Date: 1992-06-04

Title:	A Modular and Configurable Transport System Architecture for High-Performance Processing Providing a Flexible Transport Service
Source:	Germany University of Karlsruhe, Institute of Telematics T.Braun and B.Stiller
Project:	1.06.36
Status:	Expertpaper
Reference:	ISO/IEC JTC1/SC6 N 7309

This page was intentionally left blank.

A Modular and Configurable Transport System Architecture for High-Performance Processing Providing a Flexible Transport Service

Torsten Braun, Burkhard Stiller
Institut für Telematik, Universität Karlsruhe
Zirkel 2, D-W7500 Karlsruhe 1
Phone: +49 721 / 608- [3982 / 3983] Fax: +49 721 / 606097
E-Mail: [BRAUN | STILLER] @ telematik.informatik.uni-karlsruhe.de

1. Introduction

The comprising term for lower layers 1 to 4 in the ISO/OSI Basic Reference Model has been defined as ***Transport System***. It is regarded as a whole system, which encompasses the aspects of communication without internal layering. All relevant procedures for transferring transparently data between transport service users are hidden behind the transport service interface. The standard services and protocols provided, e.g., the ISO transport service and protocol (ISO 8072/73), are not well suited for emerging new applications since quite a few functional and performance oriented facts restrict the use of these transport services and protocols. Existing service classes, as specified within the 5 different OSI transport protocol classes, do not reflect the diversity of service requirements of modern applications since one of the main purposes behind their development has been to close the gap between provided network service and the required transport service. However, today there exist only a few parameters for the selection of a transport service class in transport protocols. Especially for the steadily increasing variety of application requirements, e.g., in the area of multimedia applications, more and enhanced service parameters are needed.

To overcome these restrictions, on one hand, the need for an integrated view of transport, network, and data link protocol and service respectively is necessary, on the other hand, a good structuring of these protocols is needed. Both allow **functional enhancements** in terms of new facilities and protocol functions in lower layers as well as a direct and efficient design of the enhanced transport service and protocol and, finally, an implementation. This is possible using a **function-based approach**, where the protocol functions providing a transport service are regarded as configurable and atomic units. The additional advantage of this flexible and modular function-based approach is the configuration of protocol stacks according to application requirements, which may be specified using sets of quantitative and qualitative parameters of the transport system´s service specification.

In section 2 an overview of the transport system´s service specification with the distinguished sets of quantitative as well as qualitative parameters will be given. Furthermore, in section 3 the architecture and the necessary mechanisms for future transport protocols are presented in detail. Finally, the flexible function-based approach will be described briefly in section 4 giving some important details on the assisting tools. This paper will be concluded within section 5.

2. The Transport System´s Service Specification

Standard service specifications include like the 5 different ISO transport protocol classes different facilities for different requirements. Since these existing classes do not reflect the variety of modern application types, an enhanced transport service has to be defined, on one hand, to support all currently existing applications and, on the other hand, to provide enough flexibility to reach future application requirements. Therefore, service requirements of existing applications have been investigated in great detail to extract useful service criteria. They can be classified in two categories: qualitative and quantitative criteria /ZSTa92/. The structural approach for the transport service interface led to 5 so called transport service classes, which provide a specialized transport service, but still enough flexibility for different kinds of applications.

2.1 Service Criteria

The application basically distinguishes between ***quantitative and qualitative criteria***. The first category comprises more ***performance oriented criteria***, in parts already known as quality of service parameters, which had been enhanced (cf. section 2.1.1). These parameters can be evaluated in terms of certain measures, like bit per second, number of errors, milliseconds etc. However, the qualitative parameters specify the ***functional aspects*** of an application related to the whole transport system (cf. section 2.1.2). The request of a specific functionality of the transport system without any detailed knowledge of the interiors is possible using these parameters.

