Protocolos Punto a Punto en la Capa de Enlace de Datos Modelados con Autómatas

Redes y Sistemas Distribuidos

Abril, 2005

Resumen

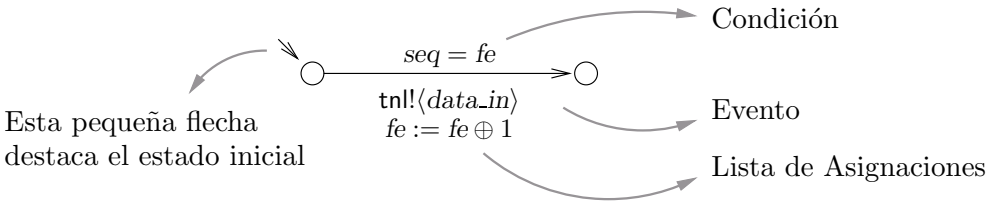
En estas notas damos un modelo utilizando una notación formal *lightweight* de los protocolos de la capa de enlace de datos que aparecen explicados en [1, Cap. 3].

Autómatas simbólicos: notación

Un autómata es un grafo dirigido donde:

- los vértices representan *estados* de un sistema y
- las arístas, que están etiquetadas, representan las *transiciones* que llevan de un estado a otro.

En particular, las transiciones que nosotros usaremos están etiquetadas de la siguiente manera:



Las *condiciones* establecen cuando la transición puede ejecutarse. Toda condición omitida se supondrá una tautología. Junto con el estado origen forman la precondicón de la transición.

La *lista de asignaciones* determina los nuevos valores para las variables. Sólo hacemos explícito la asignación que cambia de valor una variable. Por lo tanto la lista puede ser vacía. Junto con el estado destino determinan la postcondición de la transición.

En las expresiones de las asignaciones usaremos operadores como ++ o --. Además, dada una constate MAX_SEQ, las operaciones $\oplus$ y $\ominus$ son la suma y resta módulo MAX_SEQ$+1$ o binaria, según corresponda.

Los *eventos* son las "acciones" que la transición realiza. Estas pueden ser: ofrecer un servicio, pedir un servicio, o interactuar con el timer. Hay dos tipos de eventos: los generados y los esperados:

- Los eventos generados tienen la forma: nombre_evento!⟨lista de expresiones⟩.

  (Notar el signo de exclamación.) Esta clase de acción son las que toman la iniciativa como por ejemplo enviar un frame o arrancar o deterner un timer. La lista de expresiones son los datos que se envian al generar el evento. Por ejemplo tpl!⟨*seq*, *data_out*⟩ podría representar el envio de un dato en la variable *data_out* con número de secuencia *seq* a la capa física a través del SAP (Service Access Point) tpl.

- Los eventos esperados tienen la forma: nombre_evento?⟨lista de variables⟩.

  (Notar el signo de interrogación.) Esta forma de acción son las que esperan la ocurrencia de los eventos como por ejemplo esperar un dato de la capa de red o esperar un timeout. La transición *no puede ejecutarse* si el evento no se ha producido (de hecho la ocurrencia del evento también forma parte de la precondición de la transición). La lista de variables está para albergar los

datos que pudiera traer el evento esperado (la asignación sobre esta variable también forma parte de la postcondición). Por ejemplo fpl?$\langle seq, data\_out\rangle$ podría representar la espera de la recepción de un dato que se pondría en la variable *data_out* con número de secuencia, que se guardaría en *seq*, proveniente de la capa física a través del SAP fpl.

Un protocolo muy simple

Eventos:

fnl *(from network layer)*: recibe un dato

tnl *(to network layer)*: envía un dato

fpl *(from physical layer)*: recibe dato

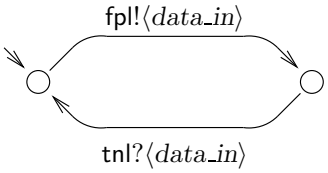
tpl *(to physical layer)*: envía dato

Variables:

data_in: información recibida de la entidad peer

data_out: información a transmitir hacia la entidad peer

Transmisor



Receptor

Protocolo *"Stop & Wait"*

Eventos:

fnl: recibe un dato

tnl: envía un dato

fpl: recibe dato o acknowledgement (dependiendo de la dirección)

