Approach to a Latency-bound Ethernet

Max Azarov, SMSC

April 21, 2006

For 802.1 AVB group

Contents

1 Introduction

1.1 Background

Low cost and reliability of Ethernet is thought to make it a technology of choice for building a back-bone networks in people homes, hotels. While in terms of bandwidth, Ethernet deems to offer a sufficient head room, it unfortunately falls short on providing adequate level of QoS (quality of service). This lack of stream-oriented QoS guarantees is characteristic of both best-effort Ethernet as well as for the Ethernet supporting priority tagging.

This lack of QoS capabilities puts in question the vision for distribution of audio/video (A/V) content over the existing network infrastructure, which requires QoS for reliable operation. To ensure its future, Ethernet needs to be extended with QoS guarantees, while keeping its cost low.

While there is a number of existing technologies (PowerLink, etc.), extending Ethernet to enable QoS, they are mostly oriented for industrial use with low-latency requirements and are not very well suited for A/V streaming, while others have higher cost.

1.2 Scope

This paper aims at providing a low-cost solution for Ethernet network, which would allow provision of deterministic stream-oriented QoS guarantees:

- support arbitrary network topology with arbitrary number of end-points and switches,
- guaranteed worst-case packet delivery latency (from this point of refereed to as just *latency*)
  - below 2 ms for any path of 7 Fast Ethernet switches,
- guaranteed inter-packet jitter (referred to as just *jitter*),
- guaranteed bandwidth.
- traffic contracts enforcement

Listed guarantees cannot be efficiently provided in the existing Ethernet technology with priority tagging, which will be illustrated in Section 2.

1.3 Paper structure

Paper begins with Section 2, where we provide analysis of short comings of the existing Ethernet networks with priority tagging and examples of inability to provide a reasonable bound for packet delivery latency. Further, in Section 4, paper proceeds to offer outline of changes that need to be made in Ethernet in order to support QoS. This is followed by Section 5 with analytical proof of the ability to offer deterministic latency and jitter guarantee for a simple network model, and then followed by Section 6, which offers analytical analysis of more realistic extended network model. Section 7 offers analysis of expressions for worst-case latency for parameter sensitivity and provides some latency figures for specific configurations. With analytical proof and analysis at hand, in Section 8 we proceed to describe in more detail proposed changes to the Ethernet enabling low-latency A/V streams.

Paper is concluded by summary of results.

2 Ethernet with priority tagging and QoS

In this section we will review short-comings of the Ethernet with priority tagging, which prevent it from providing QoS guarantees.

To clarify, we will be calling an Ethernet network with priority tagging, a network of full duplex Ethernet end-points and store-and-forward switches with certain number of prioritized queues for packets, tagged with a different priority. Each queue operates as a best-effort queue.

First and foremost, Ethernet does not have any means to control bandwidth utilization by traffic sources emitting packets. There can easily be a situation where two sources over-utilize the channel. This can be addressed on the higher level with some sort of resource reservation protocol running on end-points and on switches to provide topology information.

But even if we assume that packet sources are controlled by a resource reservation protocol and do not over utilize communication channels, Ethernet with priority tagging still fails to guarantee deterministic latency bound.

2.1 Stream distortion

If we consider a constant-rate stream of packets marked with the same priority emitted by sources, we can show that in fact, with simple output queuing rules (i.e. port priority), we can get temporary congestions 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 a network of switches with 4 input/output ports

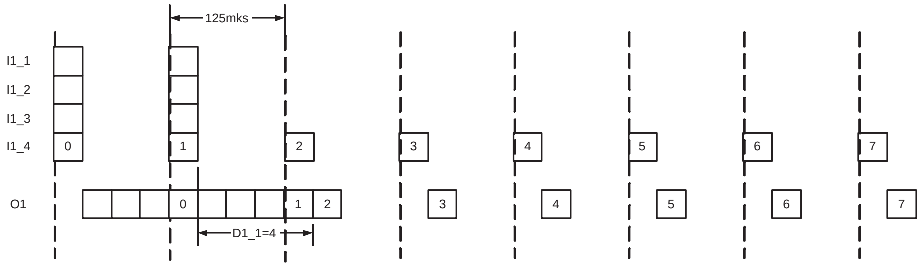

Figure 1: Stream distortion on the first switch

and packets with transmission time $31.25\mu s$ We will be measuring stream bandwidths over a period of $125\mu s$.