2.1.1 Quantitative Criteria

The list of possible quantitative parameters is valid for each transmission direction separately. Each value also includes a triple of ***values***: threshold, average, and useful. The threshold value defines a threshold for a given service parameter. Its semantic, maximum or minimum, is dependent on the service parameter. The second value defines the average value over a certain time interval. The third value specifies a limit for the appropriateness of a service parameter, e.g., an upper bound on bandwidth useful for the application. The following ***parameters*** describing the quantitative criteria can be distinguished.

- Throughput
- Burstiness
- Delay
- Response Time
- Rate
- Jitter
- Data Corruption Threshold
- Data Loss Threshold
- Costs

Throughput defines the well known value at transport service level, which is most commonly associated with the number of bits or packets per second. The ***burstiness*** describes the throughput variation of the sender depending on certain time intervals. ***Delay*** encompasses the time elapsed between sending a data unit from a service user until its reception by the destination service user. The typical requirement for real-time applications is the ***response time*** value, which is a two-way delay with the additional processing time in the destination node. The ***rate*** defines the rate at which consecutive data units have to be delivered from the transport system to the transport system user. For instance, this is the rate for delivering video frames. Additionally, the ***jitter*** specifies the acceptable rate variation, which is still sufficient for the service user. In this case, applications are concerned as isochronous data streams. The ***data corruption threshold*** is associated with errors inside a data unit. They can be detected using special error detection mechanisms. The tolerable amount of corrupted data by the service user depends on the application in terms of data or control units. The ***data loss threshold*** value defines the percentage of data losses acceptable at the service interface. These data units do not even arrive at the destination. Finally, the ***costs*** can be specified.

2.1.2 Qualitative Criteria

The second list comprises qualitative parameters. They are used to define the expected transport service related to session management, which is the relation between two or more data streams, stream management, which is the maintenance inside a single data stream, and data unit manipulation, which is the session independent as well as stream management independent processing of data units. All these terms are explained in section 4. The following list gives an overview of the distinguished ***parameters*** describing the qualitative criteria.

- Session Management
  - Inter-Stream Synchronization
  - Session Update
  - Multiplexing
- Stream Management
  - Ordered Delivery of Data
  - Guaranteed Group Delivery of Data
  - Error Tolerance
    - Loss of Data
    - Replication of Data
    - Corruption of Data
  - Priorities
  - Security Requirements
    - Data Security
    - Access Security
- Data Unit Manipulation
  - Maximum Data Unit Size
  - Segmented Delivery

The term session defines an association between two or more communicating applications. Since it might consist out of several data streams related to one another the service for the session management can be specified by the application (cf. section4). The ***inter-stream synchronization*** specifies the need of synchronizing different data streams belonging to one application. This service will be needed for the multi-media transfer of voice and moving images meeting the lip synchronization. The ***session update*** reflects the needs for adaptation of the transport service to changing network and application conditions and requirements respectively. The ***multiplexing*** allows for integrating separate transport data streams into a single network (media access) data stream.

Concerning a single data stream as a separable unit of flowing data at the application interface, the ***ordered delivery of data*** is an important factor. If there is the need for multicast as well as multipeer communication the ***guaranteed group delivery*** is necessary. Different semantics can be associated with this parameter, e.g., at most once, at least once, groups. Three different parameters can be assumed defining error tolerance. This is the ***loss of data***, which can be accepted up to a certain threshold defined by the quantitative parameter. The ***replication of data*** has to be avoided sometimes since applications like voice transmission do not accept duplicated data units. Finally, the ***corruption of data*** specifies the tolerance of corruptions of user data at the transport service interface. The support of urgent data is possible with different ***priorities***, this is, for instance, the normal and expedited data transfer if two priorities are supported. Security requirements depend on the application in terms of ***data security***, where data transmissions are ensured, as well as in terms of ***access security***, where the identification of the communication partner is relevant before establishing a connection.

The last two parameters in the category of data unit manipulation concern the maximum data unit size and the segmented delivery. The first shows the need of the transport system to segment data units handed from the application to the transport system due to the fact that the underlying network supports a ***maximum transfer unit size*** only. The second one reflects the possibility of omitting the reassembly of data at the destination since the receiving application do not need the same data unit sizes as handed by the sender to the transport system. This is called ***segmented delivery***.