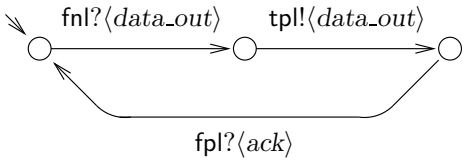
tpl: envía dato o acknowledgement (dependiendo de la dirección)

Variables:

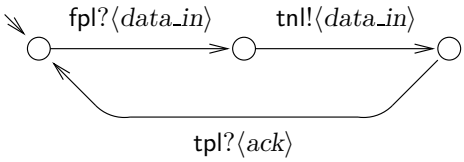
data_in: información recibida de la entidad peer

data_out: información a transmitir hacia la entidad peer

ack: variable *dummy* para indicar reconocimiento



Transmisor



Receptor

Protocolo de Reconocimiento Positivo con Retransmisión: PAR (*"Positive Acknowledgement with Retransmission"*)

Eventos:

fnl: recibe un dato

tnl: envía un dato

fpl: recibe ⟨nro. sec. frame, dato⟩ o ack último frame recibido (dependiendo de la dirección)

tpl: envía ⟨nro. sec. frame, dato⟩ o ack último frame recibido (dependiendo de la dirección)

start_timer: setea el timer

timeout: interrucpción por excedente de tiempo

Variables:

data_in: información recibida de la entidad peer

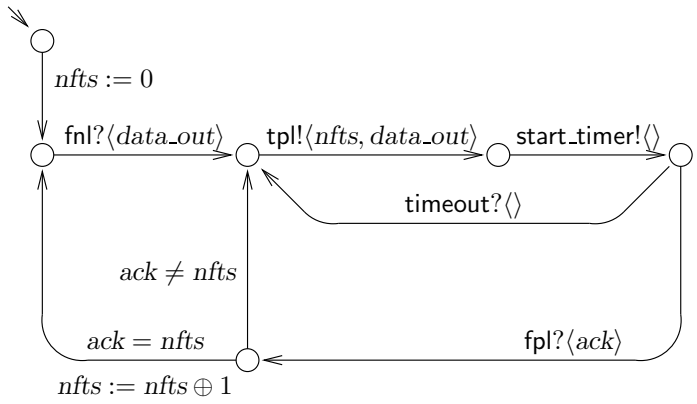
data_out: información a transmitir hacia la entidad peer

ack $\in \{0, 1\}$: nro. de frame reconocido

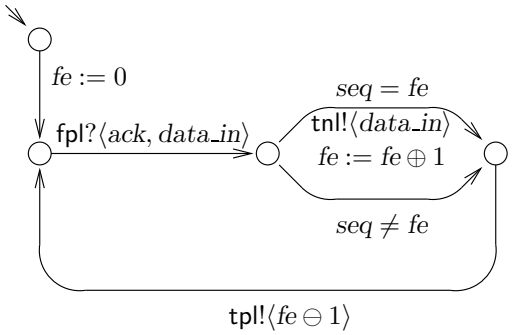
seq $\in \{0, 1\}$: nro. de frame recibido

nfts $\in \{0, 1\}$: *next frame to send*

fe $\in \{0, 1\}$: *frame expected*



Transmisor



Receptor

Protocolo de Ventana Deslizante de Tamaño 1 (con *piggybacking*)

Constantes:

MAX_SEQ: mayor número de secuencia

Eventos:

fnl: recibe un dato

tnl: envía un dato

fpl: recibe ⟨nro. sec. frm., ack último frm. rec., dato⟩

tpl: envía ⟨nro. sec. frm., ack último frm. rec., dato⟩

start_timer: setea el timer

timeout: interrucpción por excedente de tiempo

Variables:

