Integrated QoS Provision and Congestion Management for Interconnection Networks

A. Martínez[1], P. J. García[1], F. J. Alfaro[1], J. L. Sánchez[1], J. Flich[2], F. J. Quiles[1], and J. Duato[2]*

[1] Departamento de Sistemas Informáticos, Escuela Politécnica Superior
Universidad de Castilla-La Mancha, 02071 - Albacete, Spain
{alejandro,pgarcia,falfaro,jsanchez,paco}@dsi.uclm.es
[2] Dept. de Informática de Sistemas y Computadores, Facultad de Informática
Universidad Politécnica de Valencia, 46071 - Valencia, Spain
{jflich,jduato}@disca.upv.es

Abstract. Both QoS support and congestion management techniques have become essential for achieving good performance in current high-speed interconnection networks. However, traditional techniques proposed for both issues require too many resources for being implemented. In this paper we propose a new switch architecture that efficiently uses the same resources to offer both congestion management and QoS provision. It is as effective as previous proposals, but much more cost-effective.

1 Introduction

High-speed interconnection networks have become a major issue on the design of computing and communication systems, including systems for parallel computing, since they provide the low-latency and high-throughput demanded by parallel applications. The proliferation of systems based on high-speed networks has increased the researchers' interest on developing techniques for improving the performance of such networks. Moreover, due to the increase in network components' cost and power consumption, it is nowadays very important to propose efficient and cost-effective techniques, trying to use a minimum number of network resources while keeping network performance as high as possible.

For instance, many techniques have been proposed for solving the problem of network performance degradation during congested situations. Congestion is related to the appearance of Head-Of-Line (HOL) blocking, which happens when a packet at the head of a queue blocks[3], preventing other packets in the same queue from advancing, even if they request available resources. This may cause

* This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2006-15516-C04"; by Junta de Comunidades de Castilla-La Mancha under grant PBC-05-005; and by the Spanish State Secretariat of Education and Universities under FPU grant.

[3] We are considering lossless networks like InfiniBand, Quadrics, or Myrinet.

that data flows not contributing to congestion advance at the same speed as congested flows, thereby degrading network performance.

However, although congestion (and HOL bocking) in interconnection networks is a well-known phenomenon, efficient congestion management mechanisms in modern high-speed networks are very rare. On the one hand, traditional, simple solutions are not suitable for modern interconnects. For instance, network overdimensioning is not currently feasible due to cost and power consumption constraints. On the other hand, more elaborated techniques that have been specifically proposed for solving the problems related to congestion have not been really efficient until very recently.

Another way for improving network performance, from an application point of view, is to use techniques for providing Quality of Service (QoS). If communication services must be provided for several types of applications, the QoS consists in guaranteeing a minimum performance to each application, regardless the behavior of the rest of traffic classes. For instance, if we want to guarantee a minimum bandwidth to each traffic class, this minimum must be provided even if there are sources injecting more traffic than they are allowed to.

The usual solution for this problem is to provide a separate virtual channel (VC) for each traffic class. These VCs also provide separate domains of flow control, i.e. there is a separate credit counter for each VC. In this way, two objectives are achieved: firstly, there is no HOL blocking between traffic classes. Secondly, a single traffic class cannot take all the buffer space available, so buffer hogging is avoided. In this way, the performance of different traffic classes only depends on the scheduling and on the amount of injected traffic.

In this paper, we present an efficient and integrated solution for both congestion management and QoS provision problems. We propose a new switch architecture without VCs in which the buffers are managed with a novel congestion management technique: RECN (Regional Explicit Congestion Notification) [1]. Moreover, we add QoS-awareness to this scheme, so as to consider, not only congestion management, but also QoS requirements. The main benefit of this architecture is that it uses the same set of resources for both purposes: congestion management and QoS provision, thus, requiring a lower number of resources than previous proposals.

The rest of the paper is organized as follows. In the following two sections we briefly review proposals for congestion management and QoS. Next, in Section 4, the proposed switch architecture is explained in detail. Section 5 shows an evaluation of the new proposal, based on simulation results. Finally, in Section 6, some conclusions are drawn.

