2012.

[Ros10] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, April 2010.

[SCWA99] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. Tcp congestion control with a misbehaving receiver. *ACM COMPUTER COMMUNICATIONS REVIEW*, 29:71–78, 1999.