Asymmetric Best Effort Service for Packet Networks

J.Y. Le Boudec, M. Hamdi, Lj. Blazevic and P. Thiran *

Abstract

We propose a system and method for providing a "throughput versus delay" differentiated service for IP packets. We distinguish two types of traffic: type A and type B. It is expected that type A traffic receives less throughput per flow than type B. On the other hand, type A packets experience considerably smaller delay. The method is intended to be implemented in Internet routers. No bandwidth or buffer reservation is assumed in this system. The service remains a Best Effort service, thus supporting flat rates and free access to all applications. This is intended to be fully compatible with the current Internet. Furthermore, no change in the applications or protocols is required.

The system we propose is based on a collection of processing procedures to be implemented in IP routers. These procedures are organised in four modules and are designed to achieve the asymmetric performance for the two types of packets.

The originality of this work is two fold: we give the definition of the asymmetric performance in a best effort environment and propose the original algorithms and procedures to be implemented in network nodes to achieve the said definition.

1 Technical Field

- Internet Routers. Best effort service.
- Queue management, packet scheduling and bandwidth sharing between different types of traffics.

2 Goal

We propose a system and method for providing a "throughput versus delay" differentiated service for IP packets. We distinguish two types of traffic: *Type-A* and *Type-B*. It is expected that *Type-A* traffic receives less throughput per flow than *Type-B*. On the other hand, *Type-A* packets experience considerably smaller delay. The method is intended to be implemented in Internet routers. No bandwidth or buffer reservation is assumed in this system. The service remains a Best Effort service, thus supporting flat rates and free access to all applications. This is intended to be fully compatible with the current Internet. Furthermore, no change in the applications or protocols is required.

We assume that sources sending *Type-A* or *Type-B* traffic implement standard congestion control procedures mandated by the IETF (namely, they should be "TCP-friendly" as defined in [4]).

The effect of a congestion consists in delaying data packets and dropping some of them in case of lack of resources. The dropped packets are interpreted in the end systems as a negative feedback and result in reducing the emission rate of the sender application. The main novelty of our approach is to have the following two properties ensured at any time:

- the amount of negative feedback signals is unequally partitionned among the two different types of traffic in such a way that *Type-A* traffic receives more negative feedback than the other and hence receives less throughput
- *Type-A* traffic is ensured a shorter delay than *Type-B* traffic

*Copyright to EPFL-DI-ICA. All Rights Reserved. Technical Report Number SSC/1998/027

The negative feedbacks can be represented by two kinds of signals: explicit feedback signals (e.g. ECN bits in packet header) and implicit feedback (e.g. packet loss). In the remainder of this document, the proposed approach is illustrated using packet loss as a way to provide negative feedback. Nevertheless, the approach remains valid in the case of systems using explicit congestion notification (ECN) because a simple interpretation of the given embodiment allows to adapt it to ECN based networks.

More specifically, the impact of a congestion, in terms of packet losses and delay, seen by traffic of *Type-A* is different than for traffic of *Type-B*, and, the traffic that sees less loss ratio encounters more delays and vice-versa. A practical example is that during a congestion period, short delays are experienced by packets carrying voice data generated by Internet Telephony and Videoconferencing applications. Traffic belonging to non realtime connections (e.g. data using TCP, delay adaptive applications) encounters lower loss ratio but longer delays.

With this definition, the network-level quality of service (packet loss and delay) received by the two types of traffic cannot be classified in a best-worst dimension. Each traffic type receives a different quality. It is the appropriate matching between the QoS received and the application nature that allow to take advantage of this system.

These methods do not necessarily rely on a per-flow information processing. Packets treatment is differentiated only according to a generic packet classification method.

The system we propose is based on a collection of processing procedures to be implemented in IP routers. These procedures are organized in four modules named Classifier , Dropper, Scheduler and Collector.

The methods we propose are collections of algorithms for the described modules, that allow to achieve the asymmetric performance described above.

The novelty of this system is two fold:

- the definition of the asymmetric performance in a best effort environment.
- the original algorithms and procedures we describe to be implemented in network nodes to achieve the said definition

3 General Description

We define computing procedures (algorithms) to be implemented/added to routers to achieve the defined performance. These functions are performed at the output transmission port, after the routing operation as shown in Figure 1. These functions are organized in Modules.

