

# Worst-case Ethernet Network Latency for Shaped Sources

Max Azarov, SMSC

5th December 2005

For 802.3 ResE study group

## Contents

<b>1</b>	<b>Worst-case latency theorem</b>	<b>2</b>
1.1	Assumptions . . . . .	2
1.2	Theorem . . . . .	2
1.3	Proof . . . . .	2
1.4	Stream distortion and the traffic shaping . . . . .	5
<b>2</b>	<b>Generalizations</b>	<b>7</b>
2.1	Relaxing assumption of the same size packets . . . . .	7
2.2	Other traffic shaping periods . . . . .	7
2.3	Partial network load . . . . .	10
2.4	Varying number of ports for switches . . . . .	10
2.5	Adding lower-priority interfering traffic . . . . .	10
2.6	Higher-priority interfering traffic . . . . .	11
2.7	Adding routing delay . . . . .	13
<b>3</b>	<b>Worst-delay expression analysis</b>	<b>13</b>
3.1	Parameter sensitivity . . . . .	13
<b>4</b>	<b>Ethernet with latency guarantees architecture</b>	<b>15</b>
4.1	Network with two payload classes . . . . .	15
4.2	Traffic shaping details . . . . .	16
4.3	Allocation granularity . . . . .	16
<b>5</b>	<b>Conclusions</b>	<b>16</b>

# 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  $\tau$  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 implies traffic shaping on sources and re-shaping on switches (see Section 1.4).

## 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

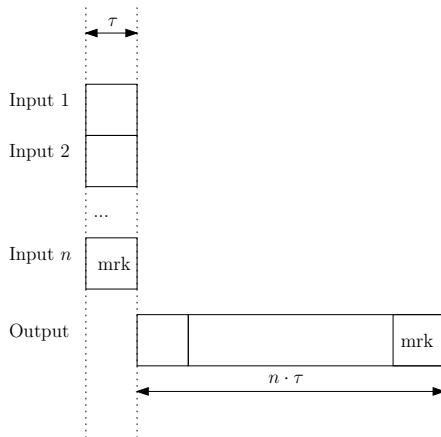


Figure 1: Worst-case switch delay scenario

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 transferred.

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.

In order to apply same speculation to subsequent switches, we arrange topology and interfering streams in 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  $\tau$  it 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 Leaky Bucket with a constant leakage rate, equal to the link capacity. Number of queued packets can be expressed as

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

,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

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

, 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  $m(t_0)$

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

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.

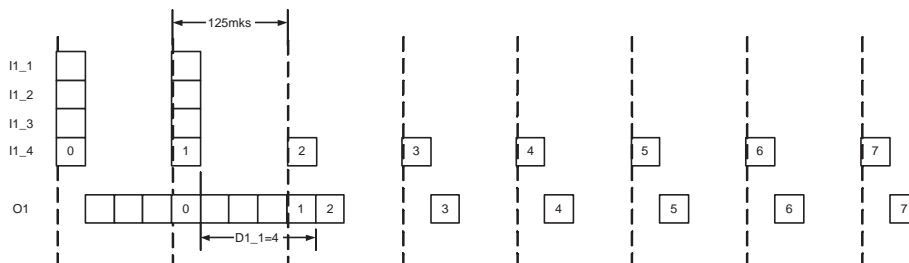


Figure 2: Stream distortion on the first switch

## 1.4 Stream distortion and the traffic shaping

Under our assumption that switches do not get congested, we were able to derive a formula for the worst case latency. If we consider a constant-rate stream of packets emitted by sources instead of carefully arranged burst, we can show that in fact, with simple output queuing rules (i.e. port priority), we can get temporary violations of the non-congestion assumption caused by a packet delay variation within the stream, which causes stream distortion. We can also show that it is possible for this distortion to get aggravated, causing excessive delays.

Let us give an example for 4 input/output ports and packets with transmission time  $31.25\mu s$  ( $n = 4$ ,  $\tau = 31.25\mu s$ ). Non-congestion criterion thus requires that each switch receives no more than 4 packets targeted to the same output port within period of time  $125\mu s$ .