On the Figure 1, we show 4 streams, $\frac{1}{4}$ bandwidth each, coming to input ports I1_1, I1_2,I1_4 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, while fourth stream continues uninterrupted.

On the output port O1, packet number 1 experienced a modest delay of D1_1=4 packets=$125\mu s$, 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 number 2 experienced much shorter delay at the switch than packet number 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 topology with 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 our numbered stream. This will give us an arrangement depicted on the Figure 2. 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 creates a backup in the switch queue. This causes packet number 1 to experience an excessive delay of D1_2=6 packets=$187.5\mu s$.

If we assume that interfering streams have stopped transmitting after the

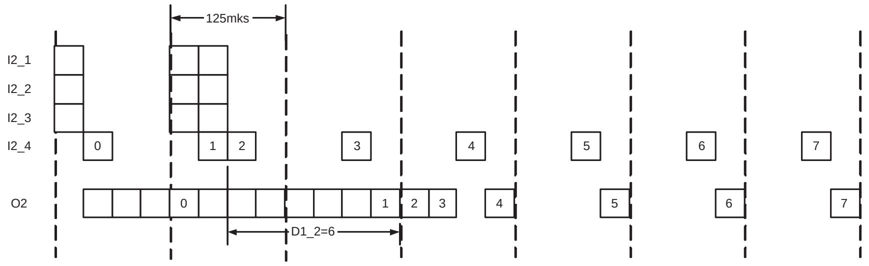

Figure 2: Stream distortion on the second switch

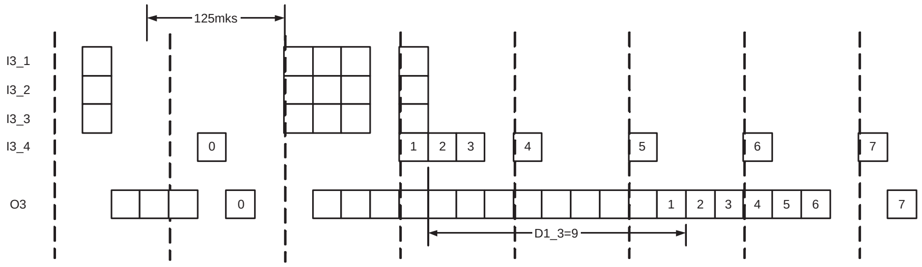

Figure 3: Stream distortion on the third switch

second 125μs as it is depicted, packets 1,2,3 and 4 get lumped together, producing even bigger distortion. Now, during the fourth depicted 125μs period, numbered stream takes up $\frac{3}{4}$ of the available bandwidth for a period of 125μs instead of the allocated $\frac{1}{4}$.

Applying the same technique to the third switch on the path, we see (Figure 3) that with four "double-distorted" streams on the input of the switch, distortion keeps increasing and packet number 1 experiences the delay of D1_3=9 packets=281.25μs.

The constructed example shows that per-switch packet delay can increase rapidly, while total latency would increase as a sum of per-switch increases. Extrapolating result to more hops, we get following numbers for the latency:

Number of hops	1	2	3	4	5	6	7
Switch delay, μs	125	187.5	281.25	406.25	562.5	750	968.75
Latency, ms	0.1	0.3	0.6	1	1.6	2.3	3.3

We see that with selected topology, very well-behaved constant-bitrate sources emitting packets precisely every 125 μs, over 7 hops we get the latency in an excess of 3.3 ms, while neglecting such phenomenon as background best-effort traffic and routing delay. If we consider less accurate sources, congestion will aggravate even further.

Consider repeating our previous constructs, but with streams emitting packets with quadrupled transmission time of 125 μs (instead of 31.25 μs) and with quadrupled bandwidth measurement period of 500 μs (instead of 125 μs). All delays will be exactly valid in terms of number of packets, except that each packet is 4 times longer now. This quadruples latency over 7 hops and brings it to 13.2 ms.

For a number of A/V applications, this latency magnitudes are unacceptable, which calls for modifications in the way Ethernet switches work to bring the latency down.

Is Section 5 we will show analytically that traffic shaping in switches which fixes traffic distortions, leads to significantly smaller latency figures.

For more scenarios on the Ethernet traffic distortion please see [3].

2.2 Topology sensitivity

In our example we used a specific topology, where measured stream mixes up with its "twin" equally distorted streams coming from other paths and after mixing gets even more distorted.

In fact, when thinking about the worst case, we should consider the topology in which on every switch the measured stream mixes up with maximum-distorted interfering streams which went through some N number of hops already. The bigger the N, the more delay will distorted interfering stream cause.

