A Protocol Architecture for a Flexible High Performance Communication Subsystem

Torsten Braun, Claudia Schmidt, Burkhard Stiller, Martina Zitterbart

Institute of Telematics, University of Karlsruhe

76128 Karlsruhe, Germany

Phone: +49 721 608-[3982|3983|4026], Fax: +49 721 388097

E-mail: [braun|schmidt|stiller|zit]@telematik.informatik.uni-karlsruhe.de

Abstract

PATROCLOS (parallel transport subsystem for cell-based high-speed networks) is a transport subsystem to support modern applications, such as multi-media applications, in high-speed network environments. PATROCLOS increases the throughput performance by the integration of protocol and implementation issues. It provides the required functionalities between the media access layer interface of cell-based networks and the transport service interface (functionality of the OSI layers 2b-4) in a single protocol component.

The key feature of PATROCLOS is its parallel and modular protocol architecture [1]. The approach combines a very fine granularity with a function-oriented decomposition into basic modular building blocks. The parallel protocol architecture allows efficient implementations on multiprocessor platforms. PATROCLOS provides a number of protocol mechanisms to support different application and network requirements. They can be selected dependent on application requirements. This paper concentrates on both issues: the protocol architecture and the implementation of the transport subsystem on a transputer based multiprocessor architecture.

Because of its modularity, PATROCLOS can be used to populate a protocol function pool for the function-based communication subsystem (F-CSS). F-CSS [2] is a framework for flexible communication subsystems, in which PATROCLOS may be integrated. F-CSS allows applications to request individually tailored services from the communication subsystem. Based on the service requirements and available resources, suitable protocol functions are selected from a protocol function pool and configured to protocol machines dependent on the service required by the application.

1 PATROCLOS Protocol Architecture

Based on the analysis of the XTP specification (version 3.4) and the performance results of the corresponding multiprocessor implementation, several design guidelines for protocol architectures appropriate for concurrent processing have been derived. PATROCLOS follows these guidelines and uses a modular design to achieve a high degree of parallelism. Redundant placement of protocol functions in several layers can be avoided, e.g., segmentation and reassembly functions in layer 3 and 4 of the OSI Reference Model.

Parallel extended finite state machines (FSMs) are the building blocks of the protocol architecture. The set of all FSMs forms a FSM system. FSMs belonging to the same PATROCLOS entity exchange messages for co-operation. All FSMs are designed as autonomous as possible to minimize interactions with other FSMs of the same entity. Communication among FSMs is uni-directional to allow a high degree of parallelism. If possible, the necessary information exchange is performed periodically to reduce the communication overhead.

PATROCLOS consists of two types of FSMs: interface and protocol FSMs (Figure 1). Interface FSMs are located at the interfaces of the transport subsystem to the application and the

network. They support interactions of PATROCLOS with the transport service user or with the underlying network unit. They are only involved in local communications within one entity. The network interface FSM (send control FSM) controls and schedules transmission of all protocol data units (PDUs) to the underlying network unit. The transport service interface FSM maps service primitives to signals for the protocol FSMs.

