Challenges and opportunities of hardware support for Low Queuing Latency without Packet Loss

Koen De Schepper (koen.de_schepper@nokia-bell-labs.com),
Olivier Tilmans (olivier.tilmans@nokia-bell-labs.com),
Gino Dion (gino.dion@nokia.com)

Traditionally low latency is achieved by giving priority to specific classes of traffic. As this might work for a limited set of applications, it will definitely make the latency of lower priorities worse and it is not always guaranteed to help the prioritized applications. Especially wireless protocols can have quite unpredictable throughput profiles, which might under harsh channel conditions even not support the throughput of the prioritized applications, even when fully starving the 'less' prioritized flows at that time. An often-missed opportunity is exploiting the adaptive nature of almost all applications. Throughput is usually not the main impact factor of the quality that users experience. It is the mismatch of throughput that the application sends and the throughput the network can sustain that causes a lot of frustration of users because of the large latencies that will build up, with eventually even packet loss as a result. Many applications have adopted this adaptiveness and try to estimate the available throughput based on latency measurements and loss detection. Even the best "estimators" are still often wrong, and the solution is a fast and explicit feedback loop from the network node managing the queue latency to the end systems and applications. Fast and accurate measurements are important to gain visibility on the different aspects of latency, but they are crucial to provide immediate feedback to applications when network devices want to control latency in their queues. In this paper we discuss what extra support will be needed in future HW pipelines to measure and achieve low latency.

Transport layer designers are well aware how to control optimally the throughput and resulting latency of flows that are congestion controlled in end systems. To give feedback (other than packet loss) to the congestion controls, and by extension the applications, for when to slow down the sending rate, active queue management mechanisms exist that only need to inspect and modify a pair of bits in the IP header (Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture). When using this explicit technique, congestion controls can keep extra queuing latency below 1ms on smooth network paths. But even this simple interaction is not straight forward to implement and to benefit from when looking at today's physical layer protocols and chip designs.

In the next sections we cover 5 hurdles that need to be reconsidered for future HW pipeline designs.

Transmit granularity

While the classic congestion controls easily "require" network queues of several tens, sometimes hundreds of milliseconds, physical layers took this abundance of available data as an opportunity to aggregate and process IP packets in large transmit frames. Doing this optimizes the throughput by reducing the lower layer header and medium access overheads. On a shared medium, a single station can easily occupy the link for 10ms, and depending on the number of other active stations, it might take several of those time intervals before the first station can send again. The price to pay today is still the extra latency that it induces. With new congestion controls, the queue can easily be reduced to a single millisecond. So future physical layers should be designed with this new reference in mind and should make sure packet bursts can be transmitted at least every millisecond. OFDMA and MIMO are now part of most shared-medium access technologies, enabling to re-design burstiness without losing throughput.

Packet aggregation in multiple stages and queues

To optimize traffic throughput on physical layers that use time slots to burst traffic, some chip designs require an extra stage (sometimes multiple stages) to process the aggregation of IP packets into larger transmit frames. Even worse if the pipeline has queues that store multiple processed transmit frames, each consisting of multiple IP packets. Without multiple stages, large aggregated packets take easily 10ms to transmit, with opportunity to transmit proportional to the number of other stations under load, each taking similar time to transmit. The total latency which could easily be limited to a few ms, now starts at for instance 10ms multiplied by the number of stations under load, multiplied by the number of aggregated data units in the pipeline (summed over all stages and queues). Multiple stages need to be avoided, and if present a tight flow control between them is required to keep the amount to a single transmit frame. Optimally, future physical layer protocols and chip designs need to allow “last milli/microsecond” packet aggregation and transmission, by pulling the available packets from a single stage queue.

Pacing when sending large bursts further on the network

When large bursts of data are received and forwarded to another interface, care must be taken to avoid bursts larger than 1ms in the bottlenecks of further network nodes. This can be due to physical layer properties in access nodes or aggregation into large data frames when received on another network interface to offload operating systems (GRO). Similarly, data in queues or bursts coming from ingress interfaces can be sent in large data chunks to egress interfaces (TSO, GSO) which are then send at line rate of that interface. HW pacing mechanisms in egress interfaces are very effective to avoid that large bursts are send further on the path and cause unwanted latency spikes that might trigger unnecessary rate reduction.

Access to IP header

The IP header is usually only accessible in the early stages of the pipeline. Here typically header fields up to transport layer and sometimes even up to application headers can be used to classify packets in different queues. Modifying fields is usually also possible. The main hurdle to achieve optimal low queuing latency is to modify at this stage the header based on the queue status and lower layer physical channel conditions. At this early stage all this information is typically not available. To optimally achieve the 1ms queue target for L4S traffic, queue and physical layer status needs to be available at least every millisecond too. Chips are more and more programmable to parse and match packets. Programmability should be extended to include queue and scheduling control that have access to real-time physical channel info, also for measurements and monitoring.

