Quality of Service Vertical and Horizontal Integration in Active Nodes

David Fuin, Eric Garcia, Hervé Guyennet and Emmanuel Jeanvoine
LIFC - Laboratoire d'Informatique de l'université de Franche-Comté
16 route de Gray 25030 Besancon Cedex - FRANCE
Tel: +33 (0) 3 81 66 20 92 - Fax: +33 (0) 3 81 66 64 50
{fuin,garcia,guyennet}@lifc.univ-fcomte.fr Emmanuel.Jeanvoine@irisa.fr

Abstract

In this paper we focus on possibilities of quality of service mechanisms implementations in active routers. We show that a different location generates a different resulting QoS. Indeed, two types of QoS can be provided in active network: one that provides "classical" network QoS and one that provides more computing resources to priority flows.
We describe a number of implementations we have realized in ANTS and Tamanoir active nodes. In these implementations based on flow differentiation, we have developed several schedulers.
We present experimentations realized in order to validate our implementations. In these experimentations, we emphasize on the two kinds of QoS provided. We show the interest of our innovative and efficient approach.
Keywords: — QoS, Active Networks, Pre/Post-Processing

1 Introduction

In active networks, classical routers have been replaced by active nodes. Active nodes execute active services that best fit applications needs. These active services are executed in an execution environment at each node.

In a first part, we present existing works in the field of QoS in active networks. We summarize our novel approach [8] for providing congestion management at "active network layer" based on flow differentiation where we have developed FIFO, PQ, FQ and WFQ schedulers in ANTS [5] and Tamanoir [9] active nodes.

The aim of this paper is to focus on a novel study: we show that it is particularly important to pay attention to the QoS implementation location relatively to the node execution environment. We have realized two implementations: one in ANTS active node and another one in Tamanoir active node.

We present experimentations in order to validate our implementations. We test the benefit of our approaches with examples which illustrate differences between our implementations. Finally, we discuss about the interest of our innovative and efficient approach.

2 QoS in active networks: relative works

We can classify relative works according to the layer where QoS is applied [3]: QoS at application layer (active application QoS), QoS re-using subjacent IP layer and QoS at "active network layer" (see figure 1).

2.1 QoS at application layer

When we design a network application, it is important to take into consideration that the network can disturb the normal application running. This is especially important in multimedia applications, for example in video distribution where latency and jitter constraints are predominant to get a good rendering.

In this context, [4, 10, 2] propose to use active networks to adapt multimedia flow sending in order to prevent congestion. In case of congestion, this kind of QoS drops first less important packets before more important one.

This method solves some problems of network congestion but there are two major drawbacks:
• when the multimedia flow is reduced at its maximum there is no more solution,
• if there are simultaneous flows, it is not possible to give priority to a particular flow. This means that, in case of important congestion, flows that do not reduce their throughput can use all bandwidth.

Generally speaking, this solution allows flows to adapt their own behaviors [4] but disallows them to influence each other (it is impossible to give a priority to specific flows).

2.2 QoS re-using subjacent network layer

Other works have developed the idea to use QoS provided by the subjacent network layer (most of the time IP layer). In this case, the interest to use active networks is to dynamically reconfigure IP QoS policies.

In [1] the aim was to mark the DSCP field of packets using active networks. In [12], the objective was to use active networks in order to dynamically modify the QoS policies in classical routers.

On the one hand, using classical networks QoS is a real advantage for interoperability. On the other hand, QoS mechanisms usable are then limited to those provided by IP QoS. Most of the time, it is impossible, on a Cisco router for example, to activate two different mechanisms at the same time (or then, the result is unforeseeable), QoS we can obtain is limited. Moreover, the granularity of these mechanisms is flow-grained.

2.3 QoS at "active network" layer

This approach consists in managing QoS at "active network" layer level. Some works in this way deal with load balancing in the network by using genetics algorithms methods [11] or ants colonies moving [7]. These methods based on load balancing are quiet efficient but there are some problems due to slow adaptation with rapid traffic fluctuations.

