Integrating Parallel Computing Applications in an ATM Scenario

Joan Vila-Sallent, Josep Solé-Pareta
Universitat Politècnica de Catalunya
Jordi Girona 1–3, Mòdul D6 (Campus Nord), 08034 Barcelona Catalunya (Spain)
E-mail: joanv@ac.upc.es, pareta@ac.upc.es

Abstract
This paper addresses the problem of supporting communications in parallel computing applications over ATM networks. We propose a mechanism specifically conceived for optimizing the cost-performance tradeoff in fairly long parallel executions. The proposed mechanism relies on a modified version of the loss recovery procedure of SSCOP, which is enhanced by means of a more intensive exploitation of ATM service categories in order to reduce the occurrence of cell loss. For this purpose, we make use of both the UBR and ABR service categories, with ABR being only introduced in the periods of high latency. These periods are determined by periodically monitoring the experienced latency. This approach can achieve equivalent latency as the plain ABR service but with a use of this service of only 30%–70% of the parallel computing traffic, depending on the load of the network and the characteristics of the application.

Keywords
ATM, Corporate Networks, Distributed Parallel Computing

1 INTRODUCTION

The availability of a high-speed network with the flexibility of ATM (Asynchronous Transfer Mode), together with the current evolution of microprocessor technology, is enabling the convergence of communications and computing. In addition, as defined by the ITU (International Telecommunication Union), ATM is the technology that will integrate the whole diversity of network-based services [1]. The combination of these three factors —high-speed networks, microprocessor technology, and integration— is facilitating the development of new applications requiring intensive communications. One of these applications is the support to distributed parallel computing, where a number of

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35388-3_42 workstations connected to an ATM network can act as nodes of a parallel computing platform. Such environments cannot reach the performance achieved by more expensive, dedicated platforms such as multiprocessors, although they can be a sufficient replacement for many applications [2]. The main bottleneck in network-based parallel computing is experienced in the network itself, and is caused by the delays produced by the protocol processing, the interface with the network, and the processing within the network [3]. These issues occur despite the bandwidth enabled by ATM.

In an integrated environment, communications in parallel computing applications are not limited to LAN environments —which can be adequately supported by Myrinet or Gigabit Ethernet— but are suitable to be extended beyond the local area. The adoption of ATM allows for parallel computing applications to take advantage of an existing network, thus avoiding the underutilization of duplicated resources that would appear with the use of dedicated networks such as Myrinet. Thus, organizations whose parallel computing needs are not very intensive will be able to achieve satisfactory performance with a more efficient exploitation of resources. In this context, parallel computing environments have to subject to a number of conditions in order to achieve sufficient performance with cost-effectiveness. The first condition we assume is that parallel computing applications will share the ATM network with traditional networking applications. Thus, the network architecture will require the presence of mechanisms enabling the support of parallel computing applications that can coexist with equivalent mechanisms for traditional networking applications. In order to preclude the increase in complexity that would arise with the enhancement of ATM with application-specific mechanisms, we consider that the adaption of parallel computing to ATM should be done with mechanisms implemented on top of ATM. Thus, ATM will solely support those services defined in the standards by ITU-T (Telecommunication Standardization Sector of ITU) and the ATM Forum [4].

Many of the applications to be integrated in ATM networks have strong bandwidth and/or delay requirements, as they manage continuous data streams. In these applications, network mechanisms should maximize the network capacity, measured by throughput. In parallel computing applications, however, communications involve the exchange of relatively short pieces of data along a relatively long execution period, so the minimization of communications time has not a tight relationship to network capacity. Thus, communications in parallel computing applications approach to the request-response model, since each task sends data to other tasks and expects other data from them. In this model, per-message overheads set a limit on the achievable performance and therefore, as discussed in [5], latency is a measure that gives a clearer idea about communication performance in parallel computing applications. We consider the latency measure as embedding all per-message communication costs which include, in addition to the costs of overheads and the delays from buffering and scheduling, the eventual need of recovering from cell loss that results from the need of sharing the network with other networking applications.

In this paper, we propose a mechanism that enables low latency operation for communications in parallel computing environments over ATM networks integrating different networking applications. In particular, we concentrate on networks spanning outside the local area, where the impact of the applications sharing the network with parallel computing can be more significant. Thus, our mechanism will provide for a strategy to minimize latency degradation caused by the presence of background traffic, which is based on periodically monitoring the latency experienced by communications in real time, in order to achieve cost-effective performance. The rest of the paper is organized as follows: Section 2 presents the general characteristics of the target environments for our mechanism. In Section 3, the particular features of the mechanism proposed in this work are extensively discussed, and their performance is evaluated in Section 4 Finally, Section 5 concludes the paper.