Explicit Congestion indicators in lower layer headers

In network nodes that don't have access to the IP header, often QoS fields were added in the lower layer headers to support prioritization of flows. As mentioned before, true QoE cannot be handled by QoS alone. When throughput is limited and IP headers cannot be accessed, the fast and explicit feedback signal towards applications needs to be coded in the lower layer protocol headers and fed back onto the IP headers when they become available in later nodes. Larger awareness is needed in the communities that are working on the lower than IP layers, before also those protocols will further evolve from plain QoS into a closed loop end user QoE.

Measuring latencies

From an application user perspective, there is usually only one latency important that can generally be specified as: how long does it take to see the impact of an interaction. Many aspects are important here, which are not the scope of this paper. The throughput and latency caused by a network device and link will have direct impact on these aspects. Diving deeper into the reasons for latency we found it useful to show detailed reporting of different aspects. Measuring Ping latencies on an idle queue is useful to identify both the minimum achievable latency and having a view on the peak latency caused by the physical layer processing and optionally congestion caused by other stations. But most of the latency only appears when the queue and link is fully saturated by a congestion-controlled flow. To measure the latencies on an interface of a NW device, we timestamp packets when they arrive from an ingress link, then have a second timestamp when the packet is offered to the physical layer for transmission, and compare both with the time that the physical layer acknowledged the successful (or failed) transmission of the packet. These measurements are kept in bins which count each the packets that experienced latency over their time interval (of 1 or 0.5 ms), which can be collected and converted into an inverse CDF for intuitive presentation. Figure 1 shows the resulting inverse CDF of the 3 experiments for an example WiFi downstream link when network conditions are less optimal (lower channel encoding due to higher SNR). The pings on an idle network (green) are send immediately to the physical layer and the plots collide, showing both MAC and transmission time on an idle queue.

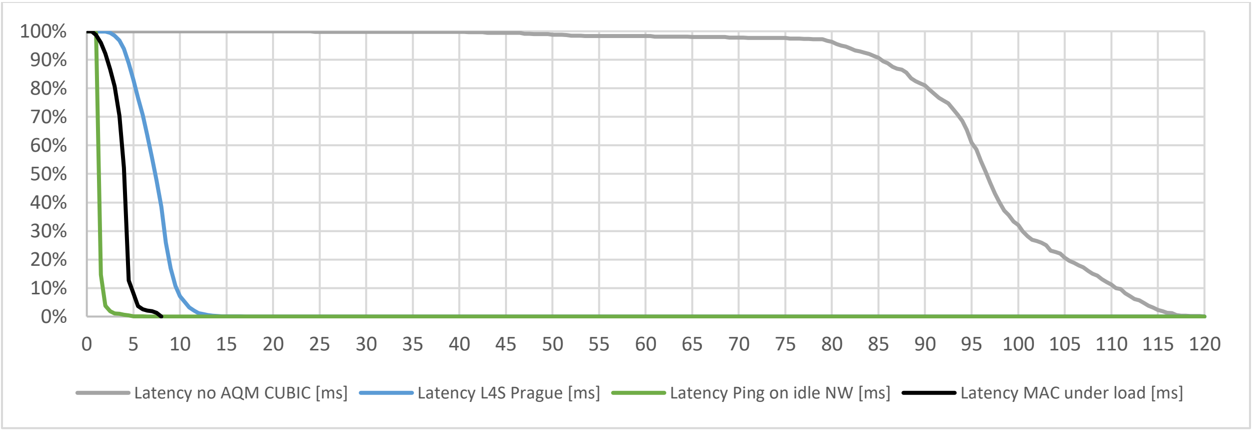

Figure 1: Latency CDF for 3 experiments over a WiFi link: Ping on idle NW; CUBIC download without AQM; Prague download with L4S AQM

A Cubic flow without an AQM (grey) will create a lot of latency (about 100ms in this case) under moderate channel conditions, while the MAC latency of this loaded NW (black) is only 4 or 8ms. When applying an L4S AQM (blue) under the same channel conditions, the queuing latency needed to be similar to the MAC time if full link utilization is wanted, due to an extra pipeline stage in the HW of this chipset. Fortunately, due to sufficient flow control mechanisms, usually only a single transmission frame can be maintained in that stage. These plots clearly show the benefits of measuring the timings of different phases in the processing and transmission of packets. Even more different processing phases would be useful to measure, like how long it takes to encode frames, access the medium, to transmit a frame and to receive the client acknowledgement, all with their distribution over fine-grained bins.

Conclusion

By applying L4S, end system congestion controls and queue management systems are capable now of keeping queues below 1ms. Accurate and detailed measurements are important to achieve these goals, but also understanding of the full processing pipeline. Next in line to improve the queuing latency are the physical layer protocols. Clear awareness in those communities is needed that congestion control and active queue management has moved on, and the existing lower layers cannot keep up the pace in many cases. We mentioned transmit granularity, packet aggregation and its pipelining/queuing designs, egress pacing, IP header and physical layer status access, support for explicit congestion signaling in lower layer headers and need for detailed accurate measurements as important points of attention. Probably other lower layer topics will be important to make the Internet a place where interactive applications can be enjoyed without user frustrations.