The ISP Column
A column on things Internet

October 2024
Geoff Huston

The Size of Packets

We've now been running packet-switched networks for many decades, and these days it's packets and not virtual circuits lie behind most of the world's digital communications service. But some very fundamental questions remain unanswered in this packet-switched world. Perhaps the most basic question is: "How big should a packet be?" And, surprisingly enough, there is no clear answer!

The pragmatic default Internet answer these days is that an Internet packet is between 20 and 1,500 bytes in size. Any bigger and the packet is likely to encounter packet fragmentation with its attendant issues of heightened risk of packet discard. Any smaller and the IP packet header is fatally truncated. Most hosts and applications stick inside the lanes and send packets within this size range.

We are going to be looking at *bits* and *bytes* here, so a quick word about terminology. I have always used the term *byte* to refer to an 8-bit grouping. Strictly speaking a *byte* is a unit of memory size, and while it is usually 8 bits in size, some computer architectures have used different grouping size. To resolve any potential ambiguity, maybe I should use the term *octet* to refer to 8-bit groupings, but I've tended by life-long habit to just use *byte* in the sense of an 8-bit word, as I have done here.

See the Wikipedia page on the subject "Byte" for a longer treatment of this topic if you are interested.

This was not always the case. In September 1981 RFC 791, the Internet Protocol Specification, was published. This specification had the advice that IP hosts must be prepared to accept IP packets of up to 576 bytes (whether they arrive whole or in fragments). Packets larger than 576 bytes were to be used only if the sending host had some assurance that the destination (and all the active network elements along the packet's forwarding path) were prepared to accept datagrams larger than 576 bytes. The document explains the rationale for this choice: "The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information. For example, this size allows a data block of 512 octets plus 64 header octets to fit in a datagram. The maximal [IPv4] internet header is 60 octets, and a typical internet header is 20 octets, allowing a margin for headers of higher level protocols."

Enter Ethernet

The original work on a radically different form of high speed networking for local area networks occurred in the mid 1970's, and the original published description, "Ethernet: distributed packet switching for local computer networks" dates from 1976.

Ethernet gathered momentum as the network technology of choice for local area computer networks as it was a simple and cost-effective high speed network solution over medium distances (of a couple of kilometres in size). The main advantage of Ethernet was its simplicity of decentralised design. In its simplest form, the network itself was a length of coaxial cable. Up to three such lengths could be joined by simple signal repeaters. There was no master controller, and each host managed its own data clocking and performed its own contention resolution. It was a common channel broadcast network, where every attached host could see every packet. Ethernet was an ideal match to the networking requirements of the emerging personal computer and workstation environment found in many computing environments as the industry moved away from the single central mainframe computer to a more distributed and diverse model of information processing

For 10Mbps Ethernet, frame (or packet) payloads were between 46 and 1,500 bytes in size, and the Ethernet framing format added a further 18 bytes (12 bytes of MAC addresses, 2 bytes of frame length and 4 bytes of CRC). These frame size numbers were the result of a trade-off between data timing and network utilization.

There is an ingenious relationship between the minimum Ethernet packet size and the original common bus (CSMA/CD) Ethernet collision sensing algorithm. The one thing Ethernet attempted to maintain was the property that a transmitter was always aware if another transmitter was active on the common wire at the same time, so that both transmitters could abort their transmission, back off and try again later. Hence, an Ethernet frame must be big enough that the leading bit of the packet must be able to propagate to the other end of the Ethernet network, and the collision with the leading edge of another transmitter must propagate back to the original transmitter before the frame's transmission ceases. That implies that the total end-to-end length of the LAN must be no longer than one half the minimum frame size.

For the maximum packet size, Ethernet opted to head down the path of maximising carriage efficiency rather than sacrificing speed and capacity for the sake of preserving implicit data timing integrity. In retrospect, it proved to be an astute design decision.

You could make the minimum Ethernet frame size smaller, but the maximal diameter of the LAN itself must shrink, or you can support physically longer LANs, but there is the consequent risk of undetected frame collisions for small frames, which will require a correction from an upper-level transport protocol.

The Speed of Light

These considerations relate to the speed of electromagnetic propagation over a copper conductor, which in turn relates to the speed to light in a vacuum.

The speed of light in a vacuum, or the physical sciences constant *c*, is probably the most researched constant in all of science. According to electromagnetic theory, its value, when measured in a vacuum, should not depend on the wavelength of the radiation. According to Einstein's prediction about the speed of propagation of light within the general theory of relativity, the measured speed of light does not depend on the observer's frame of reference; the speed of light in a vacuum is a universal constant.