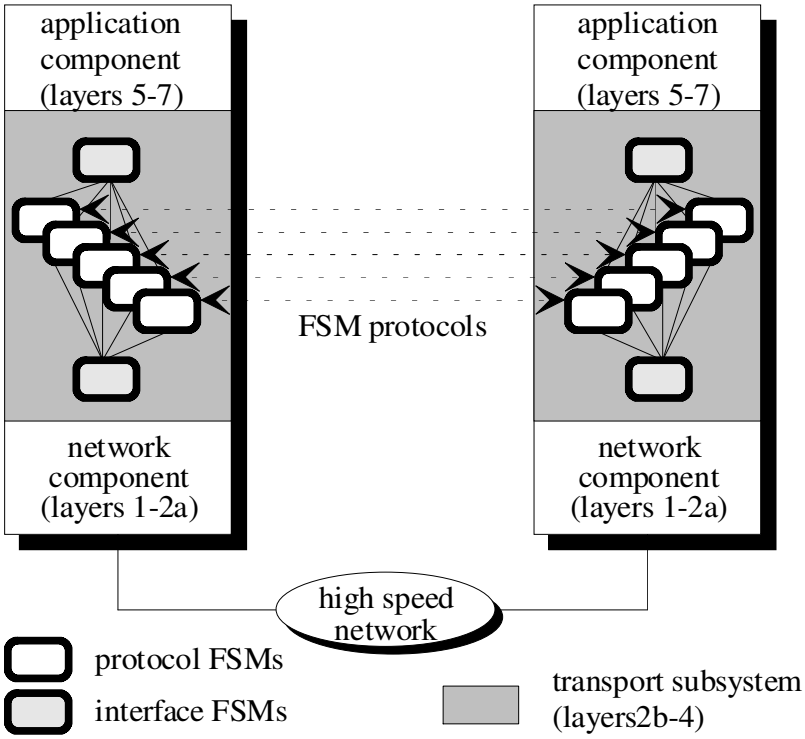

Figure 1: Protocol Architecture

The key feature of protocol FSMs is their use of separate partial protocols to communicate directly with the corresponding FSMs at the peer entity. They are designed to allow parallelism between send and receive part and to decouple the connection state information exchange from user data exchange (Figure 2). All partial protocols have special PDUs, individual error recovery mechanisms, and timers. Each partial protocol is performed by communicating peer FSMs, using PDUs containing only the necessary information for their protocol function. Multiplexing of PDUs by different FSMs is avoided to support concurrency. Together, the autonomous partial protocols build the PATROCLOS protocol. The decomposition into independent FSMs permits parallel execution and has the benefit of a highly modular system.

Currently, the following protocol FSMs are defined: The connection management FSM is responsible for establishment, management, and correct release of a full duplex connection. Therefore, it has to activate and initialize the other FSMs of the same entity for each new connection and has also to deactivate these FSMs after connection termination. The peer FSMs exchange connection establishment PDUs, initialize the connection, and forward quality-of-service requirements to other FSMs. The connection management FSMs also guarantee the unique use of connection identifiers. The transport service FSM selects handshake-based or implicit establishment procedures dependent on the required service. Additionally, explicit/timer-based and graceful/normal connection release can be selected.

User data transfer can be divided into a send path and a receive path. Data transfer FSMs send and receive user data. The main tasks of both FSMs are formatting and analyzing PDUs with user data. Additional functions are segmentation and reassembly. On sending side, transport service data units (TSDUs) are segmented by the segmentation FSM into PDUs. The user data send FSM formats the header fields of the PDUs to describe the user data (e.g., sequence

numbers). On receive side, the user data receive FSM operates on these informations to detect duplicates or errors. Furthermore, it checks the packet lifetime. The reassembly FSM reassembles PDUs to TSDUs.