2 Congestion Management

Traditionally, Virtual Output Queues (VOQs) [2] was considered the most effective way to face congestion. In this case, there are at each port as many queues as end-nodes in the network, and any incoming packet is stored in the queue assigned to its destination, thereby avoiding HOL blocking between packets addressed to different destinations. However, although this scheme is very effective, it is not efficient (and even not feasible for medium or large networks) since it requires a considerable number of queues at each switch port, and the silicon area required for implementing such number of buffers strongly increases the cost. A variation of VOQ uses as many queues at each port as output ports in a switch [3], reducing so queue requirements. However, this scheme does not completely eliminate HOL blocking since only switch-level HOL blocking is eliminated.

In contrast to these techniques, RECN [1] completely eliminates HOL blocking while requiring a small number of resources independently of network size, being so a really efficient, scalable and cost-effective technique. Specifically, in order to avoid HOL blocking between congested and non-congested flows, RECN identifies congested flows and puts them in special, dynamically-assigned set aside queues (SAQs).

RECN assumes that packets from non-congested flows can be mixed in the same buffer without producing significant HOL blocking. While standard queues will store non-congested packets, SAQs are dynamically allocated for storing packets passing through a specific congested point. SAQs can be dynamically deallocated when not necessary. Every set of SAQs is controlled by means of a CAM (Content Addressable Memory), in such a way that each CAM line contains information for managing an associated SAQ, including the information required for addressing a congested point.

In this sense, RECN addresses network points by means of the routing information included in packet headers, assuming that source routing is used. For instance, Advanced Switching [4] (AS) packet headers include a turnpool made up of 31 bits, which contains all the turns (offset from the input port to the output port) for every switch in a route. Therefore, CAM lines include turnpools which can be compared with the turnpool of any packet, in order to know if it will pass through the congested point associated to that SAQ. In this way, congested packets can be easily detected. A more detailed description of RECN can be found in [1].

3 QoS Support in Interconnection Networks

In modern interconnection technologies, like InfiniBand or PCI AS, the obvious strategy to provide QoS support consists in providing each traffic class (service level, SL) with a separate VC. This increases switch complexity and required silicon area and, therefore, very few final implementations provide all the VCs proposed in specifications[4].

In order to alleviate this problem some techniques have been proposed that reduce the number of VCs while keeping QoS guarantees. For instance, in [5], a technique for providing full QoS support with only two VCs was proposed. The key idea is that traffic has already been scheduled by the network interfaces.

[4] Note that proposals requiring many VCs could be considered if external DRAM is available for implementing the buffers. However, in this case, the low latencies demanded by QoS-requiring traffic could not be provided.

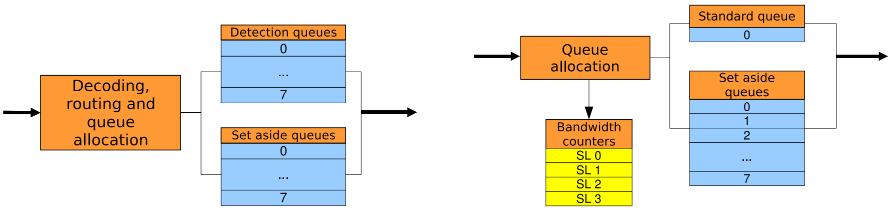
Therefore, there is information regarding the priority of these packets implicit in the order and proportions of packets leaving the end-nodes. For instance, if the end-nodes implement a Weighted Round-Robin [6] policy, packets are injected in the proportions that are configured at the scheduler.

In [7] this technique was combined with RECN by duplicating all the RECN queue structures in two VCs, in order to obtain a switch architecture offering QoS provision and congestion management at the same time.

However, we think that it is possible to achieve full QoS provision in an even more cost-effective, efficient way. Specifically, we propose in this paper a switch architecture that improves the basic RECN mechanism so it can also provide QoS guarantees without introducing additional VCs or queues. The newly proposed architecture takes into account the clear parallelism between the tasks performed by RECN and the use of VCs for QoS provision. Our proposal exploits the RECN queue structure from a new, original approach that efficiently uses these resources for managing congestion while providing QoS at the same time. The benefits of the proposal are obvious since these two important issues on interconnect design would be afforded by a single and very efficient architecture.

4 New Proposal for QoS Provision and Congestion Management

The switch organization that we propose consists of a combination of input and output buffering, which is a usual design for this kind of switches. Note that all the switch components are intended to be implemented in a single chip. This is necessary in order to offer the low cut-through latencies demanded by current applications.