If we consider growing N indefinitely, so will the latency bound *grow indefinitely.*

This highlights the sensitivity of Ethernet latency bound to the overall topology of the network. In extreme case, even if we have only one switch separating talker and listener, but we have no control over the topology of the rest of the network, *latency bound cannot be guaranteed at any level.*[1]

Such topology sensitivity is a very significant flaw. If one would try to impose topology restriction, it would require user to maintain his network restricted to a certain configuration, which can be a complicated task requiring non-trivial network engineering knowledge. On top of that, by adding just one device on the periphery of the network, one can invalidate whole configuration and prevent whole network from providing a required level of bound latency.

3 Impact of a per-port ingress/egress shaping

One modification that can be considered to enhance the Ethernet network is the addition of *traffic shapers* (see 4.2 for details) for each port either on ingress, egress or both.

As we could see in section 2.1, stream distortion can cause heavy intermittent queuing delays even in case of well-behaved sources, transmitting traffic

[1] of course in practice, because of limited switch queue size, inability to provide latency bound means that no-drop operation cannot be guaranteed, i.e. dropped packet was "delivered" with infinite latency.

compliant with its bandwidth reservations. We will show that presence of just ingress/egress shapers cannot alleviate the stream distortion. This placement of shapers does not allow for a sufficient level of discrimination of the internal structure of the traffic, i.e. more information is needed about each packet in order to make shaping effective.

3.1 Egress shaper

Consider the scenario from the section 2.1, depicted on figures 1, 2 and 3. Let's introduce a shaper on the egress and tune it for the aggregate traffic we expect on the output. Since we have four streams 1/4 bandwidth each coming in, the expected bandwidth on the egress will be 100% of the link. This means that newly introduced shaper will need to be tuned for 100% of the bandwidth, i.e. shaper will never reschedule packets, because packets will never come at the rate exceeding the 100% link utilization.

This means that such shaper will be rendered void, and stream distortion will not be alleviated.

In case when the maximum network utilization is limited below 100%, shaper will influence the distortion, but will not completely remove it.

3.2 Ingress shaper

If we introduce a shapers on the ingress and tune them for 25% bandwidth per port, this would seemingly ensure that stream distortion would get corrected before reaching the output queue. This example is limited, because in this case shaper is processing a single stream only.

Let's consider a scenario, where 4 streams 1/4 bandwidth each are coming into each input port and each get forwarded to a separate output port. Essentially, this is a scenario depicted on the figures 1, 2 and 3, repeated 4 times on each switch, forwarded to separate output ports. In this case, ingress shapers will need to be tuned for combined bandwidth, i.e. 100% of the link.

As we showed earlier, shaper tuned at 100% is effectively void as it has no effect on packets passing through it. This means that stream distortion will not be alleviated.

4 Modifications enabling low latency bound

4.1 Proposed list

As we have shown in Section 2, without modifications, Ethernet cannot provide a reasonable bound packet delivery latency. We will outline a proposed modifications, which will enable lower latencies.

1. Stream-oriented and topology-aware QoS parameters reservation protocol
2. Network consists of only store-and-forward switches with deterministic routing delay (no soft-switches)

3. Sources emit shaped A/V traffic within their reserved bandwidth
4. One or more separate priority tags above best-effort for each A/V traffic class
5. Per port pair/class ingress traffic shapers in the switch (One shaper per each input-output port pair and each A/V traffic class)

In this paper we will assume that items number 1 is addressed with some appropriate protocol. We will merely assume that this protocol allows signalling between talkers and listeners, where

- Each talker can request channel to listener with specified bandwidth, latency and jitter;
  - request can be either granted or denied,
  - in case of denial possible best-service parameters may be communicated.
- Each listener can request joining ongoing multicast stream with specified bandwidth, latency and jitter;
  - request can be either granted or denied,
  - in case of denial possible best-service parameters may be communicated.

Item 2 is essential because software switches typically fail to offer low and deterministic routing delay (not including queuing delay).

Item 3 ensures that sources initially produce shaped traffic, and thus honor made reservations and do not exceed allocated bandwidth, even temporarily.

Item 4 ensures that best-effort traffic is kept on the lower priority thus ensuring that its impact of the A/V streams latency is limited. Having more than one priority tag allows to have multiple classes of A/V traffic with different latency requirements.

Item 5 ensures that A/V stream distortions (as illustrated in Section 2) are removed before they are routed by the next switch. This ensures that distortions do not accumulate and do not cause excessive delays. Most importantly, this item removes topology sensitivity.