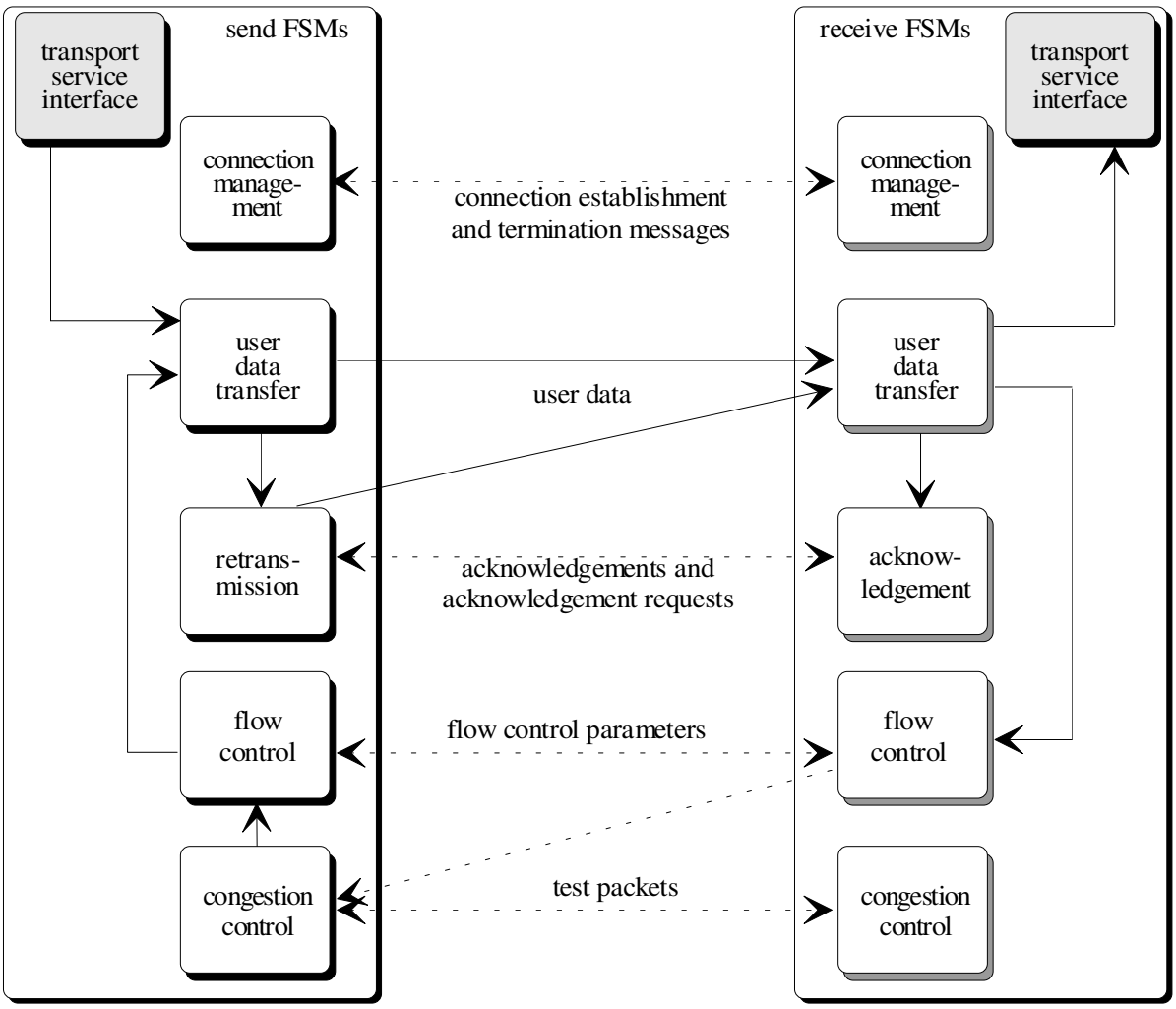

Figure 2: Protocol FSMs

Because data transfer often performs the critical path in communication protocols, retransmission and acknowledgment are handled by two special FSMs. The FSM on the send side (retransmission FSM) requests and receives acknowledgments from the peer FSM on the receive side (acknowledgment FSM). The retransmission FSM analyzes the acknowledgment PDUs, initiates retransmissions depending on the acknowledgment information and the selected strategy (selective repeat or go-back-N), and releases the buffers of acknowledged data packets. Retransmissions are sent to the peer's user data receive FSM. The retransmission FSM receives the control over sent user data packets from the user data send FSM periodically with an interval lower than one round trip time. The user data receive FSM signals informations about received data to the acknowledgment FSM periodically, which requests retransmissions based on the received results. The acknowledgment FSM sends negative selective and positive cumulative acknowledgments to the peer's retransmission FSM. Communication between the FSMs participating in user data transfer is reduced to a minimum. In the case of error-free communication, the FSMs in one entity communicate periodically instead of exchanging messages after each packet. In error cases, the information is sent directly to avoid unnecessary delays.

