¿Por qué el transmisor de stop-and-wait y Go-back-N hacen nada cuando llega un ACK dañado o duplicado?

- Caso Stop-and-wait

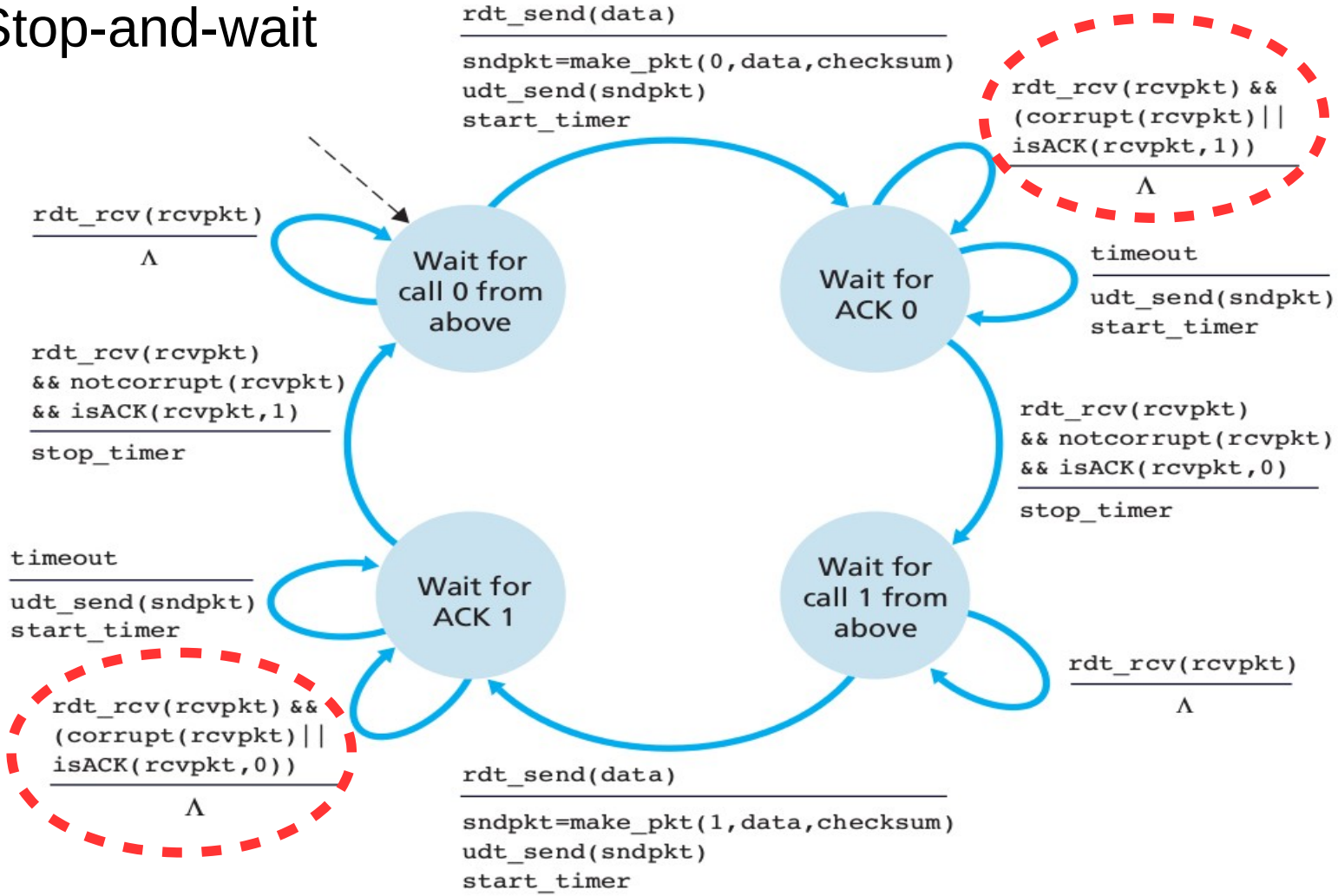
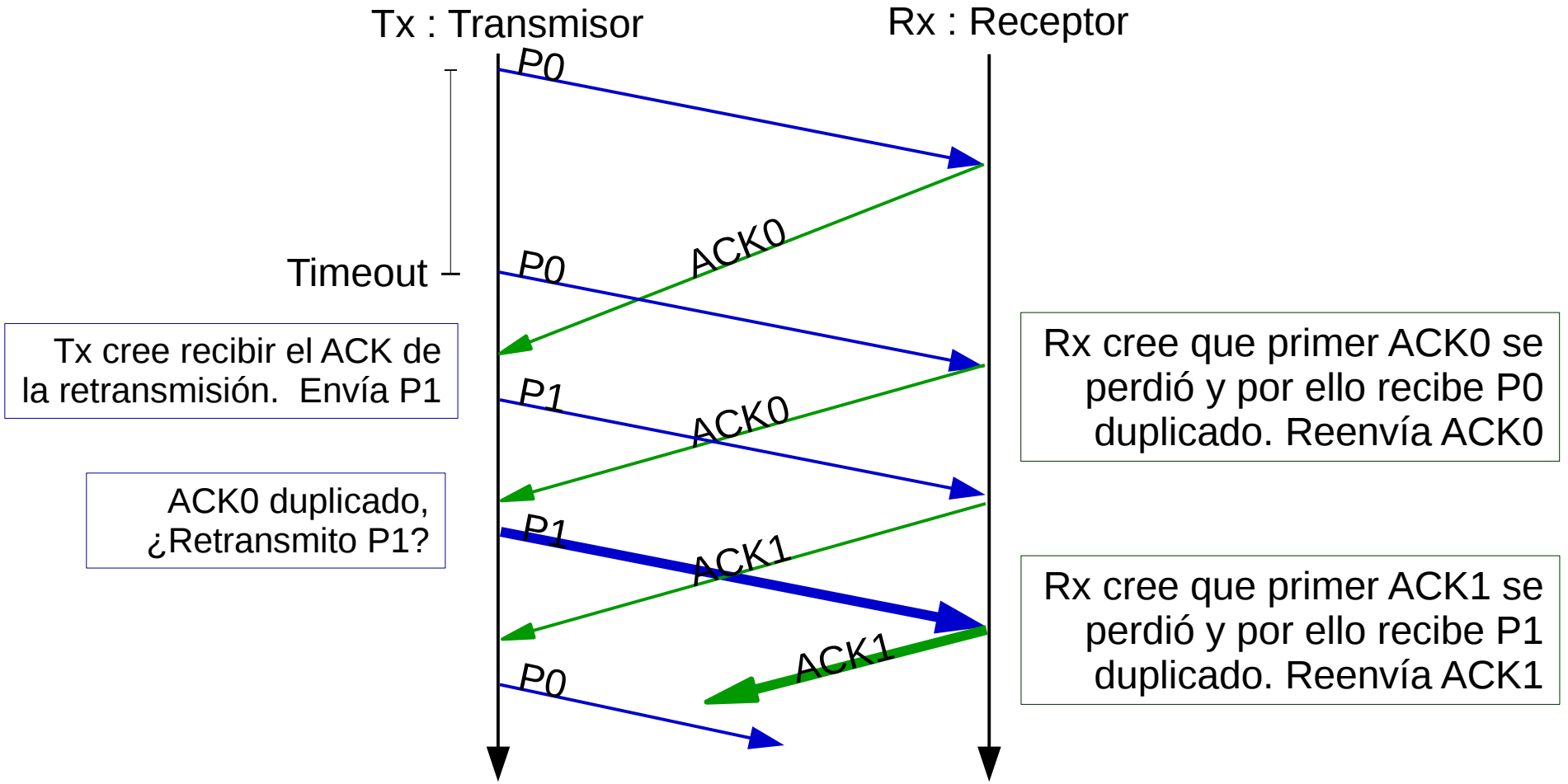