2 PARALLEL COMPUTING IN AN INTEGRATED ENVIRONMENT

Most environments to support parallel computing are addressed to operate on multiprocessors or high-speed LANs. Both of these environments can be very expensive when a large number of nodes is required by applications, so they are basically adequate for intensive use of parallel computing facilities. When the need of parallel computing support is not so intensive, it is better to allow for parallel computing environments to extend beyond the local area to provide appropriate scalability to parallel computing applications. In this case, LAN technologies like Gigabit Ethernet are not applicable, so the role of ATM as an integrating technology is more clear. Another important issue outside the local area is the greater influence of network load as more applications are then presumed to share the ATM network. In this paper, we assume that the scenario for distributed parallel computing over ATM will be based on a virtual network comprised by the endpoint hosts supporting the tasks of parallel computing applications, as well as other nodes implementing the procedures providing addressing, connection management, and other signaling functions. In this model, the endpoint hosts support the actual data transfer operations, while the rest of the nodes in the virtual network are in charge of establishing the necessary connections between the endpoint hosts in order to build the topology required for each particular parallel computing application. The signaling procedures operate before and after the actual execution, and are out of the scope of this paper.

Figure 1 displays the architecture of the endpoint part of the ATM-based platform. Data transfer mechanisms for parallel computing are supported in a specific architecture to be integrated with the specific architectures of traditional networking applications. A proposal for the architecture of the parallel computing service is discussed in [6]. Three levels are considered: (1) ***Application level***, which manages the specific requirements of parallel computing applications; (2) ***Network level***, containing the functions provided by a particular network technology, as ATM in the present work, and (3) ***Convergence level***, which includes those functions that are required for an adequate support of parallel computing applications, and are not provided by the network level as defined in the respective standards.

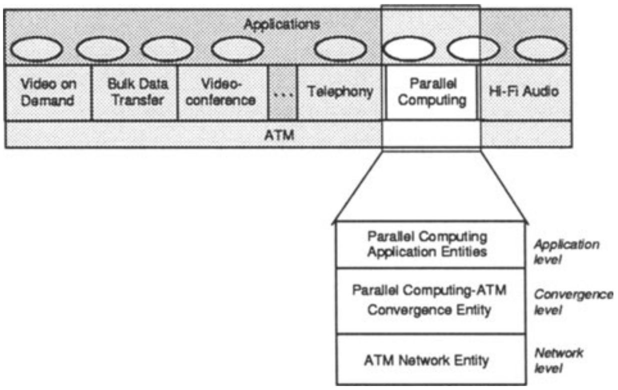


Figure 1 Integration of services over ATM.

Communications in parallel computing applications are considered to be based on sequences of elementary data types —integer, float, double, etc.—, called PC-PDUs after "Parallel Computing Protocol Data Units" which are the minimum data structures understood by parallel tasks in a logical sense. Larger structures —arrays, structs, etc.— can be broken into these elementary PC-PDUs. The needs of bandwidth are not very high on average, because communications among parallel tasks are not occurring continuously but an arbitrary period of time can separate the issue of two consecutive messages. Nevertheless, in the particular instants when a message is submitted, very low latency is required in the network in order to minimize the impact of communications on performance. As the mechanisms in the convergence level have to satisfy all these requirements with a full guarantee of data delivery, this paper adopts the specific ATM Adaptation Layer (AAL) proposed in [6], which is based on a modified version of SSCOP (Service Specific Connection Oriented Protocol). SSCOP is a protocol defined by ITU-T in the Q.2110 recommendation [7] for supporting a number of services requiring reliability on top of ATM. This specific AAL replaces AAL5 and improves performance by avoiding the retransmission of more cells than those effectively lost. With this AAL, applications are less sensitive to the network load induced by the rest of applications sharing the ATM network, and communications achieve better latency performance. This AAL, however, does not rely on any particular ATM service category from those specified by the ATM Forum [4]. In order to optimize performance, we can propose a modified version of the AAL that takes advantage of the features included with these service categories.

The fact that parallel computing applications exchange relatively short messages along a relatively long execution periods makes the use of guaranteed communications services —such as CBR (Constant Bit Rate)— not convenient. Instead, best-effort services as UBR (Unspecified Bit Rate) and ABR (Available Bit Rate) are the most appropriate service categories to support communications in ATM-based parallel computing environments, since their cost will rely mostly on the effective consumption of bandwidth, as opposed to other service categories where the length of connection period will be a more important issue. UBR is the least expensive service category, but the latency can be excessively high due to the cell loss occurring as the network load increases, while ABR is more expensive but faster, as the built-in flow control mechanism allows to achieve lower latency thanks to the fewer retransmissions needed.

In addition, because of the long execution periods, a number of high activity and low activity periods may alternate in the network, as a result of the applications sharing the network. In the periods with low network traffic, the performance of UBR may be sufficient and, as a result, the higher cost of the ABR service category would not be amortized. Thus, for achieving cost-effective performance the data transfer should be conveyed through UBR when the latency experienced in the network and, when latency through UBR is excessively high, data transfer should be moved to an ABR-based connection. A procedure to monitor latency is therefore needed in order to determine when to activate the ABR service category.