Window-based, rate-based, and stop-and-go mechanisms can be selected for flow control. Several combinations of these mechanisms are also possible. Two flow control FSMs exist in

each entity, one FSM for each data flow direction. These FSMs negotiate flow control parameters with the remote peer and use these parameters to control the data flow. The flow control send FSM controls data flow by the interpretation of the peer entity's feedback signals, e.g., window and buffer sizes, and rate values. The flow control receive FSM transmits rate parameters or credit values according to the current traffic and available buffer space. The flow control send FSM signals periodically the maximum sequence number to be sent (limited by window-based flow control) to the user data send FSM. Furthermore, this FSM starts a rate timer for rate-based flow control, calculates the maximum sequence number to be sent for the next time period, and signals this value to the send control FSM.

The congestion control FSMs control data flow by the interpretation of feedback signals from the network or the peer entity, e.g., congestion indications, round trip times, and inter-arrival times of test packets. The FSMs exchange test PDUs to identify congestions. Congestions are detected based on test packet delays and inter-arrival times, and the rate can be decreased by sending signals to the flow control send FSM. Currently, the adaptive admission control algorithm and the packet pair algorithm are used.

2 Parallel Implementation and Performance

Another advantage of the protocol architecture is its direct support of efficient parallel implementations. A hybrid multiprocessor-based implementation architecture has been developed that uses transputers as general purpose processors and several specialized hardware units (Figure 3). The implementation architecture has to be connected via a DMA unit to a host system.

The modular protocol design supports an easy mapping on multiprocessor architectures. Interface and protocol FSMs have been implemented as processes, one process for each FSM. Each process derived from the FSMs of the PATROCLOS protocol architecture can be mapped to a separate processor. Processes communicate with each other via asynchronous message passing across interconnecting inter-process channels. Timer processing is supported by special timer processes, which do not need to run on the same processor as the protocol process, and, therefore, may be performed concurrently.

Access control units to control the access of several processors to shared memory buses, and simple protocol functions on cell level (SAR processing of AAL4) have been implemented by dedicated hardware units to support the high network bandwidth. Moreover, the demultiplexing hardware on the receive side performs demultiplexing below the ATM cell reassembly level and distributes incoming ATM cells to the reassembly hardware units of the user data send processor and the processor modules of the control functions.

The achievable performance is limited by the receive part [3]. For the following analysis, rate-based and window-based flow control, selective retransmission, segmentation and reassembly have been selected. The user data receive process is the bottleneck process on the receive part. In addition to analyze the received data packets, it has to inform periodically the acknowledgment process about data packets received correctly and the flow control receive process about buffer state information. We assume that both signals are sent q times per data packet. Decreasing the value of q results in a higher performance. The throughput is higher than 5000 packets/s for $q < 0.2$, higher than 6000 packets/s for $q < 0.1$, but is limited to 6300 packets/s by the time of the data receive process to analyze a packet (159 us). The throughput of the receive path has been calculated based on the processing times of the receive processes and has been verified by real measurements. The measured and the calculated results differ only by 2 %.

Performance evaluation of the send path is similar to the analysis of the receive path. The data send process is the bottleneck process of the send path. Here, the throughput depends also on

the parameter q. For q < 0.2 more than 6000 packets/s can be achieved. The speed-up of the implementation will be 2-2.5 (receive path) and 2.5-3.5 (send path) dependent on the parameter q. A TCP/IP implementation running on the same platform as used for performance evaluation of PATROCLOS achieves less than 3000 packets/s and lower speed-up values.