4.2 Shaper definition

We will call in this paper *the shaper* to be a device or algorithm, which receives on the input a potentially arbitrary stream of packets and produces on the output a stream of packets compliant with specified bandwidth reservation characterized by parameters B and Ω.

- Parameter B is a reserved bandwidth in byte per second units.

- Parameter Ω is a period in time units, over which the bandwidth is measured.

We will define stream to be *compliant* with reservation B and Ω if measured bandwidth of the stream over any period of time of length Ω within the stream is below or equal to the B.

Definitions of compliant streams and shapers are similar to those introduced in [2].

The particular implementations of shaper depend on the known limits of input stream which is to be shaped. Shapers would typically have the ability to queue packets up to some limit, which characterizes the shaper's ability to shape severely non-compliant traffic without dropping any data.

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 misplaced packets which would cause the congestion. One important result the paper establishes is that, despite delaying some packets, correctly designed shapers do not increase end-to-end worst-case latency bound. This happens because in correct design, shaper only delays packets routed "ahead of the schedule" compared to the other packets in the stream and it doesn't cause delay more than the worst-case delay on each switch.

4.3 Sufficient level of modifications

With proposed modifications at hand, we should look into which modifications are sufficient for latency-bound Ethernet.

Our analytical analysis, presented in Sections 5 and 6, shows that items 2-5 are sufficient to enable a deterministic guarantee for low latency ($< 2\ ms$).

On the other hand results from Section 2 suggest that if higher latencies are acceptable and topology can be restricted, just having items 2-4 may be sufficient. In other words, shaping in switches may be omitted.

If this paper we will concentrate on networks with shapers, because even if higher latencies are acceptable, restricting topology seems very problematic.

5 Worst-case latency theorem

5.1 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:

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

2. All receiving and transmitting ports are functioning independently (HW router and full duplex)

3. All traffic has the same priority.

4. Processing time for packets (time between reception and start of transmission of the target port) inside switches is considered to be zero.

5. All packets have the same size.

6. PHYs on switches have identical speed and transmit/receive single packet in fixed amount of time of τ seconds.

7. Packet source and sink are separated by N switches and each switch has n input ports with one source connected to each port.

8. 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 $n \cdot \tau$.

We should note that in order to meet assumption 8 we should have sources emitting traffic which is shaped not to exceed the constant bitrate with bandwidth measurement period of $n \cdot \tau$ and reservations for bitrate are arranged is such a way that combined incoming traffic for each switch is below the network throughput. But this alone is not sufficient, because it would not address stream distortion effects reviewed in Section 2. In order to address distortions, shaping should be done on ingress of each pair of input and output ports in the switch.

This way shaped sources ensure validity of assumption 8 on the boundary of the network, while shapers inside switches ensure this assumption validity inside the network.

5.2 Theorem

With the assumptions set forth in Section 5.1 worst-case propagation delay for the packet from the source to the sink will be

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

5.3 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

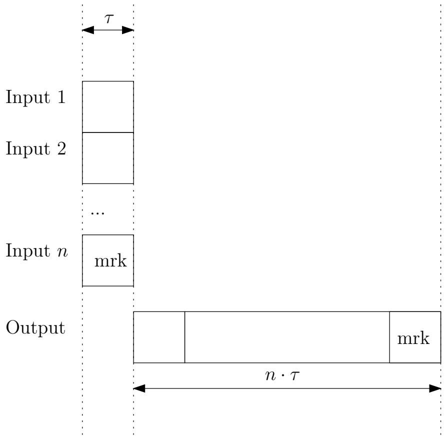

Figure 4: Worst-case switch delay scenario

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 4). 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.

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 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 us 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 5.1 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

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

,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 8 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.

6 Extended model

In this section we expand out result from the Section 5 to a more realistic assumptions about the network.

6.1 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:

- 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 5.1.

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.

