Is TCP's Nagle Algorithm Right for Me?

Martin Duke, 2015-14-12

Of all the settings in the TCP profile, the Nagle algorithm may get the most questions. Designed to avoid sending small packets wherever possible, the question of whether it's right for your application rarely has an easy, standard answer.

What does Nagle do?

Without the Nagle algorithm, in some circumstances TCP might send tiny packets. In the case of BIG-IP®, this would usually happen because the server delivers packets that are small relative to the clientside Maximum Transmission Unit (MTU). If Nagle is disabled, BIG-IP will simply send them, even though waiting for a few milliseconds would allow TCP to aggregate data into larger packets.

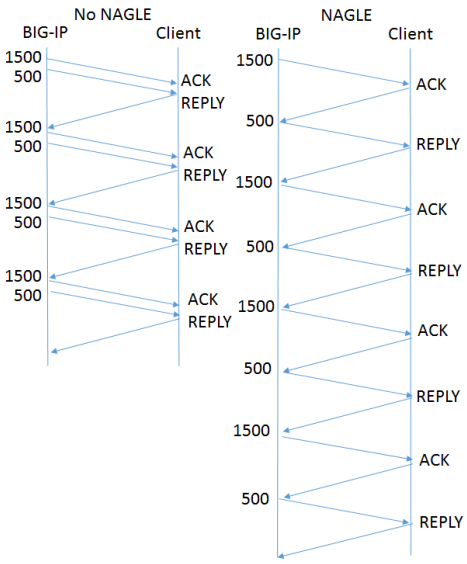
The result can be pernicious. Every TCP/IP packet has at least 40 bytes of header overhead, and in most cases 52 bytes. If payloads are small enough, most of the your network traffic will be overhead and **reduce the effective throughput** of your connection. Second, clients with **battery limitations** really don't appreciate turning on their radios to send and receive packets more frequently than necessary. Lastly, some routers in the field give preferential treatment to smaller packets. If your data has a series of differently-sized packets, and the misfortune to encounter one of these routers, it will experience **severe packet reordering**, which can trigger unnecessary retransmissions and severely degrade performance.

Specified in RFC 896 all the way back in 1984, the Nagle algorithm gets around this problem by holding sub-MTU-sized data until the receiver has acked all outstanding data. In most cases, the next chunk of data is coming up right behind, and the delay is minimal.

What are the Drawbacks?

The benefits of aggregating data in fewer packets are pretty intuitive. But under certain circumstances, Nagle can cause problems:

- In a proxy like BIG-IP, rewriting arriving packets in memory into a different, larger, spot in memory taxes the CPU more than simply passing payloads through without modification.
- If an application is "chatty," with message traffic passing back and forth, the added delay could add up to a lot of time. For example, imagine a network has a 1500 Byte MTU and the application needs a reply from the client after each 2000 Byte message. In the figure at right, the left diagram shows the exchange without Nagle. BIG-IP sends all the data in one shot, and the reply comes in one round trip, allowing it to deliver four messages in **four round trips**. On the right is the same exchange with Nagle enabled. Nagle withholds the 500 byte packet until the client acks the 1500 byte packet, meaning it takes two round trips to get the reply that allows the application to proceed. Thus sending four messages takes **eight round trips**. This scenario is a somewhat contrived worst case, but if your application is more like this than not, then Nagle is poor choice.



- If the client is using delayed acks (RFC 1122), it might not send an acknowledgment until up to 500ms after receipt of the packet. That's time BIG-IP is holding your data, waiting for acknowledgment. This multiplies the effect on chatty applications described above.

F5 Has Improved on Nagle

The drawbacks described above sound really scary, but I don't want to talk you out of using Nagle at all. The benefits are real, particularly if your application servers deliver data in small pieces and the application isn't very chatty. More importantly, F5® has made a number of enhancements that remove a lot of the pain while keeping the gain:

- **Nagle-aware HTTP Profiles**: all TMOS HTTP profiles send a special control message to TCP when they have no more data to send. This tells TCP to send what it has without waiting for more data to fill out a packet.
- **Autonagle:** in TMOS v12.0, users can configure Nagle as "autotuned" instead of simply enabling or disabling it in their TCP profile. This mechanism starts out not executing the Nagle algorithm, but uses heuristics to test if the receiver is using delayed acknowledgments on a connection; if not, it applies Nagle for the remainder of the connection. If delayed acks are in use, TCP will not wait to send packets but will still try to concatenate small packets into MSS-size packets when all are available. [**UPDATE:** v13.0 substantially improves this feature.]
- One small packet allowed per RTT: beginning with TMOS® v12.0, when in 'auto' mode that has enabled Nagle, TCP will allow one unacknowledged undersize packet at a time, rather than zero. This speeds up sending the sub-MTU tail of any message while not allowing a continuous stream of undersized packets. This averts the nightmare scenario above completely.

Given these improvements, the Nagle algorithm is suitable for a wide variety of applications and environments. It's worth looking at both your applications and the behavior of your servers to see if Nagle is right for you.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com