# Load Control: Congestion Notifications for Real-time Traffic

Z. R. Turányi [1] L. Westberg [2]

[1]Traffic Analysis and Network Performance Lab, Ericsson Research

Budapest, Hungary

[2] Ericsson Radio Systems, Ericsson Research

Stockholm, Sweden

**Abstract**

In this paper we present a new resource reservation method for Differerntiated Services domains, which uses simple bit-markings in packet headers. Our Load Control scheme operates by providing congestion notification signals to edge devices in a manner that is suitable for real-time traffic. We investigate the robustness and other properties of the method and argue that sufficent QoS can be provided with minimal complexity and no per flow processing. Simplicity is demonstrated via example router algorithms.

**Keywords:** quality of service, explicit congestion notification, resource reservation

## 1 Introduction

A recent networking trend is the development of one single, packet based transport network carrying multiple types of traffic. Although this unified transport network is founded primarily on datacom principles, it has to provide traditional telecommunication services as well. As the convergence proceeds, more and more packet networks will carry real-time, call based traffic. This traffic will consist of not only telephony calls, but other voice, audio or video based traffic such as streaming media or video conference applications. For these services to be useful, the IP network must be prepared to provide the appropriate amount of resources and quality of service (QoS) to the real-time flows.

Several alternatives exist already to provide QoS in the Internet (e.g. RSVP/Intserv, Diffserv/statistical provisioning), yet none has been proven to satisfy the requirements of all situations. The best way to provide bandwidth, delay or loss guarantees may heavily depend on the actual network and traffic conditions. A solution that is feasible in a campus environment may completely fail in a company intranet, a wireless access network or in a transcontinental backbone. The selection of the proper QoS architecture may depend on factors including user

profiles, the available equipment and management budget, the extent to which QoS should be supported, policies and the way the network is managed and built. After nearly a decade of research there is no clear consenus on how to provide QoS, which may indicate that one single solution may never solve all related problems.

In this paper we focus on a specific networking environment: an IP-based cellular radio access network. Such networks are expected to be deployed within the near future. They have the following distinctive properties:

1. a large fraction of traffic may be real-time (especially voice);
2. links in the network are often rented or implemented using microwave connections and thus the price of transmission is high;
3. different parts of the network are loaded at different time of the day (e.g., business and residental areas);
4. the time to reserve resources is very limited (it may need to be performed on each handoff); and
5. the number of nodes in the network may be very high (e.g. in the order of several thousands)

Many basic approaches to resource reservation fail in this environment. Overprovisioning is typically too costly due to property 2 above. Static configuration of available resources is not flexible enough due to property 3. Per flow signalling solutions are viable, but may result in very high equipment costs due to properties 4 and 5.

Within the Internet community there is a long history of congestion control research [12]. Most of these effors addressed bulk data transfer applications and the improvements were built into the Transmission Control Protocol (TCP). Recent results in the field introduce Explicit Congestion Notification signals that are sent by routers toward the end systems [13]. End systems are required to slow down when receiving such signals. While this approach – especially when coupled with the random method suggested for generating these signals [14] – is applicable to data transfer applications, it is less suitable for non-adaptive traffic. The utility of data transfer degrades gracefully with decreasing available bandwidth. In contrast, non-adaptive traffic has little utility even when the available resources are only a little less then required. Therefore the preferred way of avoiding congestion (and thus QoS degradation) for non-adaptive traffic is not slowing down, but blocking new incoming requests.

In this paper we describe a new scheme called Load Control which operates similarly to ECN, but allows the blocking of new flows. Load Control is aimed for real-time traffic classes that are not in the scope of current ECN work, which focuses on TCP based elastic traffic. Thoroughout the paper, we assume that all routers use Differentiated Services [18] codepoints to differentiate between various flow types.

In Section 2 we present some of the previously proposed resource provisioning methods. Section 3 contains the description of Load Control. In Section 4 we investigate some properties of Load Control partly using simulation. In this section we argue that in many cases a very simple method, such as Load Control is enough to provision resources with sufficient precision. In the Appendix we describe example Load Control router algorithms to demonstrate simplicity of implementation. Details related to the introduction of Load Control into the TCP/IP protocol family can be found in [11].

## 2 Related Work

Several resource provisioning or traffic control algorithms have been already proposed to augment Differentiated Services in providing QoS guarantees or services. One approach is to use a *signalling protocol* to communicate the resource needs from end systems to routers [2, 5, 6]. This approach is very close to the Integrated Services [19] approach, with the exception that the installed state is used only for admission control and not for scheduling packets. The advantage of this approach over Integrated Services is that packets are not classified into flows within core routers and packet treatment depends purely on the value of the DS field. The DS field of packets is set by the host or an edge device where packet classification does take place.

*Scalable Reservation Protocol* [7] is also able to augment Differentiated Services with the necessary provisioning methods. According to SRP when sources wish to open a new flow they start transmitting specially marked request packets at the rate of the flow. Routers can learn the requested rate by measuring the rate of request packets and can drop some portion (including none or all) of the them, depending on how much free resources they have. By letting through request packets at a certain rate, routers implicitly admit the flow with that rate. Receivers also measure the incoming rate of request packets and report the result back to the source, which can decide if the admitted rate is enough. If so then the source continues to transmit at that rate and marks its packets as reserved. Those packets keep the reservation in routers in a soft state manner. While this approach works very well for constant bit-rate sources, it is not possible to keep reservations without transmitting data, which may force variable bit-rate sources to transmit more than necessary and eventually lead either to loss of reservation or to

inefficient network usage.

