Buffer size and limitations

There are certai limits that affect the size of IP datagrams. We first describ these limits and then tie them all together with regard to how they affect the data an application can transmit.

1. The maximum size of an ipv4 datagram is 65535 bytes, including the ipv4 header. This is because of the 16-bit total length field in figure A.1
2. The maximum size of an ipv6 datagram is 65575 bytes, including the 40-byte ipv6 header. This is because of the 16-bit payload length field in figure A.2. Notice that the ipv6 payload length field does not include the size of the ipv6 header, while the ipv4 total length field does include the header size.

Ipv6 has a jumbo payload option, which extends the payload length field to 32 bit, bit this option is supported only on thoes datalinks with an MTU that exceeds 65535.(this is intended for host-to-host interconnects, such as HIPPI, which often have no inherent MTU)

3. Many networks have an MTU,maximum transmission unit, which can be dictated by the hardware. For example, the Ethernet MTU is 1500 byte. Other datalinks, such as point-to-point links using the PPP protocol, have a configurable MTU. Older SLIP links often used an MTU of 296 bytes.

The minimum link MTU for IPV4 is 68 byte. The minimum link MTU for Ipv6 is 576 byte.

4. The smallest MTU in the path between two hosts is called the *path MTU*. Today, the Ethernet MTU of 1500 bytes is often the path MTU. The path MTU need not be the same in both directions between any two hosts, because routing in the Internet is ofen aymmetric. That is, the route from A to B can differ from the route from B to A.

5. When an IP datagram is to be sent out an interface, if the size of the datagram exceeds the link MTU, fragmentation is performed by both Ipv4 and Ipv6. The fragments are never reassembled until they reach the final destination. Ipv4 hosts perform fragmentation on datagrams that they generate and ipv4 routers perform fragmentation on datagrams that they forward. But with ipv6 routers do not fragment datagrams that they are forwarding.

We must be careful with our terminology. A box labeled an ipv6 router may indeed perform fragmentation, but only on datagrams that the router itself generates, never on datagram that it is forwarding. When this box generates ipv6 datagrams, it is really acting as a host. For example, most routers support the Telnet protocol and this is used for configuration of the router by administrators. The IP datagrams generated by the router’s Telnet server are generated by the router, not forwarded by the router.

You may notice that fields exist in the ipv4 header(figure A.1) to handle IPV4 fragmentation,but there are no fields in the IPV6 header(figure A.2) for gragmentation. Since fragmentation in the exception, rather than the rule, IPV6 contains an option header with the fragmentation information.

6. If the DF(“don’t fragment”) is set in the Ipv4 header it specifies that this datagram must not be fragmented, either by the sending host or by any router. A router that receives a Ipv4 datagram with the DF bit set whose size exceeds the outgoing link’s MTU generates an ICMPv4 “destination unreachable, fragmentation needed but DF bit set” error message

Since Ipv6 routers do not perform fragmentation, there is an implied DF bit with every IPV6 datagram. When an ipv6 router receives a datagram whose size exceeds the outgoing link’s MTU, it generates an ICMPv6 “packet too big” error message.

The Ipv4 DF bit and its implied ipv6 counterpart can be used for *path* MTU *discovery* for IPV4 and RFC 1981. For example, if TCP uses this technique with Ipv4, then it sends all of its datagrams with the DF bit set. If some intermediate router returns an ICMP “destination unreachable, fragmentation needed but DF set” error, TCP decreases the amount of data it needs per datagram and retransmits. Path MTU discovery is optional with Ipv4 but should be supported by all Ipv6 implementations.

7. Ipv4 and Ipv6 define a *minimum reassembly buffer size*: the minimum datagram size that we are guaranteed any implementation must support. For Ipv4 this is 576 bytes.Ipv6 raises this to 1500 bytes. With Ipv4, for example, we have no idea whether a given destination can accept a 577-byte datagram or not. Therefor many Ipv4 applications that use UDP(DNS, RIP, TFTP, BOOTP, SNMP)prevent the application from generating IP datagrams that exceed this size.

8. TCP has an MSS, *maximum segment size*, the announces to the peer TCP the maximum amount of TCP data that the peer can send per segment. We saw the MSS option on the SYN segments in figure 2.5. The goal of the MSS is to tell the peer the actual value of the reassembly buffer size and to try to avoid fragmentation. The MSS is often set to the interface MTU minus the fixed sizes of the IP and TCP headers. On an Ethernet using Ipv4 this would be 1460, and on an Ethernet using Ipv6 this would be 1440. (the TCP header is 20 bytes for both, but the Ipv4 header is 20 bytes and the Ipv6 header is 40 bytes.)

The MSS value in the TCP MSS option is a 16-bit field, limiting the value to 65535. This is fine for Ipv4, since the maximum amount of TCP data in an Ipv4 datagram is 65495(65535 minus the 20-bytes Ipv4 header and minus the 20-bytes TCP header). But with the Ipv6 jumbo payload option, a different technique is used. First, the maximum amount of TCP data in an Ipv6 datagram without the jumbo payload option is 65515(65535 minus the 20-bytes TCP header).Therefor the MSS value of 65536 is considered a special case that designates “infinity”. This value is used only if the jumbo payload option is being used, which requires an MTU that exceeds 65535. If TCP is using the jumbo payload option and receives an MSS announcement of 65535 from the peer, the limit on the datagram sizes that it sends is just the interface MTU. If this turns out to be too large(i.e., there is a link in the path with a smaller MTU) then path MTU discovery will determine the smaller value.