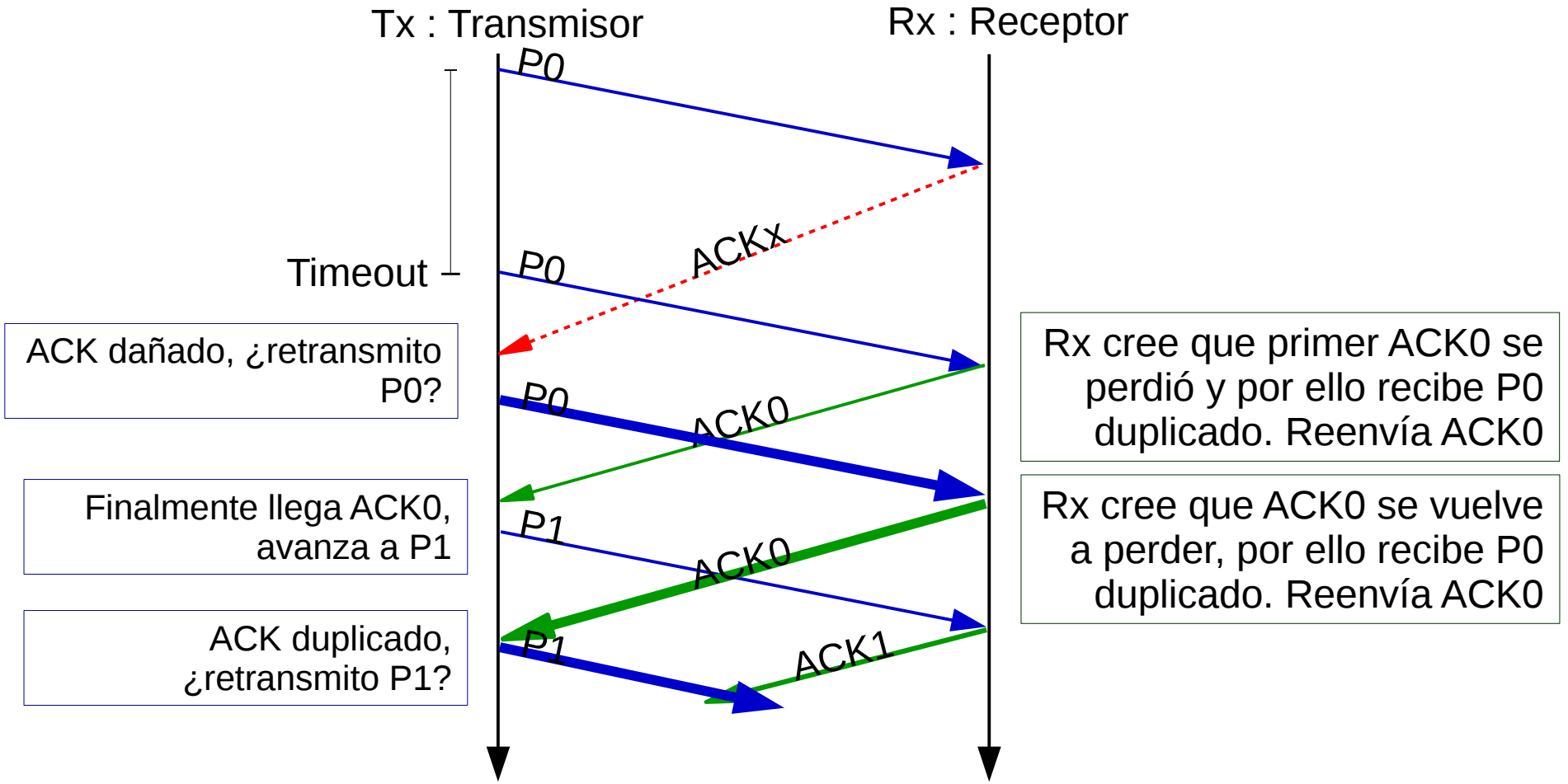
Figure 3.15 ♦ rdt3.0 sender

Analizar el caso ACK duplicado



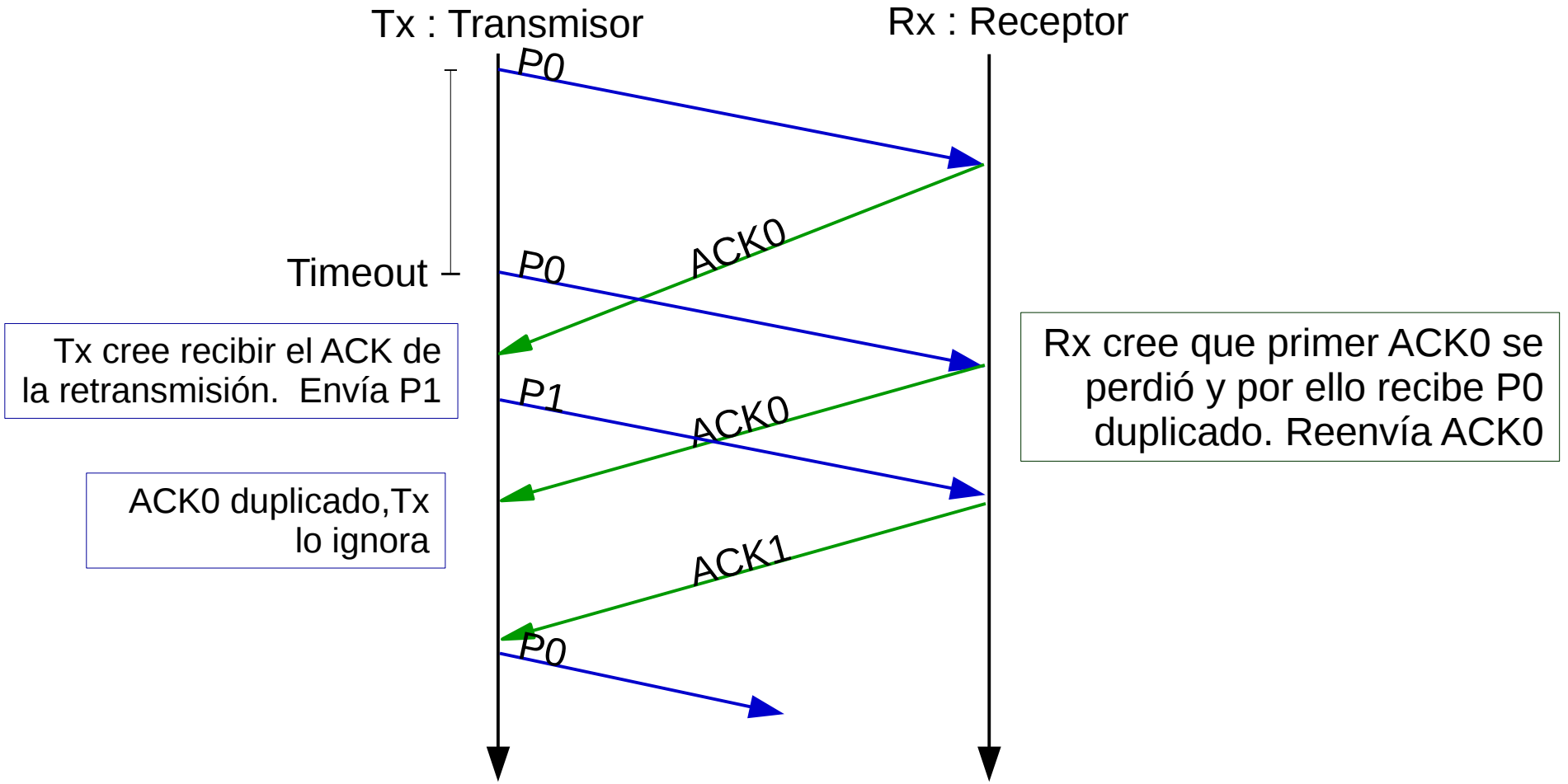
- Como los ACK se pueden perder, cuando llega un duplicado al Rx, éste debe reenviar el ACK. No tiene otra opción.
- Si Tx reenvía el paquete cuando llega un ACK duplicado, terminará enviando dos veces cada paquete. Mala idea.
- Peor aún, si no hay más datos y solo envío P0 ¿acaso debo reenviar P0 cuando llega ACK0 duplicado? Cuando el estado es “Wait for call from above”, la acción del texto es hacer nada.

