

Rule W1

After sending a basic message to a
process P_i :
 $CTR_w = CTR_w + 1$;
 $NB_w = NB_w \cup \{P_i\}$;

Rule W2

Upon receipt of a message from P_s :
CASE message of
(a) basic message :
if ($NOTDEL = TRUE$) then
/* basic message is returned */
 $CTR_w = CTR_w - 1$;
 $NB_w = NB_w - \{P_s\}$;
else
SEND($P_w, (ACK, Seq_w), P_s$);
 $NB_w = NB_w \cup \{P_s\}$;

(b) ($P_s, (TD, PI_j, BS_s), P_w$) :
 $Seq_w = \max(Seq_w, PI_j.Seq)$;

(c) ($P_s, (KM, PI_s, S_s, NB_s), P_w$) :
 $Seq_w = \max(Seq_w, PI_s.Seq)$;
 $NB_w = NB_w \cup \{S_s\} - \{P_s\}$;
if ($P_w \in PMSET$) then
/* P_w is a permanent process */
 $Deadset_w = Deadset_w \cup \{P_s\}$;
if ($P_w = S_s$) then
 $NB_w = NB_w \cup$
 $NB_s - \{P_w\} - Deadset_w$;
(d) ($P_s, (KP, PI_s), P_w$) :
while ($CTR_w \neq 0$); /* wait! */
 $Seq_w = \max(Seq_w, PI_s.Seq)$;
 $\forall P_j \in NB_w \cup PMSET$,
SEND($P_w, (KM, PI_w, S_w, NB_w),$
 P_j);
(KILL P_w);

(e) ($P_s, (ACK, Seq_s), P_w$) :
 $CTR_w = CTR_w - 1$;
 $Seq_w = \max(Seq_w, Seq_s)$;

(f) Otherwise :
Ignore the message.

Rule W3

Upon creating a process P_c :
 $NB_w = NB_w \cup \{P_c\}$;
 $NB_c = \{P_w\}$;
 $Seq_c = Seq_w$;
 $S_c = S_w$;

Rule W4

Upon deciding to kill a process P_v :
SEND($P_w, (KP, PI_w), P_v$);

Rule W5

Upon deciding to kill itself :
while ($CTR_w \neq 0$); /* wait! */
 $\forall P_j \in NB_w \cup PMSET$,
SEND($P_w, (KM, PI_w, S_w, NB_w), P_j$);
(KILL P_w);

Rule W6

Upon completing its basic computation :
while ($CTR_w \neq 0$); /* wait! */
if ($NB_w = \emptyset$) then (TERMINATE);
 $Seq_w = Seq_w + 1$;
 $LPI_w = \langle Seq_w, w \rangle$;
 $CLR_w = black$;
 $PAR_w = nil$;
 $TDSET_w = NB_w$;
 $\forall P_j \in TDSET_w$,
SEND($P_w, (TD, LPI_w, NB_w \cup \{P_w\}),$
 P_j);

Figure 2: Rules for a *white* process P_w

Rule B1

Upon receipt of a basic message from P_s :
 $SEND(P_b, (ACK, Seq_b), P_s);$
 $NB_b = NB_b \cup \{P_s\};$
 $CLR_b = white;$

Rule B2

Upon receipt of a $(P_s, (TD, PI_j, BS_s), P_b)$:
if $(PI_j > LPI_b)$ then
 $LPI_b = PI_j;$
 $Seq_b = max(Seq_b, PI_j.Seq);$
 $PAR_b = P_s;$
if $((NB_b - BS_s) = \phi)$ then
 $SEND(P_b, (RT, PI_j), P_s);$
 $CLR_b = red ;$
else $/* propagate the TD */$
 $TDSET_b = NB_b - BS_s;$
 $\forall P_k \in TDSET_b,$
 $SEND(P_b, (TD, PI_j, BS_s \cup NB_b),$
 $P_k);$
else $/* PI_j \leq LPI_b */$
if $(PI_j = LPI_b)$ then
 $SEND(P_b, (RT, PI_j), P_s);$

Rule B3

Upon receipt of a $(P_s, (RT, PI_j), P_b)$:
if $(PI_j = LPI_b)$ then
 $/* Reply to the latest TD sent */$
if $(P_s \in NB_b)$ then
 $TDSET_b = TDSET_b - \{P_b\};$
if $(TDSET_b = \phi)$ then
if $(PAR_b = nil)$ then
 $/* Termination detected ! */$
 $\forall P_k \in NB_b,$
 $SEND(P_b, (TER, NB_b \cup \{P_b\}),$
 $P_k);$
 $(TERMINATE);$
else $/* PAR_b \neq nil */$
 $SEND(P_b, (RT, PI_j), PAR_b);$
 $CLR_b = red;$

