11

Protocol support for optimized, context sensitive request/response communication over connection oriented networks

Sascha Kümmel, Tino Hutschenreuther
Dresden University of Technology Department of Computer Science
01062 Dresden Germany
kuemmel / tino @ ibdr.inf.tu-dresden.de

Abstract

This paper describes a research project on specification and implementation of a fast, reliable and context sensitive transfer system. The design considers the behavior of various connection oriented networks and the requirements of request/response based communication schemes. First we describe the current situation. This is based on recent measurements and experiences with remote procedure call (RPC) over ATM and GSM networks. The major basis is the RPC of the OSF Distributed Computing Environment. Furthermore, we discuss different approaches and existing solutions including VMTP, IIOP, RMI, IPv6 and XTP.

We then present a protocol architecture as an alternative new integrated and enhanced approach. This includes the support of QoS and the integration of the major communication layers in order to enable tuning and optimization. A discussion of the integration of multimedia communication into the system follows. This is based on experiments with video transmission over DCE pipe mechanisms. We outline the major mechanisms and algorithms. An important design goal is to consider the impact of the implementation and of the real world environment onto the protocol specification. Based on investigations of different network layers we describe some of the possible and planned performance parameters of our system. We finish with a discussion of the planned integration of our protocol suite with various existing distributed platforms. This will be possible without major changes in the systems and without any changes at the application level.

Keywords

RPC, distributed systems, high speed protocol, connection oriented networks, QoS

1 INTRODUCTION

Today's networks have a great diversity of characteristics. On one hand, „traditional“ shared media, like Ethernet, can be found in most environments, while on the other hand, line based media are becoming increasingly important due to their high bandwidth and their ability to use a connection exclusively with a reserved or fixed bandwidth.

High bandwidth becomes more and more important for applications dealing with multimedia transfer, database access and supercomputing. The exclusive use of the connection also implies low latency when no shared medium is involved. This is also necessary for efficient distributed computing like distributed file and database services, CSCW applications, workflow management, etc. But high bandwidth and fast transfer of data are not the only keys to better communication performance at the application level.

To be more precise, the way data is processed between the media and the application level is the most important factor for achieving better results at the application level. Existing protocols like TCP/IP were designed for networks, which are characterized by a shared medium, low bandwidth and high error rates. Many features of these protocols turn out to be redundant due to features provided by the lower-level network interface. ATM, for example, provides specific support for connection management, flow control, congestion avoidance, and segmentation, reassembling and routing making similar TCP/IP functionality redundant.

The work we describe in this paper focuses on designing a new protocol stack for evolving technologies, considering the provided features of line based media like ATM, ISDN and GSM. The design goals are to improve the communication behavior for RPC, and to increase the efficiency and performance of the data transfer.

For that reason we do not consider OSI layering for our approach. The work we present refers to a distinct architecture and considers network crossing or shared media not yet.

2 FOUNDATIONS

2.1 Distributed platforms and new network technologies

Major distributed platforms are OSF DCE [OSF94] and OMG CORBA [OMG95]. These platforms enable and facilitate the deployment of distributed applications and of additional middleware such as transaction monitors. Typical services offered are remote client/server communication, directory services, security services, and distributed file services. Communication in DCE is based on the RPC paradigm while CORBA uses remote object invocations. Although these two solutions are rather different from their functionality point of view (for example, dynamic invocation interface and interface inheritance are only used in CORBA), they are similar concerning their basic communication paradigm, i.e. a request-response protocol between a client and a server. For our performance investigations, we focus on DCE RPC in this paper while the general findings will also be applicable to other environments such as CORBA, Sun ONC+, ANSAware, etc. With the rapid growth of the Internet, the idea of distributing applications and data across a world wide network became much more relevant. JAVA [KRAM96] should be the basic technology for that. The Hypertext Transport Protocol (HTTP) will be used for distributing application partitions written in JAVA. In addition also a distributed object model was introduced. For communication between JAVA applets (objects) across platforms Remote Method Invocation (RMI) was developed [RMI96]. However, HTTP and RMI are also based on a request-response communication paradigm and are executed over traditional protocol stacks. We investigated also the behavior of these technologies regarding the network related performance questions. So we can say that the mechanisms of our protocol architecture are applicable to HTTP and RMI, too.

In addition to other conceptual and performance-oriented studies of DCE such as [RASC93], we especially consider the effects of different network technologies. In particular, high performance networks are emerging with the rapidly proceeding standardization of ATM by the ATM Forum and the ITU-T [I.211]. Currently, ATM UNI 4.0 (User-Network-Interface), PNNI Phase 1 (Private Network Network Interface) and LANE 2.0 (LAN Emulation) is available. Based on ATM, a physical peer-to-peer throughput of 155 Mbit/s and more is possible. Even with transport-level protocols such as TCP/IP with recent IPng (next generation) extensions [BRMA95], about 130 Mbit/s can still be reached using optimal protocol parameters. In addition, the resource reservation protocol RSVP [ZDES93] enables guaranteed resource availability in order to guarantee a maximum throughput and a minimum delay. However, traditional client/server applications can hardly exploit these new performance opportunities if they are based on conventional RPC protocols (see also 3.1). In the following section we describe briefly the basics of the considered network technologies and the underlying network protocols used by typical request-response protocols.

