ComNets
Institute of Communication Networks

TUHH
Hamburg University of Technology

Specification

"Avionic Automatic Repeat Request Protocol"

Inter Aircraft Network

Gefördert durch:

Bundesministerium
für Wirtschaft
und Energie

aufgrund eines Beschlusses
des Deutschen Bundestages

October 2020

Förderkennzeichen: 20V1708F

Project Duration:
01/2019 – 03/2022

Work Package (WP) Leader:
Prof. Dr.-Ing. Andreas Timm-Giel
Partners:
DLR, Rohde & Schwarz, NavCert,
f.u.n.k.e., BPS, iAd

Contributors:
Sebastian Lindner, M.Sc.
Konrad Fuger, M.Sc.

Contents

1 Introduction

To establish reliable communication between IntAirNet hosts, an Automatic Repeat Request (ARQ) protocol is used. The following provides a specification od the protocol as well as the documentation of the implementation in the simulator.

1.1 Description

To obtain maximal channel utilization, the IntAirNet ARQ is a version of selective repeat ARQ and therefore aims to only retransmit segments which are explicitly not received. To do so, the ARQ protocol uses cumulative acknowledgements (ACKs) as well a selective rejection list to indicate negative acknowledgment (NACK) of segments which have not been received.

1.2 Architecture

The ARQ is a link layer sublayer which resides between the Medium Access Control (MAC) and the Radio Link Control (RLC) layer. Every transmission is initiated by the MAC layer. For this purpose the MAC layer monitors the buffer status of both RLC and ARQ layer and establishes transmission oppurtunities accordingly. Whenever a transmssion oppurtunity arrives, the MAC layer will request a segment from ARQ layer which will either return an unacknowledged segment for retransmission or in turn request a new segment from the RLC layer.

1.3 Header Fields

The following header fields are required for the operation of the ARQ protocol.

`seqno_rx` The "next sequence number to receive" field `seqno_rx` carries the *next* expected sequence number at the sender of this message. This acknowledges the reception of all sequence numbers up to `seqno_rx` $- 1$.

`seqno_tx` The transmit sequence number is the sequence number of the current segment that is being transmitted.

`poll` The poll field is set to an offset in the number of slots reserved for this communication link. The slot denotes a moment in time at which an ACK is requested at the latest. If a reverse link is configured, the remote end will piggyback its ACKs onto each transmission. The slot denoted by the field is updated every time a segment is received. If ACKs continue to arrive on the reverse link, then the respective slot will never be reached. If it is reached, then ACKs stopped arriving or a reverse link is not configured or broke off. In this case, the respective slot is used for the remote end to send an ACK, and the local end abstains from transmission.

`srej_num` The `srej_num` field contains the number of entries in the `srej` field.

The `srej` field contains as many sequence numbers as specified in the `srej_num` field. Each sequence number is consequently NACK'd and should be scheduled for retransmission.

1.4 System Variables

The following system variables must be maintained at an ARQ instance.

`V_seqno_tx` The `V_seqno_tx` variable contains the sequence number of the next in-order segment to be transmitted.

`V_seqno_rx` The `V_seqno_rx` variable contains the sequence number of the next in-order segment expected to arrive.

`V_seqno_unack` The `V_seqno_unack` variable contains the sequence number of the oldest un-acknowledged segment.

`V_poll_required` Number of slots when a `poll` is required.

`V_srej_request_list` List of segments that were detected as missing but not yet requested.

`V_srej_waiting_list` List of segments that were detected as missing and requested but could not be ACK'd yet.

`V_rtx_list` List of segments that are currently required to be retransmitted.

1.5 Sending data segments

Data segments are used to send new data or to retransmit lost segments. For new data the segment is assigned the sequence number from the `V_seqno_tx` variable. The `V_seqno_tx` variable is then incremented. For retransmitted data, the segment is assigned its original sequence number and the `V_seqno_tx` is not affected.

Generally, responses to the remote end take priority over other sending activities. This is to ensure that each Protocol Data Unit (PDU)'s packet delay is minimal, as application requirements might discard packets that take too long. Also, stalls are to be avoided when a sending window is exhausted. Therefore missing segments should be retransmitted and acknowledged as quickly as possible.

Notifying the remote end of missing segments takes highest priority. If the `V_srej_request_list` list is not empty, then as many sequence numbers from it as fit are put into the `srej` field and the number into the `srej_num` field. These sequence numbers are then moved from the `V_srej_request_list` list into the `V_srej_waiting_list` list. The remaining capacity may be used for new data transmission.

The next priority is to send outstanding retransmissions. Each item in the `V_rtx_list` list is checked: if the number of attempted retransmissions exceeds the maximum allowed, then the segment is discarded. The current segment is given a header with a sequence number of the corresponding segment, but the data field is left empty.

When new data is sent, the `poll` field is filled with the current value of `V_poll_required`. This variable is then updated to the end of the sending window minus 1 ($\texttt{V\_seqno\_unack} + \frac{N}{2} - 2$) unless it is already set to a smaller value.

1.6 Receiving data segments

Upon reception, a segment's `seqno_tx` header field is compared to the local `V_seqno_rx` variable: $\texttt{V\_seqno\_rx} - 1 + \frac{N}{2} - 1 \geq \texttt{seqno\_tx} \geq \texttt{V\_seqno\_rx}$ should hold. Also the `seqno_rx` field is checked for validity: $\texttt{V\_seqno\_tx} \geq \texttt{seqno\_rx} \geq \texttt{V\_seqno\_unack}$ should hold. If either check fails, then the segment is discarded as invalid.

If the segment's sequence number is identical to the next expected segment `V_seqno_rx`, then it is passed into the reassembly buffer, and `V_seqno_rx` is incremented. The current segment is removed if present from both the `V_srej_request_list` and `V_srej_waiting_list` lists. Next the reception buffer is checked for contiguous segments. All contiguous segments are added to the reassembly buffer and `V_seqno_rx` is incremented accordingly.

If the segment is not the next expected segment, then it is added to the reception buffer. A scheduled retransmission request is canceled by removing this segment from both the `V_srej_request_list` and `V_srej_waiting_list` lists. All sequence numbers between the highest previously received segment and the current segment are now deemed as missing and are added to the `V_srej_waiting_list` list, if not already on that list.

1.7 Poll time slot arrives

When the time slot at which a poll is requested arrives, then all sequence numbers from the `V_srej_waiting_list` list are transferred to the `V_srej_request_list` list, i.e. all missing segments are re-requested. All sequence numbers from the `V_srej_request_list` list are put into the `srej` field, and `srej_num` is set to the length of `V_srej_request_list`. The `seqno_rx` and `seqno_tx` fields are set to `V_seqno_rx` and `V_seqno_tx` as usual. This is an explicit ACK, so retransmission information has priority over user data. Remaining capacity may be filled with user data.

A References