We define two types of traffic called *Type-A* and *Type-B*. For example this classification may refer to an application level classification (e.g. Real-Time and Non-Real-Time). In the defined system, the semantics of *Type-A* and *Type-B* are completely defined by the algorithms of the Classifier.

3.1 Definition of Asymmetric Best Effort Service

The following notations apply:

- $Session(T)$: an observation session of duration T. All the definitions refer to a single node.
- PLR_A : the packet loss ratio of *Type-A* traffic during $Session(T)$
- PLR_B : the packet loss ratio of *Type-B* traffic during $Session(T)$
- D_A : the average packet delay of *Type-A* traffic during $Session(T)$
- D_B : the average packet delay of *Type-B* traffic during $Session(T)$

The network node is said to offer an Asymmetric Best Effort Service to traffics of *Type-A* and *Type-B* during $Session(T)$ if and only if we have:
(*Property-1*)

$$PLR_A \geq PLR_B$$

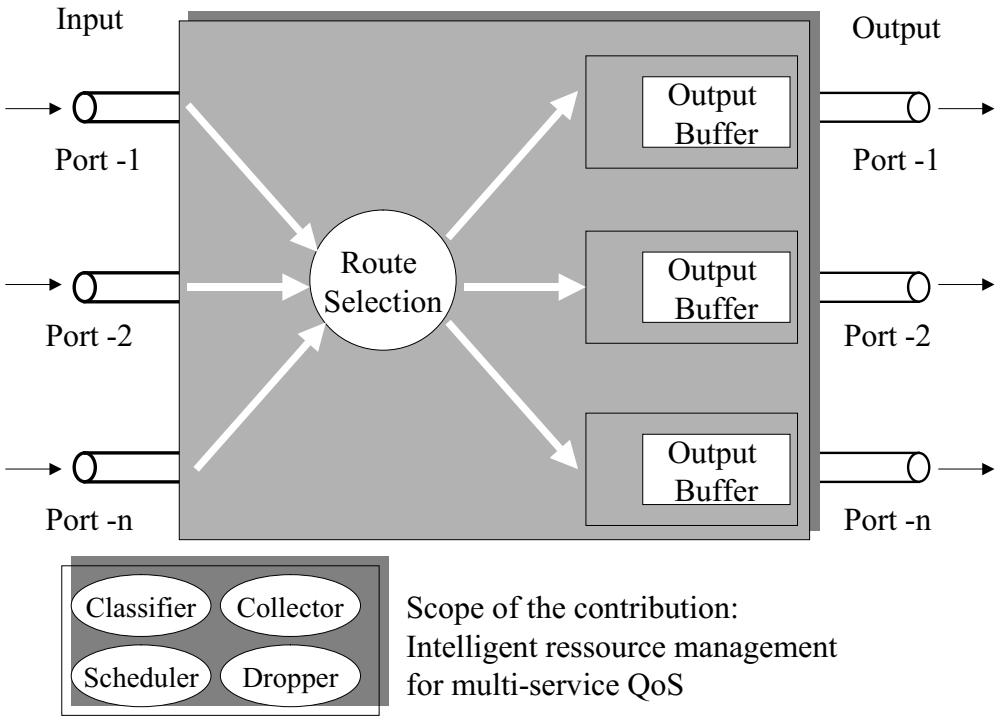

Figure 1: Scope of the contribution

and (*Property-2*)

$$D_A \leq D_B$$

In practice, the value of T is related to the statistical significance of PLR. Typical values are around few seconds.

4 Detailed Description

In this section we describe a collection of mechanisms that defines the system to be used in the routers to ensure an asymmetric best effort service for interactive and non interactive data. The design principles of this scheme are the following:

- This implementation is designed to work with existing applications and protocols. There is no assumption of new definition of IP fields or any change that may require to modify the existing applications. However, the proposed system can advantageously follow future updates (if any) of the protocols and applications.
- The incoming packets are classified into two types A and B. *Type-A* packets refer to interactive traffic where the end to end delay has to be short (e.g. 10 to 30 ms). An example of *Type-A* traffic is Internet Telephony and videoconferencing traffic. *Type-B* packets refer to non-interactive traffic

where the end to end delay can be variable. Examples of *Type-B* traffic are data traffic (e.g. TCP traffic) or delay adaptive stream-like applications (playback audio and video applications).

