1 Open Automaton for TCP Sliding Window

2 Open Automaton for Data Link Go-back-N

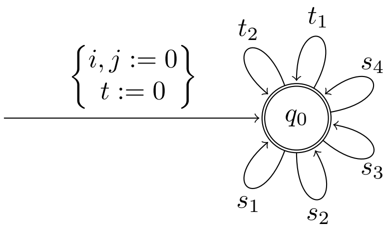
2.1 Specification

The sender maintains a circular buffer B of size N, where N is the window size. Let i and j denote the indices of the oldest unacknowledged frame and the next frame to send, respectively. Both range over $[0, N-1]$ and are interpreted modulo N. At the beginning, $i = j = 0$. The sender may transmit a frame $B[j]$ if the window is not full, i.e., $j \neq (i-1) \bmod N$. After transmission, j is incremented modulo N. The receiver maintains a single expected frame index, initially set to 0. It accepts a frame if and only if its index matches the expected i, acknowledges it, and increments i modulo N. All other frames are discarded. Upon receiving an acknowledgment for frame i, the sender removes the acknowledged frame and increments i modulo N. If an acknowledgment for frame i is not received within a timeout interval T after it was sent, the sender retransmits all frames currently in the window, starting from frame i up to (but not including) frame j.

2.2 Open Automaton

From the specification, given that Sender and Receiver as holes, two variables i and j are required. The Sender emits observable actions `send`, `wait`, `resend` and an unobservable action `getFrame`, which fetches a packet from the network layer and converts it to a frame. The Receiver emits an observable action `ack` and an unobservable action `dispatchFrame`, which sends a frame to the network layer. In addition, let $t \in [0, T]$ be the variable for waited time. We assume that for each time `wait` is emitted, t in incremented. The open automation has a single state q_0 and the transitions are

$$
\begin{aligned}
s_1 &= \frac{\{\text{Sender} \mapsto \texttt{getFrame}\}(\text{True})\{\}}{q_0 \xrightarrow{\tau} q_0} \\
s_2 &= \frac{\{\text{Sender} \mapsto \texttt{send}(j)\}((j+1)\%N \neq i)\{j := (j+1)\%N\}}{q_0 \xrightarrow{\text{frameSent}} q_0} \\
s_3 &= \frac{\{\text{Sender} \mapsto \texttt{wait}\}((j+1)\%N = i \wedge t < T)\{t := t+1\}}{q_0 \xrightarrow{\tau} q_0} \\
s_4 &= \frac{\{\text{Sender} \mapsto \texttt{resend}(i)\}(t = T)\{t := 0\}}{q_0 \xrightarrow{\text{frameResent}} q_0} \\
t_1 &= \frac{\{\text{Receiver} \mapsto \texttt{ack}(i)\}(i \neq j)\{i := (i+1)\%N; t := 0\}}{q_0 \xrightarrow{\text{frameAcked}} q_0} \\
t_2 &= \frac{\{\text{Receiver} \mapsto \texttt{dispatchFrame}\}(\text{True})\{\}}{q_0 \xrightarrow{\tau} q_0}
\end{aligned}
$$



3 Open Automaton for Data Link Selective Repeat

3.1 Specification

Selective repeat protocol allows the sender to only resend the oldest unacknowledged frame. In addition to a buffer B and two indices i and j, the sender needs an acknowledged array *Acked* whose values are boolean. Every time the receiver receives a frame correctly, it sends an acknowledgment. When an acknowledgment is received, the sender either update the acknowledgment status of the corresponding frame, or when the frame is at index i, the sender rotates the sliding window and set the acknowledgment status for this frame to False.

3.2 Open Automaton

We need an additional array *Acked* for acknowledgment status. Let the function *oldestUnacked* return the oldest unacknowledged frame index. The transitions are

$$s_1 = \frac{\{\text{Sender} \mapsto \texttt{getFrame}\}(\text{True})\{\}}{q_0 \xrightarrow{\tau} q_0}$$

$$s_2 = \frac{\{\text{Sender} \mapsto \texttt{send}(j)\}((j+1)\%N \neq i)\{j := (j+1)\%N\}}{q_0 \xrightarrow{\text{frameSent}} q_0}$$

$$s_3 = \frac{\{\text{Sender} \mapsto \texttt{wait}\}((j+1)\%N = i \wedge t < T)\{t := t+1\}}{q_0 \xrightarrow{\tau} q_0}$$

$$s_4 = \frac{\{\text{Sender} \mapsto \texttt{resend}(\mathit{oldestUnacked}())\}(t = T)\{t := 0\}}{q_0 \xrightarrow{\text{frameResent}} q_0}$$

$$t_1 = \frac{\{\text{Receiver} \mapsto \texttt{ack}(i)\}(i \neq j)\{i := (i+1)\%N; t := 0; \mathit{Acked}[i] := \text{False}\}}{q_0 \xrightarrow{\text{frameAcked}} q_0}$$

$$t_2 = \frac{\{\text{Receiver} \mapsto \texttt{ack}(k)\}(i < k \neq j)\{\mathit{Acked}[k] := \text{True}\}}{q_0 \xrightarrow{\text{frameAcked}} q_0}$$

$$t_3 = \frac{\{\text{Receiver} \mapsto \texttt{dispatchFrame}\}(\text{True})\{\}}{q_0 \xrightarrow{\tau} q_0}$$