On the Figure 2, we show 4 streams,  $\frac{1}{4}$  bandwidth each, coming to input ports I1\_1, I1\_2, I1\_3 and I1\_4. All streams get routed to the output port O1. The rest of output ports we omit from the picture. We will number packets of the stream coming to I1\_4 and non-numbered streams we will consider as interfering. As an additional twist, three interfering streams, stop transmitting after the second depicted  $125\mu s$  period.

On the output port O1, packet number 1 experienced delay of  $D1_1=4$  packets= $125\mu s$  which is not an excessive delay, but because the interfering streams have stopped, packets number 1 and 2 get much closer to each other and thus cause temporal stream distortion (this condition is also referred to as *bunching*). This happened because packet 2 experienced much shorter delay at the switch than packet 1. Since packets are much closer, stream uses  $\frac{1}{2}$  of available bandwidth during the third depicted period of  $125\mu s$  instead of the allocated  $\frac{1}{4}$ .

Now imagine that we have 3 more switches which have experienced the same situation as just was described above. Along with the packets from the output port O1, we forward the identical outputs of 3 other switches with 3 more distorted streams to separate inputs of the next switch on the path of the numbered stream. This will give us an arrangement depicted on the Figure 3.

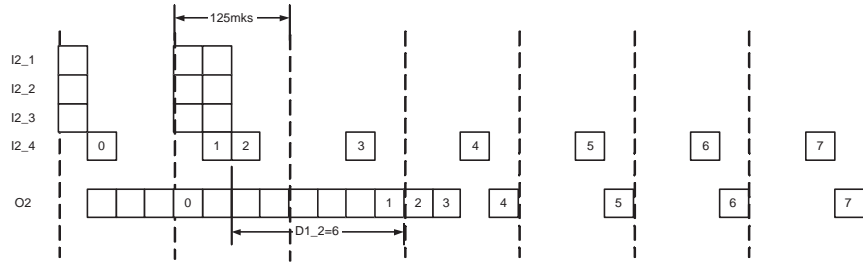


Figure 3: Stream distortion on the second switch

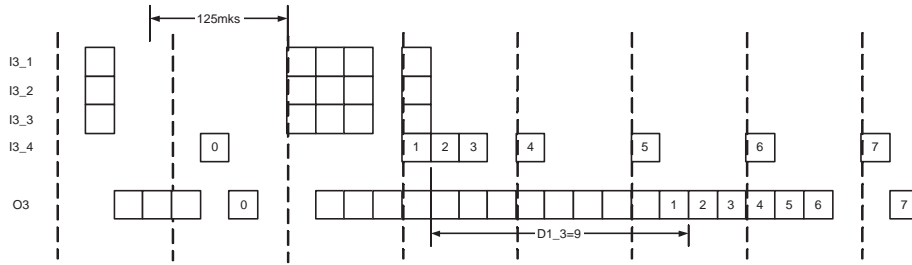


Figure 4: Stream distortion on the third switch

Note that numbered stream is deliberately shifted by 1 packet.

In order to avoid over-subscription, we assume that from each switch sending packets to inputs  $I2\_*$ , only distorted streams get routed to the output port O2, all other streams are routed to one of the remaining 3 output ports. To keep picture from getting too crowded, we show on input ports only packets getting routed to the output port O2. Our original numbered stream packets are identified with the same numbers and 3 other distorted interfering streams are not numbered.

With 4 distorted streams on the inputs, during the second depicted  $125\mu s$  period, switch receives 8 packets, which clearly violates non-congestion assumption (no more than 4). This causes packet number 1 to experience an excessive delay of  $D1\_2=6$  packets= $187\mu s$ .

If we assume that interfering streams have stopped transmitting after the second  $125\mu s$  as it is depicted, packets 1,2,3 and 4 get lumped together, producing even bigger distortion. Now, during the fourth depicted  $125\mu s$  period, numbered stream takes up  $\frac{3}{4}$  of the available bandwidth for a period of  $125\mu s$  instead of the allocated  $\frac{1}{4}$ .