3 ENHANCED PARALLEL COMPUTING AAL

We focus on the data transfers occurring during execution time by assuming that the necessary connections have been established prior to the execution. As mentioned above, our proposal is conceived to provide cost-effective performance by adapting to the latency experienced by parallel computing communications. In the following we detail the operation of our mechanism, starting with an overview and continuing with a detailed description.

3.1 Architecture

Figure 2 depicts the architecture of our mechanism when the extensions to the Parallel Computing AAL are applied. In each communicating peer, the functionality is contained in two concurrent processes: (1) The *Latency Mon-*

itoring Engine (LME), which monitors the latency in the network in order to determine the periods in which significantly high latency is experienced, and (2) the *Effective Communication Engine (ECE)*, which performs the actual data transfers according to the information supplied by the LME. The communications between the engines are served by four connections between each pair of communicating endpoints:

- A UBR-based connection with an unlimited peak rate, used by the ECE to transfer data when latency is low. We refer to this connection as the *ordinary connection.*
- An ABR-based connection with a limited peak rate and a minimum bit rate set to zero, used by the ECE to transfer data when the LME indicates that latency is high. This connection is referred to as the *backup connection.*
- A UBR-based connection like the ordinary connection, which used by the LME to monitor latency. In practice the same UBR connection is used for both purposes.
- A VBR (Variable Bit Rate) connection with a guaranteed low peak in order to support a fast and reliable delivery of feedback information in the LME.

The adoption of a VBR service category —whose cost is significantly higher than ABR— could compromise the objective for cost-effective performance of our mechanism. However, later in the paper we will observe that the adoption of a VBR-based connection does not significantly impact on performance of parallel computing applications.

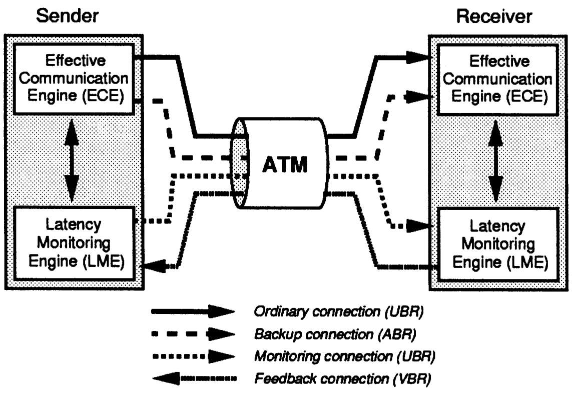

Sender
Receiver
Effective Communication Engine (ECE)
Latency Monitoring Engine (LME)
ATM
Effective Communication Engine (ECE)
Latency Monitoring Engine (LME)
Ordinary connection (UBR)
Backup connection (ABR)
Monitoring connection (UBR)
Feedback connection (VBR)

Figure 2 Mechanisms for extending the Parallel Computing AAL.

3.2 The Effective Communication Engine (ECE)

The Effective Communication Engine (ECE) consists of an extension of the Parallel Computing AAL as described in [6] that allows to exploit the information supplied by the LME in order to achieve low latency communications. The mechanism discussed in [6] is based on a modification of the selective retransmission procedure of SSCOP. The modification to SSCOP is addressed to limit the length of the frames to one cell. Thus, unlike standard SSCOP, the amount of retransmitted cells corresponds exactly to the lost cells and, as a consequence, applications become less sensitive to network load. This modification is possible thanks to the short length of PC-PDUs —corresponding to elementary data types such as integer, float, etc., as noted above. In particular, each cell encapsulates as many complete PC-PDUs as possible, so that the data can be integrated with computation as soon as received. In order to avoid the unnecessary overheads involved with the payload length and the 32-bit checksum of AAL5, the mechanism directly replaces AAL5, so it actually operates as a specific AAL. Figure 3 shows the structure of a cell generated by this specific AAL.

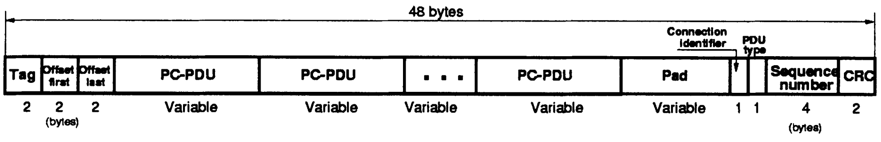


Figure 3 Encapsulation scheme of the specific AAL for parallel computing.

The cell structure shown in Figure 3 includes a significant amount of overhead. Although this overhead obviously leads to throughput degradations, we are more interested in optimizing message latency because of the small significance of throughput on the performance of communications in parallel computing. The overhead includes the following fields:

- *AAL-related fields*, which only includes the CRC field. This 16-bit CRC allows to avoid the unnecessary 32-bit CRC provided in AAL5, which is not adequate for 1-cell AAL PDU.
- *SSCOP-related fields*, which include the Sequence Number and PDU Type fields. These fields are directly inherited from standard SSCOP, but the Sequence Number allows for a larger number space because restricting PDUs to one cell will presumably lead to a higher amount of PDUs.
- *Message-passing library fields*, represented by the Tag, Offset First, and Offset Last fields. They are set to enable compatibility with the PVM (Parallel Virtual Machine) message-passing library [8], which is used by the

parallel programs we have tested. Other message-passing libraries would possibly require different fields.