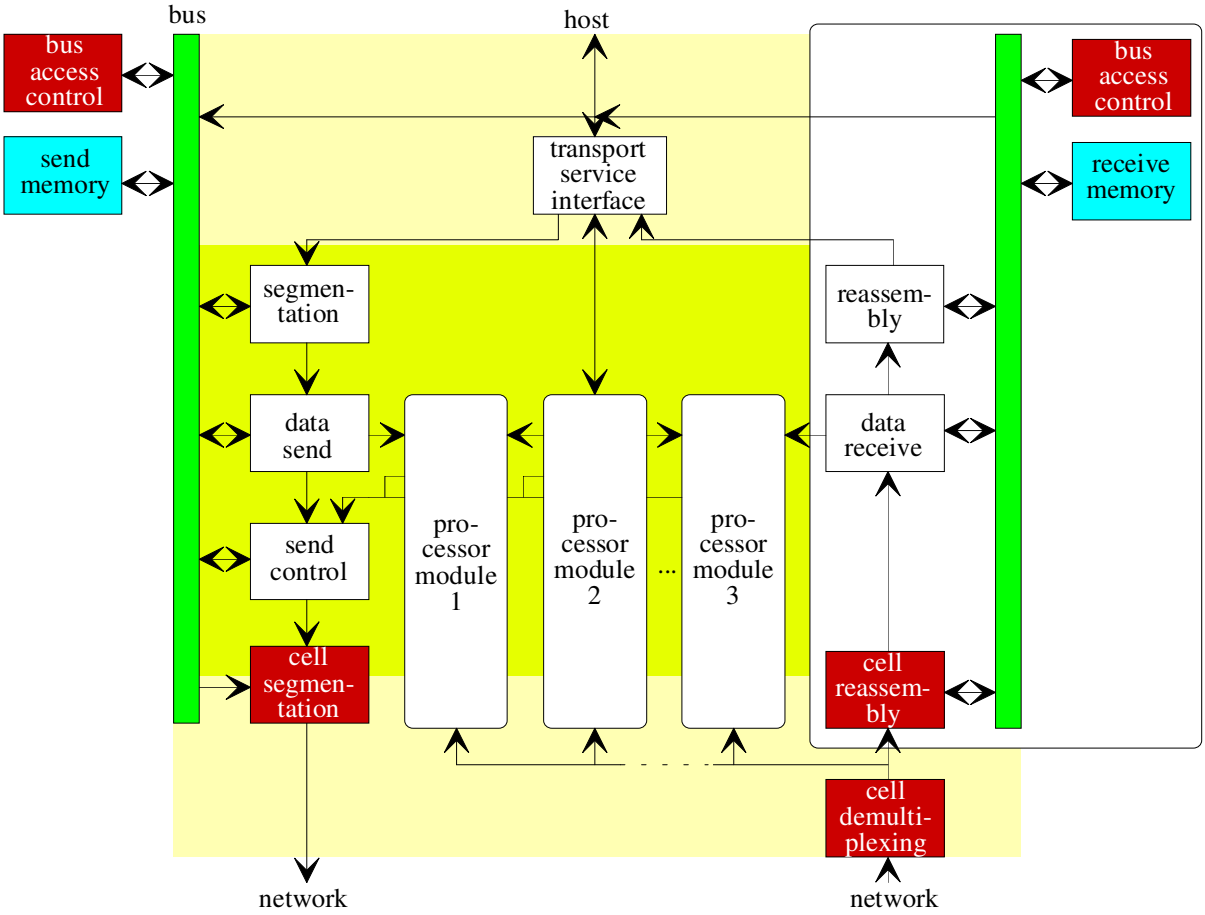

Figure 3: Implementation Architecture

3 PATROCLOS FSMs as Protocol Resources for F-CSS

F-CSS has been designed to allow adaptation of protocols to a wide variety of application requirements and different network environments. It is based on protocol functions as configurable atomic units. Another salient feature is the application interface, supporting application-driven configuration of efficient individually tailored protocol machines. F-CSS is based on functional units, not on layers. Due to the finer granularity in function definition, the service offered by the communication subsystem can be adjusted more precisely to individual application needs. Moreover, the adaptation of F-CSS to changing communication environments is supported more easily since only protocol functions have to be either added or replaced.

F-CSS provides an application interface specifically designed to support service flexibility. On one side, it relieves the application from dealing with the large number of available protocol functions by providing a small number of pre-defined service classes. On the other hand, the application interface provides the application with the possibility of formulating its individual requirements in terms of service parameters and protocol functions.