Finally, in [6], flow differentiation mechanisms have been implemented in ANTS active node. The major problem in this implementation was the very poor performance. This is in this way that we have oriented our research in paying a particular attention to QoS mechanisms horizontal integration in active router architecture.

In [8], we have implemented queues/scheduler mechanisms that directly deal with active flows. Like DiffServ, we worked on a flow differentiation based approach. Active packets are marked at their emission with a class of service identifier. To perform a different processing according to the class of service, we put a set of queues in active routers in order to classify packets. We have implemented schedulers that choose packets in the queues according to a specific queueing discipline algorithm and sends them to the next node.

To implement these QoS mechanisms in active networks we have used ANTS [5] and Tamanoir [9] active nodes. ANTS has been chosen for its standard architecture and its easiness of use whereas Tamanoir has been chosen for performance reasons.

We have realized a great number of experiments showing the efficiency of our implementation. We also show that packet grained (signaling) QoS mechanisms could improve MPEG video rendering in congestion situation. Indeed, with our mechanisms it is possible to deal with packet-grained level whereas classical QoS can only deals with flow-grained level. Our proposition is really efficient; furthermore, it is not necessary that subjacent network layer provide QoS mechanisms.

3 QoS integration in active nodes

During our research, we focused on the position of QoS mechanisms in active router architecture. We deduce that it is possible to vary this position according to two axis: vertical and horizontal (figure 1).

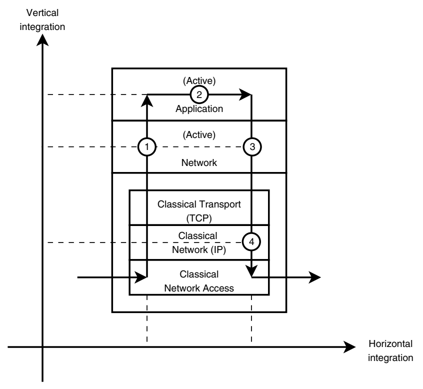

Figure 1. QoS in active networks.

In this paper, we emphasize on QoS mechanisms positioning and on the impact of this positioning on the resultant QoS.

3.1 Vertical integration

Considering the vertical architecture of an active router, we can, split it in two parts. The first part is the communication layer and the second one is the execution environment. We can consider these two levels to put our QoS mechanisms in.

Communication layer: The idea to place queues/scheduler mechanisms in the communication layer is directly issued from classical networks. Indeed, in these networks, QoS is totally transparent to the user because it is implemented under the application layer.

According to the active node used, access to the communication layer is more or less easy. For example, ANTS is based on the NodeOS layer which provides the active node CPU, memory and communication resources access abstractions. This layer is compliant with the pseudo-standard proposed by Wetherall [5] to standardize the active network architecture. This layer is particularly heavy and its modification is complex. For those reasons we have not implemented QoS mechanisms in ANTS communication layer.

In Tamanoir active node, things are different because its resource abstraction layer is minimal. Its communication layer is very light and it is relatively simple to work at this level. So, we have developed QoS mechanisms in Tamanoir communication layer.

Execution environment: The second approach consists in execution environment modification: packets are intercepted at execution environment layer output (just before sending them to NodeOS) and then sent to our queues/scheduler module. The latter send them in a particular order (depending on QoS policies) to NodeOS. We implemented this solution in ANTS (not in Tamanoir due to overhead caused by the use of an higher layer).

Discussion on vertical integration: As far as it is possible, it is better to implement QoS at the lower level in active router for performance reasons. A modification of the communication layer is then more interesting. However, it is not always simple (like in ANTS).