Applying the same technique to the third switch on the path, we see (Figure 4) that distortion keeps increasing and packet number 1 experiences the delay of  $D1\_3=9$  packets= $281.5\mu s$ .

The constructed example shows that active traffic shaping in switches is

required to avoid temporary non-congestion violations and for formula 1 to be true. Traffic shapers in each switch for each input port should be able to detect distorted streams and “fix” distortions before stream gets multiplexed with other streams.

Without the traffic shaping, worst-case latency grows much faster than linearly with the number of hops.

Particular shaping techniques are described in details in [1]. Overall, the shaper (rate regulator in this work’s terminology) implements a Leaky Bucket algorithm, which detects rate distortions and delays packets which would cause the congestion. One important result the paper establishes is that, despite delaying some packets, shaping does not increase end-to-end worst-case latency bound. This happens because by design, shaper only delays packets routed “ahead of the schedule” compared to the other packets in the stream and it doesn’t delay them to cause more than the worst-case delay on each switch.

## 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  $\tau$ . Let’s examine what happens if we relax this requirement to say that:

- The  $\tau$  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:

- 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  $\Omega$ .

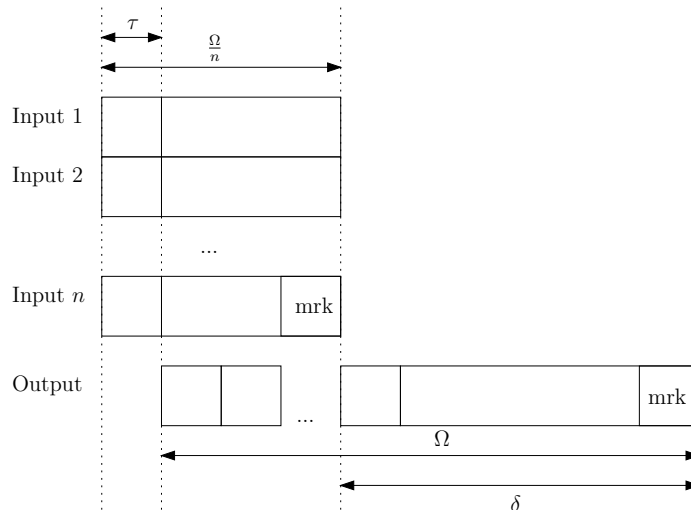


Figure 5: Worst-case scenario for sharing period  $\Omega > n \cdot \tau$

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  $\Omega$  (in terms of transmission time) evenly over all incoming ports on the switch. As illustrated on the Figure 5, this will amount to the bursts 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  $\tau$ . This will ensure that transmission on the output port will be delayed by  $\tau$ . Once transmission is started, it will take exactly time  $\Omega$  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:

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

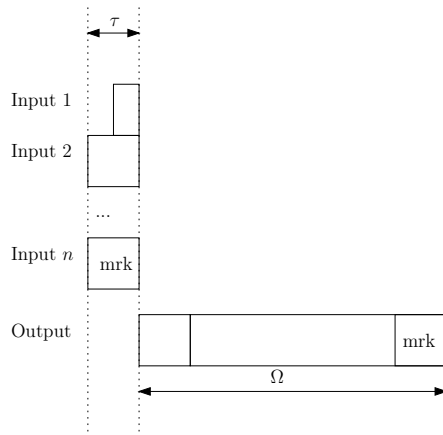


Figure 6: Worst-case scenario for sharing period  $\Omega < n \cdot \tau$

,please see Figure 5 for a graphic explanation.

To extend same speculation on the next switch on the path we ensure that only the last portion of the bursts 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  $\delta$ . Including the initial delay of marked packet from source to the first switch, we get following for the total latency:

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

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  $\Omega$ . 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  $\Omega$  (see Figure 6).

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  $\Omega$ , per-switch delay worse than  $\Omega$  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:

$$T = \Omega \cdot N + \tau, \Omega < n \cdot \tau \quad (6)$$

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

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

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  $\Omega$  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  $\Omega$  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

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

### 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  $\tau'$  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  $\tau'$ .

