Formal Specification and Analysis of an ISO Communications Protocol

J.R.Rowson
Department of Computer Science, Queen Mary & Westfield College,
University of London, Mile End Road, London E1 4NS, United Kingdom

1. INTRODUCTION

This paper demonstrates the practical use of formal methods in a very striking manner by locating an unexpected anomaly in the internationally specified protocol, ISO 8802-2 [1]. This standard describes the service and implementation of the Logical Link Control (LLC) sub-layer protocol constituting the 'upper' portion of the OSI Reference Model's data link layer. In this framework, the peer LLC sub-layer components in different computers communicate with each other by exchanging Protocol Data Units (PDUs) via the Medium Access Control (MAC) sub-layer, which constitutes the remaining 'lower' portion of the data link layer. The LLC sub-layer communicates with its users in the network layer through Service Access Points (SAPs).

This extended abstract includes some results from the full paper's systematic development of progressively more realistic CCS specifications of the service and implementation of the connection phase of the protocol. The purpose of this phase is the establishment of a connection between two user processes in the network layer. The culmination of the development is a full specification of the standard's error recovery methods designed to cope with the unreliability of the MAC. Analysis of this via the University of Edinburgh's Concurrency Workbench provides a verification of the presence of the anomaly.

2. LLC IMPLEMENTATION WITH ERROR FREE COMMUNICATION

For simplicity the CCS specifications are designed for the limited situation of just two network users and two initial assumptions are made in order to postpone consideration of the error recovery procedures. The assumptions are that the MAC provides error free communication (on rec_i and $send_i$) and that all response PDUs are received within the allowed period of time.

Each LLC component starts in the ADM state in which it is ready both to accept a Connect Request message from the local user at the SAP and to receive a SABME PDU from the MAC.

$$ADM \overset{\text{def}}{=} Connect\_Request.ADM' + rec(x).\textbf{if } x{=}sabme \textbf{ then } \overline{Connect\_Indication}.CONN$$

If it receives a request for connection from the SAP then it decides locally (the criteria are not given in the standard) whether or not to proceed with the connection. This locally non-deterministic choice is modelled by state *ADM'* using the silent action, τ.

$$ADM' \overset{\text{def}}{=} \tau.\overline{Disconnect\_Indication}.ADM + \tau.\overline{send}(sabme).SETUP$$

If the LLC component decides to proceed with the connection then it sends a SABME and enters the SETUP state where it waits for a response via the MAC. Receipt of a UA or DM PDU indicates, respectively, that the remote user has accepted or refused the connection. The third admissible type of PDU expected is a SABME, which indicates the situation of simultaneous connection. In this case UA PDUs are exchanged in both directions.

$SETUP \overset{\text{def}}{=} rec(x).$ **if** $x{=}sabme$ **then** $\overline{send}(ua).SETUP$
else if $x{=}ua$ **then** $\overline{Connect\_Confirm}.NORMAL$
else if $x{=}dm$ **then** $\overline{Disconnect\_Indication}.ADM$

In state CONN, the LLC component is waiting for a response from the local user that will indicate either acceptance or refusal of the remote request for a connection.

$CONN \overset{\text{def}}{=} Connect\_Response.\overline{send}(ua).NORMAL + Disconnect\_Request.\overline{send}(dm).ADM$

State NORMAL indicates that connection has been established and transmission of data between the network users may begin. So, $NORMAL \overset{\text{def}}{=} \mathbf{0}$.

The two LLC components LLC_i (i =1,2) are defined using appropriate relabelling functions f_i and are combined with the MAC in the implementation of the protocol as follows.

$LLC_i \overset{\text{def}}{=} ADM[f_i][rec_i/rec, send_i/send]$

$IMPL \overset{\text{def}}{=} (LLC_1 \mid MAC \mid LLC_2)\backslash\{rec_1, send_1, rec_2, send_2\}$

3. LLC SERVICE

The full paper includes the development of two alternative service specifications, each of which captures, to a different degree of abstraction, the behaviour of the protocol. The first service description is expressed as the concurrent composition of two processes, each having the same 'shape' as the LLC agents given above. This service description, which is shown to be observation congruent with the implementation, retains all details of the possible inter-leavings that can arise from the local decisions of the LLC agents over whether or not to proceed with a request to establish a connection. These interleavings, however, concern behaviour that is hidden from the network user processes, or other external observers, via communication at the SAPs. The second version of the service, expressed as a single process, makes use of the testing preorder to abstract away from these hidden details while maintaining the essence of the external behaviour of the protocol from the point of view of its users.