(a) Input port organization

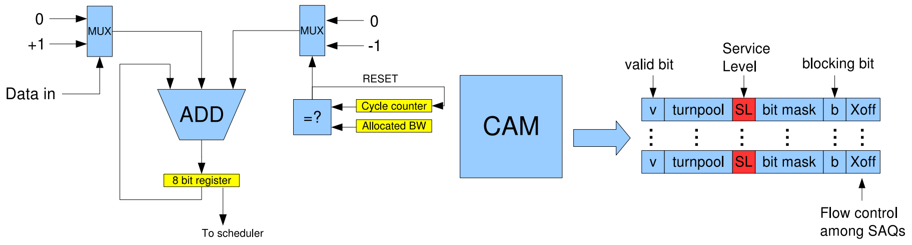
(b) Output port organization

Fig. 1. Input and output ports logical organization.

The queue organization of an input port can be seen at Figure 1 (a). This is the standard scheme for a RECN input port, so there are as many detection queues as output ports (8 in this case[5]) for storing non-congested packets, and

[5] We assume for the sake of simplicity 8 port switches. Anyway, architectures with a different number of ports could be easily deducted.

a group of SAQs (the typical number of SAQs per group is 8 or less) for storing congested packets. The use of these queues will be discussed later. A CAM is required at each input or output port in order to manage the set of SAQs. The organization of the CAM can be seen in Figure 2 (b). Each CAM line contains all the fields defined by RECN, plus a new field for storing the SL the corresponding SAQ is assigned to.



(a) Bandwidth counter

(b) CAM organization

Fig. 2. CAMs and bandwidth counters.

Figure 1 (b) shows the organization of output ports. In this case detection queues are not necessary, and a unique standard queue is used for storing non-congested packets. Following also the RECN scheme for output ports, a set of SAQs (8) is also used at the output port. In addition to this RECN queue structure, our proposal introduces at the output ports a set of bandwidth counters, one per SL. These counters must dynamically compute the difference between the reserved bandwidth and the current bandwidth consumption, and they will be used for two purposes: firstly, to perform deficit round-robin scheduling at the switches and secondly, for congestion detection (both issues will be detailed in the following subsections).

The bandwidth counters' structure can be seen at Figure 2 (a). Basically, their behavior is the following: each time a block of 64 bytes from a packet[6] is scheduled to cross towards the output port, the bandwidth counter corresponding to the packet SL is increased by 1. We have assumed a 8-bit register for implementing the bandwidth counter, so its range of values is from -8 kbytes to 8 kbytes[7]. On the other hand, the same register must be automatically decremented at a rate that matches the bandwidth we want to guarantee to the corresponding SL. For instance, if 128 Mbytes/s have been guaranteed to the SL and the internal clock is 100 MHz, the bandwidth counter must be decreased by one every 50 cycles. This decrease is implemented by configuring the "Allocated BW" register with the appropriate number of cycles. When the cycle counter matches the configured cycles, the bandwidth counter register is decreased by

[6] The unit used in our counter is 64 bytes because in PCI AS each credit is 64 bytes and, thus, packets come in 64-byte increments.

[7] These values are adequate to monitor instantaneous traffic behavior.

one and the cycle counter is reset. All these operations effectively allow to measure the difference between reserved and consumed bandwidth, while they are simple and do not introduce significant delay or require much silicon area.

4.1 Switch Scheduler

The switch scheduler implements a deficit round-robin (DRR) algorithm based on the bandwidth counters' information. This algorithm consists in giving priority to flows that are consuming less bandwidth than the amount reserved for them. Following this scheduling, for each packet ready to cross the crossbar, the scheduler checks the value of the bandwidth counter corresponding to the output port and the SL of the packet. Afterwards, packets are served in the order of these registers, first packets with the smallest values and later packets with higher values.

The scheduler needs another feature compared to a typical one. Specifically, our scheduler requires to know whether a packet is going to be stored in an output port SAQ or in the standard queue. This is calculated by comparing the turnpool of the packet with the routing and SL information at the CAM. In the case of matching, packets can only be selected if the SAQ is not filled over a certain threshold. By applying this rule, buffer hogging is prevented.