6.2 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 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 τ. This will ensure that transmission on the

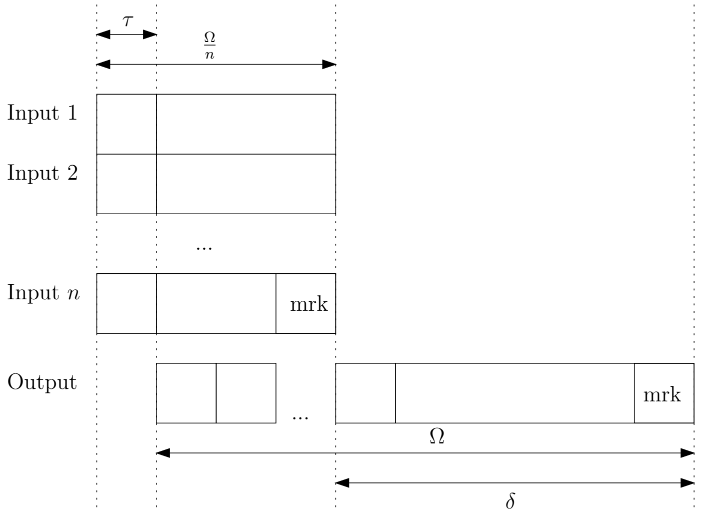

Figure 5: Worst-case scenario for sharing period $\Omega > n \cdot \tau$

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 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 δ. Including the initial delay of marked packet from source to the first switch, we get following for the total latency:

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

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

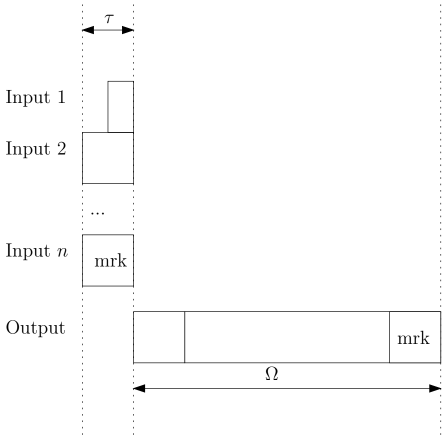

Figure 6: Worst-case scenario for sharing period $\Omega < n \cdot \tau$

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 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 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:

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

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.

6.3 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 6.2 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:

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

6.4 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

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

6.5 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 5.1.

- 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 5.3 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

$$T = \sum_{i=1}^{N} \delta_i + \tau + N \cdot \tau'. \tag{9}$$

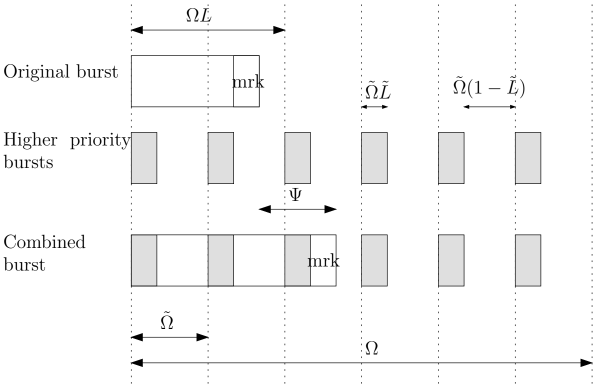

Figure 7: Higher-priority stream impact

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.

6.6 Higher-priority interfering traffic

Lets extend our model with the higher-priority (HP) interfering traffic. Besides measured traffic, running at the shaping period Ω 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 Ω for the duration of ΩL. 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 Ψ the difference between the length of the original (measured traffic) burst and the combined burst. If we find Ψ, 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, to find k, we notice that the number of HP bursts in the combined burst is the same as the number of spaces in-between HP bursts that get filled in with the pieces of the original burst. Since each space in-between HP bursts has a size of $\tilde{\Omega}(1-\tilde{L})$, we can write expression for k as

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

we use rounding to a next integer to account for a last space in-between HP bursts with a partial fill.

Now, with the expression for k we can easily rewrite Ψ as

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

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

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

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.

Now, let's get the latency impact for measured stream with two interfering streams, one of higher (HP) and another one with even higher (HP1) priorities.

For practical purposes, we will limit the scope of our analysis to the case where streams HP1 and HP have the same shaping period $\tilde{\Omega}$. Stream HP uses bandwidth fraction $\tilde{L}$ and HP1 uses $\tilde{L_1}$. Both streams are assumed to be honoring their respective bandwidth reservations, so over the period $\tilde{\Omega}$ stream HP sends no more data than $\tilde{\Omega}\tilde{L}$, while stream HP1 sends no more data then $\tilde{\Omega}\tilde{L_1}$. This means that both streams will contribute to the output port on the switch no more data than $\tilde{\Omega}(\tilde{L} + \tilde{L_1})$.

Now we need to note that for a measured stream both HP and HP1 streams can be viewed as a single combined stream of a higher priority. Indeed, the difference in priorities for HP and HP1 streams is only manifested in how these streams get multiplexed relative to each other, but not relative to the measured stream.