2.2 Transport Service Classes

Based on the above mentioned service parameters and their influences on the configuration of protocol functions (cf. section 4) for the transport system 4 predefined transport service classes (TSC) can be specified. Additionally, a complete free configurable service class can be reserved. Moreover, systems are allowed to establish local service classes as subclasses of the free class. These 4 ***classes*** are distinguished by 2 important criteria: reliability and real-time.

Class I: Unreliable Real-Time
Class II: Reliable Real-Time
Class III: Unreliable Non-Real-Time
Class IV: Reliable Non-Real-Time
Class V: Free

The classes I to IV offer an overall service. Still, they are configurable in quite a few but not all qualitative and quantitative parameters. However, the basic protocol machine (cf. section 4) providing the reliability as well as real-time capability are existent. Since some applications may require specifically tailored services, class V provides the total configuration possibilities with every single parameter separately. These protocol machine descriptions may be archived and used for further communications, but they are only known locally by the communication partners, who communicated earlier via these negotiated protocol machines.

The effects of application selections on the resulting protocol machine will be clarified. The configuration of protocol functions uses 5 different types of protocol functions. This is the set of ***kernel functions***, which have to be included in the protocol machine, e.g., send/receive data units as well as address mapping. The set of ***required functions*** encompasses functions, which have to be included for a specific service requested, e.g., acknowledgements for the reliable classes II and IV. The set of ***application requested function***s includes functions, which will be additionally used since the application made an explicit request on them. The set of ***network required functions*** are mainly dependent on the network characteristics, e.g., segmenting and reassembling. Finally, the set of ***irrelevant functions*** encompasses functions, which are not needed for the specific service.

Some ***application examples*** will be given for each transport service class. For instance, within class I the applications referred to as voice, music, and video can be satisfied since real-time is sufficient for guaranteeing a maximum jitter if the buffering schemes and sizes used are sufficient /Part91/. A simple example for applications supported by class II is process-control. The reliability is necessary as well as the real-time capabilitiy. Of course, voice and video can be transferred in this class II also, which might be important in the medical environment. Data such as text and graphics will be supported by a class III protocol machine. Transaction oriented traffic and, for instance, image requests from an image data bases can be supported by class IV. Finally, all differences of these standard configurations can be taken care of by class V.

3. Architecture and Mechanisms of Future Transport Protocols

Future transport protocols have to be designed to support the high requirements by applications, e.g., high bandwidth, short delays, small delay jitter, and short connection establishment times /Brau92/. On one hand, protocols and services must provide required functionality and flexibility at the transport service interface, on the other hand, their design must allow high performance implementations and efficiency in high-speed network environments.

New functions are proposed to support multicast as well as synchronization between and within data streams. Protocol functions of conventional protocols have to be improved or redesigned, because of changing environments. Examples are discussed below.

3.1 Problems of Conventional Transport Protocols

Today's transport protocols had been developed for networks with up to five orders of magnitude lower bandwidth and extremely higher error rates. These facts have the following impacts on the use of those protocols in high-speed networks:

- The ***protocol architecture*** in end systems often restricts efficient and high performance implementations. Improvements of the protocol syntax may have a considerable impact on the achievable performance. Considering a flexible transport service as proposed in this paper, the protocol architecture has to support the desired flexibility by modularity and its ability to be easily configurable. The high performance protocol architecture has to provide an implementation independent specification, but should not restrict the use of dedicated implementation environments like multiple processors or hardware modules.
- Some ***protocol functions and mechanisms*** are not appropriate for high-speed networks. Several standardized protocol functions assume a special class of underlying networks. Since modern networks do provide improved performance a set of even controversial protocol functions and mechanisms has to be offered satisfying different kinds of environments. These protocol functions and mechanisms must be selectable dependent on network as well as on user requirements.