2.2 Networking Background

The Asynchronous Transfer Mode (ATM) is a method of the international standard of Broadband ISDN (BISDN). An ATM network consists of a star like topology. Each computer is connected via a direct link to a switch. A connection must be established between any pair of hosts before they can communicate. Therefore, ATM provides a connection oriented service and the end system uses the physical link to the switch exclusively. The resulting delay consists of the signal speed in the medium, processing time in the switches and any potential blocking. Also the network adapters and the given infrastructure introduce delay.

ATM cells have a fixed length of 53 bytes. Five bytes of these are the ATM header, which is used for path description and connection identification. Also so called ATM Adaption Layers (AALs) for more application specific data flows are defined. The AAL5 will be used for data transmission applications. This AAL provides the transfer of up to 65.535 bytes of user data. The data is protected by a CRC checksum. No retransmission mechanism is defined. In many implementations (network interface cards) the handling of AAL5 will be performed by the hardware itself. Important characteristics of ATM with AAL5 in this context are (1) the packets are never out of order, (2) a checksum generation and check will be performed by the underlying network layers, (3) the connection is exclusive for one pair of hosts, (4) the transmission error rate is really low and (5) all problems of routing, especially QoS based routing, will be solved by the ATM network (see [ATM96a], [ATM96b]).

The second medium that has similar characteristics is ISDN (Integrated Services Digital Network). This is a standard for telecommunication, especially for the subscriber loop, based on digital transmission over traditional 2-line telephone connections. The transmission will be realized as a bit stream between sender and receiver. The European standard ISDN connection has two 64 kbit channels and 1 channel (16 kbit) for signaling. ISDN provides an unsecured connection and has no error handling mechanisms. On top of ISDN different protocols for secure data transmission can be used. We assume the use of the Point-to-Point Protocol (PPP) in our supported environments. This protocol additionally realizes the structuring of the bit stream into data blocks and the calculation of checksums. The useful characteristics of ISDN using PPP for us are similar to the points (1), (2), (3) and (4) of ATM.

A third standard we refer to is GSM [RAHN93]. It is used for wireless wide area networks and is designed for telephone connections by using digital data transmission. GSM based telephone systems are widely used in Europe and Asia. Based on the digital wireless links the transmission of data to mobile computers is also possible. Because GSM is based on a 'not fully secured' transmission procedure for error correction, the Radio Link Protocol (RLP) procedure is needed. It is optimized for GSM and provides a way to enable reliable connections. RLP ensures that there are really fewer errors when the data leaves the mobile switch.* The reached data rate is 9600 bit/s which is really small for distributed applications. Furthermore, the delay of the transmission between two stations connected via GSM is relative high (more than 500 ms in one direction). Disconnection can also occur in such an environment. As with ISDN, we assume the use of the PPP protocol on top of RLP. That's why, except these problems, the important characteristics are similar to ISDN and GSM discussed above.

All presented media types provide a line based, connection oriented service for communication. They use their own mechanisms for error-detection /-handling and provide, because of the line based character of the network, an in-order delivery of the packets. These mechanisms can be used to design a light weight protocol stack without introducing too much overhead.

On top of these transmission techniques network protocols are needed to allow the exchange of data in computer networks. In today's systems this is a layered stack. Related to a distributed system this means a request-response protocol layered over a transport protocol. The most common protocol stack in the Internet world is TCP/IP. In the following section we briefly describe the behavior of the DCE RPC in such an environment. An extended version of this study could be found in [KUES96].

3 BEHAVIOR OF A CURRENT REQUEST/RESPONSE PROTOCOL

3.1 Performance Results

We investigated RPC in broadband networks using our local ATM environment. Within our experiments, we first evaluated the performance of TCP/IP over ATM between two DEC Alpha 3000 AXP 700. Both the message sizes transferred via TCP/IP and the buffer sizes at the receiver's side were varied; initial experiments have already shown a strong influence of both parameters. The maximum achieved throughput was 16,875 KByte/s (135 Mbit/s) with optimal parameter values and ***stream-based(!)*** transmission. So the experiment has shown that the bandwidth of ATM can really be exploited based on adequate protocol parameter settings and sufficient CPU capacity. With a buffer size of less than 64 Kbytes, however, performance dropped significantly, for example down to values between 7,500 kByte/s and

* Error rate with RLP is lower than 10^{-6} vs. 10^{-3} without.

11,250 kByte/s (60 and 90 Mbit/s) for buffers of 32 Kbytes. Of course, very small messages also resulted in very poor performance.