- The amount of negative feedback (e.g. packet losses) received by *Type-A* traffic is greater than the one received by *Type-B* traffic. The fraction of *Type-A* packets that are admitted are ensured a short delay.
- The port service rate is shared between the two traffic types. No rate reservation is assumed. During a silence period of a given traffic type, the other type makes use of the whole bandwidth.

The detailed system is drawn in Figure 3.

4.1 The computing procedures

The general block diagram for packet processing within a router is illustrated in Figure 2.

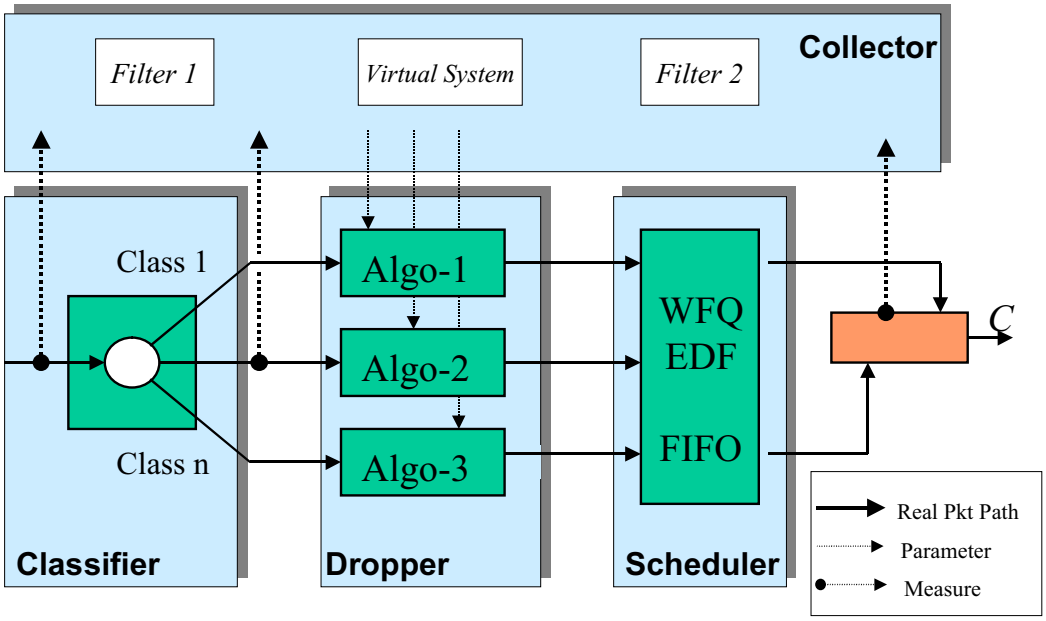

Figure 2: General block diagram of packet processing modules within a router

Basically there exist two classes of algorithms performed in a router related to traffic control. Those are queue management and scheduling. Queue management algorithm manages the router queue by dropping packets when necessary or appropriate. Scheduling algorithm determines which packet to send next while packets are in the buffer.

In this system, queue management algorithm is performed in the Dropper module while the scheduling algorithm is performed in the Scheduler module. The Classifier module identifies the traffic type of the

incoming packet. The Collector module collects information about traffic profile of incoming packets and information from the buffer. This module also performs rate estimation, averaging and filtering of measured parameters that are used by the Dropper and Scheduler modules.

4.2 The Classifier

Identifies to which type the incoming packet belongs. Two kinds of information can be used to identify the traffic class: explicit protocol information (e.g. IP fields) and implicit parameters that may help the decision (e.g. higher protocol information). Examples of explicit parameters are:

- IPv4 TOS field (e.g. short delay bit)
- MPLS label
- IPv6 Flow Type

Examples of implicit parameters are

- Packet size (voice packets are generally of small size).
- Higher layer protocols (realtime packets usually use UDP and RTP)
- Port numbers
- Any bit pattern in packet header
- Any application signature in the payload field

The determined type of the incoming packet is used by both the Dropper and the Scheduler.
An example of a classification procedure is:

IF (upperlayer protocol = UDP) AND (PacketSize $\leq$ SIZE_THRESHOLD)

- THEN Type = A
- ELSE Type = B

where SIZE_THRESHOLD is set according to statistics on interactive audio and video packets size. Its default value is set to 200 BYTES.

4.3 The Dropper

The Dropper decides whether the incoming packet is accepted in the buffer of the output port or not. If not the packet is dropped. The Dropper performs a dropping algorithm per type of traffic. The dropping algorithms are designed to meet *Property-1* of the definition above. These algorithms use some parameters that are collected and maintained by the Collector.