With this observation, we treat HP and HP1 streams as a single stream with combined load $\tilde{L} + \tilde{L_1}$ and substitute it instead of $\tilde{L}$ in formula (10) to get a new latency expression

$$\Psi = \left\lceil \frac{\frac{\Omega}{\tilde{\Omega}} L}{1 - (\tilde{L} + \tilde{L_1})} \right\rceil \cdot \tilde{\Omega}(\tilde{L} + \tilde{L_1}) \Rightarrow .$$

$$T = \sum_{i=1}^{N} \delta_i + \tau + N \cdot (\Psi + \tau') = \sum_{i=1}^{N} \delta_i + \tau + N \cdot (\left\lceil \frac{\frac{\Omega}{\tilde{\Omega}} L}{1 - (\tilde{L} + \tilde{L_1})} \right\rceil \cdot \tilde{\Omega}(\tilde{L} + \tilde{L_1}) + \tau'). \qquad (11)$$

6.7 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 5.1 with the assumption that routing is bounded by some time ξ.

Worst-case scenario described in Section 5.3 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

$$T = \sum_{i=1}^{N} \delta_i + \tau + (\tau' + \Psi + \xi) \cdot N. \tag{12}$$

7 Worst-case latency expression analysis

7.1 Parameter sensitivity

In the formula (12) 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:

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

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

$$\frac{\partial T}{\partial \tau} = \left\{ \begin{array}{cc} (N+1) & , \Omega L \geq n \cdot \tau \\ 1 & , \Omega L < n \cdot \tau \end{array} \right. ,$$

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

$$\frac{\partial T}{\partial (\Omega L)} = \left\{ \begin{array}{cc} (1 - \frac{1}{n}) \cdot N & , \Omega L > n \cdot \tau, \\ N & , \Omega L < n \cdot \tau \end{array} \right. \tag{13}$$

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 or 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.

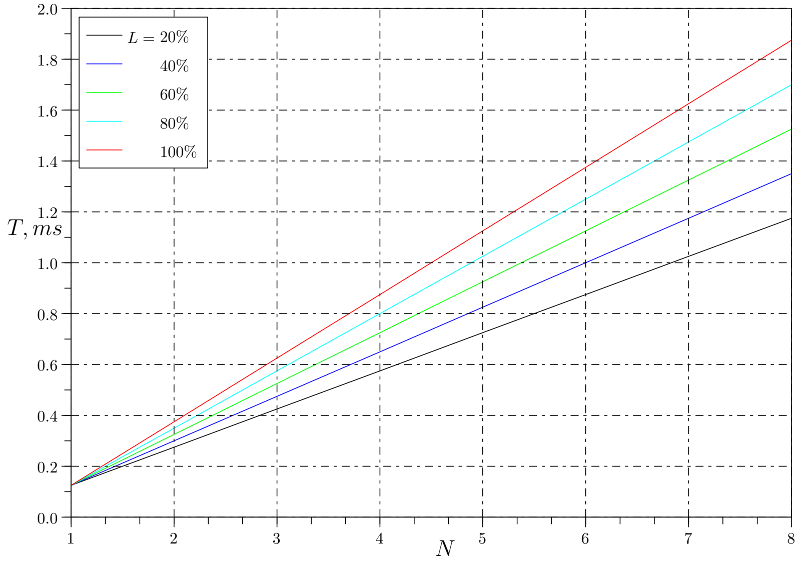

Figure 8: Latency for $n = 5$, $\Omega = 125\mu s$

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 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 $2\,ms$ 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 $8\,ms$.

7.2 Allocation granularity

Is should be separately noted that, when we shrink Ω to get lower latency bound, 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 $\Omega = 125\mu s$ we will get only $\frac{\Omega}{\tau_{min}} \approx 12$ allocation slots, while for $\Omega = 1000\mu s$ we will get ≈ 100 slots.