Rule B4

Upon receipt of a $(P_s, (KP, PI_s), P_b)$:
 $Seq_b = max(Seq_b, PI_s.Seq);$
 $\forall P_j \in NB_b \cup PMSET,$
 $SEND(P_b, (KM, PI_b, S_b, NB_b), P_j);$
 $(KILL P_b);$

Rule B5

Upon receipt of a $(P_s, (KM, PI_s, S_s, NB_s), P_b)$:
if $(PAR_b = P_s)$ then $PAR_b = S_s;$
if $(P_b \in PMSET)$ then
 $/* P_b is a permanent process */$
 $Deadset_b = Deadset_b \cup \{P_s\};$
 $NB_b = NB_b - \{P_s\};$
if $(P_b = S_s)$ then
 $NB_b = NB_b \cup NB_s - \{P_b\} - Deadset_b;$
else $/* P_b is not permanent */$
 $NB_b = NB_b \cup \{S_s\} - \{P_s\};$
if $(NB_b = \phi)$ then $(TERMINATE);$
else
 $/* Initiate new TD */$
 $Seq_b = max(Seq_b, PI_s.Seq) + 1;$
 $LPI_b = < Seq_b, b >;$
 $PAR_b = nil;$
 $TDSET_b = NB_b;$
 $\forall P_k \in TDSET_b,$
 $SEND(P_b, (TD, LPI_b, NB_b \cup \{P_b\}),$
 $P_k);$

Figure 3: Rules for a *black* process P_b

Rule R1

Upon receipt of a basic message from P_s :
 $SEND(P_r, (ACK, Seq_r), P_s);$
 $NB_r = NB_r \cup \{P_s\};$
 $CLR_r = white;$

Rule R2

Upon receipt of a $(P_s, (TD, PI_j, BS_s), P_r)$:
 if $(PI_j > LPI_r)$ then
 $LPI_r = PI_j;$
 $Seq_r = max(Seq_r, PI_j.Seq);$
 $PAR_r = P_s;$
 if $((NB_r - BS_s) = \phi)$ then
 $SEND(P_r, (RT, PI_j), P_s);$
 else /* propagate the TD */
 $TDSET_r = NB_r - BS_s;$
 $\forall P_k \in TDSET_r,$
 $SEND(P_r, (TD, PI_j, BS_s \cup NB_r),$
 $P_k);$
 $CLR_r = black;$
 else /* $PI_j \leq LPI_r$ */
 if $(PI_j = LPI_r)$ then
 $SEND(P_r, (RT, PI_j), P_s);$

Rule R3

Upon receipt of a $(P_s, (RT, PI_j), P_r)$:
 Ignore the message;

Rule R4

Upon receipt of a $(P_s, (KP, PI_s), P_r)$:
 $Seq_r = max(Seq_r, PI_s.Seq);$
 $\forall P_j \in NB_r \cup PMSET,$
 $SEND(P_r, (KM, PI_r, S_r, NB_r), P_j);$
 $(KILL P_r);$

Rule R5

Upon receipt of a $(P_s, (KM, PI_s, S_s, NB_s), P_r)$:
 if $(PAR_r = P_s)$ then $PAR_r = S_s;$
 if $(P_r \in PMSET)$ then
 /* P_r is a permanent process */
 $Deadset_r = Deadset_r \cup \{P_s\};$
 $NB_r = NB_r - \{P_s\};$
 if $(P_r = S_s)$ then
 $NB_r = NB_r \cup NB_s - \{P_r\} - Deadset_r;$
 else /* P_r is not permanent */
 $NB_b = NB_b \cup \{S_s\} - \{P_s\};$
 if $(NB_b = \phi)$ then $(TERMINATE);$
 $Seq_r = max(Seq_r, PI_s.Seq) + 1;$
 $LPI_r = Seq_r, r >;$
 $PAR_r = nil;$
 $TDSET_r = NB_r;$
 $\forall P_k \in TDSET_r,$
 $SEND(P_r, (TD, LPI_r, NB_r \cup \{P_r\}), P_k);$
 $CLR_r = black;$

Rule R6

Upon receipt of a $(P_s, (TER, BS_s), P_r)$:
 $\forall P_k \in (NB_r - BS_s),$
 $SEND(P_r, (TER, NB_r \cup BS_s), P_k);$
 $(TERMINATE);$

Figure 4: Rules for a *red* process P_r