The general expression of a dropper is specified by a *Target Loss Objective* which defines the relationship between the loss ratio seen by the two types of traffic .

This objective is a more specific formulation of *Property-1*. This objective gives an expression for the so-called Drop Probability Function of each traffic type. The Drop Probability Function gives at each packet arrival the probability that the packet must be dropped, as a function of system parameters (e.g. estimated arrival rate) that are observed or computed.

The following notations apply:

- $C > 0$ is the service bit rate of the output port
- $e(t) > 0$ is an estimation of the instantaneous arrival bit rate at time t, this comprises both types of traffic

- $e_A(t) > 0$ is an estimation of the instantaneous arrival bit rate at time t of *Type-A* traffic
- $e_B(t) > 0$ is an estimation of the instantaneous arrival bit rate at time t of *Type-B* traffic
- $q(t) \in [0,1]$ a target overall packet loss ratio for time t
- $q_A(t) \in [0,1]$ a target packet loss ratio for time t of *Type-A* traffic
- $q_B(t) \in [0,1]$ a target packet loss ratio for time t of *Type-B* traffic
- $\alpha > 1$ a drop differentiation factor

where we have:

$$e(t) = e_A(t) + e_B(t)$$

and

$$q_A(t)e_A(t) + q_B(t)e_B(t) = q(t)e(t)$$

We define the target overall packet loss ratio as:

$$q(t) = \sup(0, 1 - \frac{C}{e(t)})$$

The *Target Loss Objective* is the expression of $q_A(t)$ as a function of $q(t)$, α, the output port sevice rate and the estimated arrival rates:

$$q_A(t) = f(\alpha, q(t), e_A(t), e_B(t), C)$$

By way of the following expressions, we give two examples of this function:
Example-1

$$q_A(t) = \alpha q(t)$$

Example-2

$$q_A(t) = \alpha q_B(t)$$

Example-3

$$q_A(t) = 1 - (1 - q(t))^\alpha$$

Note that these three examples ensure *Property-1* defined above. The explicit Drop Probability Functions $q_A(t)$ and $q_B(t)$ are then derived immediately. For Example-1, their explicit expression is:

$$q_A(t) = \alpha(1 - \frac{C}{e_A(t) + e_B(t)})$$

$$q_B(t) = (1 - \frac{C}{e_A(t) + e_B(t)})(1 + (1 - \alpha)\frac{e_A(t)}{e_B(t)})$$

For Example-2, their explicit expression is:

$$q_B(t) = (1 - \frac{C}{e_A(t) + e_B(t)})\frac{e_A(t) + e_B(t)}{e_B(t) + \alpha e_A(t)}$$

$$q_A(t) = \alpha e_B(t) = \alpha(1 - \frac{C}{e_A(t) + e_B(t)})\frac{e_A(t) + e_B(t)}{e_B(t) + \alpha e_A(t)}$$

Note that expressions for target overall packet loss ratios are completely defined by $e_A(t)$, $e_B(t)$, α and C. To implement the above Target Loss Objectives, specific algorithms for the estimation of $e_A(t)$ and $e_B(t)$ are needed. Original estimation algorithms are specified in the Collector section.

4.4 The Scheduler

It decides for the queueing position of the accepted packet (that survived the dropper). Usually, FIFO scheduling is used for all the IP traffic. Known scheduling algorithms are:

- Weighted Fair Queueing (WFQ) [7] allows to define a bandwidth share (weight)for each flow (or class) and guarantees a service rate proportional to the weight.
- Class-Based Queueing (CBQ) [3] allows to reserve a percentage of the port output rate to each traffic class. It defines a hierarchical tree of traffic classes.
- Earliest Deadline First (EDF) (e.g. [5]): each packet i arriving at t_i is assigned a finishing service time deadline d_i. At any time t, the packet having the lowest value of $(t_i + d_i)$ is served first (i.e. earliest deadline).
- FIFO+ is defined in [2]. Takes advantage of multi-hop FIFO queues to give priority to the most delayed packets.
- LIFO queueing, has shown interesting performance when used as an overload strategy for time constrained traffic [6].
- Fixed priority classes allows a flow with higher priority to be served before lower priority flows.