- *Connection management fields*, which include a Connection Identifier than supports an additional addressing level together with the VCI/VPI fields, in order to facilitate the implementation of a virtual network supporting parallel computing communications.

The ECE enhances the Parallel Computing AAL described in [6] by considering two operation modes: ***low-latency mode***, and ***high-latency mode***. The extension to the AAL applies essentially to the high-latency mode, which is activated when the LME detects a significant growth in the latency experienced in the ordinary connection. In this case, the backup connection is enabled, so that the cells transmitted on the original connection are switched to the backup connection in order to minimize the impact of cell loss on performance. Figure 4 outlines the operation of the ECE.

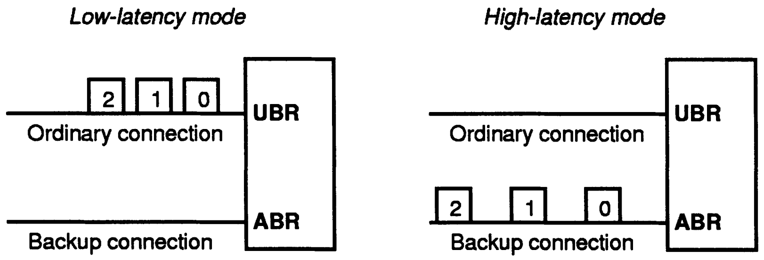


Figure 4 Operation of ECE's latency modes.

(a) Low-latency mode

In this mode, the operation of the ECE reduces to the mechanism of the Parallel Computing AAL just outlined. The transfers of data take place over the ordinary connection, so a UBR service is used. Latency monitoring by the LME takes place also over this ordinary connection using a UBR service.

When the receiver part of the LME detects that a monitoring cell has been lost, or when the ECE itself considers that the measured latency is high —i.e. it exceeds a threshold T_M, the ECE activates the high-latency mode. For this purpose, it issues a new control frame, called LSTAT, which is equivalent to a STAT frame but contains also a time stamp corresponding to the instant when the offending monitoring cell was issued from the sender. LSTAT frames are sent through the same VBR service as USTAT and STAT frames.

(b) High-latency mode

When the sender (in low-latency mode) receives an LSTAT frame, it switches to the high-latency mode and triggers the retransmission of pending data, just as a STAT frame. Then, all cells are issued through the backup connection only. As in low-latency mode, USTAT frames are generated when detecting cell loss. When the sender receives STAT and USTAT frames, the retransmission will be conveyed by the backup connection only. Thus, the operation of the ECE in high-latency mode is similar to the operation in low-latency mode, except for the fact that the backup connection (over an ABR service) is used instead of the ordinary connection (over a UBR service).

When the latency monitored by the LME falls below a threshold T_m, the low-latency mode is again activated by issuing an LSTAT frame to the sender, including again the information about the status of received cells as contained in STAT frames. The sender then retransmits the cells through the ordinary connection only. The threshold T_m should be lower than the threshold T_M in order to avoid a continuous switching between both modes. In all cases, latency is monitored by the LME over the ordinary connection only (that is, over a UBR service) regardless of the operation mode.

3.3 The Latency Monitoring Engine (LME)

The goal of the LME is to provide an estimation of the latency experienced in the ordinary connection. For its implementation we have considered three decisions:

- *Averaging vs. instantaneous monitoring.* Latency can be monitored by computing the average latency over a period of time. This is well suited for applications dealing with large chunks of data, like video and file transfers, but as this procedure has a slow response time, it is not convenient for applications generating more bursty traffic patterns. Therefore, we believe that instantaneous monitoring is a more adequate approach for parallel computing applications.
- *Asynchronous vs. periodic activation.* Latency can be monitored either before a burst of messages or in a periodic fashion. The former case forces the ECE to defer the transmission until the latency is monitored, so it involves a significant amount of latency. In contrast, the latter approach enables the ECE to avoid this delay. For this reason, we believe that a periodic LME is more adequate, despite the extra bandwidth required to support periodic monitoring.
- *The monitoring mechanism.* We can consider the following options: (1) using network-level information; (2) computing the Round Trip Time (RTT); and (3) synchronizing peers and using time-stamped information. The first approach requires the use of a ABR-like network level mechanism providing accessible feedback information, which is not currently standardized within ATM. In the second case, the computed time depends on the latencies of both the monitored connection and the returning path, which are not necessarily equivalent. In the third approach, the experienced la-

tency is monitored by the receiver LME peer, so there is no influence of the returning path on the computed value. As a result, we adopt the third approach as we find that it suits better the requirements of parallel computing communications.