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  $\tau'$  delay. On the other hand delay cannot exceed  $\tau'$  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

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

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 Higher-priority interfering traffic

Lets extend our model with the higher-priority (HP) interfering traffic. Besides measured traffic, running at the shaping period  $\Omega$  and utilization  $L$ , we assume that HP traffic is present, and is allocated a dedicated bandwidth share  $\tilde{L}$  and runs at the shaping period of  $\tilde{\Omega}$ . We assume here that  $\tilde{L} + L < 1$ , i.e. HP traffic and measured traffic combined do not exceed the available bandwidth. When considering worst-case conditions for both the HP and measured traffic, both streams will have a form of bursts of the maximum allowed size. The HP traffic will come in bursts every  $\tilde{\Omega}$  for the duration of  $\tilde{\Omega}\tilde{L}$ , while the measured traffic will come in bursts every  $\Omega$  for the duration of  $\Omega L$ . Naturally, when both streams go through the same output port, they will get combined, as it is illustrated on the Figure 7.

Let's denote via  $\Psi$  the difference between the length of the original (measured traffic) burst and the combined burst. If we find  $\Psi$ , it will give us the maximum latency impact that HP traffic will have on the measured traffic.

Since the combined burst gets bigger because on the inserted HP bursts, we can re-write  $\Psi = k \cdot \tilde{\Omega}\tilde{L}$ , where  $k$  is the number of HP bursts that get merged into the original burst. Now, since  $k$  is the number of times period of HP traffic will fit into the combined burst length, we can write the following

$$k = \left\lceil \frac{\Omega L + \Psi}{\tilde{\Omega}} \right\rceil = \left\lceil \frac{\Omega L + k \cdot \tilde{\Omega}\tilde{L}}{\tilde{\Omega}} \right\rceil = \left\lceil \frac{\Omega}{\tilde{\Omega}} L + k \cdot \tilde{L} \right\rceil.$$

This can be re-written as

$$k \geq \frac{\Omega}{\tilde{\Omega}} L + k \cdot \tilde{L} > k - 1 \Rightarrow$$

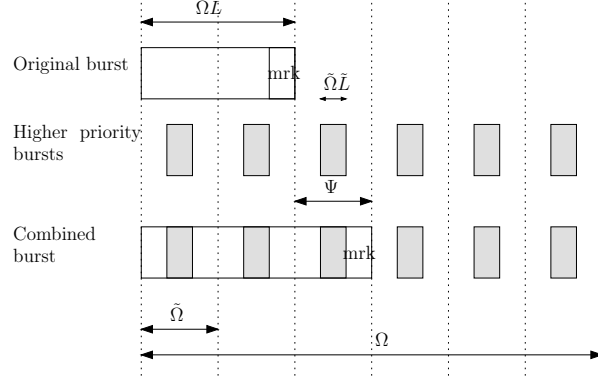


Figure 7: Higher-priority stream impact

$$\begin{aligned}
0 &\geq \frac{\Omega}{\tilde{\Omega}}L + k \cdot (\tilde{L} - 1) > -1 \Rightarrow \\
-\frac{\Omega}{\tilde{\Omega}}L &\geq k \cdot (\tilde{L} - 1) > -1 - \frac{\Omega}{\tilde{\Omega}}L \Rightarrow \\
\frac{\frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}} &\leq k < \frac{1 + \frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}} \Rightarrow \\
\frac{\frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}} &\leq k < \frac{1}{1 - \tilde{L}} + \frac{\frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}}
\end{aligned}$$

Assuming that solution exists, we can get  $k$  if we round up the lower bound, because bounds are less than a unity apart ( $\frac{1}{1-\tilde{L}} < 1$ ):

$$k = \left\lceil \frac{\frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}} \right\rceil.$$

Now, with the expression for  $k$  we can easily write

$$\Psi = \left\lceil \frac{\frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}} \right\rceil \cdot \tilde{\Omega}\tilde{L}.$$