The scheduling method is responsible for guaranteeing *Property-2* of the system specifications. We define two scheduling methods to be used in this embodiement:

- Two priorities (high and low) are defined. Each admitted packet is assigned one of these two priorities. A packet of high priority is inserted just after[1] the last packet of the same priority and before all packets of low priority. If the buffer is full, packets will be dropped from the tail of the queue as necessary (pushed-out). A packet of low priority is inserted at the tail of the queues, after the last low priority packet. If the buffer is full, the entering packet is dropped. A packet being served is never interrupted (even if it is low priority and a high priority packet arrives).

  The scheduler tags all packets of *Type-A* as high priority and all packets of *Type-B* as low priority. This algorithm ensures *Property-2*.

- Although the first algorithm can be implemented simply using static priorities, we define another algorithm based on Earliest Deadline First scheduling. Two deadlines[2] are defined to match the two priorities: d_A and d_B with $d_A < d_B$. The advantage of this scheduling is that it behaves identically to the static priority scheme as long as the packets of *Type-B* are served before $d_B - d_A$ time after their arrival. By appropriately setting the deadline values, this scheme prevents from service starvation for *Type-B* due for example to estimation errors or to bandwidth reduction (in shared medium access interfaces).

The value of d_B has to be set so that it reflects the maximum reasonable time a packet of *Type-B* can spend in the system. If t_i is the arrival time of the i^{th} packet of *Type-B*, its deadline $d_B(t_i)$ is set to the following:

$$d_B(t_i) = 2\frac{B}{(1 - q_B(t_i))e_B(t_i)}$$

where B is the buffer size of the output port. In addition, to ensure FIFO queueing between packets of *Type-B*, the packet tag $T_B(i)$ that defines the packet order in the buffer is set to:

$$T_B(i) = \max(T_B(i-1) + \epsilon, t_i + d_B(t_i))$$

[1] after means that it will be served after

[2] deadline here refers to the maximum sejourn time in the buffer, it is therefore not an absolute time. The instant at which the packet should be served is obtained by adding the arrival time to the deadline

The explanation of the $d_B(t_i)$ expression is that if we assume the arrival rate does not change too much during the buffering time of the packet, this packet should see a service rate of $(1-q_B(t_i))e_B(t_i)$ and then should be served at most within $B/(1-q_B(t_i))e_B(t_i)$. The factor of 2 is to empirically compensate the assumption error margin. The tags of *Type-A* traffic are simply set to:

$$T_A(i) = h_i$$

where h_i are the arrival instants of *Type-A* packets.

Note that when the scheduler inserts an admitted packet, the last packet of the queue (or the entering packet) is dropped if the buffer is full. This does not happen if the estimation is accurate, therefore these losses are called un-expected losses. We define $UL(s,t)$ (resp. $UL_A(s,t)$ and $UL_B(s,t)$) the amount in bits of un-expected losses in the aggregate traffic(resp. in traffic of *Type-A* and in traffic of *Type-B*) between s and t. These losses are used as a feedback signal to the estimation procedures.

4.5 The Collector

The Collector module is responsible for collecting, maintaining and processing all data that is needed by the algorithms of the Scheduler and the Dropper. An example of collected data is the estimated arrival rate. The collected data may refer to the state of a virtual system which behaviour is maintained in the Collector. Filters and estimators are examples of these procedures.

To compute $e_A(t)$ and $e_B(t)$, we define two mechanisms to be used together: an estimation procedure and an accuracy control procedure.

4.5.1 Estimation procedure

We define two estimation procedures:

- The first procedure is based on the computation of the deterministic *effective bandwidth* [1]. If $\alpha(s)$ is an arrival curve of the estimated traffic and D a delay bound , the effective bandwidth E^D is defined as:

$$E^D = \sup_{s \geq 0} \frac{\alpha(s)}{s + D} \tag{1}$$

  In order to approximate an instantaneous estimation, the effective bandwidth needs to be computed on a sliding window of time w. It is hence given by:

$$E^D(t) = \sup_{t-w \leq t_i \leq t_j \leq t} \frac{P_i + ... + P_j}{t_j - t_i + D}$$

  where P_i is the size of the packet that arrives at time t_i.