The operation of the adopted approach for the LME is as follows: the sender periodically submits a cell containing a time-stamp. When the receiver gets this cell, it compares its time-stamp to the time the receiver expected to get the cell. The measured latency corresponds to the difference between both times, and then the measurement is passed to the ECE so that it takes the appropriate action, which in the implementation of the ECE discussed above consists of replying to the sender if the monitored latency exceeds a threshold. As the time-stamped cells might be lost, when a certain amount of time T_L has elapsed since the expected time, the receiver warns the ECE of that circumstance, meaning that a monitoring cell is possibly lost. Figure 5 illustrates the operation with an example. As an enhancement to this basic procedure, the cells issued by the ECE through the ordinary connection are also monitored their latency in order to reduce the response time of the whole mechanism.

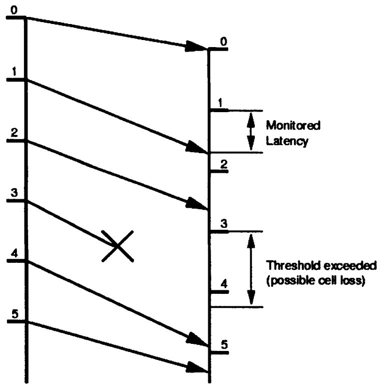


Figure 5 Operation example for the time-stamped LME.

It is important to note that both sender and receiver must be synchronized to each other in order for the measurements to be significant. For this purpose, one of the peers has to report the other one on its current time with a certain periodicity. Thus, we consider two tasks included in the time-stamp LME: (1) *Monitoring task,* which deals with both the periodic and the ECE-originated time-stamped cells; and (2) *Resynchronizing task,* which guarantees that time values are consistent for both communicating peers. We can make use of the different service categories provided by ATM in order to implement these tasks. The Monitoring Task is carried out over the same connection as the ordinary data transfers in the ECE, so it is supported by a UBR service. The Resynchronizing Task requires also high priority and, as it is periodic, a CBR service is more adequate. Note that the peak rates for the CBR service should keep low in order to avoid the allocation of an excessive amount of resources. The concrete value of the period depends mostly on the characteristics of the system clocks in both communicating peers, since the more diverging the clocks are, the more frequently the Resynchronizing Task should be activated. In the experiments presented below, we will assume both endpoints as perfectly synchronized and, therefore, no resynchronizing task is considered.

4 PERFORMANCE MEASUREMENTS

The goal of the mechanism presented so far is to allow for parallel computing applications to achieve satisfactory performance while keeping the cost not higher than strictly required. To characterize the performance, the average end-to-end latency has been measured in a simple configuration, in order to realize the impact of the mechanism. The cost of the mechanism is also determined and compared to that of the standard ABR service.

4.1 Experiment configuration

For moderate network sizes and buffer capacities, the most significant contributions to latency come from the bottleneck links in the ATM network, due to the cell loss and subsequent retransmissions occurring when becoming congested. Thus, the configuration shown in Figure 6 is sufficient for evaluating the performance of the proposed mechanism, and is simple enough to allow for simulations to keep within a reasonable duration.

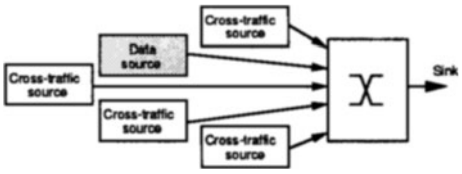


Figure 6 Simulated environment.

All the links have a capacity of 155 Mb/s. Two types of sources are considered: one *data source* modeling traffic from a real parallel computing application by means of a trace, and a number of *background sources* modeling traffic from traditional networking applications, by means of ON-OFF sources. The traffic generated by the data source corresponds to the messages generated by one task of the parallel computing application. In contrast, the traffic from each background source represents the result of multiplexing many sources of traffic from traditional networking applications.

Traces for the data source have been obtained from the execution of parallel codes from the GENESIS benchmark suite [9]. In particular, the considered codes have been *PDE1* and *PDE2*. *PDE1* is a solver of the Poisson Equation on a three-dimensional grid by using red-black successive over-relaxation. *PDE2* solves a two-dimensional Poisson equation using a multigrid method. The traffic generated by *PDE1* consists of relatively long bursts (around 8 KB). In contrast, bursts from *PDE2* are much shorter (50–100 Bytes). As a result, different behavior is expected for each code.

As far as background traffic sources are concerned, the values for the parameters of both the ON and OFF states are exponentially distributed. In the measurements, several sets of values have been used in order to obtain diverse aggregate input rates. In particular, the network utilization ρ ranges from 0.3 to 1.1, with respect to the output link capacity. ρ stands for the average network load along the execution period. A value for ρ greater that 1 indicates that the aggregate incoming traffic in on average higher that the output link capacity. As each background source models the result of multiplexing several sources we do not want a very aggressive background traffic. Thus, the parameters of the ON-OFF models generate a traffic pattern with a burstiness not higher than required to capture the characteristics of multiplexed cell streams. As demonstrated in several papers, for example [10], their burstiness decreases as long as the number of multiplexed sources grows.