A protocol function is associated in F-CSS with a protocol function task (e.g., retransmission), a protocol function mechanism (e.g., selective retransmission), protocol function parameters to specify the mechanism more in detail, and probably a protocol function information base (e.g., protocol state information). In PATROCLOS the retransmission FSM represents a protocol

function task. Selective or go-back-N mechanisms can be selected. Because of the modular, function-oriented protocol architecture, the PATROCLOS protocol can be configured easily. Selection of a protocol function or function parameters affects the behavior of one or very few protocol FSMs. E.g., the selection of the congestion control algorithm affects only the congestion control FSM.

F-CSS consists of a static part, defining the available resources, and an active part representing the protocol functions and the tools for configuration, maintenance, and termination. The static specification of F-CSS comprises a set of all available protocol functions, which form the protocol resource pool. This pool provides protocol resources for the dynamic configuration of protocol machines. A protocol machine consists of a number of protocol functions offered by the protocol resource pool. Protocol machines can be created, changed, and deleted dynamically once the resource pool is initialized. PATROCLOS FSMs are part of this protocol resource pool and can be configured to a F-CSS protocol machine.

The F-CSS session model (Figure 4) reflects the needs of complex applications, comprising different concurrent types of communication, e.g., voice, video, and data. A session is an association between two or more communicating applications and may consist of several application data streams, each of which is being processed by a separate protocol machine. Asymmetric applications with different service requirements for each communication direction have been taken into consideration in the development of the F-CSS model. A service is always associated with a single direction of data flow. In this case, the F-CSS configures different protocol machines, each being tailored to the requirements in one direction.

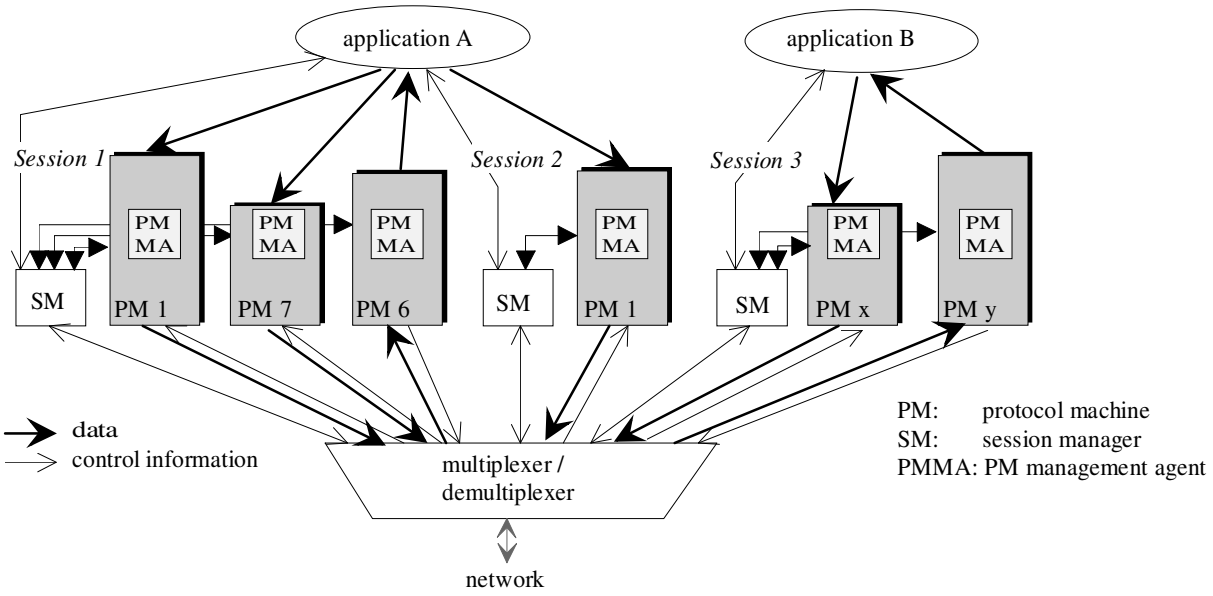

Figure 4: F-CSS Session Model

For configuration and management purposes, F-CSS provides several tools (Figure 5). The session configurator receives session-setup requests from the local or the remote application and delivers a message to the protocol configurator to start the configuration of a suitable protocol machine for each individual data stream requested in the session-setup request. The protocol configurator generates a protocol machine description reflecting the required protocol functions and their dependencies. The output may then be passed to a code configurator, which has to map the protocol machines on an implementation platform. In the case of PATROCLOS, however, the output is passed directly to the connection management FSM that initializes all other FSMs and delivers the selected protocol functions and parameters. For configuration purposes the available resources such as CPU, memory, network capacity, etc. must

be considered. The required information is collected in collaboration with network and resource management entities.