3.2 Protocol Architecture

3.2.1 High Performance

Providing network bandwidth of gigabit networks to the user, high performance of protocol processing is required /TaZi91/. Processing of a single 1 kbyte packet must be performed in a few microseconds. VLSI and parallelism are very promising approaches accelerating protocol processing.

- Conventional transport protocols have been designed for rates up to a few MBit/s. The underlying network was the limiting factor, protocol processing time of a packet could be performed fast enough. Therefore, most traditional protocols tried to minimize the bits to be transferred. This resulted often in very complex data structures, e.g., bit arrays and header fields with variable lengths and variable positions or contex sensitive semantics. This strategy requires complex encoding and decoding procedures and make the ***implementation in VLSI*** for those tasks very difficult. VLSI implementations require simple, i.e. fixed, packet formats, alignment of packet fields to 4-byte or even 8-byte words etc. These demands do not limit software implementations at all.
- Furthermore, protocol design does not take into consideration the possibility of parallel processing. ***Parallelism*** is difficult, because the protocol specifications are based on single and sequential finite state machines. Modern transport protocols have to be designed to allow parallel execution together with implementation in VLSI. Parallelism has to be supported by design and specification of the protocol.
  - A very important issue is the separation of processing of control information and user data information supported by appropriate data formats, e.g., control information packets for connection management, flow control, acknowledgement, and data packets for user data. Control data have to be transferred out-of-band and processing may be done in parallel to user data processing.

- Both directions of a duplex connection may be separated to allow parallel processing. Parallelism may be further supported by a decomposition of protocols into several highly independent finite state machines, which may be processed in parallel.
- Another important drawback of conventional protocols is their layering. Although layering is only a theoretical model for communication systems, it often influences the implementation of protocols. Layering causes overhead, because of communication among layers at the layer interfaces. Parallel processing is restricted by points, where layers synchronize their activities.

3.2.2 Flexibility

Another drawback of layering are the redundancies introduced by this concept. Several protocol functions are often performed in more than one layer. Also for flexibility reasons, communication systems should rather be decomposed into functional units than into layers. Protocol functions are the units for protocol configurations.

Functional decomposition can be done by the decomposition into modular and configurable finite state machines /Haas91/, /ZSTa92/. A ***modular decomposition into finite state machines*** (FSMs) with dedicated protocol tasks supports flexible configuration of the protocol. Single finite state machines on the level of protocol functions may be configured to a protocol machine for one data stream dependent on the network environment, dependent on the service type, and dependent on the quality of service parameters required by the application. Mechanisms, strategies, and parameters of protocol functions can be selected to provide the required service and the quality of service parameters.

We propose independent finite state machines for the following protocol functions:

- data transfer (send side), e.g., addressing, PDU generation, segmentation
- data transfer (receive side), e.g., lifetime control, resequencing, reassembly
- connection mangement (establishment, maintaining, termination)
- flow and rate control
- congestion control and round trip time estimation
- synchronization within data streams
- acknowledgement and error recovery (retransmissions or forward error correction)
- routing, relaying, and route mangement

3.3 Protocol Functions and Mechanisms

Several approaches have to be integrated to allow high-speed protocol processing with GBit/s rates. Hardware implementations with VLSI components will be required to keep up /LaSc91/. Especially in layers processed inside transit systems very simple protocols will be required. The offered network service will be very simple and limited. Functions such as flow and error control will be performed by the end systems mainly. This trend has been made possible by the decreased bit error rates of modern networks. Complex and time-consuming error control functions on a link-by-link level does not seem to be appropriate in those environments. Transport protocols become more important along with this trend, because they have to perform the main part of providing a reliable end-to-end connection.

The main problem in a high-speed network environment is the fact that the amount of data being in transfer between end systems is increasing in proportion to the increasing bandwidth. To facilitate the use of the complete network bandwidth, transmission without any breaks is necessary. Therefore, the sender's credit have to be increased to allow continuous transmission without any breaks, because of acknowledgements by the receiver.