The switch is modeled as output-queued. Two priority levels are considered: one for guaranteed service categories (in particular VBR), and the other for best-effort service categories (ABR and UBR). The buffer space is shared by the logical queues associated with each priority level. The buffering scheme is basically drop-tail, except for the case of a full switch buffer, where the arrival of a non-UBR cell forces the dropping of an UBR cell already queued in the switch. The aggregate incoming traffic is arranged in order for the switch to contemplate it as a mixture of UBR and ABR traffic. The ABR scheduling algorithm adopted in the measurements in based on ERICA (Explicit Rate Indication for Congestion Avoidance), fully described in [11]. Table 1 shows the values for the most relevant parameters in the switch and the sources, which in turn are mostly based on the defaults suggested in [4, 11, 12]. Table 2 displays the values for the parameters used in the performance evaluation study presented in this section.

Table 1 Values for the relevant parameters of the ABR service.

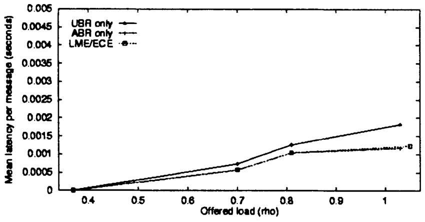
Element	Parameter	Value
Switch	Target Utilization	0.9
	Measurement Interval	30 cells
Source	Nrm	32 cells
	ADTF	0.5 sec
	Peak rate	50 Mb/s

Table 2 Parameters for our low-latency mechanism.

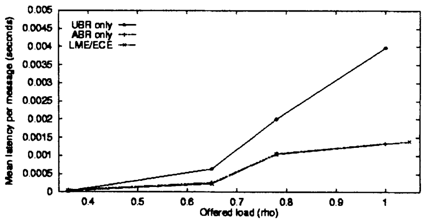
Parameter	Value
SSCOP POLL interval	0.1 sec
LME monitoring interval	0.1 sec
LME loss threshold T_L	0.1 sec
ECE latency threshold T_M	0.0001 sec
ECE latency threshold T_m	0.00009 sec

4.2 Task-to-task latency

Task-to-task latency is the measure determining the effective impact of communications on the performance of the parallel environment. As we assume that the ATM network is shared with other applications, we expect important variations on performance according to the load of the ATM network. Figure 7 shows task-to-task latency as a function of the different values for the background load. We have compared our proposal for enhancing the Parallel Computing AAL with the AAL without these enhancements, the latter by considering both UBR and ABR as the service categories conveying the data.



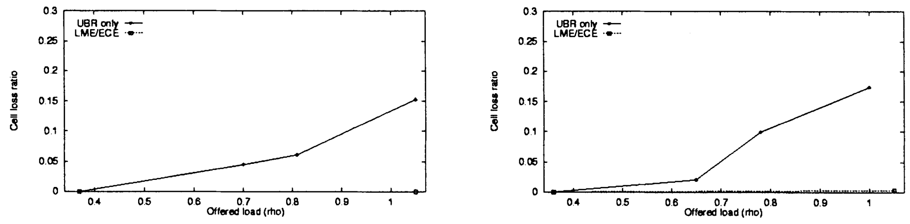
(a) Parallel code: *PDE1*



(b) Parallel code: *PDE2*

Figure 7 Latency measurements.

According to Figure 7, our mechanism achieves equivalent performance as that obtained by relying on an ABR service all the time. However, to assess the actual advantages achieved by our mechanism we have to consider other facts, such as the effective utilization of the ABR service and the bandwidth consumption. The relative performance of the measured approaches depends on the particular characteristics of the communications in each application —traffic from *PDE1* is much more bursty than traffic from *PDE2*, as stated earlier. Nevertheless, in the next subsection it is observed that the ABR service is used only by the 30%–70% messages, depending on the application and the network load. Therefore, in addition to equivalent performance, great efficiency in resource exploitation may be achieved.



(a) Parallel code: *PDE1* (b) Parallel code: *PDE2*

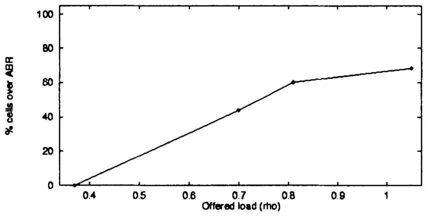
Figure 8 Cell loss ratio experienced by the application.

In order to assess the relationship between the performances achieved by both the original UBR-only mechanism and the ECE and the need of retransmissions, we have measured the experienced cell loss ratio with these configurations. The results in Figure 8 confirm that retransmissions are a major cause of latency in ATM-based parallel computing environments, as shown by the close relationship between the 'UBR only' curves in Figures 7 and 8, and also that our proposal of ECE succeeds in reducing the amount of required retransmissions, which is characterized in Figure 8 by a cell loss ratio close to zero in the 'LME/ECE' case.