Based on these initial experiments, we compared the performance of RPC over ATM versus Ethernet. The RPC protocol implementation itself was not modified; that is, the original buffer and message size settings were not adapted. The results are summarized in figure 1 [KUES96].

The maximum throughput, even for large parameters, was 4,875 kByte/s (39 Mbit/s as opposed to about 8 Mbit/s over Ethernet). With a special mass data transfer mechanism of DCE RPC (so-called RPC pipes), a slightly better result of 5,625 kByte/s (45 Mbit/s) was achieved. Nevertheless, the results were much worse than with pure, stream-based TCP/IP over ATM where 16,875 kByte/s were reached. The comparison of the roundtrip time of small calls established the following; **the time needed for a call without parameters was nearly the same for Ethernet and ATM (900 to 1000 μs)!**

Two reasons are responsible for these rather unsatisfactory results. First, the client is delayed until explicit acknowledgments are received. Even for large parameters, the time to transfer the acknowledgments and to schedule the client process upon receipt is comparably large. Secondly, the RPC protocol does not fully exploit the MTU (maximum transfer unit) size of TCP/IP due to its internal implementation. Moreover, when a multiple of the MTU size is reached by the parameter size, a performance decrease is observed due to partially filled MTUs. As shown in figure 1, data is blocked to form MTUs of 4096 bytes, although higher MTU sizes would basically be possible as discussed above. Finally, experiments with local RPC communication based on a loopback mechanism are shown in the diagram. They illustrate that even in the local case (i.e. without ATM communication), performance is not significantly better. This also confirms that protocol processing, RPC acknowledgments and process scheduling cause a major impact (rather than the actual network transfer via ATM). Changes in RPC protocols and extended mechanisms in the protocol engines are required for a better and optimized usage of the low delay and high throughput characteristics of the underlying physical network.

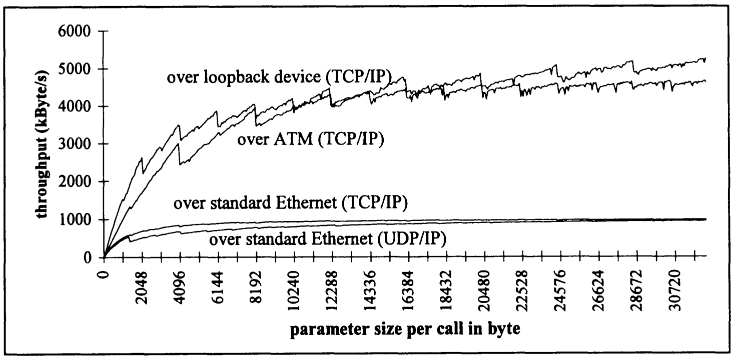

Figure 1: RPC over ATM and Ethernet: Comparison

The RPC can also be used in Mobile Environments. We were mainly interested in the RPC throughput, and also in the effects of using asynchronous calls based on threads. For our experiments we used e-plus in Germany, an DCS-1800 GSM based system.

However, it became obvious that the number of calls per second and therefore the total throughput can drastically be improved with parallel threads. For example, with 3 threads, we reached a mean throughput of about 700 byte/s unlike less than 350 byte/s with only one thread. The main explanation for these results is that the delay times while waiting for responses can be used by the client for issuing other calls. Like in conventional wide area networks, several parallel calls are required in order to really exploit the actual system capacity.

The mobile computing experiments have shown that RPC communication suffers from long delays, low throughput and potential disconnection in such environments. Threads are important as a basis for asynchronous RPCs in order to exploit phases of long delay in a productive way. We used this for instance in [KSSZ96].

3.2 Some problems with the current Transport Protocols

The implementations of current transport protocols lead to performance limitations [PORT91]. Most implementations are tied heavily into host operating systems. The heavy usage of timers, interrupts, and memory read/writes tax the main CPU and degrades the performance of the protocol. Functions such as buffer management, bus transfer time, and bus contention alone cause several processing overhead for current protocol implementations. Some of these problems may be solved with better implementations of existing protocols. However, because of the design of current protocols there are some limitations to the implementation improvements. One key design goal is to separate the protocol processing from the operating system as much as possible.

Current transport protocols (e.g., TCP or OSI TP4) were designed with different restraints and requirements compared to future high-speed networks. Older protocols were designed to minimize the number of bits transmitted to reduce insertion time. This led to packet formats with bit-packed architectures, requiring extensive decoding. Another problem with these protocols is packet field limitations. For example, the number of bits provided for window mechanisms must be increased to supply window sizes large enough to be effective at high-data rates.

The control algorithms supplied by current transport protocols will be strained by high-speed networks. For example, current protocols like TCP and ISO TP4 effectively use Go-back-N windowing schemes for error recovery. This type of algorithm will severely degrade throughput and add unnecessary network congestion. Finally, existing protocols may not be flexible enough for high-speed applications and networks.