Moreover, the scheduler does not specially treat packets coming from SAQs. However, if such packets are contributing to congestion, the corresponding bandwidth counter will have a high value and they will be penalized by the DRR algorithm. On the other hand, if the packets are no longer contributing to congestion, the corresponding bandwidth counter will have a small value and they will achieve a high priority. This effect contributes to empty unnecessary SAQs as soon as possible, thereby allowing the deallocation of these SAQs (SAQs must be empty for being dellocated).

4.2 Congestion Management

RECN detects congestion both at input or output ports of switches, always by measuring the number of packets stored in the queues. Our new proposal also detects congestion at input and output ports, but in this case these detections do not depend only on queue occupancy. Specifically, in the new proposal, input detections happen when a detection queue at an input port fills (in terms of stored information) over a certain threshold and the value of the bandwidth counter associated to the last received packet SL at the requested output port is over another threshold. The actions to take after a congestion detection at the input port are: first, a SAQ associated to the appropriate SL and with turnpool equal to the output port is allocated at the input port, and, in addition to this, another SAQ, with empty turnpool but associated to the SL is allocated at the corresponding output port. These SAQs with empty turnpool (not considered in original RECN) will store any packet belonging to the associated SL. This is necessary in order to avoid that packets from a single SL completely fill an output buffer.

On the other hand, the conditions for a detection of congestion at an output port are similar. If the arrival of a packet from an input port causes occupancy of the standard queue at this output port to be over a certain threshold and the bandwidth counter of the packet SL is over another threshold, then the detection of congestion takes place. The actions after an output detection are the same as in an input detection: allocation of two SAQs, one at the input port and another with empty turnpool (but associated to the SL) at the output port.

If congestion persists and SAQs start to fill over a certain threshold (known as the propagation level), then information about the corresponding turnpool and SL is propagated backwards the congestion flow, in order to allocate new SAQs for storing packets belonging to this flow wherever these packets are. In the case of propagation from a SAQ at an output port, SAQs are allocated at the input ports that cause the overflow of the output port SAQ. Of course, these input SAQs will have an associated turnpool with one more hop than the output SAQ. SAQs with empty turnpools (only allocated at output ports) may also produce the allocation of SAQs at the input ports, that in this case will have a one-hop turnpool.

If SAQs at input ports fill over the propagation level, a control packet is sent to the preceding switch. This packet includes the turnpool and service level associated to the filled input SAQ, in order to also have an allocated SAQ associated to this information at the receiving output port.

In this way, there will be SAQs at any point where they are necessary in order to store congested packets, thereby eliminating HOL blocking. SAQs can be deallocated when they are not necessary for eliminating HOL blocking at the point where they are allocated. The conditions for SAQ deallocation are exactly the same as in RECN [1].

On the other hand, all the thresholds and propagation levels are constant in the system, and we assume they are properly set up by the network administrator during the configuration phase of the different devices. In our experiments, we have used a set of values obtained after exhaustive tuning which are optimal for a variety of networks designs.

4.3 QoS Provision

In order to provide QoS guarantees, each traffic class is assigned a percentage of link bandwidth. For instance, if there are four traffic classes, each one could be assigned 25%. The total assigned bandwidth must not exceed the bandwidth of any link. At the end-nodes, we assume a traditional DRR implementation with a VC per traffic class, which is something feasible in these devices.

Provided that end-nodes implement this QoS policy, and as long as there is no contention, we have observed that packets pass through the switches in the same proportions as they are injected into the network. The reason is that the switches do not introduce any significant delay when links are not oversubscribed.

However, since there is no admission control, it may happen that any link of the network becomes oversubscribed. In this situation, congestion appears because at this point, one or more traffic classes introduce more traffic than their assignation.

A traditional congestion management technique would penalize all traffic regardless of their traffic class. From this point of view, all packets are equally contributing to congestion. However, with the bandwidth counters we have proposed and the switch scheduler we have presented before, only traffic classes injecting more than their allowance are penalized. Note that any traffic class can inject additional traffic if there is unused bandwidth. Therefore, problems only arise as a consequence of oversubscribed links.

Therefore, QoS guarantees are achieved in the sense that if traffic from a class is injected up to its allowed bandwidth, it will achieve maximum throughput and experience short delay. The scheme proposed for QoS provision uses the same resources provided for congestion management. Therefore, we can offer a satisfactory solution for both problems, as we will confirm in the next section.

