University of South Australia

WAP Forum Input Document
Inconsistencies in the Wireless Transaction Protocol

Cooperative Research Centre for Satellite Systems

Authors: Steven Gordon and Jonathan Billington[1]

Contact details: Steven Gordon
Cooperative Research Centre for Satellite Systems
University of South Australia
Mawson Lakes Boulevard
Mawson Lakes SA 5095
Australia
Phone: +61-8-8302-3606
Fax: +61-8-8302-3873
Email: sgordon@spri.levels.unisa.edu.au

[1]This work was carried out with the financial support from the Commonwealth of Australia through the Cooperative Research Centres Program.

Contribution Topic: Modification to the Class 2 Wireless Transaction Protocol

Executive Summary:

Formal analysis of the WAP Class 2 Wireless Transaction Protocol has revealed several inconsistencies in the specification. These are explained, and where possible, changes to the specification are proposed to improve the protocol. The inconsistencies are:

1. The counter RCR may be incremented to a value greater than RCR_MAX.
2. Two TR-Invoke.cnf primitives can be delivered to the Initiator user (within the context of one transaction).
3. The TR-Result.req primitive may immediately follow a TR-Invoke.ind primitive at the Responder user when User Acknowledgement is on.
4. A transaction may be aborted without the Responder user being notified.
5. The semantics of "Abort transaction" in the state tables is not defined.

Changes to the state tables are proposed to remedy the first 4 problems. A definition is required in the text for the final problem. Typographical errors in the state tables are also pointed out.

Contribution Category: Proposed functional modification of an existing specification: the Wireless Transaction Protocol.

Reason for Contribution: To remove ambiguities/errors from the specification.

Urgency: High. Since WAP is already being implemented, ambiguities can lead to incompatible implementations.

IP/Legal Issues: None.

1 Introduction

As part of a research project, we have performed formal analysis of the WAP Class 2 Wireless Transaction Protocol [3]. From this analysis several ambiguities in the specification have been identified. We report our findings here so that the next public version of the specification is unambiguous. The contribution category is proposed functional/editorial modifications of an existing specification.

The formal analysis has been on WTP Version 1.1 (11-June-1999) [3], the latest approved and publicly available version of the specification. Currently only the Class 2 service and protocol has been analysed. Details on the techniques used for the analysis are not discussed here. However, the modelling is at a level of abstraction that allows us to examine all possible events in the protocol e.g. at some state an event either can or cannot occur – there is no explicit modelling of time. Further information can be obtained in [1, 2] or by contacting us.

The following section lists and discusses the ambiguities found in the specification. The discussion aims to identify exactly where the ambiguities can be found and to propose solutions. Typographical errors are also reported. Possibilities for further input are discussed in Section 3. For completeness, where possible, a sequence of events from the state tables that leads to each ambiguity is given in Appendix A.

2 Contribution

There are 7 comments on the specification. For each, the area in the specification being referred to is given, the problem described and a proposed solution given. The additions to the state tables are shown in bold.

Comment No.: 1

Reference to Spec.: §10.5, page 54, WTP Initiator RESULT WAIT state table, RcvAck event with TIDve set

Problem: With the Initiator in the RESULT_WAIT state, receiving an Ack PDU (with TIDve set) may increment RCR to a value greater than RCR_MAX. This may cause errors when TimerTO_R occurs in this state. Although at an implementation level it may be obvious to limit RCR < RCR_MAX, to be consistent with other state table entries the new condition should be added.

Solution: The proposed replacement text is shown in bold in Table 1.

Table 1: State table action with new condition limiting RCR

Initiator RESULT WAIT			
Event	Condition	Action	Next State
RcvAck	TIDve Class=2|1 **RCR<RCR_MAX**	Send Ack(TIDok) Increment RCR Start timer, R [RCR]	RESP WAIT

Comment No.: 2

Reference to Spec.: §10.5, page 54, WTP Initiator RESULT WAIT state table

Problem: It may be possible that two TR-Invoke.cnf primitives are delivered to the Initiator user. From the primitive sequence table for transaction Class 2 (page 23, §7.3.2, Table 6) this is not possible. Figure 1 gives an example scenario where this error occurs. Note User Acknowledgement is off and RCR_MAX, AEC_MAX are 1.