In the late eighties, a lot of effort was applied to the development of new protocols for faster transmission of data. Also the consideration of different services and Quality of Service (QoS) parameters have played an important role. At this time different transport protocol approaches like VMTP[CHER88], XTP [XTP95] and so on were designed to overcome the lack of functionality in existing protocol stacks like TCP/IP [DOER90]. They took the evolving technologies into consideration and were adapted to special fields of communication behavior such as bulk data transfer or the support of short transaction messages. Also TCP/IP was improved over the years to adapt it to the changing demands of networking. So the window size was increased to 64 k and further adaptations were introduced to become more efficient [JABB92].

In spite of the new features and improvements, it must be stated that the protocols are not really suited for communication over line based media. The discussed protocols offer better performance due to their improvements, but they don't consider the type of media below. So several tasks are executed more than once, although the underlying medium provides this service.

There are also other projects and research groups related to efficient data transmission. For instance [CLAT90] and [EWLB94]. Also the improvement of the behavior of distributed systems is in the scope of some investigation [LIHD95], [THEK93]. These works also prefer an integrated layer approach and implement applications directly on top of AAL5. The connection-less protocol engine of the DCE-RPC was also improved with the introduction of private client sockets, multi-buffer fragments and sending message vectors, but further optimizations have not been planned up to now. Another way of improving DCE performance could be found in [GKLM93]. The researchers of Cray used reduction of data copies and MTU discovery to achieve a better performance of the original DCE kernel RPC implementation. However, the overall system performance could not reach the theoretical limit.

The widespread use of mobile computers has made the use of distributed systems in such an environment interesting. Such approaches and arising problems are described in [IMBA94] and [SCHK95]. Also the improvement of the transmission performance in wireless networks is presented in [BAAK95].

Most of the above mentioned approaches provide only solutions on specific parts of the communication behavior and were not introduced into a larger distributed environment. We have considered the above mentioned approaches and combine these in our approach, as described below.

4 ADVANCED PROTOCOL SUPPORT

4.1 Basic Concepts

Based on our investigations and on the analysis of existing approaches and protocols we started a development project for a new optimized request-response protocol. Beside the investigation of new mechanisms we also focus on the reuse of a lot of existing concepts from different fields (see above) with a special interest in the request-response paradigm. Furthermore we consider the impact of the implementation and of the real world environment onto the protocol specification. The major goal is to provide advanced efficient protocol support for optimized distributed systems.

The main component of our design is an advanced RPC protocol engine (see fig. 2). It is based on the following key ideas: (1) As opposed to TCP/IP and similar protocols, it has knowledge about the request-response behavior of the higher-level interactions and can use that for optimization. (2) Moreover, it also has knowledge of the specific underlying network, i.e. there are specific instances of the protocol engine for pure ATM, ISDN and GSM point to point connections. That means, additional protocol mechanisms such as packet reordering, routing or flow control can be omitted if already supported by the underlying network (as it is the case with ATM, for example).

The protocol engine can optionally make use of a traffic shaping component, for example for handling different priorities of traffic streams. Access from the RPC runtime system to the new components is provided via a direct programming interface or - alternatively - via WinSocket 2.0. Compatibility with existing RPC protocols is guaranteed by the direct connector component, a very simple additional RPC protocol engine. As another optimization, we include a direct memory mapping of marshalled RPC parameter data if supported by the underlying hardware and operating system (for example, under Windows NT). This is implemented by the advanced memory manager and the RPC kernel memory management.

Finally, existing protocol stacks (for example, RPC via TCP/IP) can coexist with this new solution to simplify smooth migration of existing applications. It is also possible to integrate emerging reservation protocols such as RSVP in order to guarantee resource availability. This way, resources (buffers and processing capacity) can be reserved within the RPC protocol engine during connection setup or binding.

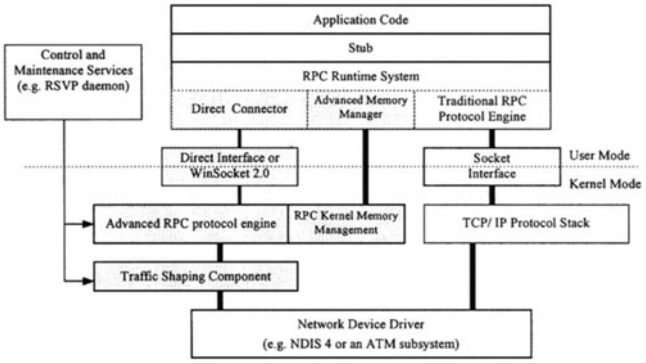
Application Code
Stub
Control and Maintenance Services (e.g. RSVP daemon)
RPC Runtime System
Direct Connector
Advanced Memory Manager
Traditional RPC Protocol Engine
Direct Interface or WinSocket 2.0
Socket Interface
User Mode
Kernel Mode
Advanced RPC protocol engine
RPC Kernel Memory Management
TCP/ IP Protocol Stack
Traffic Shaping Component
Network Device Driver (e.g. NDIS 4 or an ATM subsystem)

Figure 2: Architecture of new communication support platform

The main points of optimization are:

- With respect to real existing operating systems we favor a kernel based implementation. Interactions with the user space application must be minimized as far as possible.
- Copy processes are a potential bottleneck and must be minimized. In this context also other data touching processes are important. The impact of CPU bus, cache memory, memory subsystem and used peripheral bus systems must be considered.
- Interactions over the network between the communication partners must be minimized by using the characteristics of the underlying network.
- The amount of data to be transferred should be reduced as far as possible (header compression).