5 Simulation Results

In this section, we show the advantages of our architecture using two different tests. In the first one we have static congestion and we show that our proposal, although not using VCs, is able to isolate and guarantee bandwidth of several traffic classes. The second test shows that our proposal is also efficient when traffic conditions are dynamic and hot-spots appear and disappear quickly.

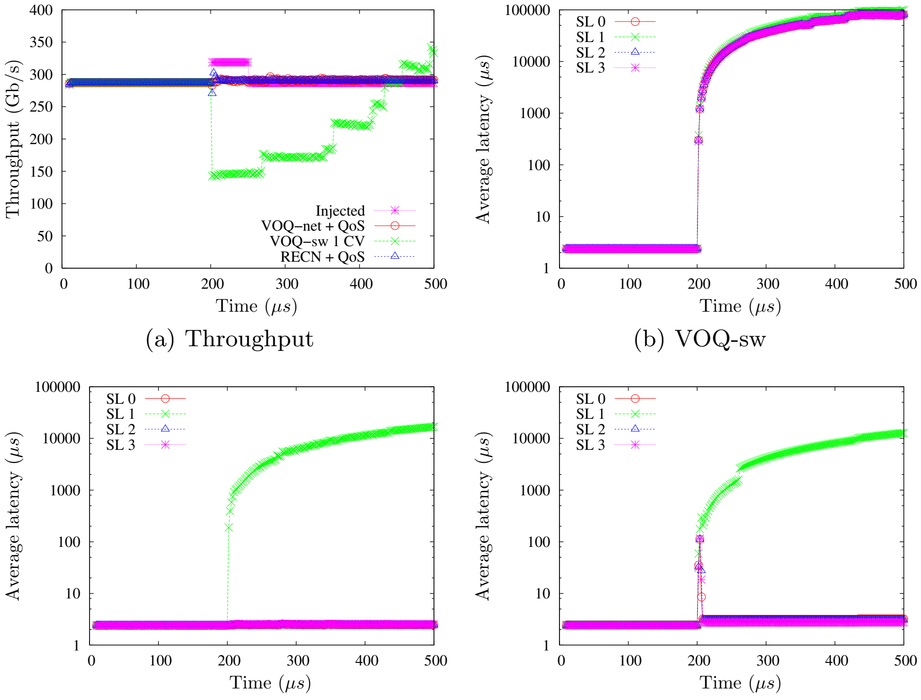
5.1 Hot-spot Scenario

For this scenario we have considered a multi-stage interconnection network (MIN) with 64 end-nodes. In this network, there is uniform traffic belonging to four service levels. However, during a small period of time, there is a sudden burst of traffic towards a hot-spot coming from a single SL. Without loss of generality, we will assume that this hot-spot is the node 5 and the SL 1. In this way, in addition to the uniform traffic, 33% of the interfaces start injecting traffic of SL 1 towards end-node 5.

In Figure 3 we can see throughput and latency results for the four aforementioned SLs, and for three different architectures: classic VOQ at switch level without VCs; classic VOQ at network level and also a VC per traffic class; and our *RECN+QoS* proposal.

Performance is very poor when using the *VOQ-switch* case since all traffic classes that do not generate congestion are penalized. The *VOQ-net* architecture offers very good performance since all the traffic classes and destinations are completely isolated, but is a very expensive solution[8]. When using our proposal, we can see that the traffic classes that are not producing congestion are only marginally affected by the hot-spot, even though they are not separated from the congested traffic class by VCs.

[8] In the evaluated architecture, there are $4 \times 64 = 256$ queues per port, one for each destination/SL combination.



(a) Throughput

(b) VOQ-sw

(c) VOQ-net

(d) RECN+QoS

Fig. 3. Results for hot-spot scenario.