- The second procedure is based on the computation of the deterministic *equivalent bandwidth* [1]. If $\alpha(s)$ is an arrival curve of the estimated traffic and B a buffer size , the equivalent bandwidth F^B is defined as:

$$F^B = \sup_{s \geq 0} \frac{\alpha(s) - B}{s} \tag{2}$$

  Similarly, the equivalent bandwidth needs to be computed on a sliding window of time w and is given by:

$$F^B(t) = \sup_{t-w \leq t_i \leq t_j \leq t} \frac{P_i + ... + P_j - B}{t_j - t_i}$$

  where P_i is the size of the packet that arrives at time t_i.

Broadly speaking, the difference between the two methods is that in the first one, the smoothing effect of the estimation results in a bounded delay D[3] while in the second method it results in a bounded buffer

[3] defined by the maximum delay encountered by the traffic if it were serviced at the rate E^D

size B[4]. The choice of the method to be used depends on whether the constraints are in terms of delays or buffer size.

The estimation of traffic *Type-A* is performed using the effective bandwidth approach where the delay parameter D is fixed according to the target maximum delay (set for example to 10 ms). However, the equivalent bandwidth is used to estimate *Type-B* traffic. The estimation parameter B then corresponds to the actual buffer size of the output port.

Exponentially Weighted Moving Average (EWMA) is a known low-pass filter that can be used to have an averaged value of a measured parameter. Given a series of measured values $\{X_i\}$, its EWMA is given by:

$$Y_i = aY_{i-1} + (1-a)X_i$$

where a is a tunable coefficient that controls the smoothing effect of the filter.

The values of $E^D(t)$ (resp. $F^B(t)$) are averaged using an EWMA filter to give a raw estimation function (for sake of simplicity, we keep the same notations $E^D(t)$ and F^B that henceforth refer to the EWMA filtered values).

4.5.2 Accuracy Control Procedure

The estimation can temporarily fail to track the instantaneous arrival rate. An accuracy control is needed because the system expressions are entirely based on the estimated values. Accuracy control is based on tracking the state of a virtual system that measures the distance between the actual traffic and the estimated values and react back on the latter to give an adjusted estimation function which was denoted $e(t)$ (the A and B indexes are removed for now). In addition, the *un-expected losses* seen by scheduled packets is used as a feedback signal. Let $E(t)$ denote the raw estimation function (either $E^D(t)$ or $F^B(t)$). The adjusted estimation function $e(t)$ is given by:

$$e(t) = E(t)\beta(t)$$

where $\beta(t)$ is the accuracy control factor. $\beta(t)$ is controlled by the state of a virtual system and the amount of un-expected losses $UL(s,t)$.

We define two original procedures for accuracy control using virtual systems:

- the first procedure is defined by a virtual buffer that is continuously emptied at rate $e(t)$ and is filled as the packets of the estimated traffic arrive by the equivalent number of bits. Let $X(t)$ denote the virtual buffer fullness at time t. $X(t)$ is updated each packet arrival as follows:

  $$X(t_i) = P_i + \max(0, X(t_{i-1}) - e(t_{i-1})(t_i - t_{i-1}))$$

  The correction factor is updated each Δ period of time using the following control:

  $$\beta(t+\Delta) = \beta(t) + K_1\frac{UL(t,t+\Delta)}{\Delta e(t)} + K_2\frac{\sup_{t\le s\le t+\Delta} X(s) - B}{e(t)}$$

  where $K_1 > 0$ and $K_2 > 0$ are constants to be tuned for the implementation.

- the second procedure uses two virtual buffers. The first one is the same as the one just defined in the first procedure. The second one is a virtual buffer that is continuously filled at the rate $e(t)$ and emptied each packet arrival by the equivalent amount of bits. Let $Y(t)$ denote its fullness at time t. $Y(t)$ is updated each packet arrival as follows:

  $$Y(t_i) = e(t_{i-1})(t_i - t_{i-1}) + \max(0, Y(t_{i-1}) - P_i)$$

  The accuracy control factor is updated each packet arrival using the following control:

  $$\beta(t_i) = \beta(t_{i-1}) + K_1\frac{UL(t_{i-1},t_i)}{(t_i - t_{i-1})e(t)} + K_2\frac{X(t_i)}{e(t_{i-1})} - K_3\frac{Y(t_i)}{e(t_{i-1})}$$

  where $K_1 > 0$, $K_2 > 0$ and $K_3 > 0$ are constants to be tuned for the implementation.

