Worst-case Ethernet Network Latency for Shaped Sources

Max Azarov, SMSC

7th October 2005

For 802.3 ResE study group

Contents

1 Worst-case latency theorem

1.1 Assumptions

In order to derive mathematically a worst-case latency we will need to make certain assumptions about the network we are dealing with. Following assumptions are set forth:

- All switches on the network are straight store-and-forward and function as output queue switches

- All receiving and transmitting ports are functioning independently (HW router and full duplex)
- All traffic has the same priority.
- Processing time for packets (time between reception and start of transmission of the target port) inside switches is considered to be zero.
- All packets have the same size.
- PHYs on switches have identical speed and transmit/receive single packet in fixed amount of time of τ seconds.
- Packet source and sink are separated by N switches and each switch has n input ports with one source connected to each port.
- Network is never congested, we will define this as sum of incoming traffic to the switch targeted on the same output port over any period of time $n \cdot \tau$ does not exceed output port capacity. This should be true for each port of each participating switch. This effectively means that no more than n packets targeted for one output port come from all input ports during the period of time $n \cdot \tau$. This assumption puts the burden of traffic shaping on sources.

1.2 Theorem

With the assumptions set forth in section 1.1 worst-case propagation delay for the packet from the source to the sink will be

$$T = (n \cdot N + 1) \cdot \tau \tag{1}$$

1.3 Proof

Proof will consist of two parts:

1. construction of example network with delay expressed with formula 1,
2. proof from the opposite that worse propagation value is not possible

First let's build the network with needed propagation delay. For this we will imagine that our source of interest emitted a packet, which we refer to as marked packet. All interfering sources on the network, i.e. all sources but the one we measure propagation delay for, emit packets in such a fashion that they, for each switch, all arrive and get queued for transmission just before the marked packet (See Figure 1). This means that when marked packet arrives, there are $n - 1$ packets queued waiting to be transfered.

Since there are $n-1$ packets in the queue, it will effectively take $\tau \cdot (n-1) + \tau = n \cdot \tau$ seconds for packet to reach the next switch or sink, in case if this switch was last on the path.

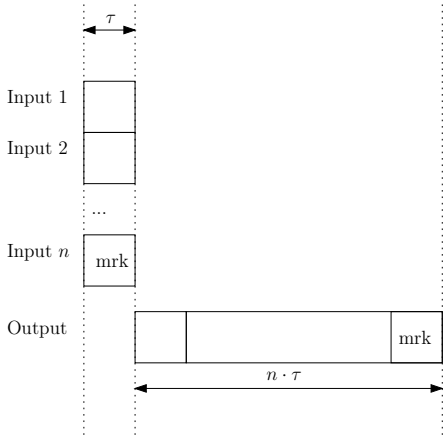

Figure 1: Worst-case switch delay scenario

Note that we can arrange topology and interfering streams is such a way that all packets except for the marked packet will get routed off the marked packet's path. This allows us to re-create the same packets arrangement as on the previous switch on the next switch without violating non-congestion condition. Since arrangement is the same, marked packet will again find $n-1$ packets waiting to be transmitted ahead of it.

Since by our assumption there are N switches between the source and the sink and every switch produces $n \cdot \tau$ seconds of delay, adding the time τ is takes for marked packet to reach a first switch, we get the value for the total propagation delay as

$$T = N \cdot n \cdot \tau + \tau = (N \cdot n + 1) \cdot \tau$$

, which is identical to the expression 1 we're trying to prove.

Now we shall show that given expression is the upper bound.

Let assume that this is not the case and there is a configuration which causes greater propagation delay $\tilde{T}$. This would mean that at least on one switch marked packet found more than $n-1$ packets enqueued when it arrived. At the minimum there was n packets queued, so adding marked packet we will get minimum $n+1$ packets in the queue total.

Lets show that no congestion assumption made in section 1.1 is equivalent to demanding that at no time any output port on switch has more than n packets in queue.