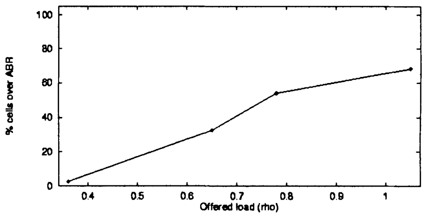
Due to the random component of the background traffic, several repetitions of the latency measurements have been performed. When considering a confidence level of 90%, the maximum radius for the confidence interval is 14% of the mean value in the worst case, which indicates a clear difference between UBR-only results and the rest.

4.3 ABR service utilization

We consider the fraction of the cells generated by a parallel task that used the ABR service as a measure of the utilization of this service. As the ABR service requires more resources from the network (a flow control mechanism, some kind of priority, etc.) than the UBR service (which just takes advantage of the bandwidth not consumed by the other service categories, so no particular resources are allocated for it), the cost of information sent through ABR is also higher.



(a) Parallel code: *PDE1*



(b) Parallel code: *PDE2*

Figure 9 ABR service utilization measurements.

Figure 9 displays the results of this measurement. *PDE1* and *PDE2* exhibit different behavior, as expected for the different characteristics of communications. The following observations can be extracted:

- In *PDE1*, the proposed mechanism for the ECE achieves 40% utilization for $\rho = 0.7$ and 70% for $\rho = 1$. These results show a highly cost-effective service achievable by our mechanism. Thus, parallel applications whose communications follow a similar appearance as those of *PDE1* can obtain a performance equivalent to that of the plain ABR service but with a higher efficiency in resource usage.
- In *PDE2*, the ECE achieves a slightly higher utilization of the ABR service —40% for $\rho = 0.65$, and 70% for $\rho = 1$. In this case, the service remains cost-effective —although slightly less than *PDE1*. The utilization of the ABR service is much less dependent on the application, as opposed to *PDE1*. Thus, applications whose communication pattern is similar to that of *PDE2* can equally achieve cost-effective communications.

In order to realize the effective cost of our mechanism, we should take into account the cost of the VBR service conveying the feedback information. As illustrated below, its performance depends on the network load as well, so we can lose some of the advantage in cost-effectiveness, specially in a highly loaded network.

4.4 Bandwidth consumption

As explained above, our mechanism allows to obtain equivalent performance as that achieved by using exclusively an ABR service, with a fairly low utilization of the ABR service. However, these features are not for free. We have seen that feedback information uses a VBR service, whose cost is higher than the ABR service.

Table 3 Average bandwidth consumption experienced by *PDE1* (Kb/s).

ρ	Service	UBR only	ABR only	LME/ECE
0.7	UBR	278.8	-	154.0
	ABR	-	275.0	122.0
	Total	278.8	275.0	276.0
	VBR	-	-	5.0
1.0	UBR	308.9	-	88.9
	ABR	-	275.0	190.2
	Total	308.9	275.0	279.1
	VBR	-	-	5.1

Table 4 Average bandwidth consumption experienced by *PDE2* (Kb/s).

ρ	Service	UBR only	ABR only	LME/ECE
0.65	UBR	285.9	-	216.0
	ABR	-	289.0	107.6
	Total	285.9.8	289.0	323.6
	VBR	-	-	7.1
1.0	UBR	338.9	-	103.6
	ABR	-	289.0	224.5
	Total	338.9	289.0	328.1
	VBR	-	-	6.0

Tables 3 and 4 reflect the bandwidth consumed in both *PDE1* and *PDE2*, considered as the total amount of bits transmitted along the execution period, by the services carrying the actual data for different two network loads in each case. The results show that, in both cases, the total consumed bandwidth is slightly higher with our mechanism than with the use of ABR only, and the difference is lower in *PDE1*. Using UBR only, as expected, yields the highest consumption due to the amount of retransmitted cells, except for *PDE2* when $\rho = 0.65$ where the cell loss ratio is not high enough for the rest of mechanisms to become advantageous. Another observation from Tables 3 and 4 is that the fraction of bandwidth spent by the ABR service is closely related to the ABR service utilization displayed in Figure 9.

Regarding the bandwidth spent by the VBR service, we recall that the VBR service conveys the STAT frames, which are periodically generated upon receipt of a POLL frame, as well as USTAT and LSTAT frames which are generated asynchronously. Thus, as expected, the spent bandwidth strongly depends on the cell loss ratio, which in turn is related to ρ. In particular, the higher the background load, the lower the consumed bandwidth, due to the increased length of high-latency periods. Note that the significance of the bandwidth consumed by the VBR service is lower than the impact of the ABR service —it is equivalent to 3%–7% of the bandwidth consumption from ABR. Thus, the total cost for the evaluated approach remains advantageous.