data_in: información recibida de la entidad peer

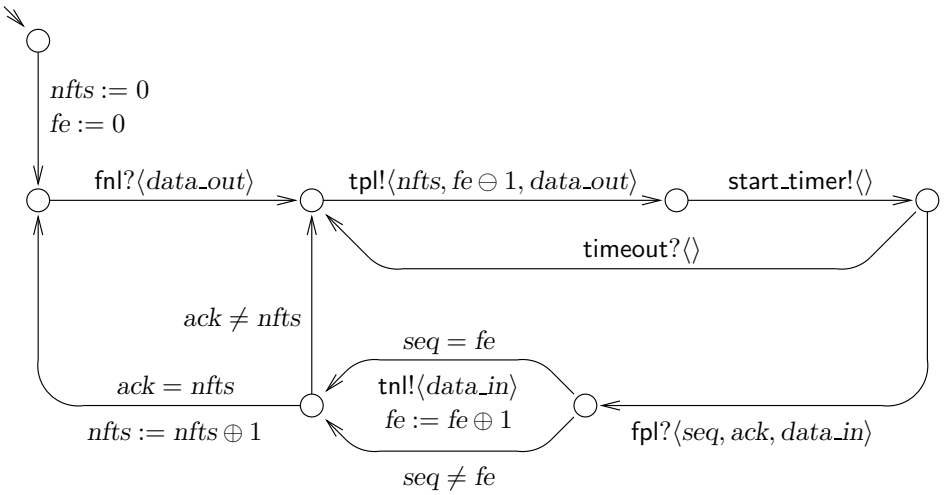
data_out: información a transmitir hacia la entidad peer

ack $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame reconocido

seq $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame recibido

nfts $\in \{0, .., \text{MAX\_SEQ}\}$: *next frame to send*

fe $\in \{0, .., \text{MAX\_SEQ}\}$: *frame expected*



Protocolo de Ventana Deslizante con n Retrocesos (*"Go back n"*)

Constantes:

MAX_SEQ: mayor número de secuencia

WIN_SIZE: tamaño de la ventana

Eventos:

fnl: recibe un dato

tnl: envía un dato

fpl: recibe ⟨nro. sec. frm., ack último frm. rec., dato⟩

tpl: envía ⟨nro. sec. frm., ack último frm. rec., dato⟩

start_timer: setea un timer dado

stop_timer: detiene un timer dado

(los timers están dados con un nro. e/ 0 y WIN_SIZE)

timeout: interrucpción por excedente de tiempo

Variables:

data_in: información recibida de la entidad peer

ack $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame reconocido

seq $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame recibido

nfts $\in \{0, .., \text{MAX\_SEQ}\}$: *next frame to send*

fe $\in \{0, .., \text{MAX\_SEQ}\}$: *frame expected*

buffer array [0..MAX_SEQ]: guarda datos que pueden necesitar ser reenviados

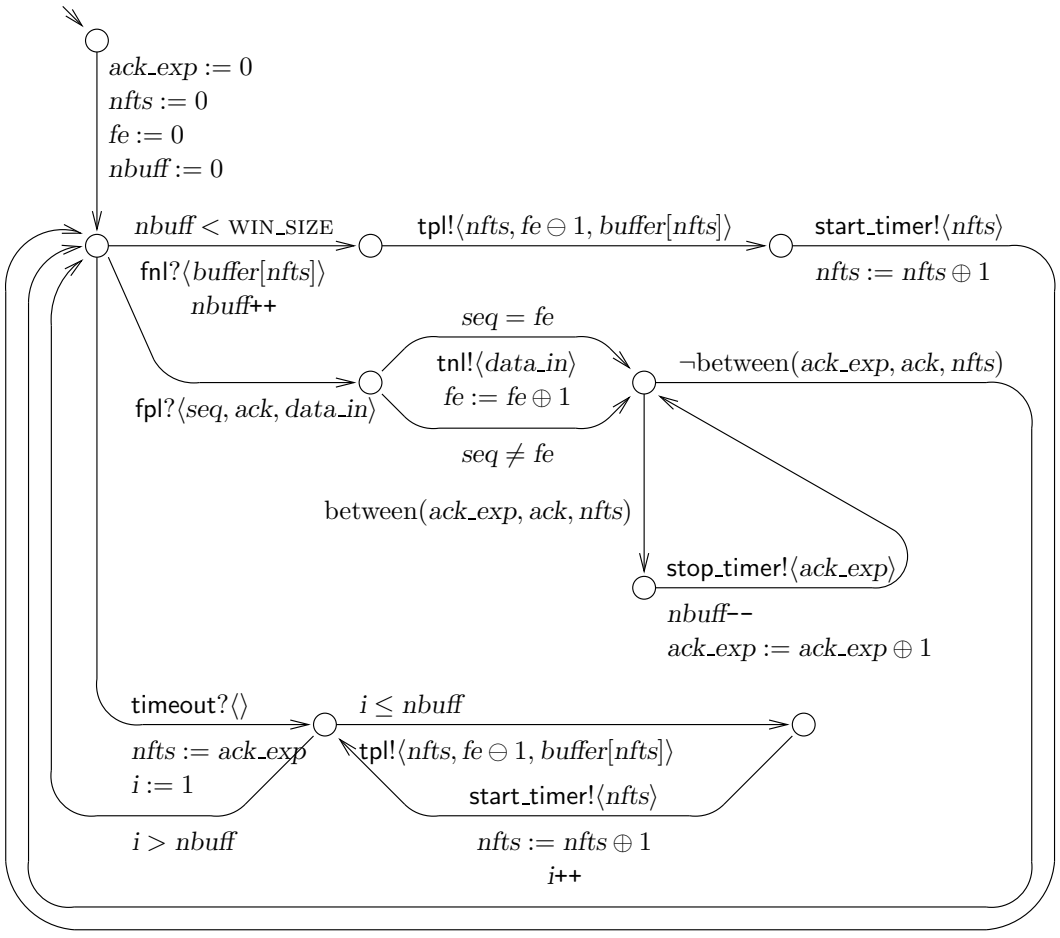
nbuff $\in \{0, .., \text{WIN\_SIZE} - 1\}$: cant. de elementos en el buffer, i.e. tamaño de la ventana de transmisión