Indeed, output port is a leaking bucket with a constant leakage rate, equal to the link capacity. Number of queued packets can be expressed as

$$m(t) = m(t - n \cdot \tau) - p_{tx} + p_{rx} \tag{2}$$

,where p_{rx} is a number of packets arrived from input ports and p_{tx} is the number of packets transmitted to the output port. By assumption of non-congested

network $p_{rx} \leq n$.

Naturally $p_{tx} \leq n$ since n is the maximum number that can be transmitted at the maximum transmission rate.

Let's consider a moment of time t_0 when $m(t_0) > n$ for the first time. This means that $\forall t < t_0, m(t) \leq n$. We can ensure that such t_0 exists if we assume than $m(t_s) = 0$, $\forall t_s < n \cdot \tau$, which means that initially router had no packets queued for the time period of $n \cdot \tau$.

This allows us to write particularly that $m(t_0 - n \cdot \tau) \leq n$. This in turn means that

$$p_{tx} \geq m(t_0 - n \cdot \tau) \tag{3}$$

, since at the minimum all packets queued at the time $t_0 - n \cdot \tau$ would have been transmitted by the time t_0 because there was less then n of them.

Using 2 we will write an expression for maximum value of $m(t_0)$

$$\max m(t_0) = \max(m(t_0 - n \cdot \tau) - p_{tx} + p_{rx}) \tag{4}$$

Now we will re-write 3, by subtracting $m(t_0 - n \cdot \tau) + p_{rx}$ as

$$-p_{rx} \leq p_{tx} - (m(t_0 - n \cdot \tau) + p_{rx}) \Rightarrow$$

$$m(t_0 - n \cdot \tau) + p_{rx} - p_{tx} \leq p_{rx} \Rightarrow$$

$$\max(m(t_0 - n \cdot \tau) - p_{tx} + p_{rx}) \leq p_{rx} \leq n$$

, here we used $p_{rx} \leq n$ (assumption of non-congested network).

Substituting to 4 we get

$$\max m(t_0) \leq n.$$

Thus we get contradiction with means that such t_0 does not exist and our initial proposition that at least one of switches would have at least $n+1$ packets queued contradicts with our assumptions of the network not getting congested.

Theorem is proved.

2 Generalizations

2.1 Relaxing assumption of the same size packets

In our theorem we assumed that all packets on the network have the same size and transmission time τ. Let's examine what happens if we relax this requirement to say that:

- The τ is the maximum size while smaller packet sizes are permitted. We will assume that there's no overhead of transmitting separate packets versus a single.

Non-congestion requirement will change its meaning. Since packets have different sizes, non-congestion requirement will mean that:

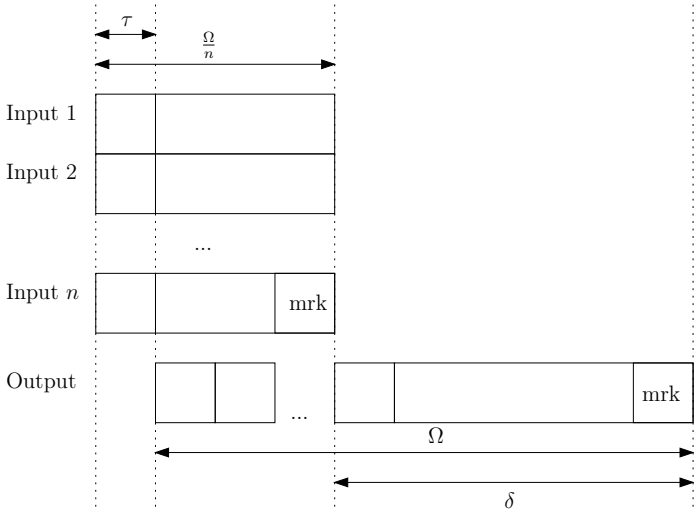

Figure 2: Worst-case scenario for sharing period $\Omega > n \cdot \tau$

- the sum of transmission times for incoming packets during the period of $n \cdot \tau$ targeted for the same output port shall not exceed $n \cdot \tau$.

This substitutes limit of n incoming packets as explained in section 1.1.