With the increasing bit rates, the delay between sender and receiver does not change in proportion to the bit rate. Considering a transmission line of 3000 km and a bit rate of 64 kbits/s, 640 bits are in transit between sender and receiver because of the signal delay. The amount of data being in transit is also called path capacity or bandwidth-delay-product. Considering a high-speed network with a bit rate of 500 MBit/s, the amount of data in transit increases to 5 Mbit, i.e. 625 kbyte. It becomes obvious that simple flow control procedures as used in today's protocols have to be replaced by advanced procedures. Especially flow control procedures seem to become one of the most important tasks in protocols for high-speed networks.

3.3.1 Error Recovery

Several years ago bit error rates have been in the range from 10^{-5} to 10^{-6}. Therefore, error correction was performed on a link-by-link level. Because of this fact and the low bit rate, simple error control like the go-back-N procedure have been sufficient. With the use of go-back-N, the lost or falsified packet and all the following have to be retransmitted. The amount of data for retransmission is at least the path capacity. In contrast to networks with low bit rates, the amount of data to be retransmitted will be very high in high-speed networks. It is obvious that this procedure can reduce throughput significantly.

Improved acknowledgement and retransmission procedures may reduce this problem. Advanced high-speed protocols use negative selective acknowledgement for falsified or lost packets and positive cumulative acknowledgement for correctly received packets. These mechanisms allow ***selective retransmission*** in the case of errors. In contrast to the go-back-N procedure, the sender has to retransmit only the packets requested by the receiver. Less bandwidth for retransmissions is required, but more buffer space is necessary at the receiver, because packets received correctly have to be buffered, if prior packets have to be retransmitted.

Nevertheless, there are some applications, which require error-free data exchange, and cannot tolerate the delay caused by retransmissions. Retransmissions delay incoming data to at least the triple time of an one-way transmission. Usually, the delay will even be higher, if packet loss is detected by expired timers. Delayed incoming data are useless in many applications, e.g., video applications.

Forward Error Correction avoids delay by retransmission. Forward error correction creates additional redundant information, e.g., h redundant packets for k packets to send. The receiver is able to recover k packets if not more than h out of (k+h) are lost.

3.3.2 Flow Control

Window-based flow control mechanisms can achieve continuous data flow in high-speed networks only with small credit values, i.e. the receiver allows the sender to transmit data up to a certain sequence number. In that case, the network bandwidth can not be used completely. Because of the path capacity of the network small credit values cause blocking situations at the sender. The time to wait for acknowledgements, which increase the credit value, may even be higher than the time for data transmission. The sender spends most of its time for waiting for new credit values, although the receiver has processed all received data in the meantime and would be able to receive further data.

In addition to window-based flow control procedures ***rate control*** can be used. At connection establishment both entities negotiate parameters like burst size and burst rate. Both parameters may be changed during the lifetime of a connection. For realisation of the rate control procedure, a byte counter is initialised with burst size value and a timer with a value of 1/burst rate is started. The sender may transmit data, if the counter value is positive. The counter is reduced by the data packet size. If the counter value is negative or equals 0, the sender has to wait for timer expiry. If the timer expires, the counter is reset to the burst size and the sender can start transmissions again.

3.3.3 Connection Management

Fast connection establishment is very important in transaction-oriented applications. Usual handshake-based establishment procedures take more time for establishing the connection than the following data transfer, e.g., establishment takes ca. 20 ms in the high-speed network mentioned above, but 10 MBit of data could be transferred during this period of time.

Implicit connection establishment procedures are proposed to support transactions. The procedure does not use special establishment packets. The sender starts user data transmission immediately and the receiver establishes a connection with the first incoming data. Sender and receiver start timers for data sent or received to ensure that the peer entity receives retransmissions and acknowledgements before timer expiry. Timer expiry triggers connection termination. The procedure supports short transactions or remote procedure calls by avoiding delays, but connections have to be maintained for a relative long period of time to ensure correct termination. After connection termination delayed and duplicated packets may cause connection establishment again or may cause confusion, if another connection with the same connection identifier has been established in the meantime. These cases may be avoided by the use of unique connection identifiers and references. The references must not be reused for a certain time period to ensure that no duplicated packets can exist in the whole network. This mechanism is well-known as "frozen references".