Solutions based on *Dynamic Packet States* [4] use free space in the IP header to store some state, most often a number. By using these states fair queueing or admission control can be implemented without flow states in the core routers. This approach is very flexible and useful, but it requires the encoding of a number into IP packets with sufficient accuracy. This may require either the introduction of a new IP option or modifications to the IP header itself, which may make actual deployment very hard.

Another set of proposals approaches the provisioning problem from an economic view and starts from the observation that if the network is congested, then the value of transmitting a bit is higher, so the cost of it also should be higher. These methods [3, 16] try to achieve control by charging for transmission depending on the congestion level. Users or protocol stacks are supposed to back off as the cost per transmission goes up. Willingness to pay determines the share the user receives from network resources. It is necessary to note that the term "cost" in this context does not necessarily mean actual money, but is rather used as a control function. Schemes based on such feedback work especially well with elastic or highly adaptive applications, where the utility of the service is close to proportional to the received bandwidth. However, they less suited to the cases where the utility is seriously degraded if the recived bandwidth is below a certain value (e.g., as in case of non adaptive media codecs). Gracefully decreasing data rate when congestion increases is not beneficial for such sources. Instead, they wish either to transmit at their specified rate or to stop entirely as for them it has no utility to to transmit below that specified rate.

*Statistical provisioning* of Differentiated Services resources is a simple and attractive solution. Network managers and traffic sources (customers) agree on long term and high level traffic descriptors and commitments (Service Level Agreements). By simple calculus network managers decide if the SLA in question can be admitted or not. If it can, then the SLA is installed in edge devices, which make admission control decisions and/or perform policing according to the SLA and ingoring the actual load of the network on shorter timescales. Also the SLAs are many cases envisioned not to contain any specific destination information, which makes exact provisioning very hard, as the traffic of each SLA may cause load anywhere in the network.

*Bandwidth Brokers* [1] are envisioned as central devices that overview the resource utilisation within a statistically provisioned domain. They are aware of the routing situation and grant or deny access to the network based on measurements and/or information about previously admitted requests. Bandwidth brokers have all the advantages of central control, but suffer from all the disadvantages as well. If requests come too frequently, the bandwidth broker may

prove to be a signalling bottleneck and it may also be a single point of failure. Measurement and routing reports may be sent to the broker only with limited frequency, limiting the efficiency and timeliness of the decisions.

In case of *total overprovisioning* users are not restricted at all to send any traffic into the network. In this case the QoS problem is solved by spending the resources of the operator not on implementing and managing a control architecture, but on increasing the raw bandwidth of the links. Customers pay based on usage, and if the prices, demand and network dimensioning are in harmony, then QoS may be very high. However, such a network may provide very little useful service in cases of extreme demand and is very easily hurt by overload.

# 3 Description of Load Control

Most of the provisioning proposals [2, 4, 5, 6, 7, 16] reuse the already existing routing infrastructure of the Internet, which provides one path to any destination. This way path selection is out of control from the QoS point of view. QoS routing proposals take the opposite approach [8] and lead to more complex solutions. They are not discussed here. When the path is determined independently by the routing protocol, resource provisioning has only two remaining tasks:

- limit the amount of traffic on each pre-determined path; and
- react appropriately to exceptional events (e.g., route changes).

The former means some form of admission control. This can be exercised not only for rigid, real-time traffic but for adaptive flows as well [15].

In the latter task the term "exceptional event" refers to any event when link utilisation unexpectedly increases to critically high values. These cases include link speed drops (e.g., as a result of link layer re-configuration or error), malfunctioning policers (which may admit too much traffic) and most notably routing changes when the load of several links are re-routed over the same path, which has insufficient capacity to carry all. The correct action in such cases depends on the policy and service definitions. The network may keep all flows and suffer higher loss and delay, or it may drop some flows to protect others according to its policy.

We note, that with the recent introduction of Multiprotocol Label Switching [20], more than one path can be established between two nodes. These paths will be most likely used for traffic engineering purposes on longer timescales and can be considered out of control from flow QoS point of view. By using MPLS the network has more options in fulfilling both tasks. When admission control failed on one path, another one can be probed and also if one path becomes overloaded flows may be re-routed to other, possibly unloaded paths.

Load Control also reuses the existing routing infrastructure. Given the path the packets of the flow will take, it performs both tasks: it can check if the network has sufficient resources along a path to the destination and can give feedback to edge devices in extreme load cases. On the other hand, Load Control does not specify what to do when extreme load occurs. Edge devices are free to take whatever action is prescribed by network administrators.

The basic idea of Load Control is to use bit markings in packet headers (e.g., in the DS byte). Thus Load Control related information can be piggybacked on existing packet traffic. Also by placing simple control information into packet headers, most Load Control related operations can be implemented in the fast data path.