Estimates of the value of *c* have been undergoing refinement since 1638, when Galileo's estimate of: "If not instantaneous, it is extraordinarily rapid" was published in "Two New Sciences". The currently accepted value is 299,792.458 kilometres per second.

The speed of propagation of electrical charge through a conductor is a related value; it, too, has been the subject of intense experimentation. Perhaps the most bizarre experiment was conducted in Paris, in April 1746, by Jean-Antoine Nollet. Using a snaking line of some 200 monks, connected by a mile-long iron wire, Nollet observed their reactions when he administered a powerful electric current through the wire. The simultaneous screams of the monks demonstrated that, as far as Nollet could tell, voltage was transmitted through a conductor "instantaneously".

The speed of light in glass or fibre-optic cable is significantly slower, at approximately 194,865 kilometres per second. The speed of voltage propagation in copper is slightly faster, at 224,844 kilometres per second.

Relating this value back to the design of Ethernet, a 10Mbps system running over copper wire will carry bits at 0.75 the speed of light in a vacuum, or at 224,844 kilometres per second. This means that 64 bytes at 10Mbps will be contained in 11.51 km of copper cable, or a "up and back" signal propagation length of 5.75km of conductor. The original Ethernet design specifications allowed for a total of sequence of three 500m runs of coaxial copper cable, plus allowance for 2 repeaters, and a generous overhead to tolerate various physical misconfigurations!

The maximal packet rate on a 10Mbps Ethernet was some 15,000 small packets per second, or a packet every 65 microseconds. With silicon processing clocking in the low MHz frequencies in the late 1980's then there is an approximate match between transmission performance and silicon switching capabilities.

What about the maximal Ethernet frame size of 1,518 bytes? The trade-off here is that longer maximal frame sizes allow for greater carriage efficiency, as the 18-byte frame overhead is amortised over a greater data payload, while shorter maximal packets reduce the average wait time where there is contention between multiple transmitters. A binary argument would propose either 1,024 or 2,048 bytes as a maximal payload size, and the 1,500 value feels like some form of compromise between these two values.

This would not be the first time that such compromises have appeared in networking technology. The design of the 48-byte ATM payload was apparently the outcome of a committee compromise between advocates of a 32-byte payload, intended to reduce potential jitter in ATM networks, and advocates of a 64-byte payload, intended to improve data carriage efficiency.

FDDI

For a short period of time in the early 1990's it looked as it the next generation of local networks would use a 100Mbps token-ring architecture, called FDDI (Fiber Distributed Data Interface). The network itself offered a payload range of zero (just the FDDI headers) to a maximally sized frame payload of up to 4,478 bytes.

In retrospect, its apparent that FDDI never really picked up a critical momentum of deployment. It was used in scenarios where aggregate network capacity in excess of 10Mbps was necessary. In many cases it did not replace 10Mbps Ethernets but acted as a common core that supported multiple 10Mbps edge Ethernets. At the time 10Mbps Ethernet adapters for hosts were far cheaper than FDDI, so individual hosts continued to use 10Mbps LAN connections while common servers may have used FDDI connections. However, mapping between Ethernet and FDDI is not a simple matter of reframing and passing the packet onward. The byte order on FDDI is "big-endian" while Ethernet uses "little-endian" bytes order (I have no idea why these two LAN technologies diverged at such a fundamental level, but it mirrored the divergence in bit-order storage models in processors that has lingered through to today). More importantly than bit twiddling, the maximum IP packet size on a FDDI network is larger than that of Ethernet, and a simple FDDI-to-Ethernet bridge unit would be forced to discard large FDDI packets.

Such hybrid FDDI/Ethernet deployments commonly used a router to perform the mapping between the two LAN technologies, and in the case of IPv4 large packets, they would be fragmented when passing a packet from a FDDI on to an Ethernet interface. Such a routing solution to interconnect "feeder" Ethernets to a FDDI "core" is by no means optimal and the overheads of router fragmentation and host reassembly eat into the underlying performance gains of the underlying 100Mbps FDDI system, and more importantly add to the cost of such hybrid LANs.