Furthermore we plan to support the observation of QoS parameters and the transfer of multimedia and mass data. In the following section we describe some of our extensions in more detail.

4.2 Memory management

One example of a performance bottleneck is the large amount of copy operations required. There are different extensions for existing implementations like in [DRUP93] ,[JABB92], [EWLB94]. Another interesting approach is Integrated Layer Processing (ILP) [CLAT90]. The main benefit of ILP is minimizing memory access. This will be achieved by processing different data manipulations, like marshalling, encryption and checksum calculation in one loop. Because all these operations will be processed over small chunks, the data could be held in CPU registers or cache memory. Our approach is similar to ILP and ALF (Application Layer Framing) ideas, but in our project we focus on the optimization below the marshalling level. We also prefer a kernel based implementation to achieve maximum possible throughput and to save CPU time for the application processing itself. In the following section we describe how our design minimizes copy operations.

In a DCE stub the marshalling procedure calls a memory allocation function to get a buffer for the marshalling result data. This buffer will be allocated in the user space memory in the traditional implementation. Then the marshalling routine puts the call parameters into this buffer (1. copy operation). After this, the send procedure is called and the data is fit into protocol dependent TSDUs (2. copy operation). The TSDUs are handed over to the kernel part of the protocol implementation, for instance to UDP by using a socket interface. In most implementations, a copy of data between user and kernel space is needed (3. copy). Now UDP performs a checksum calculation (4. not a copy but touching all data) and hands over the data to IP. Due to the *mbufs* no copy operations are needed here. The fifth copy operation is performed by moving the buffer from IP to the network card through the device driver.

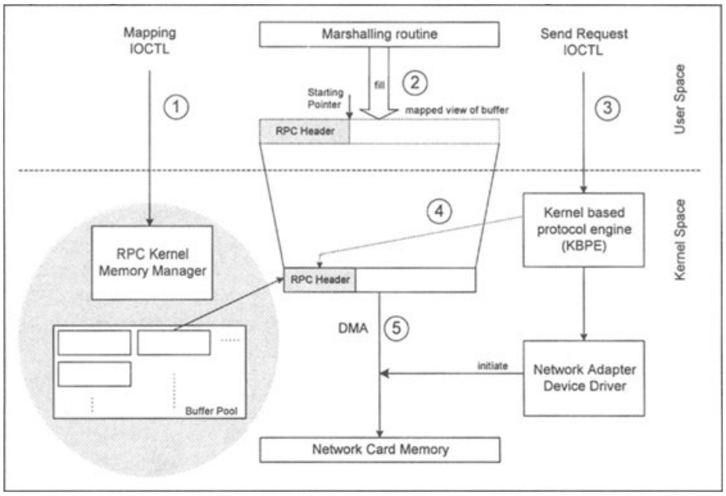

Figure 3: 2-copy memory handling scheme

In our architecture we use the schema shown in fig. 3. The kernel based protocol engine (KBPE) provides a pool of send buffers. If a marshalling procedure needs a buffer the advanced stub memory manager sends an I/O control to the RPC Kernel memory management and a send buffer will be mapped into the user space (1). Now the call parameters are put into this buffer (2). The start address for this will not be at the beginning of the buffer but rather at an offset for RPC header information. If the marshalling process is finished, the runtime must only call the KBPE via an IOCTL (3). No copy operation is needed. After preparing the RPC header information (4) the whole block will be moved to the network card. A Direct Memory Access mechanism will be used for that, in general (5). A checksum calculation is not applicable because the underlying network will perform that. In the case of ATM or 100VG this will be performed in real-time by hardware on the adapter itself. ATM also allows the use of large TSDUs (up to 64 kByte). That's why one TSDU is enough for most kinds of requests. In the term of ALF the Application Data Unit (ADU) fits into the TSDU [CLAT90]. Further segmentation and reassembling isn't necessary.

As we have shown, only two copy operations against five of the traditional version will be performed. There is also a solution for minimizing to only one operation. But this requires special hardware with large onboard data buffers. Unfortunately, the described mechanism is not meaningful without any critical points. If the request is really small, the overhead of the memory management IOCTL operations is too high. To protect the system against this new possible bottleneck we also use alternative communication mechanisms depending on the amount of data transferred.

