QoS in the Internet

Inernet approach is based on datagram service (best effort), so provide QoS was not a purpose for developers. Mainly problems are:

. recognizing flows;

. manage the issue that packets may follow different paths to reach the destination.

When new applications come, for example multimedia, was necessary provide some QoS and that realization was derived by already known concept of B-ISDN.

To provide QoS is necessary:

. **packet classification** (first postulate): done at network edge, try to maps packets on flows; in this way, switches can distinguish packets not only on the base of addresses, but also on the base of flows and, eventually, class of service; it allows to provide pricing polices, for example, if there is too much people in the highest priory class, that number can be reduced increasing the cost of having that service;

. **traffic contract verification** (second postulate): this is done to be sure that traffic generated by users is conformant form the network point of view (if there are traffic profiles defined), but also, from user's point of view, to be sure that, if their traffic is conformant, hte network gives exactly QoS negotiated; this task is performed at network ingress and borders;

. **flow isolation** (third postulate): it guarantees that, data generated by different sources are separated to provide different classes of QoS and priorities; implemented in all routers, possible algorithms are:

  . resource partitioning (trunking);
  . WRR;
  . WFQ;

. acccess control (fourth postulate): the introduction of CAC changes radically the historical behavior of Internet because it is needed to define the concept of call; on the base of this definition, it is possible to said that a call is accepted if:

  . it receives the negotiated QoS;
  . if the acceptance does not reduce QoS of other calls already accepted;

    . if it does not create congestion;

  moreover it needs a specific management of signaling:

    . how it can be exploited;
    . how it can be requested;
    . how it is possible evaluate the given QoS;

  these facts require coordination of routers while they mainly work independently;

. **obtain high resource utilization** (fifth postulate): this issue allows to have:

    . low service cost;
    . high revenues;
    . the possibility by factors to provide services with high added value;

  it can be obtain using:

    . statistical multiplexing;
    . statistical description of QoS requirement;
    . work conserving scheduling algorithms.

IETF proposal are mainly four:

. improve best effort traffic;

. define a QoS architecture:

    . integrated services: stateful approach based on RSVP, protocol for signaling;
    . differentiated services: extend the concept of traffic priority controlling the quality per traffic class; it is easier than the previous approach, but it scales well;

. MPLS (Multiprotocol Label Switching): introduce a service that is packet switching with virtual circuit thanks to labels; the approach is stateful and the routing is based on labels; mainl issues are:

    . how labels are assigned;
    . number of labels that have to be used.

. protocols for multimedia applications.

Router buffer management

Different buffer policies may affect significantly efficency and fairness; two mainly issues are:

- when packets have to be dropped:
  - drop tail approach (buffer is full, so no other choices are avaiable);
  - AQM (Active Queue Management) when there is an earlier check on buffer growness;
- when packets have to be discarded:
  - different applications have different priorities:
    - discarding a voice packet is critical;
    - discarding a TCP data transfer packet is not bad because there are good recovery mechanism;
  - packets belonging to flows that create congestion:
    - counting them;
    - measuring the rate;
  - packets at the head of the queue because those one are too old so maybe useless.

Main goals are:

- check the number of packets in the buffer;
- offer fairness to best effort;
- protect responsive flows (TCP make congestion control, but UDP no, so if in a congested link both TCP and UDP packets are present, TCP reduce its rate and this allows to UDP to increase it);
- obtain an high output link utilization.

Drop tail buffer management

When there is congestion, the more aggressive sources get more the buffer obtaining an higher throughput; infact, buffer policies such as, increase the buffer size, allows only to deal with short term congestion.

Drop tail approach is easy to implement and, as a general fact, the more is large a buffer, the more is low the probability of losses. Moreover, since buffer has a cost in terms of managing, exploit them at the most is a benefit.

Mainly drawbacks are:

- flow punishment reguardless of:
  - their own behavior;
  - their service requirement;
- bad for TCP for:
  - synchronization;
  - timeouts.

AQM buffer management

These techniques start drop packets earlier than Drop Tail approach. The most important algorithm is RED (Red Earlier Detection). It is simple to implement, guarantees fairness in terms of dropping (no penalization of any kind of flows) and it works with a single queue. The purpose of RED is having a low average occupancy so that:

- low delays are guaranteed for TCP and multimedia applications (very useful);
- an high output link utilization is exploited.