Load Control has two variants. The *measurement based variant* does not do any reservation: load measurements are used to make admission decisions in the routers instead. In contrast, when using the *refreshment based varinat* resources are explicitly reserved along the path. While the semantics of the two are different, the basic idea and the implementation are very similar in the two cases. In the following we briefly present these two variants and then give a short description of the extreme load case.

### 3.1 Measurement based Load Control

When a new flow or resource request arrives to the edge of the network, a *probe packet* is sent through the network. This packet is specifically marked as being a probe packet, but otherwise can be any packet including any higher level signalling message (e.g., a TCP SYN or a SIP INVITE packet). If the routers along the path can carry more traffic than they currently do, they let the probe packet pass unmodified. However, if a router decides that more traffic would disturb the QoS of ongoing flows, it changes the marking of the packet from probe to *marked probe*. When the probe packet reaches the egress edge device, that device signals the result of the probing (marked or not) back to the originating edge. The originating edge device then can take the appropriate action and eventually reject the flow. The backward signal itself sent by the egress edge device can be used as another probe packet to check the resources on the backward path if the flow to be established is bi-directional.

The probing mechanism can be used to probe resources on a per diffserv class basis as well. The DS field of the probe packet contains the diffserv class the flow will use, and the header field is available to core routers. In this case core routers mark or do not mark probe packets according to the resource situation in the DS class of the probe packet.

Load Control does not specify how a core router decides whether to accept more traffic or not. One natural approach is the use of measurements. Several measurement-based admission

control algorithms have already been proposed, some of which can be used even in this context where the router knows almost nothing about the flows [9, 10]. A simple algorithm that is used throughout the simulations of this paper is described shortly in Appendix A.

The originating edge device does not specify in the probe packet how much traffic will be injected into the network if the probe succeeds. This means that core routers have to decide admission or rejection without knowing the traffic. This problem is discussed in detail in Section 4, here we only note that it can be solved by using DS codepoints to identify flow types. In a network where only a small number of flow types exist (such as in a cellular access network where the majority of flows are voice calls), a separate codepoint could be allocated for each flow. This codepoint indicates not only the PHB the packets should receive, but the resource usage of the flow as well. Core routers then can take flow type into account when marking or not marking probe packets.

### 3.2 Load Control using refreshment packets

While measurement-based admission control has its benefits over non-measurement-based counterparts, it has several drawbacks as well. One notable weakness lies in the probabilistic nature of measurements. Another – sometimes undesirable – effect is that if a source temporarily decrease transmission rate the measurement process implicitly assumes that it will transmit less in the future as well. For this reason it might be beneficial if sources could reserve some amount of resources even when they are not transmitting that much. This is what the second variant of Load Control makes possible.

The host or edge device can periodically mark packets of each already admitted flow as *refreshment packets*. Refreshment packets indicate resource reservations to the core routers. One refreshment packet indicates the reservation of one *unit of resources* for the time of one *refreshment period*. Routers along the path detect and count the number of passing refreshment packets that arrive during refreshment periods and thus determine the amount of resources reserved. When a probe packet arrives routers base admission decisions on number of refreshment packets counted instead of traffic load measurements. A simple router algorithm is shown in Appendix B. We note that unmarked – and thus admitted – probe packets also perform the reservation of one unit of resources for one refreshment period and are therefore counted together with refreshment packets.

The amount of resources one refreshment packet reserves – the unit – may consist of any type of resources. It may be an equivalent bandwidth value, a peak rate value or something more network-specific such as "one call". It must be uniform throughout the entire domain and can

be specified by the network administrator. Different types of flows may require different amount of resources and thus may occupy different number of units. If a flow is larger than one unit then it sends more than one refreshment packets per refreshment period. These packets should be evenly spread in time across the refreshment period to reduce potential errors resulting from the different time alignment of refreshment periods in hosts and routers. If a flow requires a fractional number of units then it may send different number of refreshment packets in consecutive refreshment periods, averaging in the correct value on the long run. If the network has sufficiently large amount of such flows than the number of refreshment packets passing the core routers in any refreshment period will be close to the sum of the fractional flow sizes.

Similarly to the size of the unit, the length of the refreshment period can be selected by the network operator and must be uniform throughout the whole network. Discussion of the length of the refreshment period can be found in Section 4.3.

### 3.3 Overload situation

We mentioned earlier that besides checking the resources along a path, Load Control can also give feedback in case of extreme load. Upon receiving this feedback, edge devices can take appropriate steps, e.g., reduce the amount of traffic on the problematic path. This is important to make the network robust to exceptional events, such as link or equipment failures.

Feedback is given by using specially marked *overload packets*. When a link is overloaded or when the traffic of a Diffserv class seriously overuses the resources it is entitled to, the router starts marking data packets as overload packets. This is very similar to other forward explicit congestion notification schemes, such as the FECN bit in Frame Relay or ECN in IP [13]. In case of Load Control, however, the exact nature of the overload the signal indicates is defined locally within a domain by the network administrator and can range form light congestion to serious overload. Also the appropriate action to be taken is operator defined and is not part of the Load Control mechanism. The overload signals keep flowing as long as the overuse persists. Packets can be marked as overload packets with both versions of Load Control, i.e., with or without the use of refreshment packets. This overload mechanism is not investigated further in this paper. For more details refer to [11].