Faster Ethernet

In 1995 the IEEE 802.3u specification for 100Mbps Ethernet was published ("Fast Ethernet"). The system dispensed with a passive common bus and replaced it with an active switching hub to which hosts were attached (the use of Ethernet Switches had already happened in many 10M Ethernet deployments in any case as a means of improving overall network capacity). The Ethernet framing protocol was maintained, and the 10Mbps Ethernet packet size ranges were preserved. The potential peak packet rate lifted by a factor of 10 to 150,000 small (64-byte) packets per second to 823 large (1,518-byte) packets per second.

Three years later, in 1999, the IEEE 802.3ab was released, specifying 1Gps Ethernet. In another three years, in 2002, 10Gbps Ethernet was specified. The next factor of 10 speed increase took a little longer, and in around 2015 100Gbps Ethernet was entering the market. Current efforts are focussed on completing the Terrabit Ethernet (TbE) specification.

Across this program of speed increases for Ethernet, there is no basic change to the supported frame sizes. Once Ethernet dispensed with the common bus model and the associated contention detection and management and turned to what is in effect a collection of point-to-point serial connections using a packet switching hubs, there was no need to tie Ethernet packet sizes to a particular set of network constraints, and the desire to support backward compatibility, which supported plug-and-play hybrid Ethernet networks, far outweighed any marginal advantages in carriage efficiency by changing the base Ethernet packet size specification.

The result is that Ethernet at Tb speeds imply a peak packet rate of some 1.5B small packets per second, and 82M large packets per second.

Jumbo Packet Sizes

In 30 years, we've managed to push transmission speeds in local networks up by an astounding 100,000-fold. At the same time processor clock speeds have increased from some 100Mhz to around 5Ghz, or a far more modest (but still impressive) 50-fold increase. Today's silicon switching systems can only keep pace with network transmission systems as long as the majority of the packets are large.

It's not as if the issues of the increasing disparity between transmission and silicon processing clock speeds have gone completely unnoticed, particularly in the context of high-density datacentres. For more than two decades some vendors of Ethernet switches and network interfaces have supported Ethernet frame sizes larger than the IEEE 802.3 standard 1,518-byte maximum frame size. The scale of this change is not dramatic, and the common 9,000-byte maximum frame size is these so-called Ethernet *jumbo-frames* is just a 6-fold increase in frame size.

There are, however, a number of issues with these jumbo frames, including the inability of the IEEE to provide a single definitive standard for jumbo frames on 802.3 networks. Some network equipment supports larger jumbo frames, some smaller. The construction of end-to-end paths that use a variety of transmission technologies also does not help. Many of these links may use a common Ethernet frame format, but that does not mean that there is a consistent end-to-end maximum frame size beyond the 1,518-byte 802.3 standard. Hosts could perform a form of path MTU discovery if they so desired, but this discovery process consumes time. In many scenarios, the fastest approach is to avoid this MTU discovery step and just work with 1,500-byte packets as the effective MTU.

It is also worth noting that much of the host-based pressure to introduce larger frames was dispelled with the introduction of network interface cards that perform TCP segmentation offload. The host can send and receive large frames to the network interface, offloading a high per-packet processing load from the host processor, as the incremental load of interfacing to the network with smaller packets is handled in the network interface processor. With large send offload for example, a 65,535 byte data bundle, can be passed from the host to the network interface, which then performs segmentation into 45 1,460-byte TCP segments which are passed into the network.

The Internet Protocol does not have a clear story when it comes to large packets. IP (both V4 and V6) supports packets of up to 65,535 bytes in size (due to the 16-bit packet length field in the IP headers for both protocols), but in practice very large IP packets are not a robust choice for the public Internet. The problem lies in IP fragmentation. As we've already noted the path of greatest assurance without resorting to incurring the costs of path MTU discovery is to assume a MTU size of around 1,460 bytes. Larger packets are more likely to require fragmentation to be passed through the network, and issue is that trailing fragments do not contain the transport headers and present a problem for various forms of security-related middleware found in networks. The trade-off here is to incur the incremental cost of data segmentation in the sending host and send packets that have a high probability of not requiring any form of packet fragmentation, and to avoid this cost and run the risk of session performance degradation when recovering from silent packet discard.

Transmission vs Silicon

It seems to be somewhat curious that story of the Internet parallels the story of Ethernet, where the large scale increases in the clocking speed of data, from 10Mbps to 1Tbps, has been achieved within the same packet size range. The implication is that each increment in network speed comes at a cost of greater processing intensity, where the overall picture of processor clock speed improvements and memory cycle times are not increasing at anywhere near the same rates.