4.3 Minimizing of transmission interactions

A further optimization is based on the reduction of interactions between the involved hosts. This is important for wireless communication lines. The delay for message delivery is really high in this case and every unnecessary interaction will drop the throughput dramatically. But also in high speed networks like ATM the reduction of interactions will increase the performance significantly. According to our measurements, a simple request/response at the kernel level needs approximately 200 µs. This includes the simple transmission of one ATM cell from one node to another and back again without any changes. This time can only be reached on fast platforms!

Fig. 4 shows the standard process of a non-idempotent DCE call [OSF93]. Non-idempotent means that a call should and must be executed only once. Therefore a server/client verification for each call is needed (conversation manager in fig. 4). Also the reception of the response by the client must be acknowledged. In the case of a slow call much more interactions will be performed. This is to verify that client and server are still alive and the server is working on the request. Because a data connection with a higher error rate could be used, it should be possible that much more interactions for realizing a reliable communication are required. Processing this scheme on our development machines required approximately 600 µs over ATM, but only if the protocol engine is realized in the kernel. A user space implementation approach, like the UDP connectionless protocol engine of the DCE RPC, requires more than 1000 µs.[†]

The reduction of these interactions is based on another view of the client/server relationship. The RPC itself is a kind of connection less communication. But **one** non-idempotent call needs a connection oriented processing schema. That's why something like the conversation phase in fig. 4 is required.

[†] All these measurements are performed on Digital DECStation 3000/700 connected via ATM 155 Mbit/s MMF.

In our approach, which is focused on connection oriented networks, we need to establish a virtual connection between the client and server (machine). This connection will be reused by different following calls. Furthermore, we assume that in most cases more than one call will be performed between client and server during its lifetime. We verify the personality of client end server implicitly in every call based on an ID. This ID will be negotiated during the connection establishing period. Based on this, two interactions per call could be saved; the whole conversation manager phase (see fig.4).

Also the acknowledgment to the response message could be saved. This will be realized based on a machine to machine connection control mechanism. The description of the whole mechanism is beyond the scope of this paper. In brief, the protocol engine checks the connections between the computers if established. No additional transmission is required in general. The monitoring of the whole traffic of **all** client/server relations between the two engines will be enough. Only if there was no interaction for a long time a ping procedure will be started. However, we didn't monitor the separate client/server pair connection. To protect every single call context against the failure of client or server we check the existence of them by using local communication mechanisms. The detection of client or server shutdown produces a signal to the protocol engine of the another partner. This leads to cleaning up the connection context. Such a connection context is only an internal context of the protocol engine and it's not similar to the *context handle* of DCE!

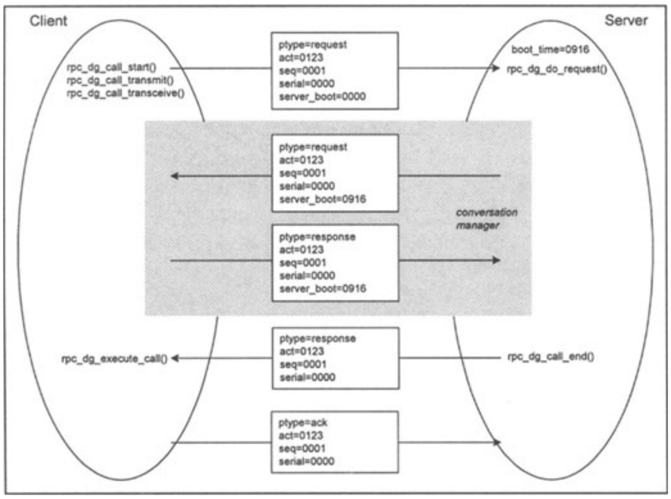

Figure 4: A Non-Idempotent Call [OSF93]

In most cases our system will reduce the interaction between client and server machine to only two messages exchanged via the network per non-idempotent RPC. By using an ATM network this leads also to only two AAL5 PDUs in general.

4.4 Reduction of data amount

The amount of transferred data has a significant influence on the performance of mobile distributed systems. Useless overhead is more disturbing here. Also an ATM based system is sensitive to data overhead. Beside the problem of transmission, the problem of processing also must be considered. That means all information in a header must be touched, for instance. The size of the header of a DCE RPC is 80 bytes [OSF93]. Furthermore, 28 bytes for underlying IP and UDP header must be added. If the parameter size is relatively small, this leads to an overhead of 50 percent and more. We assume that there are more than one call per client/server pair during their lifetime. This allows us to reduce the RPC-header information to 24 (28) bytes beginning with the second call. Also the header of IP and UDP are not needed. So the header could shrink from 108 to 24 bytes. How does this reduction work? The original header does not include unnecessary information. But some of the information has a relatively static nature (relative in this context means static for the lifetime of a client/server pair), for instance, the interface and object UUIDs, boot time of the server, RPC version, etc.

As described in section 4.3 we introduce a short negotiation phase if the connection between client and server will be established. In this phase all static parameters of such a logical connection (represented by a binding handle) will be introduced to both sides. The resulting identifier of this negotiation will be used by client and server to represent the original static data.