It is obvious that our initial worst-case example can still be used since all packets of the maximum size still represent a valid case. This means that per-switch worst-case delay is no less than $n \cdot \tau$.

Expressions 2-4 can be rewritten in terms of transmission times of packets rather than number of packets. We will come to the conclusion that under assumption of shaping interval being $n \cdot \tau$, the sum of transmission times for packet queued at each router will not exceed $n \cdot \tau$. This means that transmission delay per hop will still be bound by $n \cdot \tau$ and latency formula 1 will still be correct.

2.2 Other traffic shaping periods

We've assumed in theorem above that traffic is shaped to not exceed network capacity on intervals of time $n \cdot \tau$. Lets relax this restriction and derive latency for an arbitrary shaping time period of Ω.

Let consider a case where $\Omega > n \cdot \tau$.

This condition allows each input port on a switch to have more than one incoming maximum size packet back-to-back. Short of a strict proof we will construct a worst case latency scenario for a switch based on the intuitive assumption that in order to provide maximum queueing delay we need to produce combined burst with the maximum data rate on input ports targeted to the same output port and make sure marked packet get queued last.

Maximum burst data rate will occur when all input ports are receiving simultaneously. This can be achieved by spreading total budget of incoming data size Ω (in terms of transmission time) evenly over all incoming ports on the switch. As illustrated on the Figure 2, this will amount to the busts with combined size of packets $\frac{\Omega}{n}$ coming from the each input port.

Because of the store-and-forward nature of the switch, we can introduce maximum queueing delay if initial packets on all bursts are maximum-sized packets with transmission time τ. This will ensure that transmission on the output port will be delayed by τ. Once transmission is started, it will take exactly time Ω for output port to transmit all the incoming data including the marked packet at the end. On the other hand marked packet will get queued in the switch only after all burst is received on the input port, which will take $\frac{\Omega}{n}$. So marked packet will be transmitted after $\Omega + \tau$ from the beginning of the burst, but it will only get queued at the time $\frac{\Omega}{n}$ since the beginning of the burst. Putting it all together we get a delay marked packet will experience on this switch as:

$$\delta = \Omega + \tau - \frac{\Omega}{n} = \Omega(1 - \frac{1}{n}) + \tau$$

,please see Figure 2 for a graphic explanation.

To extend same speculation on the next switch on the path we ensure that only the last portion of the burts with the size $\frac{\Omega}{n}$, including marked packet, will be routed on the output port on the marked packet's path, while the rest of the data is routed elsewhere. This will allow us to recreate exactly the same scenario on the next switch and get the same expression for the per-switch delay δ. Including the initial delay of marked packet from source to the first switch, we get following for the total latency:

$$T = (\Omega(1 - \frac{1}{n}) + \tau) \cdot N + \tau \,, \Omega > n \cdot \tau. \tag{5}$$

Note that on the edge $\Omega = n \cdot \tau$ formula turns into the original formula 1.

Lets inspect the case where $\Omega < n \cdot \tau$.

This essentially means that not all input ports on a switch are allowed to have maximum-sized packet queued up simultaneously. At least one port will have a smaller packet or no packet at all, and total maximum size of packets will be Ω. To ensure that output port doesn't start forwarding packets from incoming burst before marked packet is received, we will align all incoming packets' ends with the end of the marked packet. Now if we arrange that marked packet gets queued last, we will get queuing delay of exactly Ω (see Figure 3).

To obtain an exact worst-case proof in this case one can rewrite expressions 2-4 in terms of transmission times versus packets count and show that with shaping period Ω, per-switch delay worse than Ω is not possible.

