1. SSCOP (and AAIF)

Author: Juhana Räsänen

SSCOP module implements a connection oriented protocol for assured data delivery between (signalling) AAL connection endpoints.

AAIF defines the SSCOP service interface (AA-primitives).

1.1 1 Introduction

SSCOP (Service Specific Connection Oriented Protocol) is a link layer protocol that offers the signalling protocols a reliable transport of data over an AAL5 link to the neighbouring system. Its functionality is defined in [1].

SSCOP uses CPCS adapter to access AAL5 service offered by the ATM Network Interface Card, and the service interface of SSCOP is implemented in AAIF module, that contains OVOPS++ messenger classes for AA primitives. Usually the applications (eg signalling protocols) don't use SSCOP directly, but through a suitable SSCF (Service Specific Coordination Function). Currently there are SSCFs defined for UNI and NNI signalling protocols.

1.2 2 Architecture

Architecture of SSCOP implementation is a text-book case of a protocol conduit. Q.2110 Recommendation defines 10 states for the protocol state machine and the inputs driving the state machine (as well as the primitives sent by SSCOP) are represented in Figure 1.

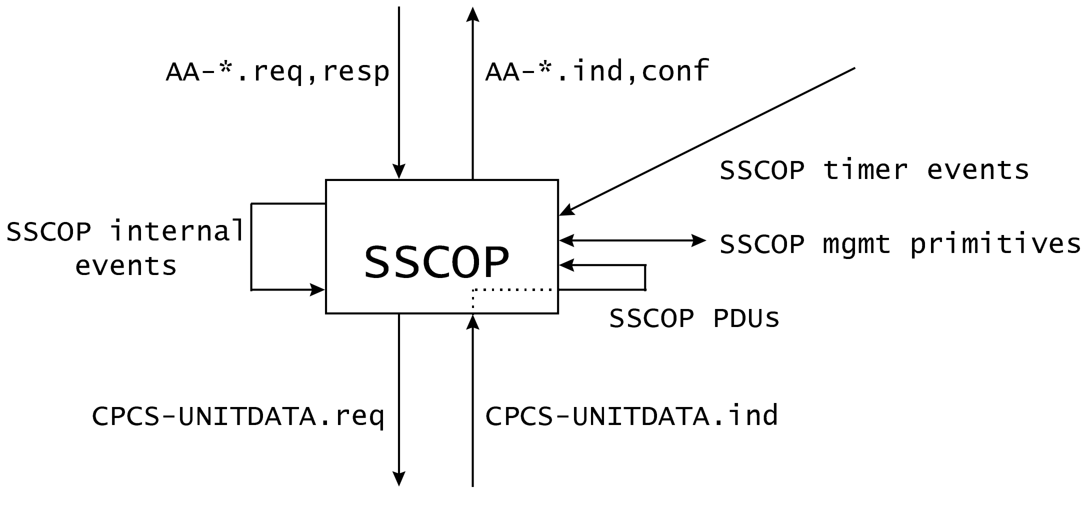

Figure 1 SSCOP inputs

The main inputs for SSCOP are the primitives from the SSCOP user and data indications from AAL5 CPCS layer. Other sources of events are the primitives from layer management entity (MAA-* primitives), timeouts from SSCOP timers and internal events generated by SSCOP itself (these events are generated eg when a data chunk is accepted for transfer, but it is first buffered inside SSCOP and the actual transfer takes place later). Also the PDU inputs can be viewed as internal events, since on an arrival of a CPCS-UNITDATA.indication the contents of the AAL5 data chunk are decoded into a SSCOP PDU, which is then accepted as a new event into the message queue of SSCOP. This is indicated with the dotted line in Figure 1.

1.3 3 Implementation details

SSCOP implementation is a good example of the capabilities of the OVOPS++ framework. No special tricks were needed to implement SSCOP functionality, it was very straightforward to transform the protocol specification to working code. To demonstrate this, Figure 2 has a small part of SSCOP SDL diagram.