3.2 Horizontal integration

Once the QoS mechanisms vertical positioning is defined, we have to specify their location according to the execution environment. Naturally, it seems that QoS is different if packets are processed by QoS mechanisms before (① on figure 1) or after (②) their processing in the execution environment. According to active nodes used, it is not always possible to realize both implementations. Indeed, with ANTS it is only possible to put mechanisms after the execution environment. Due to the NodeOS layer, it is impossible to intercept packets before their processing in the execution environment. At the opposite, with Tamanoir we have implemented both versions.

Pre-processing implementation: In this approach (figure 2), QoS policies are applied before active services execution. In a congestion situation, the possibly dropped packets have not executed their service yet. This can be considered as CPU saving. It is also possible with this approach to give a priority on a specific flow.

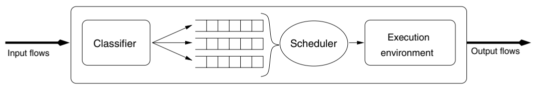

Figure 2. Pre-processing implementation.

Post-processing implementation: On the contrary, if QoS policies are applied after active service execution (figure 3), packets, in a congestion situation, will be processed whereas they might be dropped at the output of the router. The higher the executed service consumes resources, the more this approach is problematic. However, this method is similar to the one used in classical QoS and permits to obtain a more shaped output traffic.

Discussion on horizontal architecture: Whereas the first method offers a kind of QoS saving computing resources; the second one offers a kind of QoS only acting on network resources. It seems that the first one would be more efficient: in active networks, one important factor is computing resources. Indeed, the more efficient active nodes have gigabit throughput (with the simplest active service possible) whereas classical network have now ten or so gigabit throughput. The second method allows us to better manage flow where size of packets can change (packets can be enriched on active nodes).

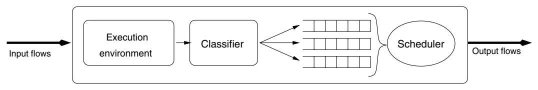

Figure 3. Post-processing implementation.

3.3 Discussion on implementations

Then we have focused on QoS mechanisms positioning in active nodes architecture. We have shown that it is particularly important to pay attention to the QoS location relatively to the execution environment of the node.

In pre-processing implementation, packets are processed by QoS mechanisms as soon as they arrived on the node and CPU is not wasted. Indeed, in a congestion situation, packets are dropped as soon as possible, before the processing of their active service. In post-processing implementation, packets are processed immediately after their active service processing: CPU resources can be wasted but at this moment, we know the real packet size (if packets have to be enriched, this has been done at execution environment level). Thus, bandwidth is best managed with post-processing solution (for example if packets are enriched on active nodes).

4 Experimentation and performance

In order to validate our approaches, we have realized a large number of experimentations (not simulation). We have tested QoS algorithms we have implemented: FIFO (No priority), Priority Queueing (Absolute priority), Fair Queueing (Throughput balancing) and Weighted Fair Queueing (statistical priority). We also have tested active QoS interest with real use cases in order to show differences between pre-processing and post-processing implementation.

4.1 Test architecture

Our test architecture is composed of two computers, each one sending a different class of service flow, an intermediate active router and a receiver. There is a throughput limitation on the link between intermediate active router and the receiver. This allows us to create an artificial congestion on the intermediate active router in order to observe queueing discipline algorithms behavior.

4.2 Heavy processing

Our goal was to show active QoS interest in real use case. We have tested a case where processing time on intermediate router is much longer with one flow than with another.

We have compared results between a FIFO scheduling and a Priority Queueing scheduling (absolute priority is given to the heavy processing flow). Furthermore, we have used two implementations of QoS mechanisms in Tamanoir: pre-processing mechanisms and post-processing mechanisms. Let's call A the heavy processing flow and B the light processing flow.