ack_exp $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame más viejo sin ack

i: var auxiliar para retransmisión

Funciones:

between(a, b, c) verifica que $a \leq b < c$ circularmente, módulo MAX_SEQ $+ 1$.



Protocolo de Ventana Deslizante con n Retrocesos (*"Go back n"*) con acknowledgements independientes

Constantes:

MAX_SEQ: mayor número de secuencia

WIN_SIZE: tamaño de la ventana

DATA: indica que el frame contiene datos

ACK: indica que el frame solo contiene un ack (sin datos)

TACK $\notin \{0, .., \text{MAX\_SEQ}\}$ es el número correspondiente al timer del ack

Eventos:

fnl: recibe un dato

tnl: envía un dato

fpl: recibe ⟨nro. sec. frm., ack último frm. rec., dato⟩

tpl: envía ⟨nro. sec. frm., ack último frm. rec., dato⟩

start_timer: setea un timer dado

stop_timer: detiene un timer dado

timeout: interrucpción por excedente de tiempo

(los timers están dados con un nro. e/ 0 y WIN_SIZE, y existe un timer más, llamado TACK, que corresponde al ack)

Variables:

data_in: información recibida de la entidad peer

ack $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame reconocido

seq $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame recibido

nfts $\in \{0, .., \text{MAX\_SEQ}\}$: *next frame to send*

fe $\in \{0, .., \text{MAX\_SEQ}\}$: *frame expected*

buffer array [0..MAX_SEQ]: guarda datos que pueden necesitar ser reenviados

nbuff $\in \{0, .., \text{WIN\_SIZE} - 1\}$: cant. de elementos en el buffer, i.e. tamaño de la ventana de transmisión

