QoS in Frame Relay

Frame relay characteristics are:

. packet switching with virtual circuit service (virtual circuits are bidirectional);

. labels are called DLCI (Data Link Connection Identifier);

. for connection is meant virtual circuit, so QoS is provided per virtual circuit;

. no error control (optional in edges), no flow control;

. LAP-F protocol with packet size up to 4096 Byte.

Parameters negotiable at setup with contract basis are:

. CIR (Committed Information Rate): this is the bit rate that network guarantees, measured in bit/s;

. CBS (Committed Burst Size): this is the amount of data always accepted and trasmitted at highest priory measured in seconds;

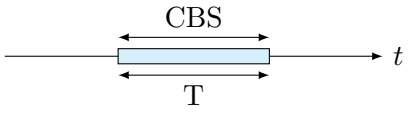
. EBS (Excess Burst Size): this is the amount of data trasmitted with best effort approach, accepted only if it is possible; it is measured in bit.

Data exceeding the size CBS+EBS is always discarded for default: this is an example of policing.

The time in which the CIR is measured is T, defined as:

$$T = \frac{\text{CBS}}{\text{CIR}} \qquad \frac{\text{bit}}{\text{bit/s}} = \text{s}$$

Graphically:



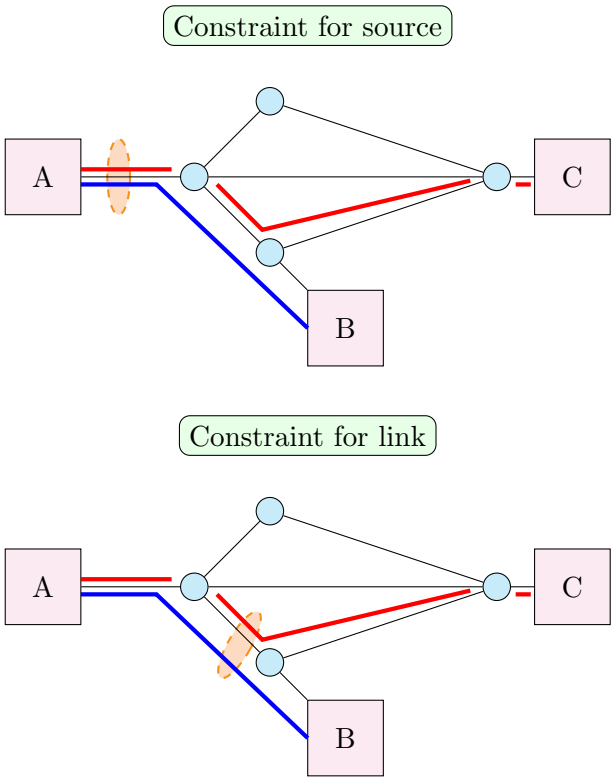
Practically, T is the period in which CBS is transmitted.

There are two fundamentals constraints have to be respected:

. the sum of all CIR of a **source** must be lower than the **access** speed rate;

. the sum of all CIR of a **link** must be lower than the **link** speed rate.

The following pictures describe the two constraints:



Congestion

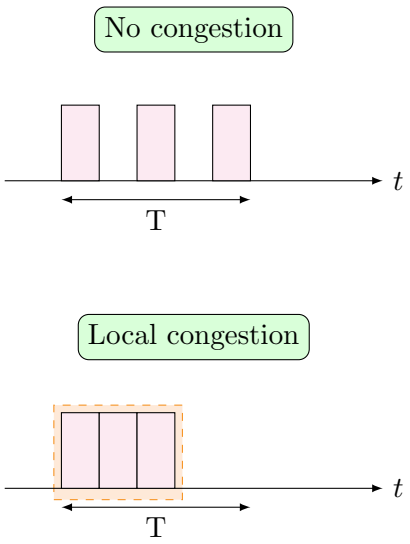
If the sum of CIR over a single link exceed the global capacity this create a long term congestion. It can be done in order to increase network utilization in two ways:

. transmitting packets with low priority: this traffic is sent only if there are avaiable resources so it does not create unsatisfaction to users;

. overbooking: this solution violate the second constraint previously declared because sooner or later high priority packets will be dropped creating unsatisfaction to users.

If it is traffic burstiness who create congestion, there is short term congestion, so it is not critical beause this fact is well controlled if burstiness is the small as possible.

Congestion depends on CIR, link speed and T. Look at the following examples:



Algorithms

Policing algorithms will be discussed are:

. leaky bucket;

. token bucket.

In order to manage congestion problem, there are two possible algoritms:

. backward;

. forward.

Policing

Confermance verification is realized differentiating the traffic characterization over the period T:

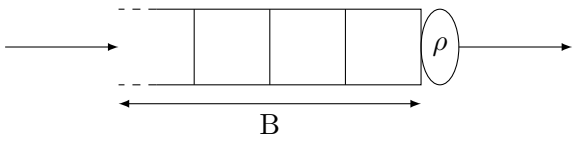
. packets conformant to the CBS constraint are always sent at high priority (Discard Eligibility bit set to 0);

. packets conformant to the EBS constraint are always sent at low priority (Discard Eligibility bit set to 1);

. packets non conformant to the CBS+EBS constraint are always dropped.