These above presented alternative protocol function mechanisms and strategies for the discussed protocol functions have to be selected by a configuration component in order to provide the requested quality of service. The choice also depends in some cases on the network characteristics.

4. The Flexible Transport System

In this section a brief overview of the model for a high performance transport system will be presented. The transport system is based on a decomposition of its tasks into well defined and separated protocol functions. The layer oriented structure has been logically replaced by a protocol function oriented structure, which describes in a fine granular abstraction level the tasks of transport systems. The flexibility of this transport system, provided by separate ***protocol functions***, which are defined as basic elements, allows for the suitable configuration of protocol machines consisting of a set of protocol functions. In this case the ***protocol machine*** is regarded as an integrated view of transport, network, and data link protocol functionality. Each protocol function is associated with a specific ***protocol function task*** that it provides. The inherent ***protocol function mechanisms*** can vary widely (cf. section 3) and are user selectable by a set of configuration parameters (cf. section 2). Additionally, a protocol function is uniquely defined by two sets of input and output parameters.

4.1 Transport System´s Model

The model of this flexible transport system consists out of three parts. First, there is the ***transport system architecture***, second the ***transport system session model***, and, third, the ***transport system tools***.

4.1.1 Transport System Architecture

The transport system architecture includes the ***protocol resource pool***, which is regarded as a static specification and comprises sets of all available protocol functions and their mechanisms. However, the active part within the transport system is called ***protocol machine*** and is an instantiation of a certain number of protocol functions. It includes a protocol machine management agent, who handles management issues.

4.1.2 Transport System Session Model

The transport system session model encompasses all different types of communication -video, voice, and data- and their complex abstractions -symmetrical and asymmetrical- in terms of manageable units. These manageable units are called ***application data streams,*** which are associated with an unidirectional transmission of data from one transport system user to another. For any bidirectional traffic type two distinct protocol machines with possibly different requirements will be configured, each representing one single application data stream inside the transport system. The association of several data streams between two or more communicating applications is called ***session*** and will be maintained by the session manager.

4.1.2 Transport System Tools

Three different ***phases*** can be associated with a protocol machine. This is the configuration phase, the operation phase, and the termination phase. An application requests a dedicated transport service, which results in the configuration of a protocol machine. Any changes in the network environment or of the application will result in an additional update request. After the configuration phase is finished, in the operation phase protocol processing takes place for exchanging application data between transport system users. Finally, in the termination phase a data stream will be released and, therefore, the allocated resources also.

The ***Protocol Configuration Manager*** (PROCOM) is regarded as a configuration tool for protocol machines inside the transport system assisted by other tools like shown in figure 1. The service primitive for a session request will be handled by the transport service interface (TSI). This primitive contains specific quality of service (cf. section 2) as well as configuration parameters from the upper layers and has to be forwarded with a protocol machine profile to PROCOM. These proceedings represent the request of specific application requirements. Network parameters will be provided via the network service interface (NSI). This information is forwarded to the ***System Resource Manager*** (SYSREM), who keeps track of these data in the System Resource Data Base.

PROCOM administrates default values of configurations, initializes the configuration process, and processes the configuration results in term of a protocol function dependency graph. Possible responses of PROCOM contain the refused parameters, delivered back to the application, or the protocol functions that have to be configured to a running protocol machine, delivered to the ***Code Configuration Manager*** (CODCOM). This resulting set of needed protocol functions is specified in the protocol machine description and includes a protocol function dependency graph, where separate protocol functions are interconnected by their input and output parameters. They form altogether the executable protocol stack after CODCOM has generated the protocol machine implementation out of the protocol function dependency graph.

In the following an overview of the assisting tools (cf. figure 1) for PROCOM and CODCOM are given. The ***Protocol Resource Manager*** (PROREM) maintains a single pool consisting of three distinctive parts.

