Sorcerer's Apprentice Syndrome

Sorcerer's Apprentice Syndrome (**SAS**) is a network protocol flaw in the original versions of TFTP. It was named after Goethe's 1797 poem "Der Zauberlehrling" (popularized by the "Sorcerer's Apprentice" segment of the 1940 animated film *Fantasia*), because the details of its operation closely resemble the disaster that befalls the sorcerer's apprentice: the problem resulted in an ever-growing replication of every packet in the transfer.

The problem occurred because of a known failure mode of the internetwork which, through a mistake on the part of the TFTP protocol designers, was not taken into account when the protocol was designed; the failure mode interacted with several details of the mechanisms of TFTP to produce SAS.

Statue of Mickey Mouse as the Sorcerer's Apprentice in *Fantasia* at Hong Kong Disneyland. In the story, the apprentice's enchanted broom multiplies uncontrollably.

Contents

Technical background

TFTP operates in simple lock-step: there is only ever one packet outstanding at any time, and every packet received by either party caused one packet to be sent in reply (until the termination of the transfer). The TFTP specification said that any time *any* packet was received, the receiver was *required* to send the appropriate reply packet. Thus, the receipt of a block of data triggered the sending of an 'acknowledgement', and the receipt of an acknowledgement triggered the sending of the next data block.

TFTP also, like all protocols designed to operate across an unreliable network, includes timeouts. After sending a packet, it expects a reply, so it starts a timer. If the timer expires with no reply received, it takes some action; typically re-sending the original packet.

Details

SAS occurred when a packet was not *lost* in the internetwork, but rather simply *delayed*, and later successfully delivered, after a timeout had occurred (on either side).

The timeout causes a second copy of the previous packet to be sent to replace the 'lost' packet. However, the first copy was not lost, and since, according to the TFTP specification, receipt of any packet *always* forced the generation of a reply packet, two replies were generated (one to each copy). Those forced the generation of two replies to them, and so on. A typical scenario was as follows:

- Computer S (source) sends data block X to computer D (destination)
- Computer D receives block X, and sends an acknowledgement for X back to S
- The packet containing the acknowledgement for X is delayed in the internetwork
- Computer S times out, and resends data block X to D
- Computer S receives the delayed acknowledgement for X, and sends data block X+1
- Computer D receives the second copy of block X, and sends another acknowledgement for X back to S
- Computer D receives block X+1, and sends an acknowledgement for X+1 back to S
- Computer S receives the second acknowledgement for X, and sends a second copy of data block X+1
- Computer S receives the acknowledgement for X+1, and sends data block X+2
- Computer D receives the second copy of block X+1, and sends another acknowledgement for X+1 back to S
- Computer D receives block X+2, and sends an acknowledgement for X+2 back to S

It will be seen that at this point the situation is now stable, and repeats; *every packet* from then on is duplicated (that is, two identical copies are sent across the internetwork).

Even worse, the increased number of packets being sent around the internetwork was likely to cause congestion, which was likely to cause a packet to be delayed past the timeout yet again, which would then cause yet *another* duplicate packet to be generated by a timeout, and from then on a *third* copy of each packet would be sent. Needless to say, at that point, the situation would usually snowball, and *further* copies would be generated — hence the name given to this pattern of behaviour.

For a small file, the transfer would complete, and the duplicate packets would eventually drain from the internetwork. If the file were large, however, congestive collapse would result, and only when the transfer failed would the mass of packets drain from the internetwork.

Solution

The fix to SAS involved modifying the TFTP specification to break the loop.[1] Only the *first* instance of a received acknowledgment should cause the next data block to be sent; further copies of the acknowledgment for a particular data block would be ignored, thus breaking the retransmission loop. In the new version of the protocol, a block would only be retransmitted on timeout.

This change also makes it possible to simplify the implementation of the receiving end (often, a bootstrap program written in a low-level language) by omitting the retransmission timer, as any lost packet would cause retransmission of the last packet sent by the sender. However, keeping the timer has its benefits, such as dealing with lost ACKs more efficiently.

References

1. Braden, Robert, ed. (October 1989). "Sorcerer's Apprentice Syndrome" (https://datatracker.ietf.org/doc/html/rfc1123#section-4.2.3.1). *Requirements for Internet Hosts -- Application and Support* (https://datatracker.ietf.org/doc/html/rfc1123) (rfc). IETF. pp. 43–45. sec. 4.2.3.1. doi:10.17487/RFC1123 (https://doi.org/10.17487%2FRFC1123). RFC 1123 (https://datatracker.ietf.org/doc/html/rfc1123). Retrieved 2012-10-05.