With the expression for the latency impact on a per-hop basis, we can write a full latency with higher-priority traffic interference added:

$$T = \sum_{i=i}^N \delta_i + \tau + N \cdot (\Psi + \tau') = \sum_{i=i}^N \delta_i + \tau + N \cdot \left( \left\lceil \frac{\frac{\Omega}{\tilde{\Omega}}L}{1 - \tilde{L}} \right\rceil \cdot \tilde{\Omega}\tilde{L} + \tau' \right). \quad (10)$$

If our analysis we disregarded the fact that bursts actually are composed of packets and during merging, higher-priority stream will not merge perfectly

at the time they arrive on the input, but may get pushed off to the end of the pending packet from the measured stream. If we take this into consideration, it is easy to see that the impact of the HP traffic will not increase and in fact may decrease when HP burst on the right edge of the measured traffic burst get pushed off to the side. In other words, when packetization is considered,  $k$  may decrease, but not increase, which means that our initial analysis holds true as a upper bound.

We should note that result obtained assumes that there's no traffic of even higher priority than the HP traffic. In order to take this into account, additional analysis should be done.

## 2.7 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  $\xi$ .

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  $\xi$  at each hop. Adding this delay to formula 9 we get

$$T = \sum_{i=1}^N \delta_i + \tau + (\tau' + \Psi + \xi) \cdot N. \quad (11)$$

## 3 Worst-delay expression analysis

### 3.1 Parameter sensitivity

In the formula 11 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:

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

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

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

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

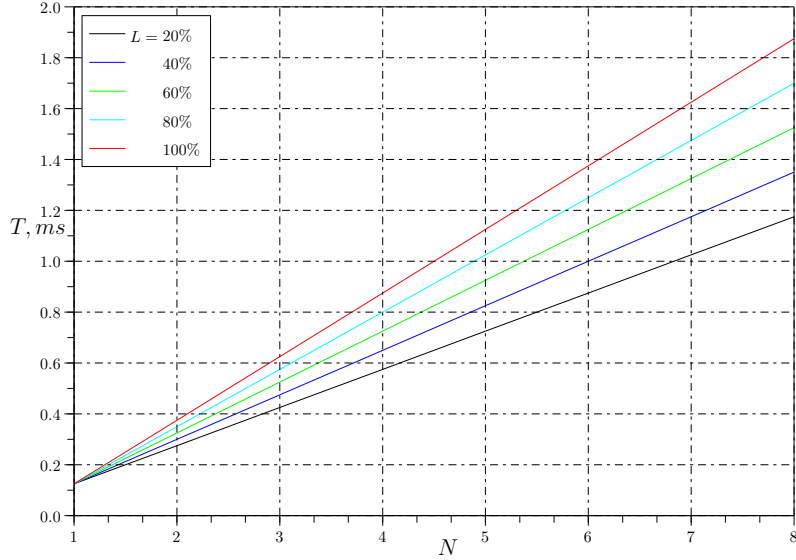


Figure 8: Latency for  $n = 5$ ,  $\Omega = 125\mu s$

$$\frac{\partial T}{\partial(\Omega L)} = \begin{cases} (1 - \frac{1}{n}) \cdot N & -\Omega L > n \cdot \tau, \\ \frac{1}{N} & -\Omega L < n \cdot \tau \end{cases} \quad (12)$$

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  $\Omega L$  is bigger in value we should adjust it first. Once  $\Omega L$  is the same or less than  $\tau$ , both variables should be adjusted. Finally, assuming that  $\xi$  is small, it will provide very little room for improving a latency figure.

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

Figures 8 and 9 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  $\tau = \tau' = 125\mu s$ .

From figures we can see that the worst-case latency of  $2\text{ ms}$  through 7 hops is achievable only for shaping period  $\Omega = 125\mu s$ . For bigger period of  $\Omega = 1000\mu s$ , the latency is bounded by  $8\text{ ms}$ .