The investigation of another, more complex mechanism with renegotiation phases is in progress now. With this algorithm a reduction of the header to 8 or 12 bytes is expected. This mechanism provides much more processing overhead and is only useful in small bandwidth environments.

4.5 Support of QoS

The handling of QoS parameters isn't supported by many distributed platforms. There are a lot of papers describing QoS problems. Most of them providing a solution for operating systems or network transport but only few consider the „traditional" client-server computing. However, the guarantee of QoS is also needed in distributed systems. For instance the remote access to a real-time database requires an adequate remote procedure call. It's not enough to guarantee the execution time of the database query. Furthermore it must be possible to guarantee the delivery of the request and the response in a determined time. The guarantee of a minimum or constant throughput could also be required in file server applications.

Delay guarantee for message delivery is possible with network techniques like ATM. We introduce an additional interface for describing and negotiating QoS parameters for a specific client-server binding in our architecture. Currently we plan to consider delay, throughput and jitter. In the current state of our system we can provide such „soft" guarantees on the transport level. It is much more difficult to realize this for the whole call. There are lot of problems concerning the existing operating systems regarding scheduling, memory management, etc. Furthermore, we couldn't calculate the execution time of the marshalling procedures especially for the analysis of complex linked data structures.

For distributed platforms we also need a guarantee for the server application execution time. But this could only be provided by real-time operating systems, real-time hardware (harddisk etc.!) and real-time application level systems like special database systems [GOPL96]. Not only the realization of QoS in such a whole system is difficult, but also the negotiation, meaning and transformation of parameters is not clear today. We plan to investigate the requirements of the mapping of user-QoS or application specific data onto abstract parameters of network, transport system, operating system and underlying applications. This is also needed to decide which parameters are necessary. Also the value range and unit of such parameters for the network and RPC layer must be made clear.

Furthermore, QoS support for multimedia transfer is also required if such a communication form will be supported. However, the question remains; Is the transfer of multimedia data possible with the mechanisms of an existing distributed platform like DCE? We think yes, but extensions are needed.

The DCE RPC provides a mechanism for mass data transfer called pipes. Three kinds of pipes exist; in pipe, out pipe and in/out pipe. Data will be transferred as so called atoms. This is an application specific data unit, for instance the size of a file block. If the application starts using a pipe it must not know how much data will be transferred. Also not all data must exist in the main memory if the pipe transfer starts. This makes the pipes a candidate for multimedia transfers too. In experiments with a video conferencing tool we used DCE pipes for the transfer of video and audio information. No significant problems occurred by using an ATM network. If there are bandwidth limitations it doesn't work as well because pipes provide a reliable transmission only.

However, the pipe mechanism must be extended. First, we plan to introduce a variable atom size depending on the size of an ADU, for instance a single compressed frame of a video stream. This makes it possible to handle the transmission in an application specific manner. Second, we need a mechanism for dropping atoms (frames) depending on the delays and regarding the requirement of continuous delivery to the receiver process (for instance display process). This will be allowed by using variable atoms. The transport system knows what the unit is that could be dropped. And third the mechanisms must be extended by specification of QoS parameters and control procedures to guarantee the negotiated QoS.

5 OUTLOOK

The operating systems we support are Windows NT and DEC UNIX. We plan to finish the determination of the exact behavior of the different communication levels, based on a simple request - response protocol. We placed the measurements directly on top of the device driver, kernel based IP and user space IP to determine the maximum accessible speed and the consumption of operating system methods, such as IO control. Furthermore, we plan to finish a first integration of our improvements into DCE-RPC mechanism in march 1997.

Our first implementation is based on the free sources of the DCE-RPC. The main part of our protocol engine is not limited to DCE because our improvements are generic for request - response communication schemes. An integration of this protocol into other systems will be possible. So an integration into ORBIX of IONA is planned by exchange parts of the transport class. Furthermore, we look into an integration of our approach into JAVA RMI. Like in the DCE implementation no changes at the user interface are made and also the usage of old binaries could be possible.

By using our protocol stack in an ATM environment, we expect a maximum time for one call of less than 500 µs. Further, the increasing of throughput up to 12 MByte/s is expected.

At the end of 1997 we plan the full integration of QoS and multimedia support into our reference implementation.