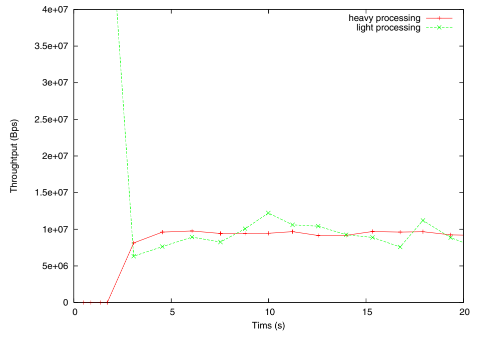

Figure 4. Heavy processing and FIFO pre-processing mechanisms

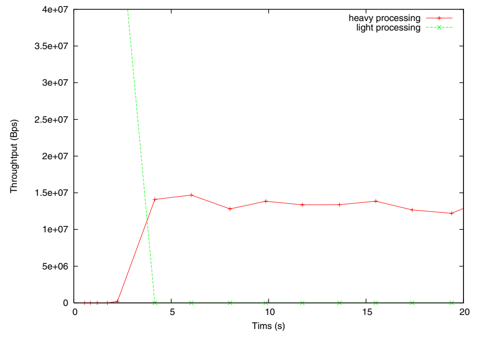

Figure 5. Heavy processing and PQ pre-processing mechanisms

Pre-processing QoS: With a pre-processing QoS we have said that it was possible to give priority to a flow processing. When we use a FIFO scheduler (figure 4), A and B have each about 10 Mbps throughput. With a Priority Queueing scheduler (figure 5), when the transmission starts, there are no more other B packets received. In this case, even if there is less consumed bandwidth, Priority Queueing gives all resources to A for the best processing. We can observe that A has a 15Mbps throughput. An absolute priority on the heavy processing flow is advantageous if A is very important and if it must be processed with high priority. The drawback is that bandwidth is wasted because it cannot be used by B.

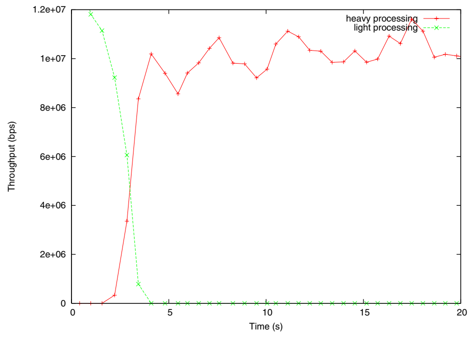

Figure 6. Heavy processing and PQ pre-processing mechanisms

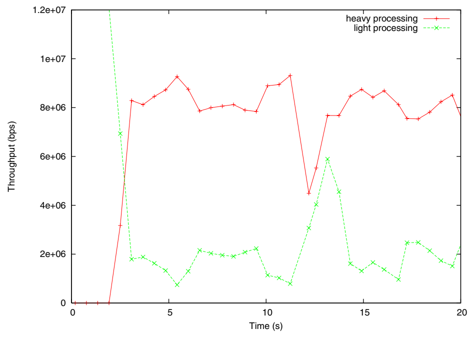

Figure 7. Heavy processing and PQ post-processing mechanisms

Post-processing QoS: We have measured totally different results with a post-processing QoS implementation. Even with an absolute priority on A, B throughput is not reduced to 0. In this test, we have used a 10 Mbps network. With our pre-processing implementation (figure 6), A uses entire bandwidth (10 Mbps) whereas with our post-processing implementation A uses only 8 Mbps.

We can see here that it is really important to take into account the processing time of packets. Indeed, in this test, pre-processing implementation allows us to give absolute priority to a heavy processing flow whereas it is not possible with post-processing QoS.

4.3 Enriched flow

We have performed a test where the first flow is enriched on intermediate router. The consequence is that its throughput is higher at the output than at the input. Let's call C the enriched flow and D the normal flow. We reserve 30% of the bandwidth to C and 70% to D and we limit global bandwidth to 70 MBps.