## 4 Analysis

Most of the mechanisms of Load Control can be implemented in hardware using elementary arithmetic operations. Per packet work in the core routers requires no per flow state or process-

ing. Load Control is insensitive to the number of edge devices as well, unlike approaches where resource tunnels are set up between edge devices.

However, this simplicity does not come free. Routers making admission decisions are in extreme shortage of information, which may imply higher probability of erroneous decisions. In this section we investigate three aspects of Load Control operation and argue that in large aggregation areas, where scalability is essential, the assurances provided by Load Control are enough.

Section 4.1 is applicable only to the measurement based variant and investigates the problem of probe packets arriving in batches. Section 4.2 is applicable to both variants and discusses resource reservation for flows of multiple classes. Finally, Section 4.3 gives more insight into the proper selection of the length of the refreshment period.

Throughout this section we assume that the routers have enough resources configured to provide the appropriate QoS if the arriving traffic is below the admission control target. Therefore, from QoS point of view, the most important performance metric of Load Control is the amount of traffic admitted. If the adimitted amount of traffic is below the admission target, then QoS guarantees are assumed to be met.

### 4.1 Batch flow arrivals

The problems discussed in this section are relevant only to the the measurement-based variant of Load Control and relate to the fact that with this variant, routers do not perform resource reservations.

When a core router inside a Load Control domain forwards a probe packet unmarked, it declares the acceptance of the corresponding flow. If the packet makes its way through the domain to the edge still unmodified, a notification is sent to the ingres edge device about flow acceptance and the edge device will start transmitting packets of the accepted flow. Thus the router in question expreriences traffic no sooner than one round-trip time after it has forwarded the probe packet. During this round-trip time the router may let any number of probe packets pass unmarked as its measurement of the traffic has not changed due to the newly accepted flow. In other words even if only little resources are free, a large batch of arriving probe packets will all pass unmarked leading to the –erroneous– acceptance of all. The larger the round-trip time the higher the chance of such false admissions.

By taking admitted probe packets into account and reserving some resources for a round-trip time, routers could help the situation. Unfortunately routers do not know the bandwidth of the flows to be accepted, nor the round-trip time, so such reservations are highly problematic.

However, even if the routers do not reserve and make false admissions, the number of such erroneous admissions is small and easy to estimate and control.

It is well known that in systems with a large user population, independent session arrivals tend to form a Poisson process [21]. If we assume exponential session holding times with parameter $\mu$, homogenous connections and a link which can serve $N$ sessions, then the system can be modelled as an M/M/1/N queue. Assume further that at time $t_0$ the router accepted the $N$th connection. During the next round-trip time $T$ the router will forward probe packets unmarked, even though it should refuse new connections from that point on. To estimate the impact of erroneous admissions due to batch parobe packets, we calculate the probability that at time $t_0 + T$ more than $N + K$ sessions are in the system. We denote the arrival rate by $\lambda$.

The probability of having exactly $i$ arrivals during time $T$ is

$$p_i = e^{-\lambda T} \frac{(\lambda T)^i}{i!}$$

If we approximate the departure rate by $\mu N$, then the chance of having $i$ departures is:

$$q_i = e^{-\mu N T} \frac{(\mu N T)^i}{i!}$$

So the probability of having more than $N + K$ connections at $t_0 + T$ is

$$P = \sum_{i=0}^{\infty} \left( q_i \sum_{j=K+i}^{\infty} p_j \right) = 1 - \sum_{i=0}^{\infty} \left( q_i \sum_{j=0}^{K-1+i} p_j \right) \tag{1}$$

Table 1.: Safety Margin Required

| $T$ | 10ms | | 30ms | | 100ms | | 300ms | | 1s | |
|---|---|---|---|---|---|---|---|---|---|---|
| *blocking* | 1% | 50% | 1% | 50% | 1% | 50% | 1% | 50% | 1% | 50% |
| 100 | 3 | 3 | 3 | 4 | 4 | 5 | 5 | 7 | 8 | 11 |
| 300 | 3 | 4 | 4 | 5 | 5 | 7 | 8 | 10 | 12 | 18 |
| 1000 | 4 | 5 | 6 | 7 | 8 | 11 | 12 | 18 | 20 | 36 |
| 3000 | 6 | 7 | 8 | 10 | 12 | 18 | 20 | 33 | 34 | 73 |
| 10000 | 8 | 11 | 12 | 18 | 21 | 36 | 35 | 73 | 62 | 177 |

The value of $P$ can be approximated numerically. To get some insight to the magnitude of admission errors, we fixed $P = 10^{-5}$ and calculated the value of $K$. (See Table 1..) By setting the admission control target utilisation to $N - K$ instead of $N$ we assure that false admission decisions will happen rarely.

We varied the round-trip-time and the size of the link (the value of $N$). The flow arrival rate $\lambda$ was selected such that the resulting blocking probability are close to 1% or 50%. The former models normal operational conditions, while the latter is true rather in times of excess demand. $\mu$ was set to 1/100, resulting in a mean flow holding time of 100 seconds. From the table we

can observe that the value of $K$ is small compared to the system size. Thus only small fraction of the link resources needs to be sacrificed to prevent false admissions.