4. IMPLEMENTATION OF ERROR RECOVERY METHODS

It is possible to distinguish in the standard three distinct methods for recovering from the following three types of error which may occur:

(a) the loss of SABME PDUs by the MAC
(b) the loss of response (UA and DM) PDUs by the MAC
(c) the failure by a LLC component to receive a response to a SABME PDU within a predetermined, finite period of time.

From the point of view of the sender of a SABME all three types of error are encompassed by (c) and are dealt with identically by retransmission of the SABME when the wait for a reply is timed out. Both the amount of time that a LLC component waits before retransmission of a SABME and the maximum number of retransmissions are undefined parameters of the standard. The only reference to them is that for "*the proper operation*" of the protocol the amount of time that elapses before expiry of the acknowledgement timer is required to be greater than the maximum time between sending a SABME and receiving a response during "*normal network operation*".

The paper progressively extends the CCS specifications to imitate the error recovery methods. This includes modelling the unreliable MAC and acknowledgement timers (which communicate via *at*) and the extension of the SETUP state to retransmit SABMEs as follows:

$$\begin{aligned} SETUP \overset{\text{def}}{=}\ & (rec(x).\ \textbf{if}\ x{=}sabme\ \textbf{then}\ \overline{send}(ua).SETUP' \\ & \quad \textbf{else if}\ x{=}ua\ \textbf{then}\ \overline{at}(stop).\ \overline{Connect\_Confirm}.NORMAL \\ & \quad \textbf{else if}\ x{=}dm\ \textbf{then}\ \overline{at}(stop).\ \overline{Disconnect\_Indication}.ADM\ ) \\ & + expired.ADM' \end{aligned}$$

$$ADM' \overset{\text{def}}{=} \tau.\ \overline{Disconnect\_Indication}.ADM + \tau.\ \overline{send}(sabme).\ \overline{at}(start).SETUP$$

SETUP now waits either for a PDU or for the timer to expire. If the timer expires first then the decision whether or not to retransmit is imitated by the modified version of *ADM'*. *SETUP'* (given only in the full paper) is a new state introduced to cope with simultaneous connection.

The different types of error demand different recovery methods in the accepting component, however. It is in considering the loss of a UA that a situation of particular interest arises. Here, the sender of the UA that is lost will eventually receive any retransmitted SABME in state NORMAL. The recovery method invoked here forms a small part of the *resetting* phase of the protocol and is exactly that used in normal connection except that a *Reset_Indication* (instead of *Connect_Indication*) is sent to the local user which may then reply with a *Reset_Response* (instead of *Connect_Response*). For the subsequent analysis the artificial action *Ready* is now included in *NORMAL* to indicate that the LLC agent is ready to start transmitting data.

$$NORMAL \overset{\text{def}}{=} (rec(x).\textbf{if}\ x{=}sabme\ \textbf{then}\ \overline{Reset\_Indication}.RESET\_CHECK) + Ready.\mathbf{0}$$

$$RESET\_CHECK \overset{\text{def}}{=} CONN[Reset\_Response/Connect\_Response]$$

5. DETECTION OF THE ANOMALY

While the CCS specification of this error recovery method was being developed, the suspicion grew that there was an anomaly in the standard's method of dealing with this aspect of type (b) transmission errors. Our concern was that the method seemed to overlook the fact that the decision by the initiating component to stop retransmission of SABMEs may occur when the accepting component has already entered state NORMAL. This would lead to the situation in which the initiator is in state ADM (where no connection is established) and the acceptor is in state NORMAL (where connection is assumed to be established).

To verify this, a *partial* specification of the required behaviour is developed in the process logic of the Concurrency Workbench. The formulation of (half of) the relevant correct behaviour is: "in all reachable states the system is not ready to accept both *Connect_Request*$_1$ and *Ready*$_2$".

$$\nu X.(\neg\ (<Connect\_Request_1>\textbf{true}\ \wedge\ <Ready_2>\textbf{true})\ \wedge\ [.]X)$$

The Workbench confirmed that the extended version of *IMPL* did *not* have this property, implying that it is possible for just one component to have 'established' a 'connection'. We do stress that this will only occur when the MAC is so unreliable as to lose in succession one UA followed by *n* SABME or UA PDUs, with *n* the given maximum number of retransmissions of SABMEs. We do not know if this case is one that might be considered by the standard as being outside "*normal network operation*". However, since the other recovery methods seem to cater for abnormal operation, it is surprising to us that the standard does not assume the worst here as well. One reason, perhaps, is because there appears to be no obvious cure for this 'bug'.

REFERENCES

1. International Standard ISO 8802-2: 1989. Part 2: Logical link control.