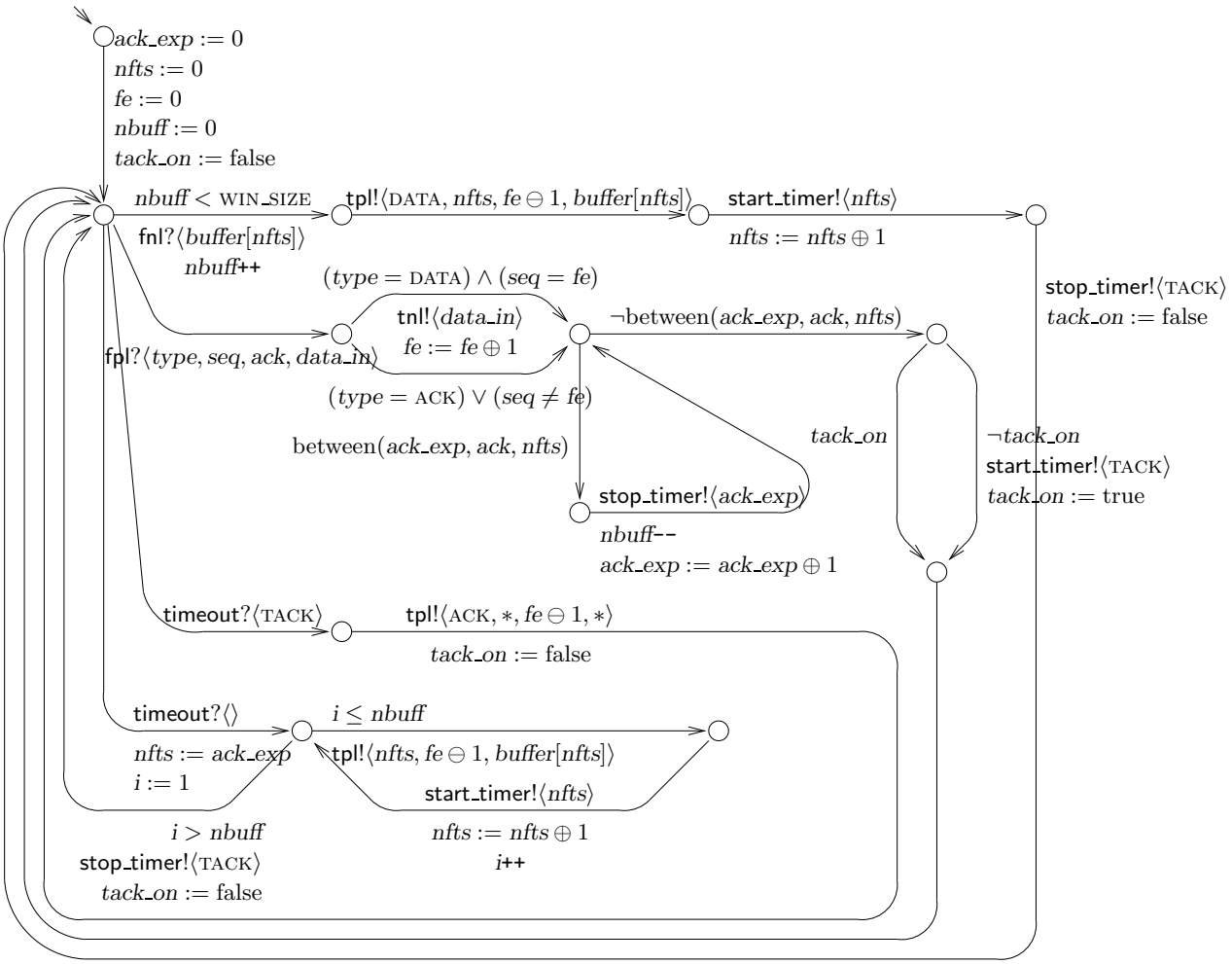
ack_exp $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame más viejo sin ack

tack_on: registra si el timer del acnowledgment está corriendo

i: var auxiliar para retransmisión

Funciones:

between(a, b, c) verifica que $a \leq b < c$ circularmente, módulo MAX_SEQ $+ 1$.



Protocolo de Ventana Deslizante con Retransmisión Selectiva

Constantes:

MAX_SEQ: mayor número de secuencia

WIN_SIZE: tamaño de la ventana

DATA: indica que el frame contiene datos

ACK: indica que el frame solo contiene un ack (sin datos)

TACK $\notin \{0, .., \text{MAX\_SEQ}\}$ es el número correspondiente al timer del ack

Eventos:

fnl: recibe un dato

tnl: envía un dato

fpl: recibe ⟨nro. sec. frm., ack último frm. rec., dato⟩

tpl: envía ⟨nro. sec. frm., ack último frm. rec., dato⟩

start_timer: setea un timer dado

stop_timer: detiene un timer dado

timeout: interrucpción por excedente de tiempo de un timer en particular

(los timers están dados con un nro. e/ 0 y WIN_SIZE, y existe un timer más, llamado TACK, que corresponde al ack)

Funciones:

Variables:

data_in: información recibida de la entidad peer

ack $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame reconocido

seq $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame recibido

nfts $\in \{0, .., \text{MAX\_SEQ}\}$: *next frame to send*

fe $\in \{0, .., \text{MAX\_SEQ}\}$: *frame expected*

nbuff $\in \{0, .., \text{WIN\_SIZE} - 1\}$: cant. de elementos en el buffer, i.e. tamaño de la ventana de transmisión

ack_exp $\in \{0, .., \text{MAX\_SEQ}\}$: nro. de frame más viejo sin ack

nr_timer $\in \{0, .., \text{MAX\_SEQ}\} \cup \{\text{TACK}\}$: nro. de timer sobre el cual se efectuo un timeout