The table indicates that overload puts more stress on the admission control. As we are primarily interested in the robustness of Load Control during the rest of this section arrival rates will be set to produce approximately 50% blocking. We will see that even if the offered traffic is twice the link capacity, with proper link dimensioning, appropriate guarantees can be given.

To further investigate the problem of missing reservation we performed packet level simulations using the measurement based algorithm in Appendix A. The traffic was simulated on one link, that connected a source and a sink. The source randomly generated on/off flows. The flow interarrival time was exponentially distributed with mean of 0.5 seconds, holding time was also exponential with a mean of 100 seconds. Within a flow the length of on and off periods were both following a Pareto distribution with mean of 5 seconds and parameter $\alpha$ of 1.1. During "on" periods a 40-byte-long packet was sent in every 20 ms for each flow generating 16 kbit/s peak and 8 kbit/s average load. The sources roughly imitated the behavior of voice connections coupled with heavy tailed on and off periods. Probe packets were 20 bytes long. We performed two sets of simulations, one with the target of the admission control set to 800 kbit/s and another with 8 Mbit/s. In both cases the round-trip time was varied between 30 and 300 ms. The saturation probability of the admission control ($\epsilon$) was set to 1% and simulation time was 3 hours in each simulation run.

On average the two systems contained roughly 78 and 930 flows. The link was heavily overloaded: flow blocking probability was in all cases more than 50%. We measured the incoming throughput on the link in 20 ms slots and calculated the histogram of the slot measurements (see Fig. 1.). The mean and the 99% quantile of the histograms are shown in Table 2.. The $N + K$ values shown in the table are the theoretical 99% quantile computed the same way as in Table 1. using (1). To calculate this value for the two set of simulations $N$ was set to the maximum number of admittable flows based on average bitrate, namely $N = 100$ and 1000 respectively; and one flow was assumed to occupy 8 kbit/s bandwidth.

Table 2.: Properties of Link Utilisation Histograms

| | 800 kbit/s link [kbit/s] | | | 8 Mbit/s link [Mbit/s] | | |
|---|---|---|---|---|---|---|
| *RTT* | Mean | 99% | $N + K$ | Mean | 99% | $N + K$ |
| 30ms | 628 | 813.6 | 832 | 7.472 | 7.986 | 8.056 |
| 100ms | 631 | 818.5 | 840 | 7.439 | 7.976 | 8.088 |
| 300ms | 630 | 820.3 | 856 | 7.436 | 8.024 | 8.144 |

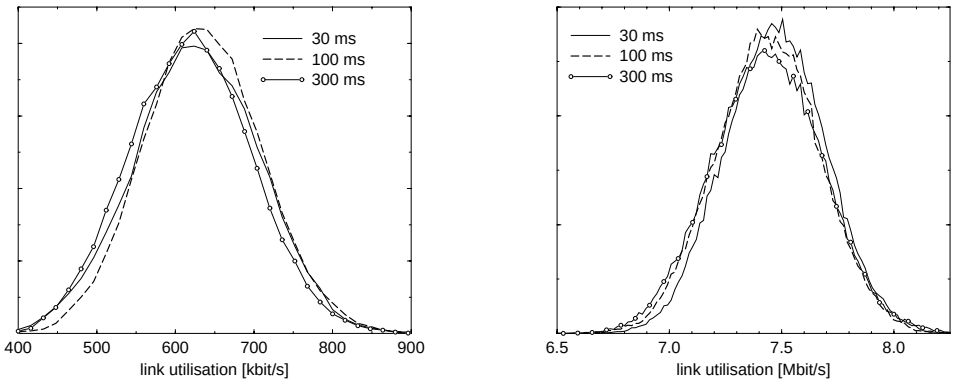


Figure 1.: Histogram of link utilisations

The histograms of the larger links are relatively narrower, as statistical multiplexing gain is higher with higher number of flows. It can be seen that the round-trip time has little effect on the long term performance. If the admission algorithm worked perfectly then the measured 99% quantile values would all be 800 kbit/s or 8 Mbit/s. Larger values are the result of erroneous admission decisions. We can see that generally larger round-trip times result in more bad decisions, but this is not so characteristic if we have a larger number of flows. We can also see that the 99% values are below the theoretical values of $N + K$. This can be attributed to the fact that there are always periods during which the admission control does not have a flow to admit, no matter how overloaded the link is. Thus there were times when the link utilisation could have been higher by the judgement of admission control, but there was no flow to let in. This results in lower values then the worst case figures on Table 1. and indicates that the values of the table are safe to use.

### 4.2 Multiple flow classes

Load Control does not allow edge devices to inform the core routers about the exact bandwidth and QoS requirements of the flows. Although some rough QoS (loss or delay) requirements can be expressed by the DS codepoint the flow will use and which is available to the routers, the probe packet tells nothing about the size of the flow.

In the measurement-based variant, the problem can be addressed by setting the target utilisation of the admission algorithm to the real target minus the size of the most bandwidth demanding type of flow. This way the router will forward probe packets unmarked only if it has enough space even for the largest type of flow, thus admitting one will not cause overload. Although by lowering the target some resources become unavailable to smaller flows and are wasted, the amount of waste is at most the size of the largest flow. In a large network network

serving a lot of flows we expect that the links can carry many even from the most bandwidth demanding type of flow. For example, if the link can serve 100 from the largest flow, the waste is less than 1%.