8 Architecture of Ethernet with a bound low-latency

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

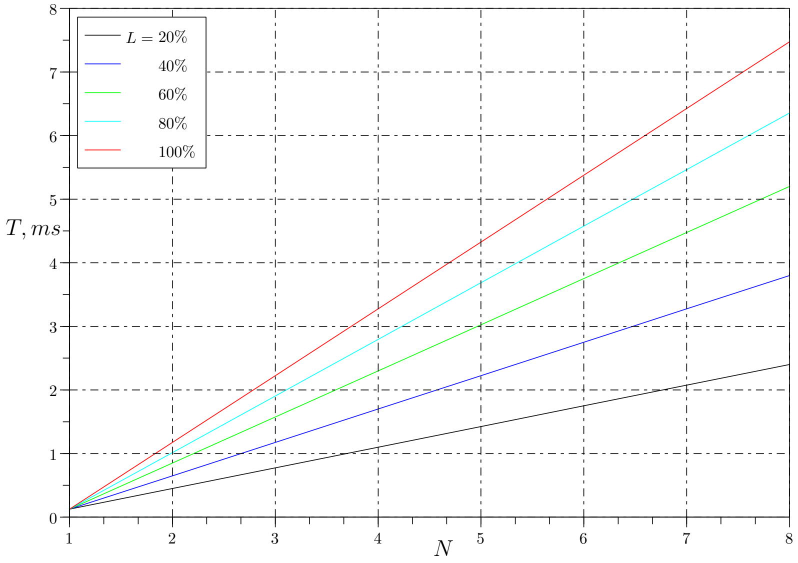

Figure 9: Latency for $n = 5\,, \Omega = 1000\mu s$

is defined elsewhere.

8.1 Architecture drivers

One main driving force will be an assumption that the modified Ethernet should be able to provide at least 2 ms latency guarantee over 7 Fast-Ethernet hops.

Another assumption we make is that only a limited number of applications will actually need a 2 ms latency guarantee, while most applications will function with higher, 10 $ms+$ guaranteed latencies.

We can see from the analysis on the latency formula in Section 7, that latency varies greatly for different shaping periods Ω. In order to get latency figure below 2 ms, we would have to operate on a very stringent shaping period of 125 μs, which imposes restriction of packetization, as well as leaves only a limited number of bandwidth allocation slots.

Since number of low-latency streams is expected to be small, to address this issue streams are divided into two separate classes with different latency requirements and different shaping periods.

8.2 Network with two payload classes

This type of network consists of compliant end-points and switches, connected into the arbitrary topology with no loops (if loops exist, we assume they are resolved somehow). Network has the following traffic classes:

- Best effort

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	8kHz	TBD	TBD

Table 1: Traffic classes in Two Payload class network

- Payload 1 — higher latency guaranteed
- Payload 2 — low latency guaranteed
- Control (timing, reservation signalling?)

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

For both classes, Payload 1 and Payload 2, guaranteed latencies are calculated using the formula (12).

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 Ω, which is a shaping period for a specific class this stream belongs to.[2] In this case, shapers inside switches will not drop any data during re-shaping of the streams.

For both end-points and switches, shapers can be implemented using scheduling algorithm of Leaky Bucket. This can be a simple credit-based algorithm, where credit of at least the size of a pending packet is needed for transmission to occur. Once transmission occurs, used credit is subtracted and next packet doesn't go out until the credit is restored to be at least the size of the packet again. Credit is linearly adjusted periodically with the appropriate increment, depending on the combined target rate and Ω until it gets saturated at some point. For details please refer to [1].

In order to effectively use the bandwidth, source will have to packetize it payload into the equal-sized packets. If this is not true, source will have to request more bandwidth than it will actually use to account for odd-sized packets.

[2] 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.

8.3 Switch architecture

A network with priority classes, described above would require switches on the network to implement additional set of traffic shapers. Rough outline of such switch architecture is depicted in the Figure 10.

As one can see from the diagram, switch would require to inject shapers in-between a switching fabric and the output queues to handle shaping of the latency-bound traffic. For each output port, there is one shaper per input port and per latency-bound traffic class. Each of those shapers need not be a separate physical device, assuming packets forwarded to a specific port are serialized when they come out of the switching fabric . In this case a single physical set of shaper logic can be used with multiple RAM slots for separate shapers' state machines. For packet storage, all shapers can utilize the same RAM pool, shared with TX queues.

In the diagram, TX queues for regular traffic and TX queues for a latency-bound traffic are separated because they have different size requirements, otherwise, latency-bound TX queues can utilize identical logic with regular TX queues. TX queues for latency-bound traffic will need substantially less RAM, because the latency-bound traffic is well-shaped compared with the regular traffic.

Despite the fact that each shaper would require a separate queue, overall per-port RAM utilization for all shapers will not be significantly higher than a per-port RAM utilization of a conventional switch. This is due to the fact that separate shapers use smaller queues, because they deal with only a fraction of a bandwidth of the output port. Counting in smalled size requirements for latency-bound TX queues, overall RAM requirements for the switch should not be substantially higher when compared to a conventional strict priority switch.