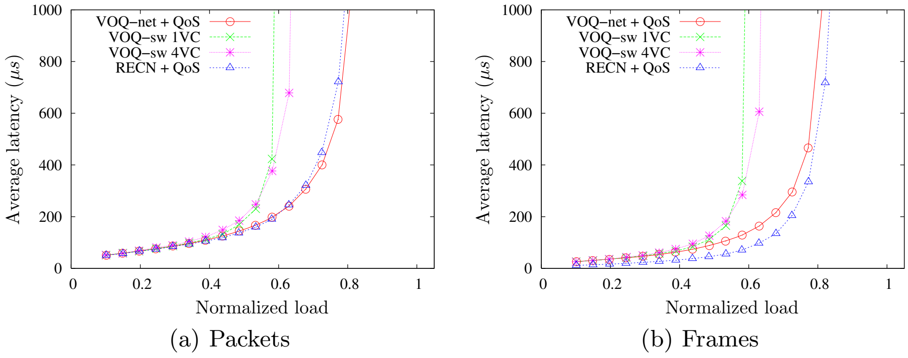
From this test, we can conclude that our proposal is able to guarantee bandwidth to several traffic classes, even if there is another traffic class generating a hot-spot.

5.2 Multimedia Traffic Scenario

In this scenario, we have also considered 64 end-nodes connected through a MIN, but now we assume that sources transmit MPEG-4 video sequences. There are four classes of sequences, each with a guaranteed 25% of the throughput. The video sequences consist of frames, each with a size ranging from a few Kbytes up to 150 Kbytes. These frames are produced each 40 milliseconds for every sequence. Note that many of these trasmissions are held in parallel.

The bursty nature of video transmission produces a lot of congestion and an efficient network architecture is necessary to obtain the maximum throughput. Note that many hot-spots appear and vanish over time. Moreover, congestion may originate at the middle of the network, instead of at an end-node.

For this test, we have considered an additional switch architecture, based on having one VC per traffic class, and each VC is further divided in VOQs at the switch level. This will be noted in the figures as *VOQ-sw 4VC*.



(a) Packets

(b) Frames

Fig. 4. Performance in video transmission scenario.

We can see at Figure 4 two types of latency results. At the left, we have typical per packet latency. At the right, we have latency of whole video frames, made up of several packets. Note that this is the value that counts from the application point of view.

Regarding individual packets latency, the best results are for the *VOQ-net* architecture, with the *RECN+QoS* architecture very close. The other cases are not able to handle properly the bursts of packets.

If we look at frame's latency, the best results are for our *RECN+QoS* architecture, even better than the *VOQ-net* case. The reason is that our *RECN+QoS* proposal is able to cope better with congestion at any point in the network, while VOQs are designed to handle congested end-nodes. As a consequence, large bursts of packets (like big video frames) progress faster, and hence the better frame-level latency results.

6 Conclusions

Current high-speed interconnection networks demand adequate QoS support and congestion management techniques for achieving good network performance. In this paper we propose a new switch architecture able to face the challenges of congestion management and, at the same time, QoS provision, while being more cost-effective than other proposals, since it uses the same resources for both purposes.

Results have shown that, by means of affordable mechanisms and techniques, we can manage the buffer space and the queues in a very efficient way for addressing our goals. Without VCs and with just some additional queues per port, we can guarantee QoS while eliminating congestion.

References

1. García, P.J., Flich, J., Duato, J., Johnson, I., Quiles, F.J., Naven, F.: Efficient, scalable congestion management for interconnection networks. IEEE Micro **26, 2006** (2006) 52–66
2. Dally, W., Carvey, P., Dennison, L.: Architecture of the Avici terabit switch/router. In: Proceedings of the 6th Symposium on Hot Interconnects. (1998) 41–50
3. Anderson, T., Owicki, S., Saxe, J., Thacker, C.: High-speed switch scheduling for local-area networks. ACM Transactions on Computer Systems **11** (1993) 319–352
4. ASI SIG: Advanced switching core architecture specification. (2005)
5. Martínez, A., Alfaro, F.J., Sánchez, J.L., Duato, J.: Providing full QoS support in clusters using only two VCs at the switches. In: Proceedings of the 12th International Conference on High Performance Computing (HiPC). (2005) 158–169 Available at `http://www.i3a.uclm.es`.
6. Katevenis, M., Sidiropoulos, S., Courcoubetis, C.: Weighted round-robin cell multiplexing in a general-purpose ATM switch. IEEE J. Select. Areas Commun. (1991) 1265–1279
7. Martínez, A., García, P.J., Alfaro, F.J., Flich, J., Sánchez, J.L., Quiles, F.J., Duato, J.: A cost-effective interconnection architecture with QoS and congestion management support. (2006) Available at `http://www.i3a.uclm.es`.