6 REFERENCES

[ATM96a]	ATM User-Network Interface Signalling Specification Version 4.0, ATM-Forum, 1996, af-sig-0061.000
[ATM96b]	PNNI Phase 1, ATM-Forum, March 1996, af-pnni-0055.000
[BAAK95]	Balakrishnan, H., Amir, E., Katz, R.H.: Improving TCP/IP Performance over Wireless Networks, Proceedings of the 1st ACM Mobicom Conference, 1995, pp. 124-31
[BRMA95]	Bradner, S.O., Mankin, A.: IPng - Internet Protocol Next Generation; Addison-Wesley, 1995
[CHER88]	Cheriton, D.R.: VMTP: Versatile Message Transaction Protocol; Protocol Specification, RFC 1045, Stanford University, 1988
[CLAT90]	Clark, D. D. and Tennenhouse, D. L.: Architectural considerations for a new generation of protocols. SIGCOMM '90, pages 200-208, Philadelphia ACM.
[DOER90]	Doeringer, W.A. et al.: A survey of Light-Weight Transport Protocols for High Speed Networks; Transactions on Communications, November 1990, pp. 2025-2039
[DRUP93]	Druschel, P., Peterson, L.: Fbufs: A High-Bandwidth Cross-Domain Transfer Facility, SOSP 1993
[EWLB94]	Edwards, A., Watson, G., Lumley, J., Banks, D., Calamvokis, C., Dalton, C.: Users-Space Protocols Deliver High Performance to Applications on a Low Cost Gb/s LAN, SIGCOMM'94
[GKLM93]	Gaffey, B., Kimlinger, P., Lord, S., Mostek,J., Reinart, J.: The Performance of OSF DCE Distributed File System (DFS) at Cray Research, Inc., http://www.cray.com/PUBLIC/product-info/sw/dce/perf.html
[GOPL96]	Goebel, V., Plagemann, T.: Data Management and QoS in Distributed Multimedia Systems - Towards an Integrated Framework, 4. Workshop on Quality of Service, IWQoS, Paris, March 1996, pp. 79-82
[I.211]	CCITT I.211: INTEGRATED SERVICES DIGITAL NETWORK (ISDN); B-ISDN SERVICE ASPECTS; Recommendation I.211
[IMBA94]	Imielinski, T., Badrinath, B.: Mobile Wireless Computing: Challenges in Data Management; Comm. ACM, Vol. 37, No. 10, Oct. 1994, pp. 18-28
[JABB92]	Jacobson, V., Braden, R., Borman, D.: TCP Extension for High Performance, RFC 1323, May 1992
[KRAM96]	Kramer; D.: The Java Platform, A White Paper; Sun Microsystems, Inc., 1996 http://java.sun.com/doc/whitePaper.Platform/CreditsPage.doc.html
[KSSZ96]	Kuemmel, S., Schill, A., Schumann, K., Ziegert, T.: An Adaptive Data Distribution System for Mobile Environments, IFIP'96 World Mobile Communications Conference, Canberra, Australia 1996
[KUES96]	Kuemmel, S., Schill, A., Volkmann, G.: RPC over Advanced Network Technologies: Evaluation and Experiences; Third International Workshop on Services in Distributed and Networked Environments (SDNE'96), Macao, June 3-4, 1996, pp. 68-75
[LIHD95]	Lin, M., Hsieh, J., Du, D.: Distributed Network Computing over Local ATM Networks, IEEE journal on selected areas in communications, vol. 13, no. 4, May 1995, pp. 733-748
[OMG95]	OMG: The Common Object Request Broker: Architecture and Spec.; Rev. 2.0, 1995
[OSF93]	Open Software Foundation: DCE RPC Internals and Data Structures; OSF, 1993
[OSF94]	Open Software Foundation: DCE 1.1 New Features; OSF, 1994
[PORT91]	La Porta, T.F., Schwartz, M.: Architectures, Features, and Implementation of High Speed Transport Protocols; IEEE Network Magazine, May 1991, pp. 14-21
[RAHN93]	Rahnema, M.: Overview of the GSM System and Protocol Architecture; IEEE Communications Magazine, April 1993, pp. 92-100
[RASC93]	Rabenseifner, R., Schuch, A.: Comparison of DCE RPC, DFN-RPC, ONC and PVM; DCE - The OSF Distributed Computing Environment, LNCS 731, Springer, 1993
[RMI96]	Java^{TM} Remote Method Invocation Specification; Sun Microsystems, Inc., 1996; http://chatsubo.javasoft.com/current/doc/rmi-spec/rmiTOC.doc.html
[ROBD96]	Roca, V., Braun, T., Diot, D.: Efficient Communication Architectures for Open Systems, Submitted to IEEE Networks journal
[SCHK95]	Schill, A., Kümmel, S.: Design and Implementation of a Support Platform for Distributed Mobile Computing; Mobile Computing Special Issue of Distributed Systems Engineering; Sept. 1995
[THEK93]	Thekkath, C.A.: Limits to Low-Latency Communication on High Speed Networks, ACM Transactions on Computer Systems; Vol.11, No.2, 1993
[XTP92]	XTP Protocol Definition Revision 3.6; Protocol Engines Incorporated, Santa Barbara, CA, 1992
[XTP95]	Xpress Transport Protocol Specification, XTP Revision 4.0; XTP-Forum 1995
[ZDES93]	Zhang, L., Deering, S., Estrin, D., Shenker, S., Zappala, D.: RSVP: A New Resource Reservation Protocol; IEEE Network, Sept. 1993, pp. 8-18