1. CPCS (and CPCSIF)

Author: Juhana Räsänen

CPCS module provides the higher layers an interface to AAL5 Common Part Convergence Sublayer services. Actual CPCS/SAR functionality is implemented in hardware, so this protocol acts only as an wrapper to ATM Network Interface Card and Linux kernel ATM services.

CPCSIF defines the AAL5 CPCS service interface (CPCS-primitives).

1.1 1 Introduction

This protocol is the lowest layer of the TOVE protocols. Its basic function is to provide higher layers transport of AAL5 SDUs over an ATM connection as defined in [1], Chapter 6. A suitable subset (that is also supported by the ATM NIC and Linux ATM interface) for signalling use has been selected, see Chapter 4 for a complete list of implemented features.

The users of the CPCS (in practice SSCOP, Service Specific Connection Oriented Protocol) access its services with the help of CPCS interface module (CPCSIF), which define the service primitives of CPCS.

Conceptually one CPCS instance represents one ATM point-to-point connection and is bound to a single VPI/VCI triple in the ATM interface of the controller workstation. From the signalling software point of view this means that a CPCS instance (together with SSCOP and SSCF instances associated with it) represent one signalling link.

From the software architecture point of view CPCS can be seen as the boundary between OVOPS++ world and operating system.

1.2 2 Architecture

The architecture of CPCS is very simple due to the fact that it acts only as an wrapper to the Linux kernel ATM services. CPCS consists of a protocol conduit (class cpcsAdapter) that has a pfDevice instance as a class member. The state machine of the protocol is trivial, because CPCS is stateless; likewise the service interface is equally trivial having only two primitives (classes cpcsUNIDATAreq and cpcsUNITDATAind).

Interface from the OVOPS++ to the communications channel (eg AAL5 socket) is implemented with help of pfDevice class instance. Device classes act as wrappers to Unix file descriptors and facilitate asynchronous operation by registering themselves to OVOPS IOHandler.

Thus, the role of the protocol is on the other hand to process incoming CPCS user requests and send the data to kernel, and on the other hand to transform incoming messages from kernel to CPCS primitives and send them to CPCS user.

1.3 3 Implementation details

Although CPCS resembles an adapter conduit (connects to other conduits only on A-side), it is implemented as a protocol conduit, because the functionality of cpcsAdapter was easiest to implement with a state machine, which is not used in adapters.

CPCSIF messenger classes are inherited from pfMsgTransporter instead of pfMessenger. In this simple case the same functionality is achieved with a small performance gain, when unnecessary creations/deletions of pfMessenger classes is avoided.

1.4 4 Features implemented

Only message mode without corrupted data delivery option is implemented in the service interface (CPCSIF). The protocol itself does not implement CPCS functionality, but provides a wrapper to the actual AAL5 CPCS function implemented in hardware.

- CPCS primitive parameters: CPCS-ID, CPCS-LP, CPCS-CI and CPCS-UU are present in cpcsUNITDATAreq and cpcsUNITDATAind classes, but only CPCS-ID (Interface Data) is actually used, others are set to default values.

1.5 5 Known bugs and flaws

None known at the time of this writing.

1.6 6 Future development

No needs for future development is seen at the moment, but if OVOPS++ framework is developed for instance by allowing adapters have state machines or making a new conduit class of pfDevices, the implementation of CPCS will get a little bit simpler.

1.7 7 Statistics

CPCS module development was spread over a long time. First version that offered a “virtual” AAL5 channel between two conduit stacks was completed in the summer 1996 and a full version supporting pfDevices was completed later in the autumn.

Activity	Research	Design	Coding	Reviews	Total
Duration (h)	5	10	25	20	60

Table 1 Duration of activities

Lines Of Code (LOC)	Number of files	Number of classes
517	11	7

Table 2 Metrics

1.8 8 References

[1] ITU-T Recommendation I.361, *B-ISDN ATM Adaptation Layer (AAL) Specification*, Helsinki, March 1993.