[4] defined by the maximum buffer size that arises if the estimated traffic were serviced at the rate F^B

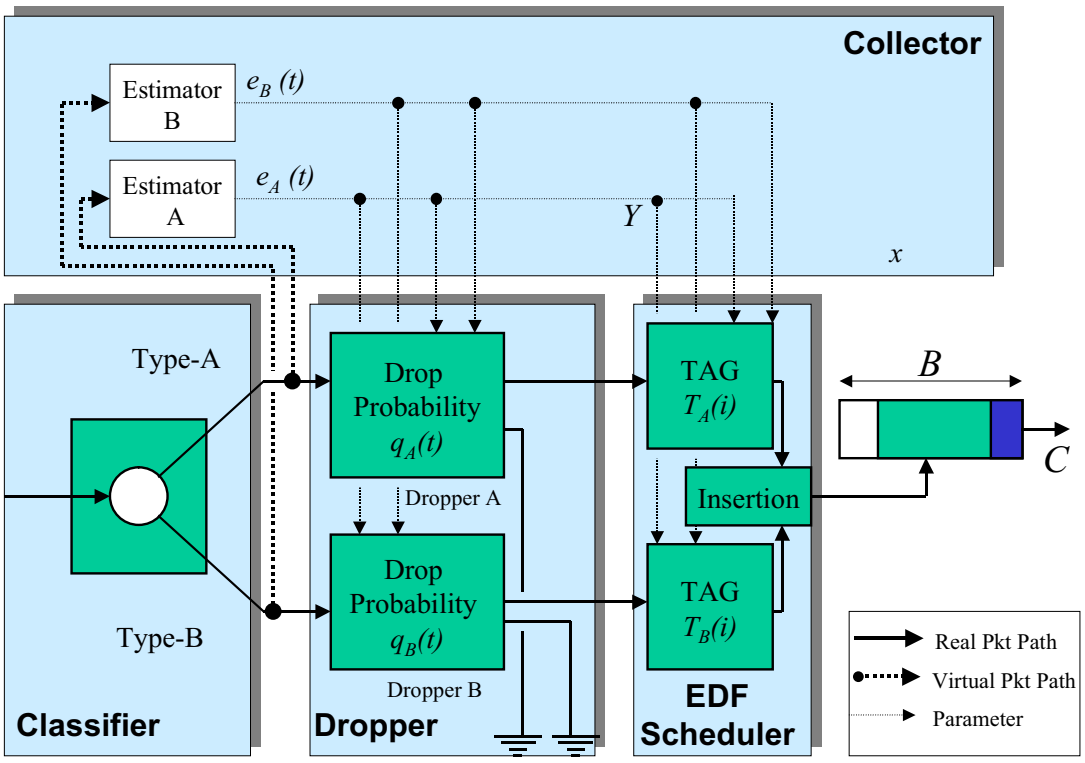

Figure 3: Detailed Drawing

5 Conclusion

We propose a method within routers for providing a "throughput versus delay" differentiated service for IP packets. The novelties of this approach are the following

- decoupling delay objectives from loss objectives
- the definition of the Asymmetric Best Effort Service where a traffic type receives less throughput and shorter delay than traffic of the other type
- the dropping mechanism is based on estimation of deterministic effective bandwidth and deterministic equivalent bandwidth
- the estimation methods are controlled by one or a set of virtual systems
- combined dropping mechanism with drop rates proportional to fixed weights

References

[1] J.Y Le Boudec. Application of Network Calculus To Guaranteed Service Networks. *IEEE Transactions on Information Theory*, 44(3), May 1998.

[2] D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of SIGCOMM*. ACM, August 1992.

[3] S. Floyd and V. Jacobson. Link Sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4), August 1995.

[4] Sally Floyd and Kevin Fall. Router Mechanisms to Support End-to-End Congetion Control. *Technical Report, Network Research Group, LBL, Berkeley CA*, February 1997.

[5] R. Guerin and V. Peris. Quality-of-Service in Packet Networks - Basic Mechanisms and Directions. *Computer Networks and ISDN Systems. Special issue on multimedia communications over packet-based networks*, 1998.

[6] M. Hamdi, R. Noro, and J-P. Hubaux. Fresh Packet FirstScheduling for Voice Traffic in Congested Networks. *EPFL Technical Report No. SSC/1997/034*, December 1997.

[7] A.K. Parekh and R.G. Gallager. A Generalized Processor Sharing Approch to Flow Control In Intergated Services Networks-The Single Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.