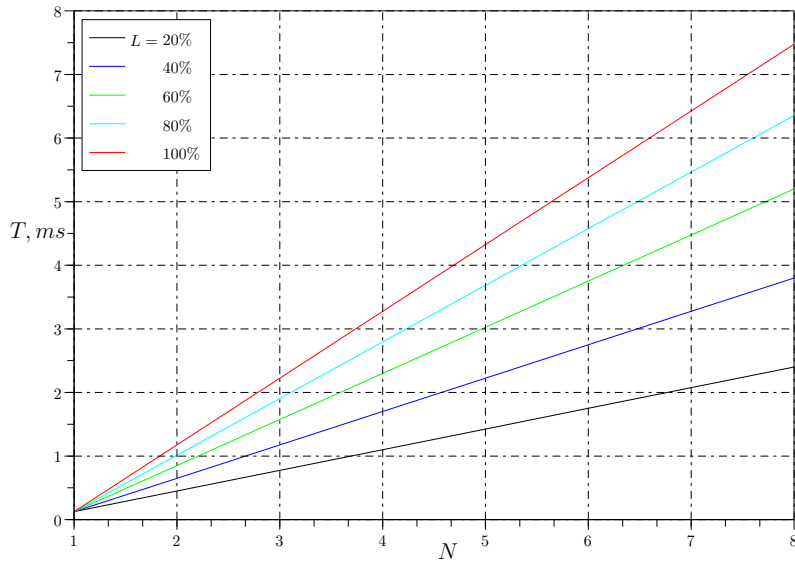


Figure 9: Latency for  $n = 5, \Omega = 1000\mu s$

## 4 Ethernet with latency guarantees architecture

Here we will outline an architecture option for designing the Ethernet LAN with latency guarantees. We assume that bandwidth reservation signaling is defined elsewhere.

### 4.1 Network with two payload classes

*Notice: this section had barely been started.*

This type of network consists of compliant end-points and switches, connected into the arbitrary topology (are loops allowed?). Network has the following traffic classes:

- Best effort
- Payload 1 — higher latency guaranteed
- Payload 2 — low latency guaranteed
- Control

Switches on the network schedule packets using a strict priority algorithm and do the re-shaping of flows before transmission for each input port and each guaranteed latency class. Each traffic class uses it's own priority as depicted in Table 1.

Traffic class	Priority	Shaping period	Bandwidth reserved	Latency guaranteed
Best Effort	Lowest	N/A	N/A	N/A
Payload 1	+1	1kHz	TBD	>2ms ?
Payload 2	+2	8kHz	TBD	≤2ms ?
Control	+3	TBD	TBD	TBD

Table 1: Traffic classes in Two Payload class network

For both classes, Payload 1 and Payload 2, guaranteed latencies are calculated using the Formula 11.

As part of the process of reserving the bandwidth along the delivery path, each switch provides a value for a worst-case latency bound pertaining to this switch. Once all switches along the path provide the latency values, end-points use those values to obtain the total path guaranteed worst-case latency. This approach allows for heterogeneous networks with the mix of the hardware (i.e. Fast Ethernet and Gigabit Ethernet).

It is assumed that end-points generate streams abiding their traffic contracts. In our case this means that streams do not exceed reserved bandwidth over any period of time  $\Omega$ , which is a shaping period for a specific class this stream belongs to.<sup>1</sup>

## 4.2 Traffic shaping details

To be added...

## 4.3 Allocation granularity

It should be separately noted that, as we shrink  $\Omega$ , 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  $\Omega = 1000\mu s$  we will get  $\approx 100$  slots.

## 5 Conclusions

We have produced an exact upper 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 11 suggests that:

- traffic shaping inside switches is essential to enable linear-only growth of guaranteed worst-case latency with the number of hops,

<sup>1</sup>If it found to be useful, this requirement can be relaxed to allow some stream distortions similar to the one that may occur in the switch during the packet multiplexing.

- delay can be varied effectively with changing link utilization level and shaping period,
- worst-case latency guarantee on 2ms is achievable with the shaping period  $\Omega = 125\mu s$ ,

## References

- [1] Peter Kim, Resource Reservation in Shared and Switched Demand Priority Local Area Networks, PhD dissertation, 1998