Policing works marking and dropping packets but if instead perform drop operation packets are delayed the same algorithm can be used as shaping in order to adapt traffic to be conformant to the previous constraints.

Leaky Bucket Leaky bucket behaves like a traffic regulator because translate in a CBR source at rate ρ if source is greedy (it means that generates packets countinously). If it is used like a policer, when packets arrive earlier respect ρ they are dropped; they are also dropped if user traffic exceed the buffer size B.

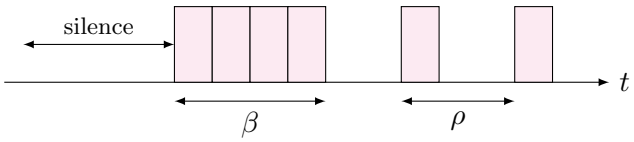


Over a period T is not possible to sent more traffic than $T \cdot \rho$; what concerning the bit rate:

. if ρ can be approximate with the average rate too much traffic could be discared;

. if ρ can be approximate with the peak rate, resources are underutilized because there is a waste of link bit rate.

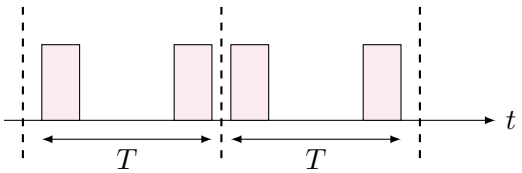
Token Bucket If with leaky bucket burstiness can not be generated, token bucket allows it: paying a silence for β packets (the size of the token buffer) the buffer is full up, so the transmission rate will be, over a local period, more than ρ (rate in which token are generated). Otherwise if the token is released as soon as is generated the behavior is, again, the one of a CBR source. The following picture show what it is said before:



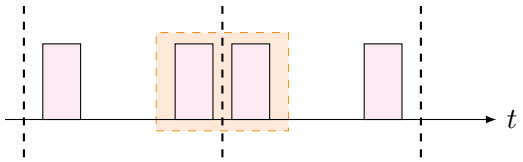
Users can send data only if there is a token present into the buffer and the maximum traffic which they can generate over a T period is always lower or equal to $T \cdot \rho + \beta$ (β is related to the burst duration).

Measurement problems

Take measures of rate over an asynchronous packet flow is a complex problem: the simplest solution is takes measures over fixed lenght intervals:



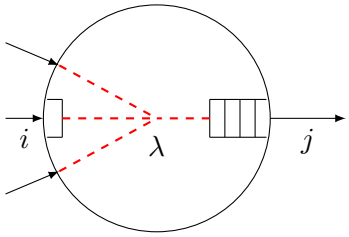
It is important notice that over each T period is not possible send more than CBS and probably this constrain is respected in average, but maybe locally can not:



The larger is period T the most critical is *border effect* between adjacent intervals. Another solution is a temporal sliding window: with this approach rate is measured accounting for the amount of byte received during the window as soon as a packet arrive. The most important disadvantage is that implementation is too much complex because it is necessary to store information about packet time arrivals. It is possible exploit *fluid approximation*, approach in which it is not necessary store information about packet time arrivals, but only calculate and update the rate thinking that all bytes arrived are spreaded over the sliding period.

Congestion Control

The scenario in which congestion control is discussed is the *outer buffer architecture*: it provides the best performance because, in a packet time, only one packet has to be forwarded to the proper outer port.



Congestion problem is completly avoided if all sources are sending at CIR, but due to overbooking is possible that there is congestion; in this case, first of all, delay increases, then buffers fill up and consequently packets will be dropped causing a reduce of QoS. Algebraic condition is:

$$\sum \lambda_{i,j} \geq 1$$

if all paths are normalized: $0 < \lambda < 1$. This condition may create a *long* congestion problem or *short* congestion problem: it depends on which time it is measured and it is a critical parameter.

When a node detects congestion have to signal it setting one of the possible two bits declared by the standard:

. FECN (Forward technique);

. BECN (Backward technique).

These techniques are different only for the direction in which congestion is signaled.

The two constraints that must be respected in order to avoid packet lost are **fairness** and **maximun network utilization**: they are always in conflict because satisfying one of them the other will be violated. A typical example is the scenario of only one user in CSMA-CD: the network utilization will be maximum because there are no collisions, but this is completly unfair for others users.

Max-min fairness

This algorith tries to maximize the bandwidth allocation to flows that receive the minimum allocation.

With this algorithm, to increase the bandwidth already allocated to a flow, it is necessary to decrease the bandwidth of another flow and this is a very critical disadvantage for QoS. The second disadvantage is that, max-min algorithm can not be implemented looking locally in the network otherwise it does not work: a global view is needed.