If unit-based reservations with refreshment packets are used then another possibility exists: multiple probe packets can be sent per flow. This helps the problem because in this case probe packets actually reserve resources. In this case the flow is accepted only if all the probe packets pass through the network unmarked. This way larger flows are admitted only if there is enough free capacity to service them. If any of the probe packets is marked, then the flow is rejected and the resource units allocated by the unmarked probe packets are wasted for one refreshment period. Again, the amount of waste is proportional to the size of the largest flow.

To investigate the effects of sending more than one probe packets per flow, we performed simulation on the simple network shown in Fig. 2.a. Three sources, $S_A$, $S_B$ and $S_C$ were generating traffic towards destination D. These sources can be hosts or edge devices as well. The sources were similar to the source used in Section 4.1, with the exception that the generated flows this time were not on/off, but constant bit rate sources. Source A, B and C generated 16 kbit/s, 160 kbit/s and 1.6 Mbit/s flows with average flow interarrival time of 15 ms, 150 ms and 1.5 second, respectively. Thus the load generated by the three sources was equal. The sources were periodically transmitting refreshment packets. Link bandwidth was 160 Mbit/s on each link. The refreshment period was 1 second and the unit which one refreshment packet reserves was set to 16 kbit/s. Thus source A, B and C sent 1, 10 and 100 refreshment packet per flow per second. Refreshment packets belonging to one flow were equally distributed within the refreshment period.

Admission control was exercised on link RD. The target was set to 160 Mbit/s or 10000 unit. Two cases were simulated, both runs were 3 hours long. In the first case, sources generated 1, 10 or 100 back-to-back probe packets on flow arrival according to the size of the flow. In the second case only one probe packet was generated on flow arrival even for the largest flows, but the target was lowered to 9900 units. Again, the utilisation was measured in 20 ms slots and a histogram of the slot measurements were calculated. Fig. 2.b shows the histogram of both simulation runs and Table 3. summarizes the tail behavior of the utilisation.

It can be observed that by using multiple probe packets some bandwidth is really wasted. The amount of waste is not more than the size of the largest flow, and in many cases smaller. This can be attributed to the fact that if for a longer time no flow arrives from the largest type, then smaller ones can fill up the link better. In the single packet case many wrong decisions were made, the 160 Mbit/s threshold has been violated many times, even though the target

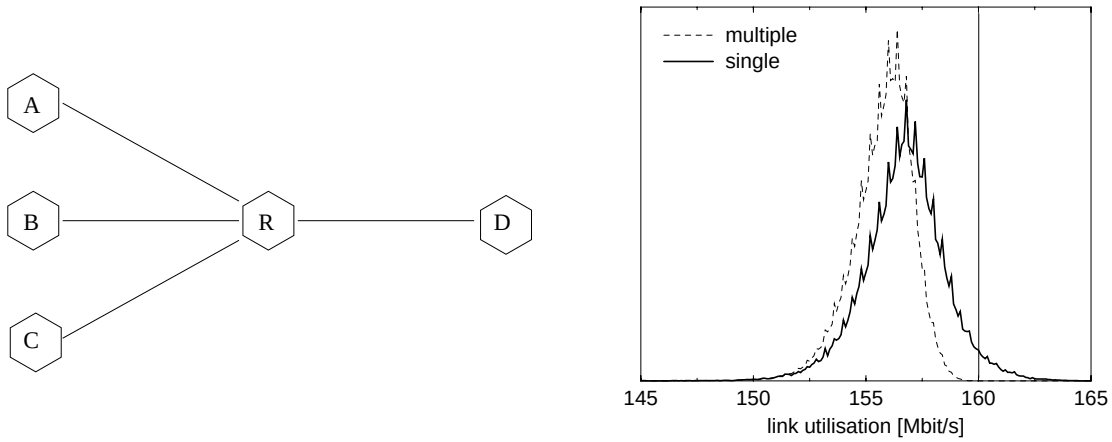


Figure 2.: Simulation topology and histogram of link utilisations

Table 3.: Properties of link utilisation histograms

| | multiple probe packet case | single probe packet case |
|---|---|---|
| average | 155.6 Mbit/s | 156.4 Mbit/s |
| 99% quantile | 158.3 Mbit/s | 161.2 Mbit/s |
| maximum value | 159.8 Mbit/s | 166.7 Mbit/s |
| type A (16 kbit/s) | 61.0 Mbit/s | 52.2 Mbit/s |
| type B (160 kbit/s) | 59.3 Mbit/s | 51.5 Mbit/s |
| type C (1.6 Mbit/s) | 35.3 Mbit/s | 52.8 Mbit/s |

was set to 158.4 Mbit/s or 9900 units. This is the result of the effect described in the previous section. When a probe packet is passing, the router cannot tell what will be the size of the flow and reserves only one unit of resources until the refreshment packets arrive. During the first round-trip time other flows may be –falsely– admitted. From this we can conclude that both methods are robust and efficient, but in the single probe packet case upon determining the threshold, the arrival process has to be taken into account.

