1. Call Control

Call Control is the top layer of TOVE architecture providing call controlling with connection to a possible IN-architecture (Intelligent Network). CC also controls a switch fabric through VE (Virtual Exchange).

1.1 1 Introduction

This call control implementation is based on the ITU-T standard Q.1224 (draft).

1.2 2 Architecture

The Call Control module includes following classes:

- ccProtocol
- Acknowledge Object
- Originating and Terminating states and detection points
- Cross Connector Mux
- SCF Adapter
- Multi Point Mux
- Multi Point Mux accessor
- Detection Point data Objects
- Triggers

At run time the ccProtocol has references to messenger objects (e.g. SETUPind), because some of the operation is hidden to these objects.

Figure 1 shows relations between different objects in call setup situation: the SETUPind messenger has arrived, the protocol is initialized and the connection is established to the terminating side. Next chapter describes modules more closely.

1.3 3 Implementation details and features implemented

1.3.1 3.1 Protocol

The CC Protocol is parametrized with a state structure which makes it an Originating side or a Terminating side protocol. O-side is the starting side and T-side the receiving side in switch's perspective.

1.3.2 3.2 Acknowledge Object

Acknowledge object is a callback object used by VE. Object is located in the CC protocol class providing function interface for VE. VE function calls are translated to messengers and sent to the CC protocol. By this the communication can be asynchronous.

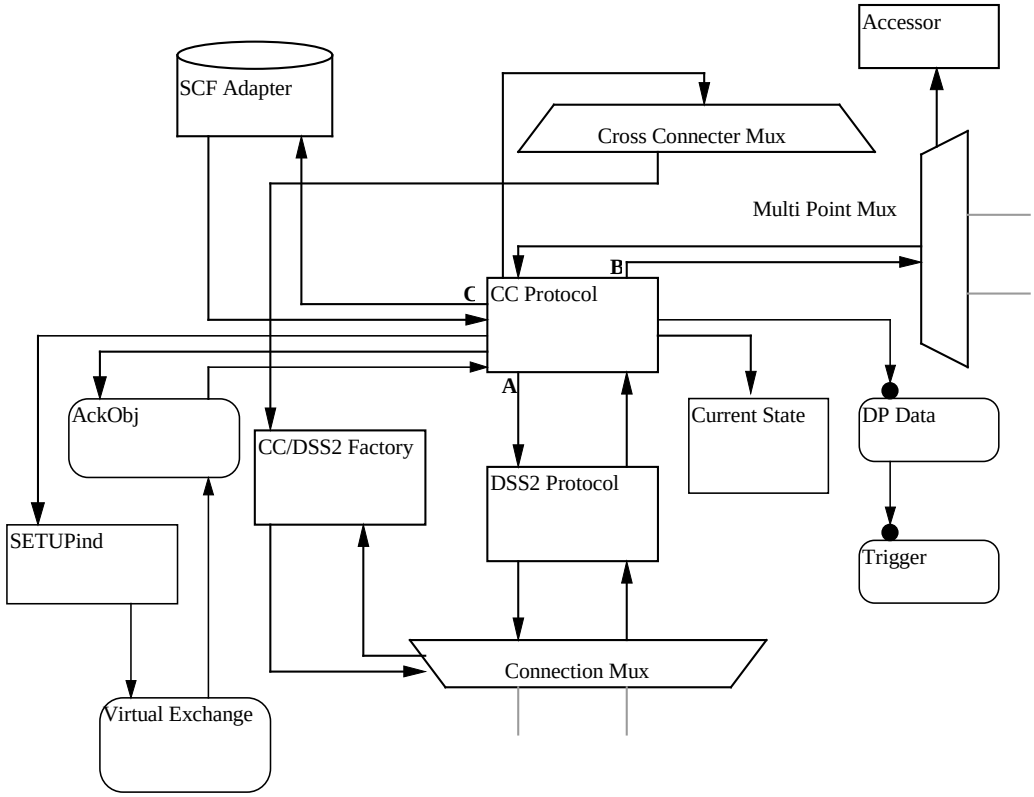

Figure 1 Object relations at run time

1.3.3 3.3 States

Figure 2 presents a CC state structure. In CC there are two kinds of states: Point In Call states (PIC) and Detection Point states (DP). Both state structures are derived from interface classes up, down and binap, which present possible messengers and their inputs. Up interface presents messengers coming from a down layer (e.g. DSS2). The down interface is included because "down" messages (e.g. requests) are used between originating and terminating CC protocols. CC states are also derived from a common ccState class and from ccOstate or ccTstate classes. These classes implement common functionality.