Algoritm step:

. at first all flows are marked as unsatisfied;

. bandwidth is increased over each flow by the same value until the first bottleneck link is satisfied;

. when a bottleneck link is saturated it is mark as satisfied and it can not increase more his bandwidth;

. the procedure runs until all flows are marked as satisfied.

Forward congestion

This approach is used when a node detects congestion and, in order to signal it to the network, sets FECN bit to 1. Congestion, in this way, is signaled to all nodes until the receiver, and it has to redirect this information to the transmitter because only that node can reduce its traffic in order to react to the problem. The signalling from the transmitter to the receiver is sent through the same path of intermediate nodes because virtual circuit are bidirectional.

Algoritm property are:

. implementation very easy: the only task have to be performed is the set of FECN bit;

. slowly because if the node who detect congestion is closed to the source, a RTT is paid to propagate signalling to the receiver and then to the transmitter;

. no additional traffic is introduced because ACKs are used from destination to source, so the reverse signalling can be done in this way.

Backward congestion

This approach is used when a node detects congestion setting BECN bit to 1. It is much more complex than forward algoritm because an ad-hoc signalling is needed in order to reach the source. The other issue is that the signalling have to be propagated to all congested virtual circuits, so it is required to mark all of them. Main properties are:

. fastly than forward algoritm: in the worst case time needed to signal congestion is half a RTT (if the node who detects congestion is closed to the receiver);

. much more complex (store a list of congested DLCI and wait or create traffic to reach the source).

Source behavior

The capacity of network to react to congestion problem depends on the RTT and it is very different chosing backward or forward technique.

In FECN technique the react is slow and the source does not know where and how congestion is detected over the paht: it simply compute the percentage of packets received with FECN bit set to 1 over a time interval. Notice that if the source goes out it can not receive packets with FECN so congestion problem is not solved.

BECN technique react istantly reducing the rate when a frame with BECN bit is set to 1, otherwise rate in increased.

Issues of congestion control

The first problem is fairness: there are lots of differences among the two techniques. For example, with BECN approach, if a source receives a lot of congestion signals decrease a lot bit rate for all flows when is possible there

was only short term congestion. In general all flows who generate more are penalized, but in some sense, can be positive block anyway all virtual circuits who create too much traffic. Otherwise, it is possible that a source keep silent for a long period, wake up and generates lot of packets for a short period and then come back silent: if in the period in which it want to transmitt flows are block that transmitter will be too much penalized.

Any kind of reaction to congestion is always slower than a RTT because the signalling takes at minimun half a RTT then verification process (when source has already received the signalling and reduces its transmission rate) takes another RTT.

Another issue to take care is the RTT duration: for example, a source who generates a lot of traffic and have a small RTT, in BECN technique, receives a lot of signals, so reduce bit rate more than a source who generates few packets with bigger RTT. In this case, the first one is penalized.

Detect congestion is a very difficult problem to solve, infact if rate measures are used there are following constraint to keep in mind:

. the interval over which measures are taken depends on the RTT, but nodes do not know this value; moreover to take a relevant average measure the period considered have to be greater than a RTT;

. since there is only one bit to signal congestion, it is impossible to send information on the proper rate to use: the source knows only if congestion is present or not; probably takes complex measures not really used is an useless operation;

. rate measures are not so relevant looking to the wide network: a virtual circuit congestioned somewhere probably in a previous link should send at an higher speeed.

Another approach to detect congestion is looking at the buffer size of nodes (indirect measure): considering a threshold, if the occupancy is greater than that threshold the node is congestioned. This is an indirect measure and is more performant since comparisons are done with integer numbers while rate measure with real numbers. Moreover this approach does not distiguish about flows but looks at aggregate traffic.

There are two methods who can implement buffer occupancy:

. *istantaneous buffer occupancy* is faster but unstable;

. *average buffer occupancy* reacts slowly but it is stable.

For the second method time interval have to be determinated and this is an issue while the istantaneous buffer occupancy can not distinguish among situation very different: assume that at time t_x there is congestion over two links estimated in 120%; the link A previously presented a situation of 80% at time t_{x-1} while link B was 200%: the first implies that load have increased too much while in the second situation there is congestion but the link is underloaded. Also average buffer occupancy can not distinguish among these situations if the period considered is too small.

Another approach is *derivative occupancy* which is more complex and precise than others approaches takes care to consider the period in which congestion problem is detected.

The threshold value have to be chosen close to empty buffer in order to have more free space to collect packets but in this way if in certain time arrive a lot of packets congestion is signaled although the situation is temporary.

All parameters, threshold, windows size, RTT change in time so if algorithm is very fastly but sensitive reacts not well than a slowly but robust parameter. For buffer size the only constraint to be respected is the cost since in Frame Relay there is no delay contract; in order to avoid packets loss it is just need to have large buffer size as possible. In this way if there is a large number of connections, large RTT or large connection rates packets will never be lost.