Transducer Models of Retransmission Protocols for Noisy Channels

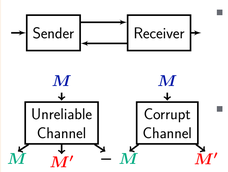
Jay Thakkar and Aditya Kanade

Indian Institute of Science, Bangalore

Work submitted to Tools and Algorithms for the Construction and Analysis of Systems (TACAS)

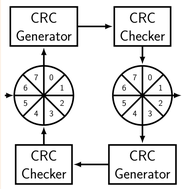
Introduction

Communication System



- Communication channel between sender and receiver can be unreliable.
- Aim: To transfer the message successfully from sender to receiver, without any error.

Retransmission Protocols



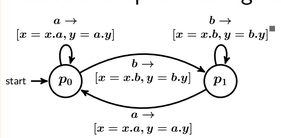
- They are used to ensure that a sequence of frames is delivered in order and without errors despite of transmission errors.
- Reliable delivery is achieved using acknowledgements with re-transmission paradigm.

Cyclic Redundancy Check

- The *cyclic redundancy check (CRC)* is a common technique for detecting errors in data transmission.
- Given a fixed generator polynomial, we build an SST to encode a message string into a cyclic redundancy check (CRC) codeword.

Streaming String Transducer

- An SST reads an input string in a single left-to-right pass and produces the output string using a finite set of states.
- It also uses a finite set of string variables to store strings over the output alphabet.
- On each input symbol, it may transition to another state and update string variables.

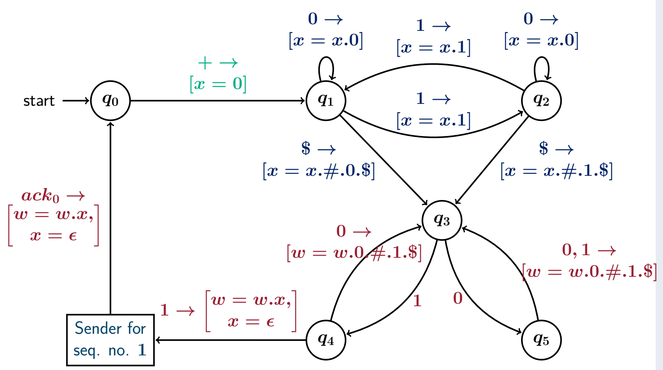


- A string variable can be used at most once across the right-hand side of the parallel assignment.

Protocol Models

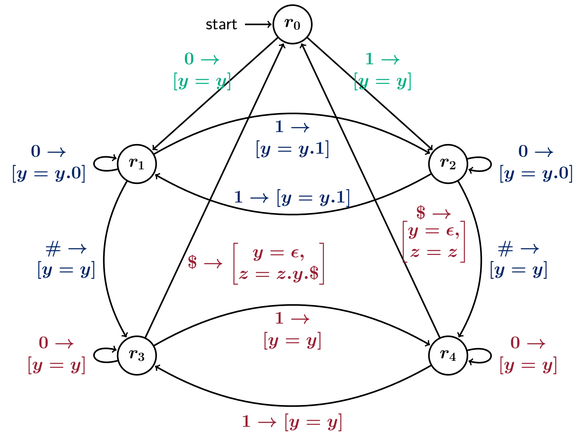
Sender SST for SWP

- The sender receives a message and encodes it into a frame with sequence number, message content and checksum.
- The sender outputs a correct frame upon receiving a positive ACK and a corrupt frame upon receiving a negative ACK.
- String variable x plays the role of the sender's buffer.



Receiver SST for SWP

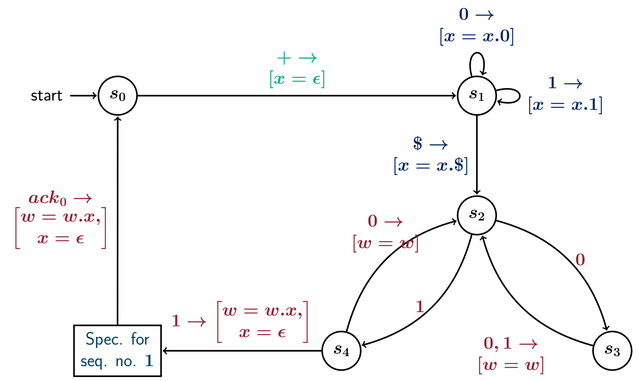
- It distinguishes between the correct and corrupt frames.
- For the correctly received frame, it removes sequence number and checksum and passes the message to its client.



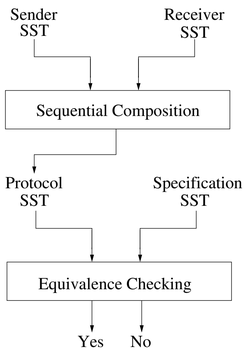
Verification Approach

Specification SST for SWP

- Key Property: Messages acknowledged by the receiver are delivered to the receiver's client correctly and in the same order in which the client of the sender handed them to the sender.
- The *specification SST* does not encode checksum computation, noisy outputs and repeated retransmissions.
- The input to the specification transducer is a sequence of messages and acknowledgements (similar to the sender) and the output is a sequence of correctly delivered messages (similar to the receiver).



The Verification Layout



- Verification Problem: To check *functional equivalence* between the specification and the protocol model.
- The output of the sender is the input of the receiver.
- The model for the entire protocol is obtained by sequentially composing the two.

Wrapping Up...

Case studies

1. **TinyOS** : The SerialP [1] software module of TinyOS computes the checksum and uses the stop-and-wait protocol in the host-to-mote direction.
2. **HDLC** : HDLC [2] is a bit-oriented protocol, that operates at data link layer. Its software implementations compute checksum and use go-back-n protocol.

Conclusions and Future Work

- Our work allows explicit modeling of message contents yet enables algorithmic verification of the resulting protocol models.
- In the future, we will try to permit arbitrary bit corruption using non-deterministic version of SSTs.
- Bounding the number of retransmission rounds can also be useful to extend the protocol models.

Related Work

- Several automated techniques abstract messages to model and verify these protocols [3], whereas we model a message as a bit stream. Hence, we use SSTs to model these protocols.
- SSTs are closed under sequential composition [4] and the equivalence problem for SSTs is decidable [5].

References

[1] http://www.tinyos.net/tinyos-2.x/doc/html/tep113.html.

[2] ISO, "Data Communication - HDLC Procedures - Elements of Procedure," Tech. Rep. ISO 4335, International Organization for Standardization, 1979.

[3] F. Babich and L. Deotto, "Formal Methods for Specification and Analysis of Communication Protocols," *IEEE Comm. Surveys and Tutorials*, vol. 4, no. 1, pp. 2–20, 2002.

[4] R. Alur and P. Cerný, "Expressiveness of streaming string transducers," in *FSTTCS*, pp. 1–12, 2010.

[5] R. Alur and P. Cerný, "Streaming Transducers for Algorithmic Verification of Single-pass List-processing Programs," in *POPL*, pp. 599–610, 2011.