1.3.4 3.4 Cross Connector Mux

There is only one Cross Connector Mux (CCM) in the whole system. CCM holds references to all link factories. It is used to route messengers to a correct link. Every messenger holds an address which is translated to a link number in a routing service outside CC. CCM uses this link number to guide messengers to the correct link.

1.3.5 3.5 SCF Adapter

SCF (Service Control Function) adapter serves as an interface to IN-architecture.

1.3.6 3.6 Multi Point Mux

Multi Point Mux is used in point-to-multipoint call to route messengers to a correct session of current call. See chapter future development.

1.3.7 3.7 Detection Point Data Objects

Every instance of CC protocol has its own DP Data Objects. These objects hold specific run time information of DPs: status, type and triggers.

1.3.8 3.8 Triggers

Each Detection Point Data Object can hold limited amount triggers. Triggers form DP a criteria, which are conditions that must be met in order to notify the SCF that the DP was encountered. The DP criteria can be controlled by INAP (Intelligent Network Application Protocol) messages.

1.3.9 3.9 Messages and transitions

The CC protocol accepts all primitives described in a standard Q.2931 (DSS2). These primitives are also used between O- and T-sides (internal messages). The signaling interface primitives (classes) encapsulates some of the CC-operation, e.g. establishment of connection, reply messages etc. Figure 3 and 4 describe indications (primitives) and state transitions for both O- and T-sides. The CC protocol also accepts and sends some of INAP messages.

1.3.10 3.10 Fabric interface

The CC protocol is connected to a Virtual Exchange (VE) interface. See document X.