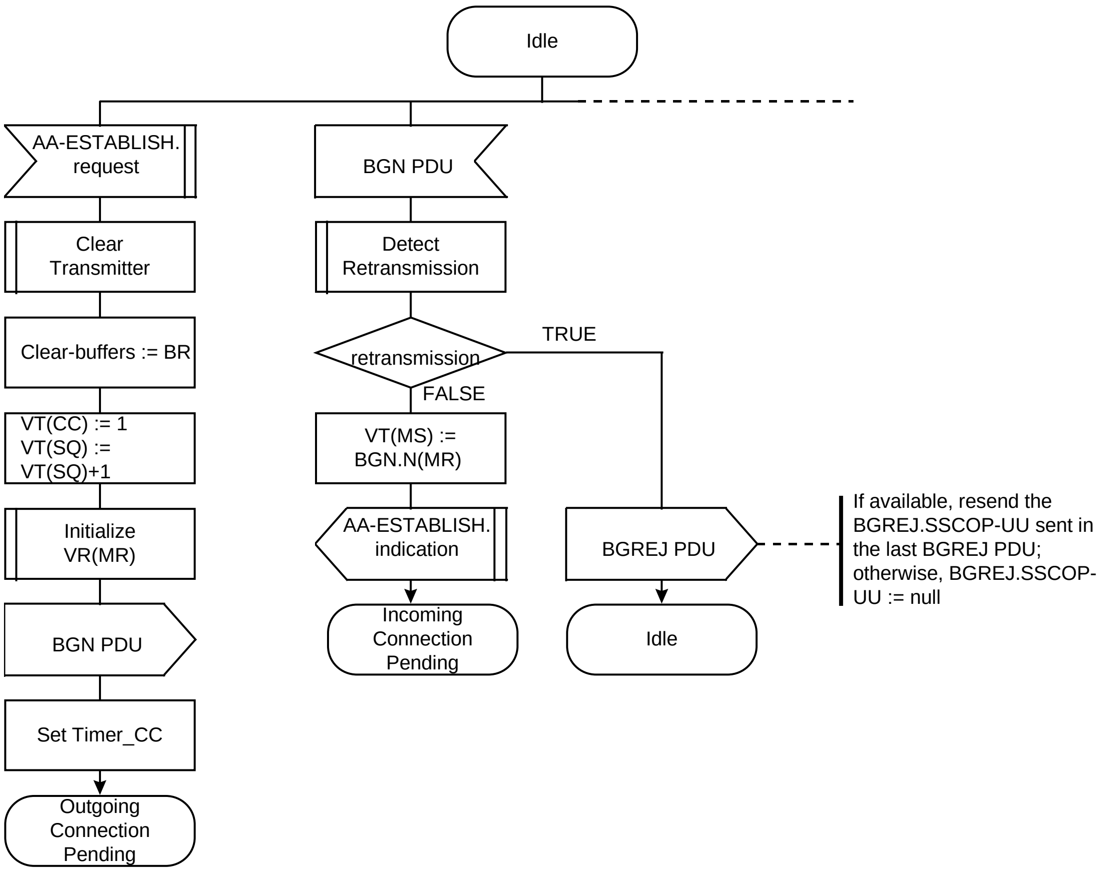

Figure 2 Part of SSCOP SDL diagram

The code snippet below contains the corresponding input methods from SSCOP state machine code. One can see that one SDL block translates into approximately one line of code, so reading code with the SDL diagram is quite easy.

```
void sscopIdle :: aaESTABLISHreqAct(
    aaESTABLISHreq *messenger_,
    pfProtocol *protocol_) const
{
    sscopProtocol *sscop = (sscopProtocol *) protocol_;
    clearTransmitter(sscop);
    sscop->_clearBuffers = messenger_->getBufferRelease();
    sscop->_VT_CC = 1;
    sscop->_VT_SQ++;
    initializeVR_MR(sscop);
    sscop->sendBGNpdu(messenger_->getSSCOP_UU());
    sscop->_timerCC.start();
    changeState(protocol_,
                sscopOutgoingConnectionPending::instance());
    return;
}


void sscopIdle :: sscopBGNpduAct(
    sscopBGN_PDU *messenger_,
    sscopProtocol *protocol_) const
{
    if (detectRetransmission(protocol_,
                             messenger_->getN_SQ()) != 0)
    {
        protocol_->sendBGREJpdu(protocol_->_lastBGREJsscop_uu);
    }
    else
    {
        protocol_->_VT_MS = messenger_->getN_MR();
        protocol_->
            sendAaESTABLISHind(messenger_->getSSCOP_UU());
        changeState(protocol_,
                    sscopIncomingConnectionPending::instance());
    }
    return;
}
```

1.4 4 Features implemented

Most of the features of SSCOP are implemented in this initial release, but some details still remain partially or totally unimplemeted. See TODO file in SSCOP module directory for a detailed list of missing features.

1.5 5 Known bugs and flaws

This version of SSCOP is not yet rigorously tested, so hidden bugs may remain in the code. One known bug in SSCOP is that the handling of SSCOP PDU sequence numbers is not currently correct. Sequence numbers are integers (modulo 2^24), but the present code can't handle the situation where sequence numbers wrap over 2^24, so if an SSCOP instance runs over a long time, it will eventually encounter this bug.

1.6 6 Future development

Because SSCOP is a relatively critical part of the signalling stack (all actual signalling protocols depend on the reliable data transport service provided by SSCOP), the bugs and missing features should be corrected. TODO file of SSCOP module contains some development ideas, more may arise if SSCOP gets tested more formally.

1.7 7 Statistics

SSCOP development was concentrated into July 1996. The initial implementation took about one man-month and further development took the rest of the time seen in the table below.

In the metrics table, the great number of PDUs and primitives contribute to high number of classes. Most of them are nearly identical and very simple at the same time, however.

Activity	Research	Design	Coding	Reviews	Total
Duration (h)	30	30	170	40	270

Table 1 Duration of activities

Lines Of Code (LOC)	Number of files	Number of classes
5253	31	59

Table 2 Metrics

1.8 8 References

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