If we again arrange that on the next switch all packets except for the marked packet are routed off the marked packet's path, we can extend same speculation for every following switch, which puts total latency at:

$$T = \Omega \cdot N + \tau \,, \Omega < n \cdot \tau \tag{6}$$

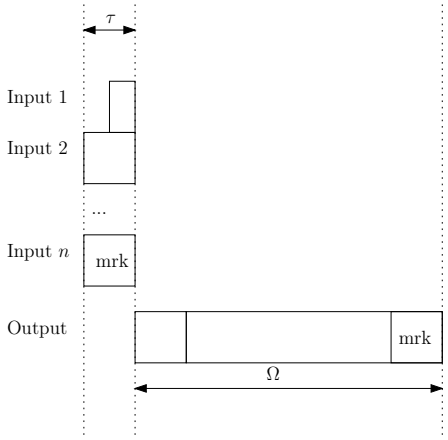

Figure 3: Worst-case scenario for sharing period $\Omega < n \cdot \tau$

Now we can combine all together formulas 5, 6 and original formula 1 (for the case $\Omega = n \cdot \tau$) we get:

$$T = \delta \cdot N + \tau,\ \delta = \begin{cases} \Omega(1 - \frac{1}{n}) + \tau & -\Omega \geq n \cdot \tau \\ \Omega & -\Omega < n \cdot \tau \end{cases} \tag{7}$$

Formula 7 suggests that if we shape traffic at sources more coarsely, propagation delay upper bound will increase.

2.3 Partial network load

We assume in all our speculations above that network can be fully loaded. In fact, designer may choose to limit the load on the network to some specific value $L \in (0, 1]$.

We will define limited network load with coefficient L as

- network where during any period of time Ω any switch can receive on all input ports packets targeted for the same output port with aggregate size of up to $\Omega \cdot L$ (in terms of transmission time).

With this definition we can repeat same speculations we had in section 2.2 for fully loaded network, but instead of Ω we will need to substitute $\Omega \cdot L$ because this will be our maximum burst size now. With this in mind formula 7 will become:

$$T = \delta \cdot N + \tau,\ \delta = \begin{cases} \Omega L(1 - \frac{1}{n}) + \tau & -\Omega L \geq n \cdot \tau \\ \Omega L & -\Omega L < n \cdot \tau \end{cases}$$

2.4 Varying number of ports for switches

We can further generalize formula for networks with switches each having different number of ports. Let's denote number of ports switch number i has as n_i. Using this we can easily generalize formula 7 to

$$T = \sum_{i=i}^{N} \delta_i + \tau,\ \delta_i = \begin{cases} \Omega L(1 - \frac{1}{n_i}) + \tau & \text{—}\Omega L \geq n_i \cdot \tau \\ \Omega L & \text{—}\Omega L < n_i \cdot \tau \end{cases},. \tag{8}$$

2.5 Adding lower-priority interfering traffic

It is very easy to extend equation 8 to take into account presence of an interfering traffic of a lower priority. To do that we add two more assumptions about our network in addition to assumptions spelled out in section 1.1.

- Network has a lower priority interfering traffic with the maximum packet transmission time τ' and this traffic is serviced in the router using a strict priority scheduling. Essentially this means that this traffic has a separate output queue which is serviced only when our higher-priority output queue is empty.
- Lower-priority frame transmission is not interrupted by the arrival of higher-priority frames into the higher-priority output queue .

It is easy to see that with this model at each hop we will incur at the maximum an additional delay τ'.

It will happen when our burst constructed in section 1.3 gets queued up when lower-priority frame transmission just got started. And this means indeed an additional τ' delay. On the other hand delay cannot exceed τ' since this would mean that more than one lower-priority packet got serviced while at least one higher-priority packet (marked packet) was queued up, which is impossible with strict-priority scheduling.

With this in mind we can extend expression 8 to

$$T = \sum_{i=i}^{N} \delta_i + \tau + N \cdot \tau'. \tag{9}$$

Note that lower-priority traffic is not expected to abide any bandwidth restriction. In particular parameter L only limits bandwidth for a higher-priority traffic.

2.6 Adding routing delay

We can take into consideration the fact that routing does not generally happen instantaneously we can replace assumption of zero-time routing set forth in section 1.1 with the assumption that routing is bounded by some time ξ.