5 CONCLUSIONS

In this paper, we have described and evaluated a mechanism to integrate communications generated by parallel computing applications in a private virtual network environment based on ATM. This mechanism has been designed to enhance the operation of a novel, specific AAL for parallel computing that was suggested in a previous work, which is based on a modified version of SSCOP. The mechanism presented in this work exploits the service categories provided by ATM. Typically, data applications use an ABR service to reduce the occurrence of cell loss, but the use of a UBR service when the network is unloaded can lead to similar performance. Thus, our mechanism for supporting parallel computing applications uses UBR as the basic transfer service but, when latency experiences a significant increase, an ABR service is introduced. By means of this operation, we achieve low latency in communications and a cost-effective service.

To evaluate the performance of our mechanism, we have undertaken a number of simulation-based experiments. In particular, we have measured the end-to-end latency and cell loss ratio experienced by communications, the utilization of the ABR service category, and the bandwidth consumption, particularly of the VBR service category. In view of the results yielded by these measurements, we observe that (1) the latency achieved by our mechanism is equivalent to the latency experienced when conveying all communication through ABR-based connections; (2) as in the worst case only 70% of cells use the ABR service category, the cost of communications with our mechanism is much lower than the cost inherent to the full use of ABR-based connections; (3) the LME succeeds in determining the high-latency periods, since our mechanism has been able to avoid most of cell loss; and (4) the bandwidth consumption is moderate and the requirements for the VBR service category are sufficiently low, so the cost of communications is not significantly affected. As a summary, our mechanism allows for parallel computing applications that execute for a significantly long period to achieve cost-effective performance.

As introduced earlier, the mechanisms suggested in this work implement only the data transfer part of ATM-based parallel computing environments. Given that we want these platforms to extend beyond the local area, the mechanisms to build and manage parallel computing environments should be defined. In particular, these mechanisms should include a user interface in order to facilitate the platform setup, as well as intelligent load balancing algorithms so that optimal performance can be achieved at each time according to the available resources. For longer term research, we believe that applications other than parallel computing may also benefit from similar mechanisms and architectures, and therefore these can be adapted in order to advance in the integration of services.

6 ACKNOWLEDGEMENTS

This work has been supported by CICYT (Spanish Education Ministry) under contract TIC97–1054–CO3–03.

REFERENCES

[1] ITU-T, Recommendation I.121. *Broadband Aspects of ISDN.* Geneva, April 1991.

[2] T. E. Anderson, D. E. Culler, D. A. Patterson, et al. A Case for NOW (Networks of Workstations). *IEEE Micro*, 15(1):54–64, February 1995.

[3] K. Castagnera et al. NAS Experiences with a Prototype Cluster of Workstations. In *Proceedings of Supercomputing'94*, pages 410–419, 1994.

[4] ATM Forum Technical Committee. *Traffic Management Specification, Version 4.0.* Document ATM_Forum/95-0013R10, February 1996.

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture. A Quantitative Approach.* Morgan Kaufmann, 2nd edition, 1996.

[6] J. Solé-Pareta and J. Vila-Sallent. Network-Based Parallel Computing over ATM Using Improved SSCOP Protocol. *Computer Communications*, 19(11):915–926, September 1996.

[7] ITU-T, Draft Recommendation Q.2110. *B-ISDN ATM Adaptation Layer - Service Specific Connection Oriented Protocol (SSCOP).* Geneva, March 1994.

[8] A. Geist et al. *PVM 3 Users' Guide and Reference Manual.* Oak Ridge National Laboratory, 1994.

[9] C. A. Addison, V. S. Getov, A. J. G. Hey, R. W. Hockney, and I. C. Wolton. The GENESIS Distributed-Memory Benchmarks. *Advances in Parallel Computing*, 8 (Computer Benchmarks):257–271, 1991.

[10] J. Solé-Pareta and J. Domingo-Pascual. Burstiness Characterization of ATM Cell Streams. *Computer Networks and ISDN Systems*, 26(11):1351–1363, August 1994.

[11] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan. *The ERICA Switch Algorithm: A Complete Description.* ATM Forum, Contribution ATM_Forum/96-1172, August 1996.

[12] R. Jain, S. Kalyanaraman, R. Viswanathan, and R. Goyal. *A Sample Switch Algorithm.* ATM Forum, Contribution ATM_Forum/95-0178R1, February 1995.

7 BIOBGRAPHIES

Joan Vila-Sallent received his Master's degree and his Ph.D. in Computer Science in 1994 and 1997 respectively, both from the Universitat Politècnica de Catalunya (UPC). After reciving the Ph.D. he joined the Advanced Broadband Communications laboratory (CCABA) of the UPC. Currently he is doing research tasks for this laboratory in R&D projects. Joan Vila-Sallent is member of the IEEE.

Josep Solé-Pareta received his Master's degree in Telecommunication Engineering in 1984, and his Ph.D. in Computer Science in 1991, both from the Universitat Politècnica de Catalunya (UPC). In 1984 he joined the Computer Architecture Department of the UPC. Since 1992 he is an Associate Professor with this department. His currently research interests are in ATM Networks, IP over ATM and Optical Packet Networks, with emphasis on traffic engineering, traffic characterization and traffic management. Josep Solé-Pareta is member of the IEEE and the ACM (Sigcomm).