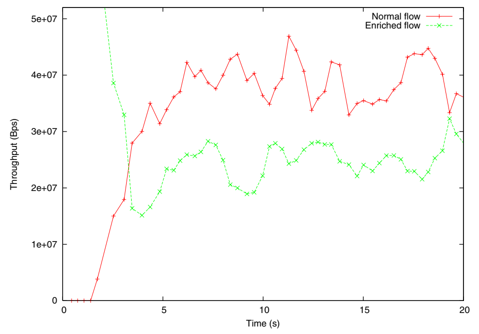

Figure 8. Enriched flow and WFQ pre-processing mechanisms

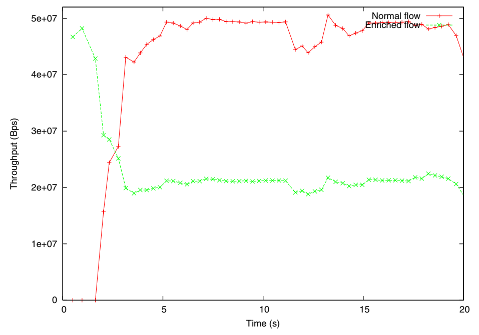

Figure 9. Enriched flow and WFQ post-processing mechanisms

Pre-processing QoS: With pre-processing QoS, we can observe on figure 8 that the bandwidth division is not very clear. Indeed C uses between 22 to 31 MBps of the bandwidth whereas we have only reserved 21 MBps (30% of 70 MBps).

Post-processing QoS: With post-processing QoS, we can observe on figure 9 that bandwidth division is much clearer. C uses 21 Mbps of the bandwidth and D uses 49 MBps. That corresponds exactly to the 30% of our reservation.

In this case, we show that our post-processing implementation is more efficient than pre-processing one. Our post-processing version permits to apply a real network QoS.

4.4 Cost in Terms of Delay

In this test, a computer sends a flow crossing two active nodes towards a receiver. Clocks of these four computers are synchronized using NTP (Network Time Protocol).

We measure time between packets sending and receiving at application level. We compare the Tamanoir original version (without QoS support) with our versions including our QoS management.

We measure the average time for 1000 packets (as well as standard deviation) and for packet size varying from 50 bytes to 64000 bytes (see table 10).

Tamanoir without QoS		
Size (Byte)	**Average (ms)**	**Standard Deviation (ms)**
50	1.3	7.6
500	1.8	7.4
1000	2	7.5
1400	2	7.4
1500	2.2	7.4
10000	4	8
64000	19.7	8.3

Tamanoir with QoS		
Size (Byte)	**Average (ms)**	**Standard Deviation (ms)**
50	3.6	8.8
500	3.9	9.2
1000	4.1	9.4
1400	4.4	9.2
1500	4.6	9.8
10000	8.3	14.9
64000	23.3	17.1

Figure 10. Delay in Tamanoir.

Standard deviation is significant when compared to the average. Our implementation does increase the standard deviation, but in a reasonable way. By looking more closely at measured delays, in both cases (with or without our QoS), we notice that most packets have a latency far below the calculated average.For example, in the case of 1500 bytes sized packets, more than 95% of packet require 3ms, however, sometimes, some packets take longer. It is especially the case of the first packets (this is due to the service installation process) where delays are sometimes longer than 50ms.

In some rare cases, packets can require more than 15ms. This explains the significant value of the standard deviation. Because phenomenon affects Tamanoir with or without our implementation of QoS, we suppose that these specific delay increases (i.e. except those of the first packets) are related to Java and/or to the operating system (garbage collector, cache-miss...). Thus, it should be understood that for more than 90% of the packages, effective routing time is actually quite lower than that shown in this table.

However, 10kB packet sized delay becomes really significant and must be reserved for applications having low delay constraint (although this is already true without our QoS implementation). Therefore, we can conclude that our implementation added very little delay (less than 3ms).

4.5 Discussion on experimentations

We have implemented few standard QoS mechanisms to compare them with classical QoS mechanisms; it is also possible to implement other ones.

We have shown that QoS mechanisms positioning in active router architecture is really important because the QoS we obtain can be totally different. Indeed, we have shown with examples that our pre-processing implementation can save computing resources (allocating more computing resources to more priority flows) whereas our post-processing one can give results closer to QoS in classical network. According to active services used, one approach will be more efficient than the other.