Worst-case scenario described in section 1.3 have the same effect on the maximum queue occupation. Since all packets including marked packet are getting delayed some amount of time before being queued in the output queue, we can always arrange them on the wire such that interfering packets will queued just a moment before our marked packet exactly reproducing same worst-case configuration. This means that marked packet at the maximum will experience an additional delay of ξ at each hop. Adding this delay to formula 9 we get

$$T = \sum_{i=i}^{N} \delta_i + \tau + (\tau' + \xi) \cdot N. \tag{10}$$

3 Worst-delay expression analysis

3.1 Parameter sensitivity

In the formula 10 delay is a linear function of all variables. For the sake of simplicity we will consider all switches having the same number of ports $n_i = n$. This will simplify latency expression to:

$$T = \delta \cdot N + \tau + (\tau' + \xi) \cdot N,\ \delta = \begin{cases} \Omega L(1 - \frac{1}{n}) + \tau & \text{—}\Omega L \geq n \cdot \tau \\ \Omega L & \text{—}\Omega L < n \cdot \tau \end{cases}.$$

Looking at n,N as given parameters of network topology, sensitivity to other variables changes can be easily obtained by getting partial differentials:

$$\frac{\partial T}{\partial \tau} = \begin{cases} (N+1) & \text{—}\Omega L \geq n \cdot \tau \\ 1 & \text{—}\Omega L < n \cdot \tau \end{cases},$$

$$\frac{\partial T}{\partial \xi} = N,$$

$$\frac{\partial T}{\partial (\Omega L)} = \begin{cases} (1 - \frac{1}{n}) \cdot N & \text{—}\Omega L > n \cdot \tau, \\ N & \text{—}\Omega L < n \cdot \tau \end{cases} \tag{11}$$

From these expressions we can conclude that with given topology (given n and N) all variables $(\tau, \xi, \Omega L)$ have roughly the same influence on the overall latency. This means we should start adjusting the one with the maximum absolute value since this will provide more headroom for the adjustment. When ΩL is bigger in value we should adjust it first. Once ΩL is the same of less than τ , both variables should be adjusted. Finally, assuming that ξ is small, it will provide very little room for improving a latency figure.

We skipped partial differential over τ' because adjusting a best-effort traffic packet size is very hard in practical terms. Changing this size would cause backward compatibility issues.

Figures 4 and 5 show increase of T with respect to number of hops N for different network utilization levels L. In both graphs we have neglected the routing delay and maximum packet size for all traffic is set to $\tau = \tau' = 125\mu s$.

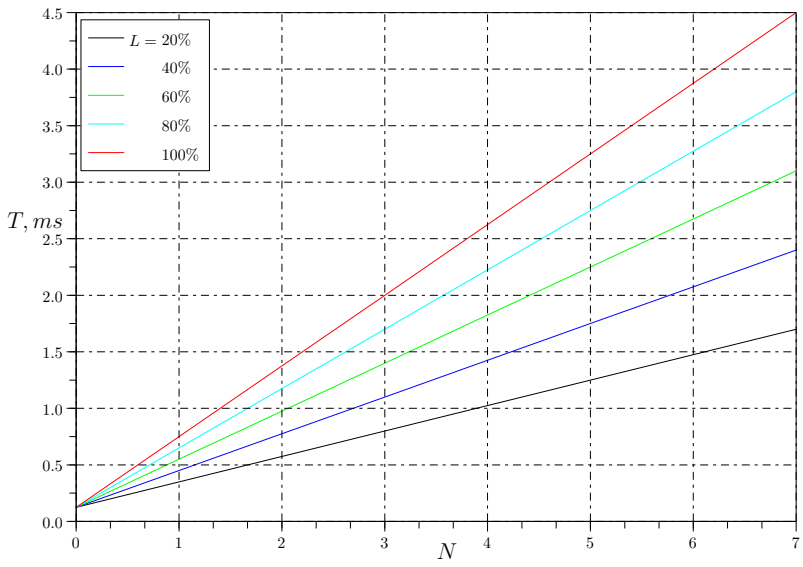

Figure 4: Latency for $n = 5$, $\Omega = 500\mu s$

From figures we can see that the worst-case latency of $2\,ms$ through 7 hops for both 5-port and 2-port switches seems achievable only if we utilize link for $L = 20\%$. At the same time with full utilization and 5-port switches the latency is bounded by $4.5\,ms$.