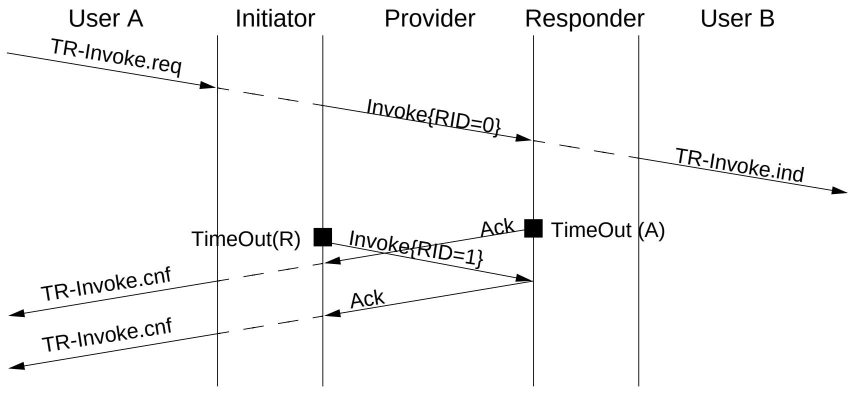

Figure 1: Error scenario: Two TR-Invoke.cnf primitives

An Ack PDU from the Responder results in the delivery of a TR-Invoke.cnf primitive to the Initiator user. However, previously the Invoke PDU was retransmitted, and the Responder replies with an Ack PDU, resulting in a second TR-Invoke.cnf primitive. The second Ack PDU is retransmitted to cope with the situation when the first Ack PDU is lost. If it were lost, the first TR-Invoke.cnf would not be generated and everything would proceed correctly.

Solution: To remedy this situation we suggest that once a TR-Invoke.cnf has been generated and a second Ack PDU is received, simply do not deliver the TR-Invoke.cnf (and optionally, log the receipt of the Ack PDU). This will require a variable to indicate whether the TR-Invoke.cnf has been sent. For example, the entry in Table 2 could be added to the Initiator RESULT WAIT state table.

Table 2: New state table action disallowing two TR-Invoke.cnfs

Initiator RESULT WAIT			
Event	Condition	Action	Next State
RcvAck	**Class==2** **TR-Invoke.cnf sent==True**	**Stop timer** **HoldOn=True** **Log Ack PDU**	**RESULT WAIT**

Comment No.: 3

Reference to Spec.: §10.6, page 56, WTP Responder INVOKE RESP WAIT state table, TR-Result.req event

Problem: When User Acknowledgement is on it is possible for a TR-Result.req to immediately follow a TR-Invoke.ind primitive. From the primitive sequence table for transaction Class 2 (page 23, §7.3.2, Table 6) this is not possible.

Solution: This can be fixed by introducing a new condition in the state tables as shown in Table 3.

Table 3: State table entry with new condition restricting TR-Result.req

Responder INVOKE RESP WAIT			
Event	Condition	Action	Next State
TR-Result.req	**Uack==False**	Reset RCR Start timer R[RCR] Send Result PDU	RESULT RESP WAIT

Comment No.: 4

Reference to Spec.: §10

Problem: The semantics of “Abort transaction” as an action in the state tables is not defined. We have assumed it is only used to describe an abstract procedure. It does not correspond with sending of PDUs, issuing primitives, or modifying variables/timers.

Solution: A definition in the specification (§10) is necessary.

Comment No.: 5

Reference to Spec.: §10.6, page 56, WTP Responder INVOKE RESP WAIT state table, TimerTO_A event with AEC==AEC_MAX

Problem: When User Acknowledgement is on a transaction may be aborted with the Responder user not being notified (i.e. by a TR-Abort.ind). For example, Fig. 2 shows a sequence that constitutes a transaction. After the number of timeouts at the responder reaches the maximum, the transaction is aborted. The retransmitted Invoke PDU initiates a TID verification which fails. At the end of the sequence the initiator is in the NULL state and the responder in the LISTEN state, indicating that both entities have discarded any state information for that transaction. However, the responder user has been issued a TR-Invoke.ind but no other primitive to indicate the end of the transaction.

Solution: After the second timeout occurred at the Responder, a TR-Abort.ind primitive should have been delivered to the user. The proposed change is given in Table 4.

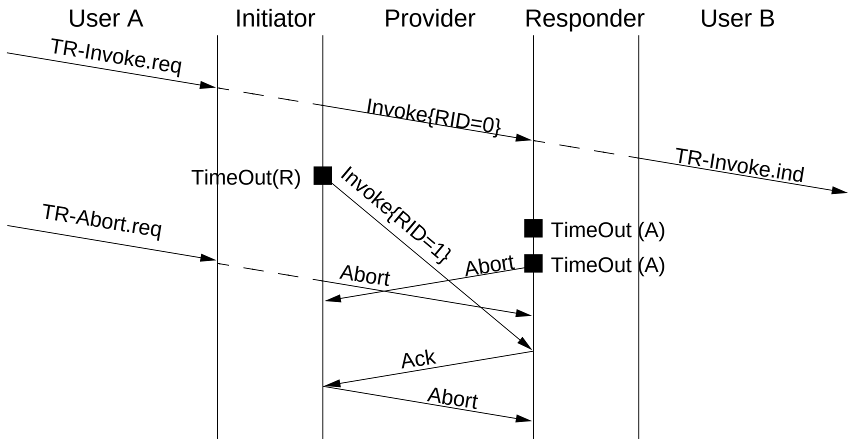

Figure 2: Error scenario: Invalid halt state

Table 4: State table entry with new action generating TR-Abort.ind

Responder INVOKE RESP WAIT			
Event	Condition	Action	Next State
TimerTO_A	AEC==AEC_MAX	Abort transaction **Generate TR-Abort.ind** Send Abort PDU (NORESPONSE)	LISTEN