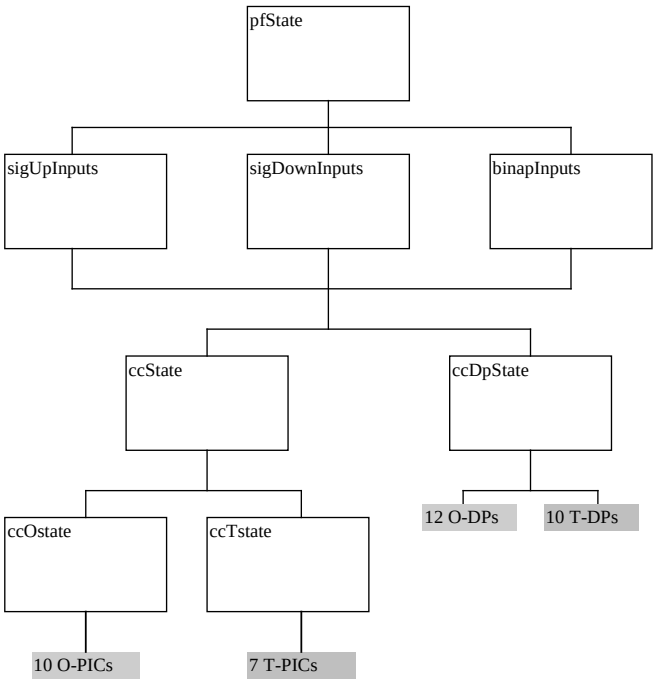

Figure 2 State structure

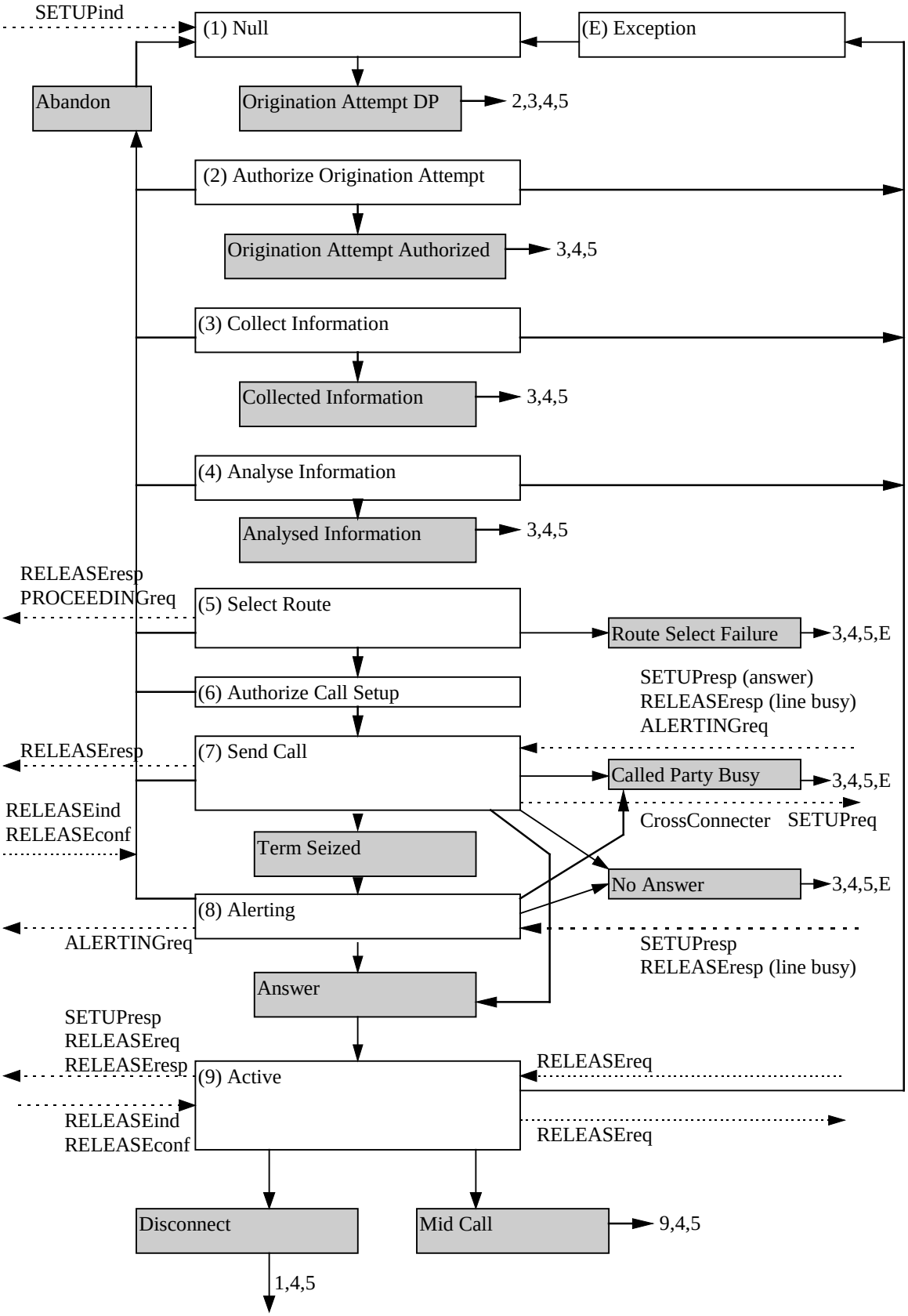

Figure 3 Set of transitions and indications for the originating BCSM

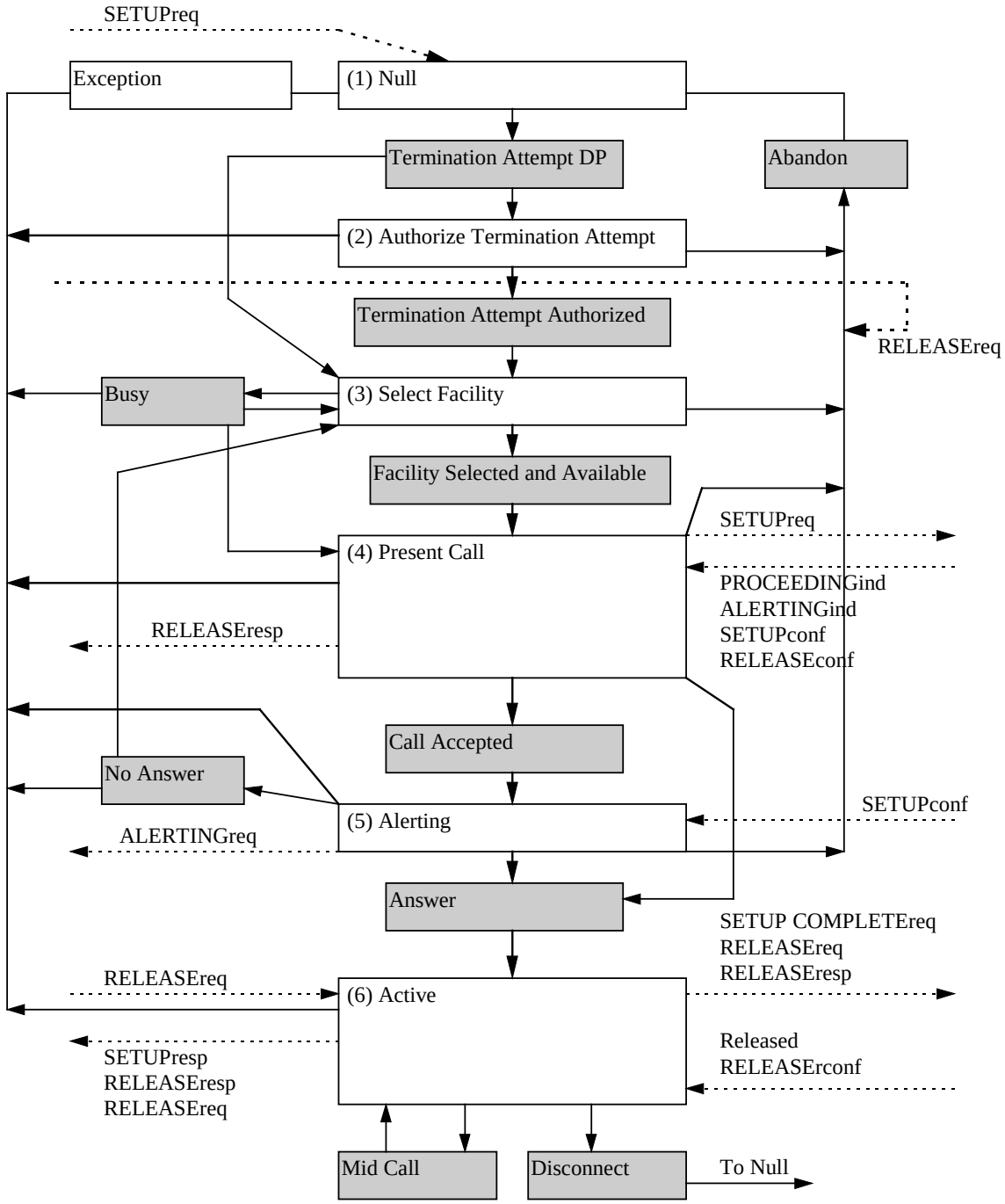

Figure 4 Set of transitions and indications for the terminating BCSM

1.4 4 Known bugs and flaws

The current implementation of setup message handling is optimized for speed. One data block travels from O-DDS2 to T-DSS2 in different messengers. This solution is not very robust and it should be redesigned simultaneous with new primitive/PDU design.

1.5 5 Future development

The basic architecture supports point-to-multipoint calls, but in lack of specifications the messages invoking this kind of call are not implemented. The IN part of the CC protocol needs further development, e.g. management interface for DPs and more INAP messages. Possible IDL-interface between CC and INAP-gateway is also planned.

1.6 6 Statistics

Following tables describe duration of activity and software metrics.

Activity	Prototypes	Design	Coding	Reviews	Total
Duration (h)	160	200	160	470	830

Table 1 Duration of activities

Lines Of Code (LOC)	Number of files	Number of classes
4160	133	48

Table 2 Metrics

1.7 7 References

1. ITU-T, *Q.1224 CS-2 Draft*.