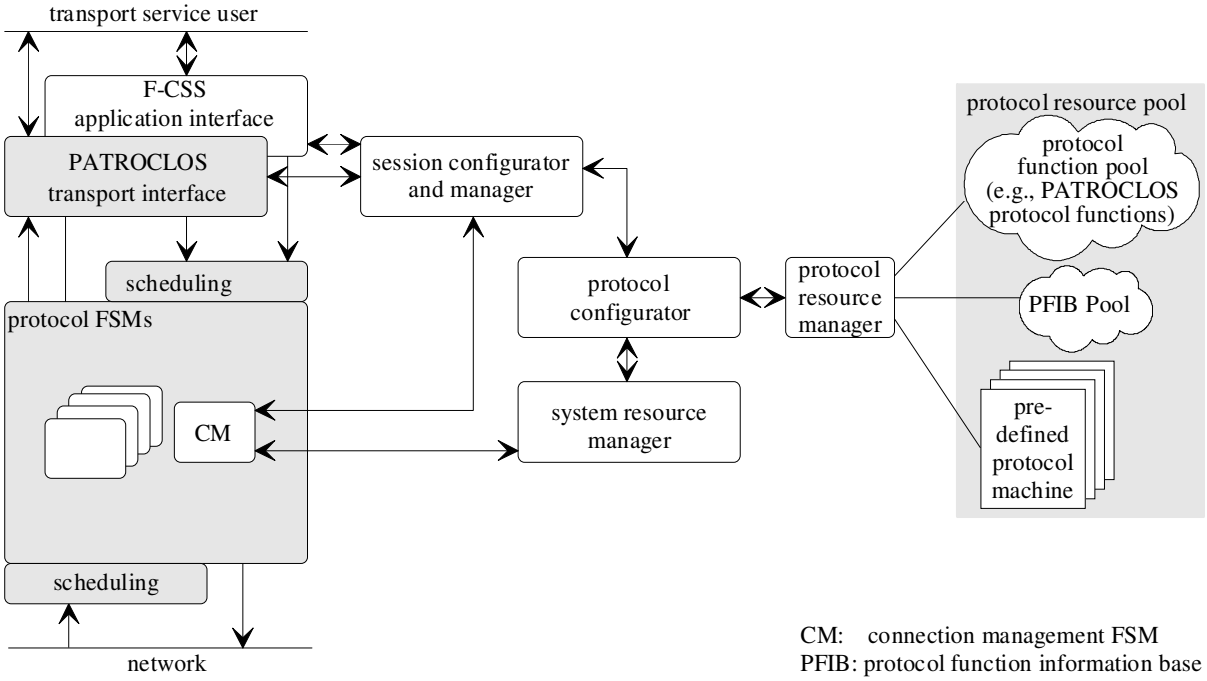

Figure 5: PATROCLOS and F-CSS Tools

References

[1] Braun, Torsten; Zitterbart, Martina
Parallel Transport System Design
in: Danthine, A.; Spaniol, O. (eds.): High Performance Networking, IV, North-Holland, 1993, pp. 397-412

[2] Zitterbart, Martina; Stiller, Burkhard; Tantawy, Ahmed
A Model for Flexible High-Performance Communication Subsystems
Journal of Selected Areas in Communication, Vol. 11, No. 4, May 1993, pp. 507-518

[3] Braun, Torsten
Performance of a Parallel Transport Subsystem Implementation
Second IEEE Workshop on the Architecture and Implementation of High-Performance Communication Subsystems, Williamsburg, Virginia, USA, September 1 - 3, 1993