3.2 Considerations of traffic shaping

As it was noted earlier in our model parameter Ω defines the time period over which sources do not produce more traffic than the available fraction of the network throughput. Is is similar to assuming that sources are expected to abide to their traffic contracts (aggregate, for sources with multiple streams) when bandwidth is calculated over the time Ω.

This can be achieved with sources using transmission algorithm which implements leaky basket. This can be a simple credit-based algorithm, where credit of at least the size of pending packet is needed for transmission to occur. Once transmission occurs used credit is subtracted and next packet doesn't go out until credit is restored to be at least the size of the packet again. Credit is linearly adjusted periodically with the appropriate increment which depends on the target rate and Ω. Credit also gets saturated at some point.

When such leaky basket is used, in order to effectively use the bandwidth, source will have to packetize its payload into the equal-sized packets. If this is

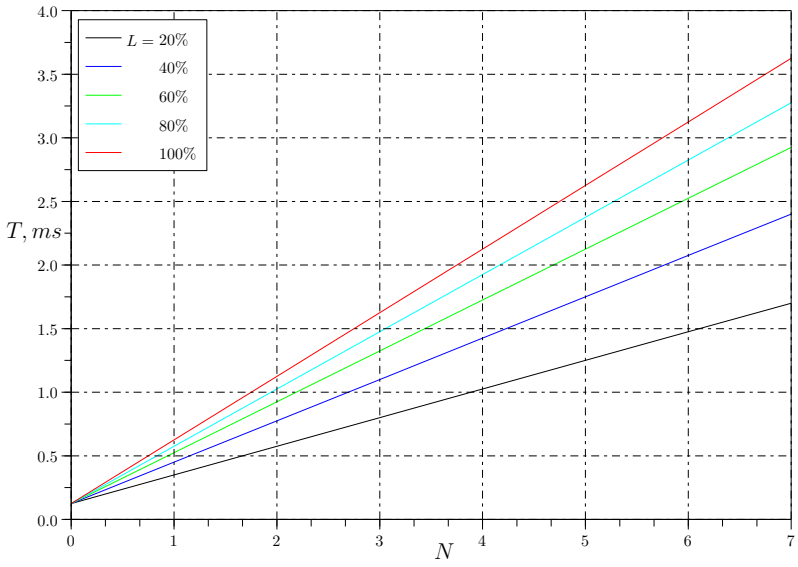

Figure 5: Latency for $n = 2\,, \Omega = 500\mu s$

not true, source will have to request more bandwidth than it will actually use to account for odd-sized packets.

It is also worth noticing that as we shrink Ω, granularity of bandwidth allocation will increase since it is defined by the minimum packet transmission time $\tau_{min} \approx 10\mu s$. For the case $\Omega = 75\mu s$ we will get only $\frac{\Omega}{\tau_{min}} \approx 7$ allocation slots, while for $\Omega = 500\mu s$ we will get ≈ 50 slots.

4 Conclusions

We have produced an exact upped bound (worst-case) for propagation delay under assumptions outlined in section 1.1 as well as generalizations for different shaping time periods, best-effort traffic, different packet sizes, non-zero routing delay. Formula 10 suggests that:

- delay can be varied effectively with changing link utilization level and shaping period,
- assuming shaping period of $500\mu s$ and 5-port switches with 7 hops on 10/100 Ethernet:
  - achieving propagation delay of $2\,ms$ is only possible with 20% link utilization,

  - latency is bounded by $4.5\,ms$ when utilization is 100%,

- results assume adequate traffic policing (See section 3.2). Without policing worst-case latency will be "even worse",
- sources will need to packetize payload into equal-sized packets for efficient bandwidth utilization,
- to further reduce latency figures network will have to employ some form of synchronized media access arrangement or arbitration for sources, switches or both. Better results are not achievable for completely asynchronous network.