9 Conclusions

We have produced an exact upper bound (worst-case) for propagation delay under assumptions outlined in Section 5.1 as well as generalizations for different shaping time periods, best-effort traffic, different packet sizes and non-zero routing delay. Resulting latency bound can be found using derived formula (12).

Our analysis of the formula suggests that:

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

Based on the results, we have produced an example of a modified Ethernet architecture with two traffic classes, allowing deterministic low-latency guarantees below 2 ms over 7 hops. In this architecture traffic shaping inside switches is essential to prevent stream distortion, topology sensitivity, and indefinite growth of guaranteed wost-case latency with arbitrary topology.

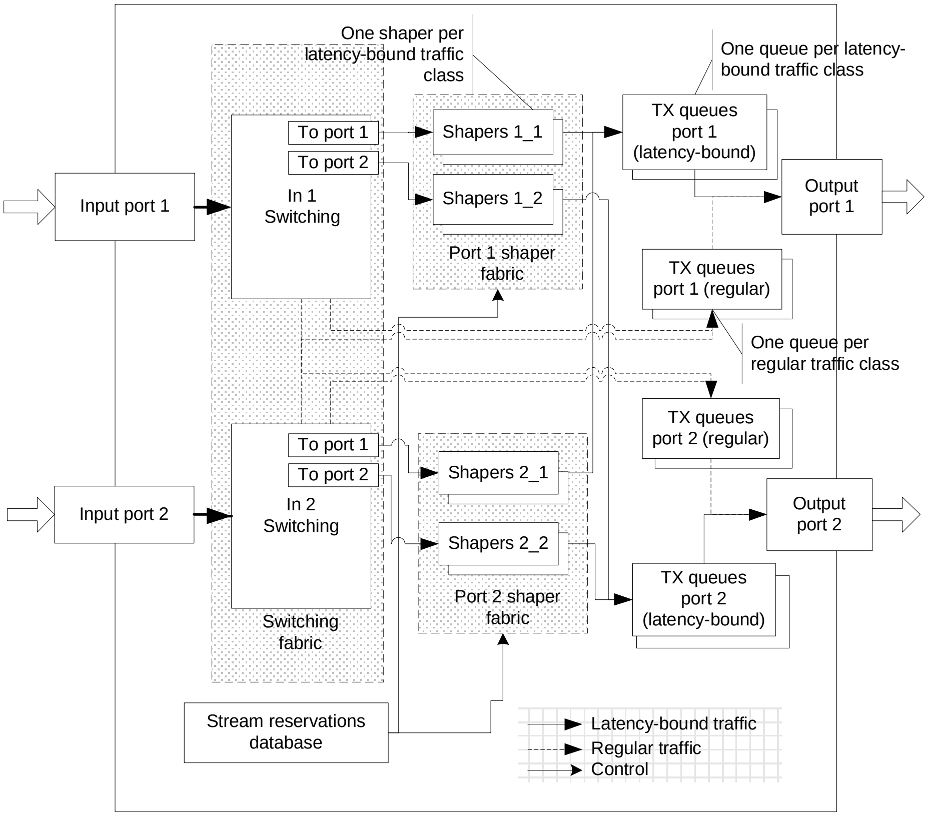
One shaper per latency-bound traffic class
One queue per latency-bound traffic class
Input port 1
Input port 2
To port 1
To port 2
In 1 Switching
To port 1
To port 2
In 2 Switching
Switching fabric
Shapers 1_1
Shapers 1_2
Port 1 shaper fabric
Shapers 2_1
Shapers 2_2
Port 2 shaper fabric
TX queues port 1 (latency-bound)
Output port 1
TX queues port 1 (regular)
One queue per regular traffic class
TX queues port 2 (regular)
Output port 2
TX queues port 2 (latency-bound)
Stream reservations database
Latency-bound traffic
Regular traffic
Control

Figure 10: Latency-bound switch architecture diagram

We noted that if higher latencies and restricted topology are acceptable, architecture without shapers inside switches can be sufficient, but topology restriction requirement is very undesirable.

References

[1] Peter Kim, Resource Reservation in Shared and Switched Demand Priority Local Area Networks (ftp://cs.ucl.ac.uk/darpa/pk_phd_thesis.ps.Z), PhD dissertation, 1998

[2] David E. McDysan, Darren L. Spohn, ATM Theory and Application, McGraw-Hill, 1994

[3] Dave V James., AV Bridges White Paper