The best solution would be to use both solutions at the same time; but this is not usable because the implementation would use too many threads and this overhead would lead to performance reduction.

We have also shown that our solution is efficient and keeps delay low when using packet sizes below 10kB.

5 Conclusion

We have presented existing works in the field of QoS in active networks by classifying them into three categories: active application approach, re-use of QoS mechanisms provided by subjacent network and QoS mechanisms implemented at "active network" layer.

We have developed QoS mechanisms based on the latter in ANTS and Tamanoir active nodes. In our implementations, we pay a particular attention to the positioning of these mechanisms (in particular vertical and horizontal location of QoS mechanisms in active nodes).

We have validated our approaches with a set of experimentations. These experimentations have shown the interest of each one. Both are powerful in two different ways. When applying QoS before executing active services, we provide QoS saving processing resources. We also have shown that we can provide QoS closer to classical network QoS by implementing queues/scheduler mechanisms after execution environment. Our approaches for providing QoS in active nodes is innovative and powerful.

We have shown that the best solution would be to use both solutions at the same time but it is not possible without reducing network performances. Pre-processing is the solution to choose unless active services alter packet size (active services can enrich flow, compress them...).

References

[1] N. Achir, N. Agoulmine, M. Fonseca, Y. Ghamri-Doudane, and G. Pujolle. Active technology as an efficient approach to control DiffServ networks: The DACA architecture. *Lecture Notes in Computer Science*, 2496, 2002.

[2] N. Achir, M. D. de amorim, O. C. M. B. Duarte, and G. Pujolle. Active bandwidth sharing for multi-layered video in multi-source environments. In *Video Data*, July 2002.

[3] Y. Bai and M. Ito. Qos control techniques for video and audio transport over non-active and active networks. In *ICN'2004, 3rd IEEE International Conference on Networking, Pointe-a-Pitre, Guadeloupe, French Caribbean*, March 2004.

[4] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. On active networking and congestion. Technical Report GIT-CC-96-02, Georgia Institute of Technology. College of Computing, 2002.

[5] J. V. David J. Wetherall and D. L. Tennenhouse. Ants : A toolkit for building and dynamically deploying network protocols. *IEEE OPENARCH'98, San Fransisco, CA*, Apr. 1998.

[6] R. K. Eric Horlait, Nicolas Rouhana. An implementation of differentiated services using active networks. *IEEE Workshop on Networking, Beyrouth, Liban*, 1999.

[7] G. D. Fatta, S. Gaglio, G. L. Re, and M. Ortolani. Adaptative routing in active networks. *IEEE Third Conference on Open Architecture and Network Programming, OpenArch 2000, Tel Aviv*, March 2000.

[8] D. Fuin, E. Garcia, and H. Guyennet. A Novel Approach to Quality of Service in Active Networks. In *in proceedings of the 3rd IEEE Workshop on High-Speed Local Networks HSLN'04, Tampa, Florida*, November 2004.

[9] J.-P. Gelas, S. E. Hadri, and L. Lefevre. Towards the design of an high performance active node. *Parallel Processing Letters journal*, 13(2):149–167, June 2003.

[10] D. He, G. Muller, and J. L. Lawall. Distributing MPEG movies over the Internet using programmable networks. In *The 22nd International Conference on Distributed Computing Systems*, pages 161–170, Vienna, Austria, July 2002. IEEE.

[11] I. W. Marshall and C. Roadknight. Provision of quality of service for active services. *Computer Networks (Amsterdam, Netherlands: 1999)*, 36(1):75–85, June 2001.

[12] P. Vicat-Blanc Primet. Dynamic control and flexible management of quality of service in grids: the qosinus approach. In *in proceeding of the 1st IEEE Broadnet Conference, GridNets workshop, San Jos*, October 2004.