The following are typographical errors in the state tables:

Comment No.: 6

Reference to Spec. : §10.5, page 54, Initiator RESULT WAIT state table, TimerTO_R event with RCR<RCR_MAX.

Problem: There is a duplicate entry.

Solution: Delete the duplicate entry.

Comment No.: 7

Reference to Spec. : §10.6, page 57, Responder RESULT WAIT state table, RcvInvoke event with RID=1, Ack PDU already sent

Problem: Typo: “Resent Ack PDU”

Solution: Replace with “Resend Ack PDU”

3 Other Issues

The findings presented here have resulted from initial analysis of WTP. The work is continuing, with the effects of multiple transactions and errors yet to be analysed. Further results will be reported to the WAP Forum if necessary. Indication of any similar work being performed by the WAP Forum would be appreciated.

A Example Sequences

Where applicable, an example sequence of events is given that leads to the inconsistency described in the corresponding point in Section 2.

Comment No.: 1

Conditions: UserAck==False, RCR_MAX==1

Sequence:
1. Initiator NULL: TR-Invoke.req
2. Responder LISTEN: RcvInvoke (InvalidTID)
3. Initiator RESULT WAIT: TimerTO_R (RCR<RCR_MAX) /* RCR==1 */

4. Initiator RESULT WAIT: RcvAck (TIDve) /* RCR==2 */

Comment No.: 2

Conditions: UserAck==False

Sequence:
1. Initiator NULL: TR-Invoke.req
2. Responder LISTEN: RcvInvoke (ValidTID)
3. Responder INVOKE RESP WAIT: TimerTO_A
4. Initiator RESULT WAIT: TimerTO_R (RCR<RCR_MAX)
5. Initiator RESULT WAIT: RcvAck
6. Responder RESULT WAIT: RcvInvoke (RID=1, Ack PDU already sent)
7. Initiator RESULT WAIT: RcvAck

Comment No.: 3

Conditions: UserAck==True

Sequence:
1. Initiator NULL: TR-Invoke.req
2. Responder LISTEN: RcvInvoke (ValidTID)
3. Responder INVOKE RESP WAIT: TR-Result.req

Comment No.: 4

Sequence: Not applicable.

Comment No.: 5

Conditions: UserAck==True

Sequence:
1. Initiator NULL: TR-Invoke.req
2. Responder LISTEN: RcvInvoke (ValidTID)
3. Initiator RESULT WAIT: TimerTO_R (RCR<RCR_MAX)
4. Responder INVOKE RESP WAIT: TimerTO_A (AEC<AEC_MAX)
5. Responder INVOKE RESP WAIT: TimerTO_A (AEC==AEC_MAX)
6. Initiator RESULT WAIT: TR-Abort.req
7. Initiator (Table 33, §10.2, page 50): Abort PDU, no matching outstanding transaction
8. Responder LISTEN: RcvInvoke (InvalidTID)
9. Initiator (Table 33, §10.2, page 50): Ack PDU, TIDve flag set, no matching outstanding transaction
10. Responder TIDOK WAIT: RcvAbort

Comment No.: 6

Sequence: Not applicable.

Comment No.: 7

Sequence: Not applicable.

References

[1] S. Gordon and J. Billington. Modelling the WAP Transaction Service using Coloured Petri nets. In Hong-Va Leong, Wang-Chien Lee, Bo Li, and Li Yin, editors, *Proceedings of the First International Conference on Mobile Data Access*, LNCS 1748, pages 105–114, Hong Kong, 16-17 December 1999. Springer-Verlag. ISBN: 3-540-66878-0.

[2] S. Gordon and J. Billington. Analysing the WAP class 2 Wireless Transaction Protocol using Coloured Petri nets. Submitted to *21st International Conference on Application and Theory of Petri Nets*, Aarhus, Denmark, 26-30 June 2000.

[3] WAP Forum. Wireless application protocol wireless transaction protocol specification. URL: `http://www.wapforum.org/what/technical/WTP-11-jun-99.pdf`, June 1999.