In the first part, the supported *protocol functions* are collected. They are defined by their input and output parameters as well as by their configuration parameters. Input and output parameters are needed to specify the protocol data manipulation done by the protocol function and, therefore, describe the effects of protocol processing. Compared to this, configuration parameters are used for configuration purposes only, which is, e.g., the selection of a protocol function inherent mechanism.

The second part consists out of a number of *protocol function information bases*, in which an abstract set of information concerning protocol control data as well as user data will be concentrated. This is, e.g., a routing table or connection state information. Any access to data inside any protocol function information base does not imply any sequential relation between protocol functions.

Finally, the third part manages the set of *pre-defined protocol machines*. These are already existing protocol machine descriptions. They belong to the predefined transport service classes as well as to special protocol machines, which had been configured earlier and kept in a local archive. They are useful only if two dedicated comunication partners know their special requirements from earlier sessions.

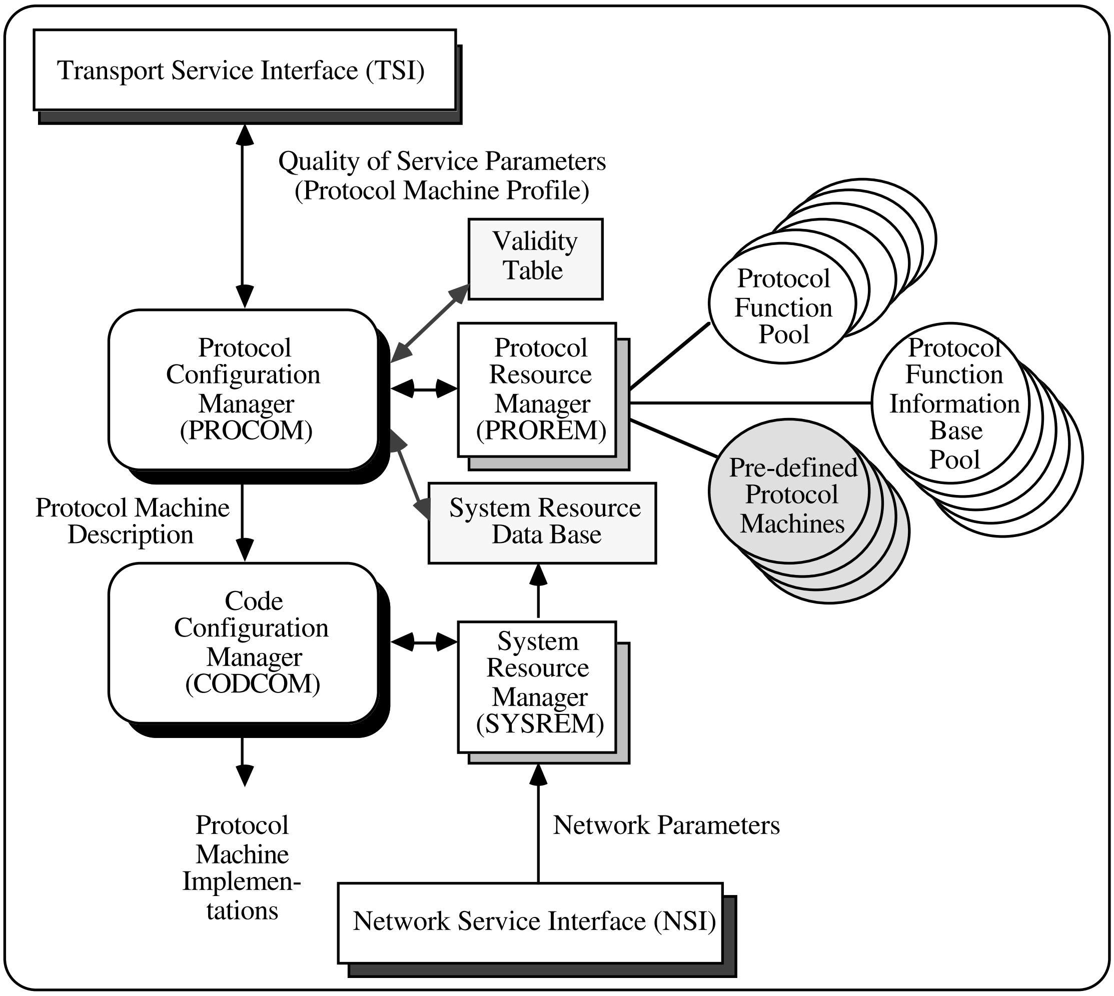

Figure 1: The Flexible Transport System

The first two parts have to be used from PROCOM via the Protocol Resource Manager (PROREM) since no transport service class has been selected explicitely at the transport service interface. In this case, the configuration process of PROCOM calculates the dependencies of protocol functions.

In a much less complicated case, a transport service class has been selected by its number. PROREM just takes a pre-defined protocol machine description out of the set of possible one -third part of the pool- and hands it to PROCOM. This protocol machine description just bypasses CODCOM since the protocol machine implementation already exists. Therefore, the configuration delay has been minimized to the optimal lowest time possible.

PROCOM provides the configuration task by checking the configuration request against a validity table, in which the valid protocol function configurations as well as protocol function interconnections are stored. It uses PROREM for defining the protocol machine description in consideration of the actual system resource data base.