Moreover, RED does not cause any kind of problem with TCP because just one packet is dropped per time window, therefore TCP can exploit a fast restart (problems occurs when burst of packets are lost). Principles of RED:

- detect congestion through measurements of the average buffer occupancy (not at the istantaneous occupancy therefore a time window have to be defined: it means that it is a parameter to set up); the detection takes place earlier, as a consequence the dropping phase occurs earlier therefore TCP reduces its rate;
- drop more packets if the congestion is more severe, but it does not mean that all packets are dropped as the Drop Tail approach;
- drop more packets from more active flows: for example, for TCP, shortest is the RTT, highest is the throughput:

$$Th = \frac{L|_{\text{bit}}}{\text{RTT}}$$

  receiving earlier ACKs means transmitting another packet earlier;
- drop packets even if the buffer is not full.

The dropping scheme is probabilistic and that probability depends with the level of congestion: it increases when the buffer occupancy increases.

The scheme is exploited by means of two threasholds: before the first there is no dropping phase, between the two the probability increases and after the second there is a full dropping phase. The probabilistic scheme is adopted to:

. avoid dropping several adjacent packets in the same flow;

. active flows are statistically penalized;

. TCP connection synchronization probability is reduced.

In short term there is some unfairness for those flows that are penalized at that moment, but in long term no because, not penalized flows, will create more load so, if congestion occurs again, the will be statistically more penalized.

Problems:

. difficult set up correctly parameters;

. for an high number of TCP flows the probability is more or less p_{max} and the algorithm becomes unstable: possible avoid using 3 threasholds instead of 2;

. if flows are *non-responsive* the algorithm does not work properly (the assumption is that only for TCP standard flows it works correctly);

. even if only responsive flows are present the low pass bass filter, used to estimate the buffer occupancy, introduce a delay.

RED extensions are vary: one of the most important is WRED (Weighted RED) that distinguishes flows in classes and, for example, the premium class starts dropping packets later.

Internet QoS architectures

These architectures require signaling and they can be divided into:

. IntServ: integrated services;

. DiffServ: differentiated services.

Integrated services

Here flows are recognized and:

- QoS is provided and negotiated for each application flow;
- traffic is policed for each flow;
- nodes reserve needed resources for each flow;
- flows are accepted on the base of signaling procedure where each application tries to open a separate flow that could be accepted or refused.

In IntServ:

- traffic flow is characterized by a vectorial representation called T-spec;
- QoS requirements are characterized by a vectorial representation called R-spec.

Nodes use T-spec and R-spec to establish if resources are avaiable to satisfy simultaneously a pair of T-spec and R-spec.

RSVP It is a protocol (Resource reSerVation Protocol) for IntServ that, hop by hop, carries signaling messages over IP. A set of parameters are not specified:

- multicast routing protocols;
- CAC;
- node resouce reservation algorithm;
- how provide QoS.

Properties:

- support for both unicast and multicast: sources sending at the minimum means that they adapt the rate at the minimum of all receivers and everyone get the minimum; it is possible that when users join a renegotiation takes place to adapt the sender at the newest lowest rate: it happens also when users leave the communication for the same reason;
- support heterogeneous receivers, but they have to confirm their capability of receiving that quality recognizing the code used;
- adaptation to flow modifications, done in *soft-state*:
    - nodes keep information only for a limited amount of time;

- resources are not explicity freed;
- each reservation have to be refreshed or, after some time, it will be dropped; this is done because if the path changes the reservation has to change accordingly.

The last property can introduce an overhead if the path does not change and it is relevant because the time for which the reservation is maintained must be short otherwise the detection will be too long causing a damn for the connection. The key point, infact, is that the routing is assumed to be static; in the other case:

- the new path may not have the same quality reserved;
- users do not have a connection guaranteed;
- resources are not avaiable in the new path.

Each data flow issues its own signaling request and what is a data flow depends on user's decisions and applications.

Control messages are encapsulated in IP packets and, in general, ACKs are not necessary end to end to confirm a reservation, but they are mandatory for failures. The complexity given by a *many to many* multicast communication is treated as many *one to many* communications.

The soft approach does not require strictly a message for the end of the session, but a TEARDOWN message is sent anyway because it is important for the source. The soft state, moreover, guarantees quality for the whole duration of the session only if routes do not change.