Modelling and analysis of a telecommunications protocol

In this exercise, you will model the DECT (Digital European Cordless Telephone) error-detecting data transfer protocol. This protocol is called "unilateral jump". Unilateral jump is based on alternating bit protocol, but is somewhat more versatile.

The idea behind alternating bit protocol is very simple. Each data packet has a number (modulo 2). What the protocol ensures is that a new data packet will not be transferred until the current one is positively acknowledged. Hence, the "Transmitter" will not send packet 0 until packet 1 has been acknowledged. NOTE: Here "Transmitter" is understood as the transmitter of DATA. Naturally, the transmission link has to function also in the other direction. This is because the "Receiver" must acknowledge the data packets sent by the "Transmitter".

Unilateral jump adds one feature to the plain alternating bit protocol. If the "Transmitter" does not receive a positive acknowledgement for a packet x (possibly after a number of retransmission attempts), it may spontaneously start to send data packet (x+1) (modulo 2). When this happens, the transmitter jumps over data packet x without getting a positive acknowledgement for it. This is the basis for the name of the protocol.

The behaviours of "Transmitter" and "Receiver" are described in the following section. The descriptions come directly from the ETSI specification prETS 300175-3.

Receiver:

The three state variables at the receiver are:

• LAST-BCK meaning "what packet number was transmitted in the last BCK bit to indicate the number of the next expected I_P segment";

• LAST-PKT meaning "what I_P packet number appeared in the last databurst with correct received A-field and containing I_P data";

• THIS-PKT meaning "what I_P packet number appears in the databurst just received with correct received A-field and containing I_P data".

The receiver, after evaluating the A-CRC of the received I_P packet evaluates the state variables following table 1 to know how the transmitter acted; after that, it evaluates the Rb-CRCs of all the B subfields and shall act as indicated in table 2.

THIS-PKT=LAST-PKT	THIS-PKT=LAST-BCK	How transmitter acted	Resulting case
yes	yes	retransmit	(a)
no	yes	normal advance	(b)
yes	no	unnecessary retransmit	(c)
no	no	jump	(d)

Table 1

Table 1 State	Rb-CRC result	how Receiver shall act
(a)	Rb-CRC passed	LAST-BCK=(LAST-BCK+1) MOD 2; RX requires a new I_P packet
(a)	Rb-CRC failed	LAST-BCK=LAST-BCK; RX requires the retransmission of the I_P packet
(b)	Rb-CRC passed	LAST-BCK=(LAST-BCK+1) MOD 2; RX requires a new I_P packet
(b)	Rb-CRC failed	LAST-BCK=LAST-BCK; RX still requires the same I_P packet
(c)	Rb-CRC passed	LAST-BCK=LAST-BCK; RX still requires the same I_P packet
(c)	Rb-CRC failed	LAST-BCK=LAST-BCK; RX still requires the same I_P packet
(d)	Rb-CRC passed	LAST-BCK=LAST-BCK; RX requires a new I_P packet (alignment to TX jump); The I_P packet just received is accepted
(d)	Rb-CRC failed	LAST-BCK=(LAST-BCK+1) MOD 2; RX requires the retransmission of the I_P packet (alignment to TX jump); The I_P packet just received is rejected

Table 2

Transmitter:

Define: LTI_P=packet number of the last transmitted I_P segment.

There are two state variables at the transmitter:

• $\Sigma ACKN$ meaning "has the receiver acknowledged the LTI_P at least once?"; For duplex bearer (as in this case) the LTI_P is acknowledged by receiving a packet with $Q2$ bit set to 1. *Note:* $\Sigma ACKN$ is reset to "no" if the transmission of a new I_P packet starts.

• $LAST-BCK$ meaning "what was the setting of the last correct received BCK bit (with the BCK bit the receiver reports the next expected I_P packet)?"; *Note:* If $\Sigma ACKN$ switches to "yes" also a new BCK is received.

On receipt of new segments, the transmitter shall look at the values of its state variables and act according to table 3 In state (b) the transmitter can choose between retransmission and jump. Its choice is reflected in the pkt number chosen. If jump the pkt number toggles, if retransmit the pkt number is unchanged.

Case	$\Sigma ACKN$	LTI_P=LAST-BCK	How transmitter shall act
(a)	yes	no	normal advance or retransmit
(b)	yes	yes	retransmit/jump
(c)	no	no	retransmit
(d)	no	yes	retransmit/jump

Table 3

Your Task

The file `ujump.net`, downloadable from the course homepage contains a partial `PROD` model of the previous specification. There are 12 transitions missing from it. Your task is to complete this `PROD` code. Generate the reachability graph of your model and make sure that there are no deadlocks in it. In your answer, return the complete `PROD`-code and the statistics of the reachability analysis. You can obtain the statistics of your reachability graph with `probe`'s command `statistics`.

N.B.: The home assignment is an individual task. No signs of cooperation shall be visible in the returned assignments.

The deadline for the Home Assignment #5 is Dec 1, 2000; use the post box between rooms B336 and B337 in the Computer Science building. You can also return the home assignment by email in PostScript or LaTex formats. The homework assistant is Jukka Honkola (email: Jukka.Honkola@hut.fi).

Hints

- All the places and transitions of the "Receiver" side contain the prefix rx_ in their name.

- The state variables on the "Receiver" side are represented as places in the `PROD` model. The mapping from state variables to places is: LAST-BCK=rx_last_bck, LAST-PKT=rx_last_pkt. There is no separate place in the `PROD` model for THIS-PKT, because the number of the current message may be read off the packet itself.

- Transitions rx_case_?_detect take a 5-tuple from place "ether" and place it into one of the local places on the "Receiver" side. This selection is based on Table 1 test. Transitions rx_case_?_send place either a negative or a positive acknowledgement into place "ether" according to the cases indicated in Table 2.

- All the places and transitions on the "Transmitter" side contain the prefix tx_ in their name.

- Mapping between the "Transmitter" state variables and the places in the partial `PROD` code is: $\Sigma ACKN$=tx_acks, LTI_P=tx_last_tx_packet, ja LAST-BCK=tx_last_bck.

- Transitions tx_detect_acks_prev_acks and tx_detect_acks_no_prev_acks update the state variables (ie. places) on "Transmitter" side. The first one takes care of the situation, where the incoming acknowledgement message is unnecessary (ie. it repeats a previously received acknowledgement message). The second handles the situation where the acknowledgement received is the first one received for that particular packet.

- Errors are simulated through the Transmitter's possibility of sending a good or a bad CRC with each message.

- In this model, it is assumed that the mobile station (PP=Portable Part in DECT terminology) acts as the "Transmitter". Similarly, the base station (FP=Fixed Part in DECT terminology) is assumed to act as the "Receiver".

- 'ether' place models air interface between the "Transmitter" and the "Receiver". It contains a 5-tuple. The first field in the tuple indicates the packet number. The second tells who sent the message (naturally, the "Receiver"=FP listens only to packets originating from the "Transmitter"=PP and vice versa). The third field contains the bck-bit. It tells the "Transmitter" which packet the "Receiver" is expecting next. The fourth field contains the Rb-CRC (checksum). An error-free Rb-CRC is modelled by the value 1, whereas value 0 indicates an error. The last field contains the Q2-bit. If a data packet is flawlessly received by the "Receiver", it shall send back a message with Q2 bit set to 1. Otherwise, it will send a message where Q2 bit equals 0 (look at the transition rx_case_a_send).