CODCOM will be supported by a ***System Resource Manager*** (SYSREM). Maintaining the system resource data base the System Resource Manager keeps track of the utilization of system resources, like processors and memory. Additionally, the availability of network resources will be taken care of by System Resource Manager provided from the network service interface. This is for instance the maximum available bandwidth, which will be reserved by SYSREM, added into the system resource data base, and requested from this data base by PROCOM for configuration purposes. Another maintained value is the maximum data unit size for transmitting data.

5. Conclusions

This paper proposed a function-based approach for the design of future high speed transport services and protocols. The service was described by two sets of quality of service parameters, which can completely specify application´s requirements on the transport system. They provide a detailed and a more higher level selection of protocol functions as well as of protocol function mechanisms. This approach for an advanced architecture based on the logical decomposition of transport protocols into relevant functions and protocol function mechanisms for the use in high-speed network environments has been discussed by pointing out the problems of conventional protocol functions and addressing some issues on better suitable protocol functions and mechanisms respectively. The achieved flexibility does not add additional overhead to protocol processing itself. Only for configuration as well as update requests with special application requirements some overhead occurs. Finally, the transport systems model and the needed tools have been described briefly.

6. Acknowledgements

The authors are grateful to Dr. A.Tantawy and Dr. M.Zitterbart for their highly valuable contributions to major parts of the presented work. Additionally, thanks to Professor Dr. G.Krüger for his support and encouragement of this work at his Institute of Telematics at the University of Karlsruhe.

7. References

/Brau92/ T.Braun: *A Parallel High Performance Transport System for Metropolitan Area Networks*; in 5th IEEE Workshop on Metropolitan Area Networks, Taormina, Italy, May 1992.

/Haas91/ Z.Haas: *A Protocol Structure for High Speed Communication over Broadband ISDN*; in IEEE Network Magazine, January 1991.

/LaSc91/ T.La Porta, M.Schwartz: *Architectures, Features, and Implementation of High Speed Transport Protocols*; in IEEE Network Magazine, January 1991.

/TaZi91/ A.Tantawy, M.Zitterbart: *Transport Service and Protocols for High Speed Networks*; IBM Research Report RC17074, Juli 1991.

/ZiSt91/ M.Zitterbart, B.Stiller: *A Concept for a Flexible High Performance Transport System*; in Informatik Fachberichte 293, Springer Verlag, Heidelberg, GI Jahrestagung Darmstadt, October 1991.

/ZSTa92/ M.Zitterbart, B.Stiller, A.Tantawy: *A Model for Flexible High-Performance Communication Subsystems*; IBM Research Report RC17801, February 1992.