The processing response has been to make up the difference in increasing levels of parallelism in processors and load distribution through offloading. So far it appears that processing has been able to roughly keep pace with the network, but it's unclear how long this can last.

The pressures on processing speeds would be relieved, to some extent, if there was a shift to supporting some increase in maximum packet sizes in the network, but it's not clear if there is a critical mass of support behind such changes. Path MTU discovery has not been enthusiastically embraced.

I noticed in the Proceedings of the November 1989 IETF Meeting than the MTU Discovery Working Group was chartered to work in this problem, and anticipated some form of completion of this work by April 1990!

It appears that for many end-host network implementations the faster approach is to just pick an MTU with a high assurance of working through most networks and leave Path MTU Discovery as an option for those applications which could make productive use of the larger packet size even at the incremental cost of performing the discovery.

In that vein it's interesting to note that the IEEE 802.11WiFi specification defines an MTU of 2,304 bytes, yet it appears that most host implementations use an MTU value of 1,500 to reduce the potential packet loss pitfalls when moving from the WiFi access network to the next hops in the full path. It is also interesting to note that the QUIC transport protocol takes this one step further and by default uses an MTU value of 1,200 bytes. Yes, it is an option for QUIC to use path MTU discovery, but it appears that the default behaviour is to simple use this option. It's just quicker and simpler!

While the platforms continue to scale in terms of speed, it appears that the network stacks are reluctant to take on the agenda of effective and efficient path MTU discovery. Indeed, the current view is to trim packets down in size to avoid any need for IP level packet fragmentation. It seems odd in an environment of continually increased transmission speeds, but when packet size is concerned, we appear to be saying that 1,500 bytes is a pragmatic ceiling for packet sizes, and there are no signs of imminent movement in this position for the larger Internet. I'm not sure that the original Ethernet designers guessed that their initial choice of 1,500 bytes was going to be sticky for the ensuing fifty years, and likely longer!

It appears that the engineering consensus in the public Internet the size of packets lies between 20 and 1500 bytes, based on the original 10Mbps Ethernet. But the opening question was: "How big **should** a packet be?"

The larger the packet payload, the greater the carriage efficiency. With IPv4 a 1,500 byte is 97% efficient (payload to total IP packet size) while in IPv6 it is 96% efficient. The use of 802.3 Jumbo packets, at 9,000 bytes is 99.6% efficient (V4) and 99.3% efficient (V6). So bigger is better – right?

On the other hand, the larger the packet the greater the likelihood that noise will add bit errors into the payload, and if a constant size cyclic redundancy checksum is being used, the larger the packet the greater the possibility of undetected bit errors. In a single channel media larger packets block access to the media for all others while the packet is being used which adds jitter to network paths. In an ACK-paced sliding window protocol, such as TCP, where the sender infers the state of the network path from the implicit signalling within the ACK stream. Reducing the density of these ACK signals, as is the case with larger packets, reduces the ability of the sender to adjust its sending behaviour to attempt to match the available network conditions.

If we accept the design trade-offs of the original 10Mbps Ethernet then the comparable packet size range for a 1Tbs Ether would be 6.4M bytes to 151M bytes. This seems like an insane volume of padding to place a 40 byte ACK packet in a 6.4M byte frame! The alternative is to keep the original minimum packet size of 64 bytes, which implies that the receiver needs to process incoming packet rates of between 823 (large) to 1.5B (small) packets per second.

If we are not willing to change the minimum frame size, what should the maximum frame size be?

If hosts (and applications) are unwilling to perform path MTU discovery due to the time overheads, and the application is comfortable with the efficiency level provided by a 1,518-byte frame size, then why not just use this value as the host's interface MTU? The advantage of this approach is that there is a high assurance that this frame size will work across the entire spectrum of Ethernet-framed networks. If hosts (and interface cards) use this size as the default network MTU size, then they will not incur any reliability issues, nor need to cope with size adaptation issues when the local attached network MTU does not match the MTU of other path components. Here I am referring specifically to the IPv6 fragmentation implementation and the robustness issues of signalling fragmentation needs between network and attached host. All of these issues are avoided if the host simply uses a 1,500-byte MTU.

So how big **should** a packet be? Today's answer is the same as the answer given for 10Mbps Ethernet some 50 years ago. Any size between 46 and 1,500 bytes is a reasonable answer for use within the public Internet.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net