The second part of Table 3. shows how much bandwidth the different flow types occupied on the long run. The offered traffic of the 3 sources was the same. In the single probe packet case the admission control achieved the expected fairness as each flow type received roughly one third of the link capacity. This is because each flow sends only one probe packet that gets marked with the same probability, so blocking is the same for all type of flows. However, in the multiple packet case large sources have a higher chance of getting rejected due to the many probe packets needed for admission. This is a general property of multirate systems and results in a clear unfairness. We note, however, that in the multiple probe packet case the target is never violated no matter what the arrival process or intensity is.

### 4.3 Length of the refreshment period

This section is relevant only for the variant that uses refreshment packets, as it discusses issues related to the length of the refreshment period.

The reservation performed by a refreshment packet persists in the routers for one refreshment period even if the flow departs sooner. This results in a resource waste that depends heavily on the length of the refreshment interval. Assuming that average flow holding times ($T_h$) are much longer than the refreshment periods ($T_r$), we can estimate the amount of resources wasted. If the last refreshment packet is forwarded at $t_0$ flow departure can happen anywhere between $t_0$ and $t_0 + T_r$, depending on the length of the flow. When using the algorithm in Appendix B the resources are freed between $t_0 + T_r$ and $t_0 + 2T_r$ depending on the offset between the refreshment period of the source and the router. We approximate both values with uniform distributions, and conclude that on average resources are wasted for approximately $T_r$ time for each flow. As $T_h \gg T_r$, the portion of resources wasted is approximately $T_r/T_h$.

Simulation gives similar results. To demonstrate, we simulated one link with one type of constant bit-rate flows, each occupying one unit of link resources. The holding time was exponential again with 100 seconds mean. Two sets of simulations were run with different link capacities (100 and 1000 units) and the refreshment period was varied between 100 ms and 3 seconds each case. In Fig. 3. the histograms of utilisation and on Table 4. the mean utilisation values are shown for both set of simulation runs.

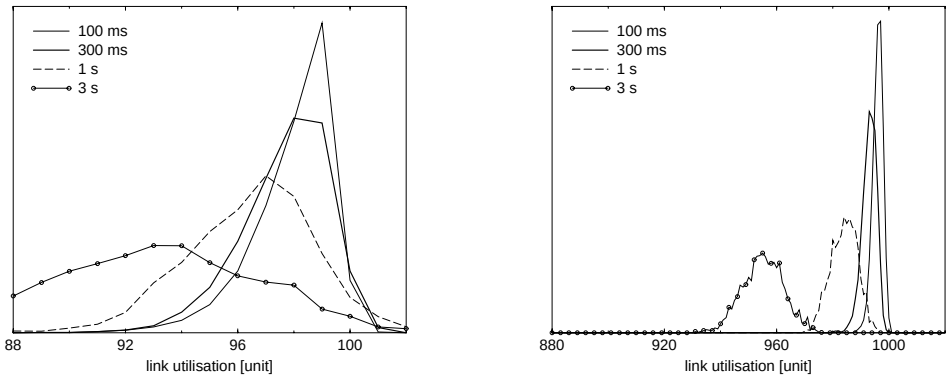


Figure 3.: Histogram of link utilisations for different refreshment periods and link capacities

The two graphs show that larger refreshment periods result in lower utilisation and higher waste. From the table we can conduct that the average relative utilisation is always below the theoretical approximation. This is because the real systems does not always have traffic to fill the link and use all available resources. It can be observed, however, that even in case of longer

Table 4.: Average Utilisation for Different Refreshment Periods and Link Capacities

| | 100 unit | 1000 unit | $1 - T_r/T_h$ |
|---|---|---|---|
| 100 ms | 98.0% | 99.8% | 99.9% |
| 300 ms | 97.7% | 99.5% | 99.7% |
| 1 sec | 96.4% | 98.6% | 99.0% |
| 3 sec | 93.0% | 95.7% | 97.0% |

refreshment periods high utilisation can be achieved.

# 5 Conclusion

In this paper we describe and analyse Load Control, a lightweight resource provisioning method for Differentiated Services domains. It performs admission control and provides feedback to edge devices in times of serious overload by using simple bit-markings in packet headers. The congestion notification signals provided by Load Control are similar to existing explicit congestion notification schemes but are more suited to real-time traffic. Its main advantages are simplicity, scalability and easy implementation in hardware. By investigating its properties we show that it is robust even when facing serious overload.

# References


[1] Internet 2 QBone Bandwidth Broker Advisory Council homepage, —http://www.merit.edu/working.groups/i2-qbone-bb—

[2] B. Braden B. Ed., *et. al.*, *Resource Reservation Protocol (RSVP) - Version 1 Functional Specification*, RFC 2205, Sept. 1997

[3] I. Stoica, H. Zhang, *LIRA: A Model for Service Differentiation in the Internet*, NOSSDAV'98, `http://www.cs.cmu.edu/~hzhang/LIRA/`

[4] I. Stoica, H. Zhang, *Providing Guaranteed Services Without Per Flow Management*, SIGCOMM'99, Boston, Oct. 1999

[5] G. Fehér, K. Németh, M. Maliosz, D. Ahlard, J. Bergkvist, I. Cselényi, T. Engborg, *Boomerang - A Simple Protocol for Resource Reservation in IP Networks*, IEEE Workshop on QoS Support for Real-Time Internet Applications, Vancouver, Jun. 1999