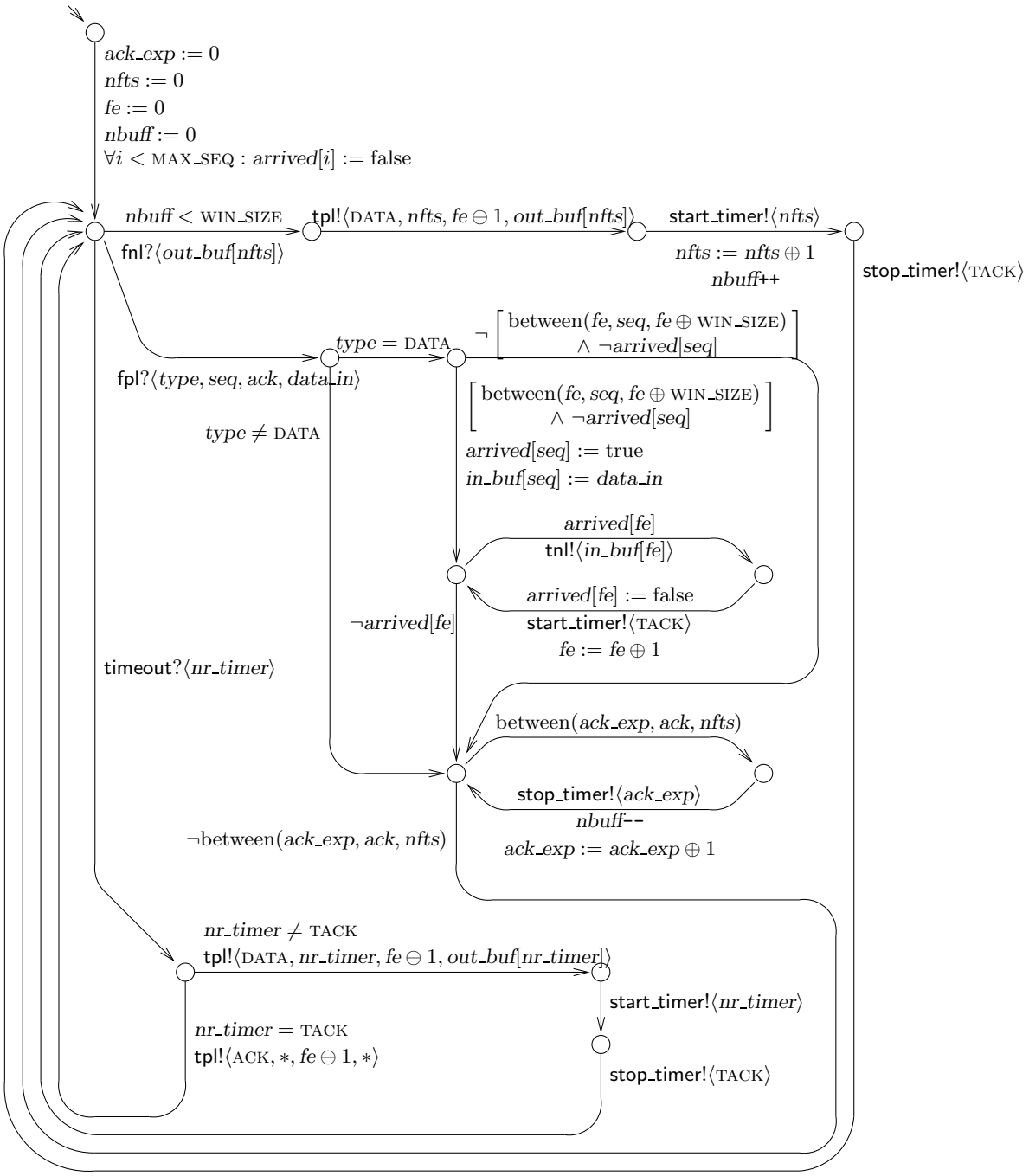
type $\in \{\text{DATA}, \text{ACK}\}$: tipo de frame recibido

out_buf array [0..MAX_SEQ]: guarda datos que pueden necesitar ser reenviados

in_buf array [0..MAX_SEQ]: guarda datos que han arribado pero aún no pueden enviarse a la capa de red

arrived array [0..MAX_SEQ] de booleanos: indica que datos en *in_buf* son válidos

between(a, b, c) verifica que $a \leq b < c$ circularmente, módulo MAX_SEQ $+ 1$.



Referencias

[1] A. Tanenbaum. *Computer Networks.* 3ra y 4ta edición. Prentice Hall. 1996.