[6] P. Pan, H. Schulzrinne, *YESSIR: A Simple Reservation Mechanism for the Internet*, in ACM Computer Communication Review, vol. 29, pp. 89-101, Apr. 1999

[7] W. Almesmerger, T. Ferrari, J.Y. Le Boudec, *SRP: a Scalable Resource Reservation Protocol for the Internet*, preprint 1999, `http://lrcwww.epfl.ch/srp/`

[8] G. Apostolopoulos, R. Guerin, and S. Kamat, *Implementation and Performance Measurements of QoS Routing Extensions to OSPF*, IEEE INFOCOM'99, New York, Mar. 1999

[9] M. Grossglauser, D. Tse, *A Time-Scale Decomposition Approach to Measurement Based Admission Control*, IEEE INFOCOM'99, New York, Mar. 1999

[10] Z.R. Turanyi, A. Veres, *A Family of Measurement/based Admission Control Algorithms*, IFIP PICS'98, Lund, May 1998

[11] L. Westberg, Z.R. Turanyi, *Load Control of Real-Time Traffic*, Internet Draft, Jun. 1999

Table 5.: Algorithm parameters

| | | |
|---|---|---|
| $T_h$ | average flow holding time | 100 sec |
| $T_c$ | sliding window of average calculation | $T_h/\sqrt{N}$ |
| $T_s$ | slot time | 20 ms |
| $\epsilon$ | tail probability | 1% |
| $target$ | the target utilisation | |


[12] V. Jacobson, *Congestion Avoidance and Control*, SIGCOMM'88, Palo Alto, 1988

[13] K. Ramakrishnan, S. Floyd, *A Proposal to add Explicit Congestion Notification (ECN) to IP*, RFC 2481, Jan. 1999

[14] S. Floyd, V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, vol. 1 N. 4, pp. 397-413, Aug. 1993

[15] L. Massouliè, J. Roberts, *Arguments in favour of admission control for TCP flows*, International Teletraffic Congress, ITC 16, 1999

[16] R.J. Gibbens, F.P. Kelly, *Resource pricing and the evolution of congestion control*, Automatica 35., 1999

[17] J.H. Saltzer, D.P. eed, D.D. Clark, *End-to-end arguments in system design*, reprinted in ACM Transactions in Computer Systems 2, 4, pp. 277-288, Nov. 1984

[18] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *An Architecture for Differentiated Services*, RFC 2475, Dec. 1998

[19] B. Braden, D. Clark, S. Shenker, *Integrated Services in the Internet Architecture: an Overview*, RFC 1633, Jun. 1994

[20] E.C. Rosen, A. Viswanathan, R. Callon, *Multiprotocol Label Switching Architecture*, Internet Draft, Aug. 1999

[21] L. Kleinrock, *Queueing Systems, Volume 1, Theory*, John Wiley & Sons, 1975, ISBN 0471491101


# A Measurement-based router algorithm

In this appendix we describe a measurement based algorithm, that is used in our simulations. The algorithm is based on [9]. It does not assume any signalling facility other than probe packets described in Section 3.1

Parameters of the algorithm and the values used in this paper are summarized in the table 5.. $N$, the number of flows were estimated to be 100 and 1000 in the two simulation runs and the link capacity was 800 kbit/s and 8 Mbit/s (see Fig. 1.).

The algorithm calculates the average incoming rate within slot times, by counting transmitted bytes and dividing the counter by the slot time. These rate values are then smoothed with an exponentially weighted moving average filter, whose window size is $T_c$. The variance of the difference between the actual slot measurements and the filtered average is also calculated and smoothed with a filter of $T_h$ window size. Then the tail of a normal distribution with the smoothed average and variance is estimated. All complex calculations take a fixed number of operations are performed at slot time, per packet work is very small.

Fig. 4. shows the algoithm. the function $EWMA(a,t)$ calculates the smoothed value of $a$ with the window size $t$ and function $TAIL(m,\sigma,\epsilon)$ estimates the tail of the normal distribution $N(m,\sigma)$.

**On initialisation**
```
bytes = 0
avg = 0
variance = 0
```
**On arrival of a packet**
```
bytes += length of the packet
```
**On the end of the slot**
```
rate = bytes/T_s
avg = EWMA(rate, T_c)
diffsq = (rate-avg)^2
variance = EWMA(diffsq, T_h)
bandwidth = TAIL(avg, variance, ε)
bytes = 0
```
**On arrival of a probe packet**
```
if bandwidth > target then
   Mark Packet   //rejection
endif
```

Figure 4.: Measurement based router algorithm

# B Router algorithm for refreshment packets

The simple algorithm on Fig. 5. works by counting probe and refreshment packets. It ensures that only so many probe packets are forwarded unmarked as many units were left unutilised in the previous refreshment period. Its only parameter is *target*, which is the number of units that the link can carry.

**On initialisation**
```
last = 0
count = 0
```
**On arrival of a refreshment packet**
```
count ++
```
**On arrival of a probe packet**
```
if last < target then
   last ++       //acceptance
   count ++
elseif
   Mark Packet   //rejection
endif
```
**On the end of the refreshment interval**
```
last = count
count = 0
```

Figure 5.: Router algorithm for refreshment packets