Analizar el caso ACK dañado



- Si Tx reenvía el paquete cuando llega un ACK dañado en este escenario, también terminará enviando dos veces cada paquete. Mala idea.

Supongamos Tx, hace nada ...



- Si Tx ignora el ACK duplicado, todo se comparta como se desea. Buena idea.

Caso Go-Back-N

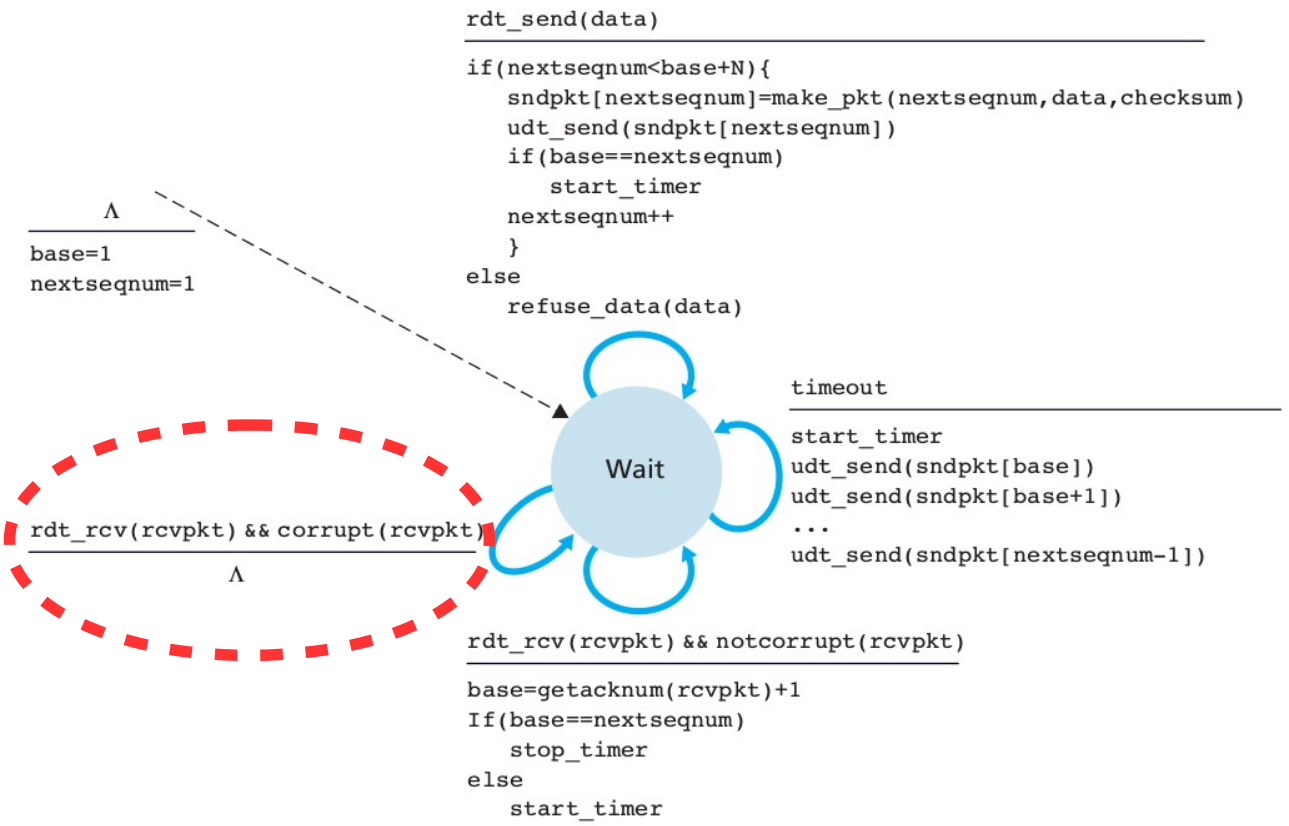
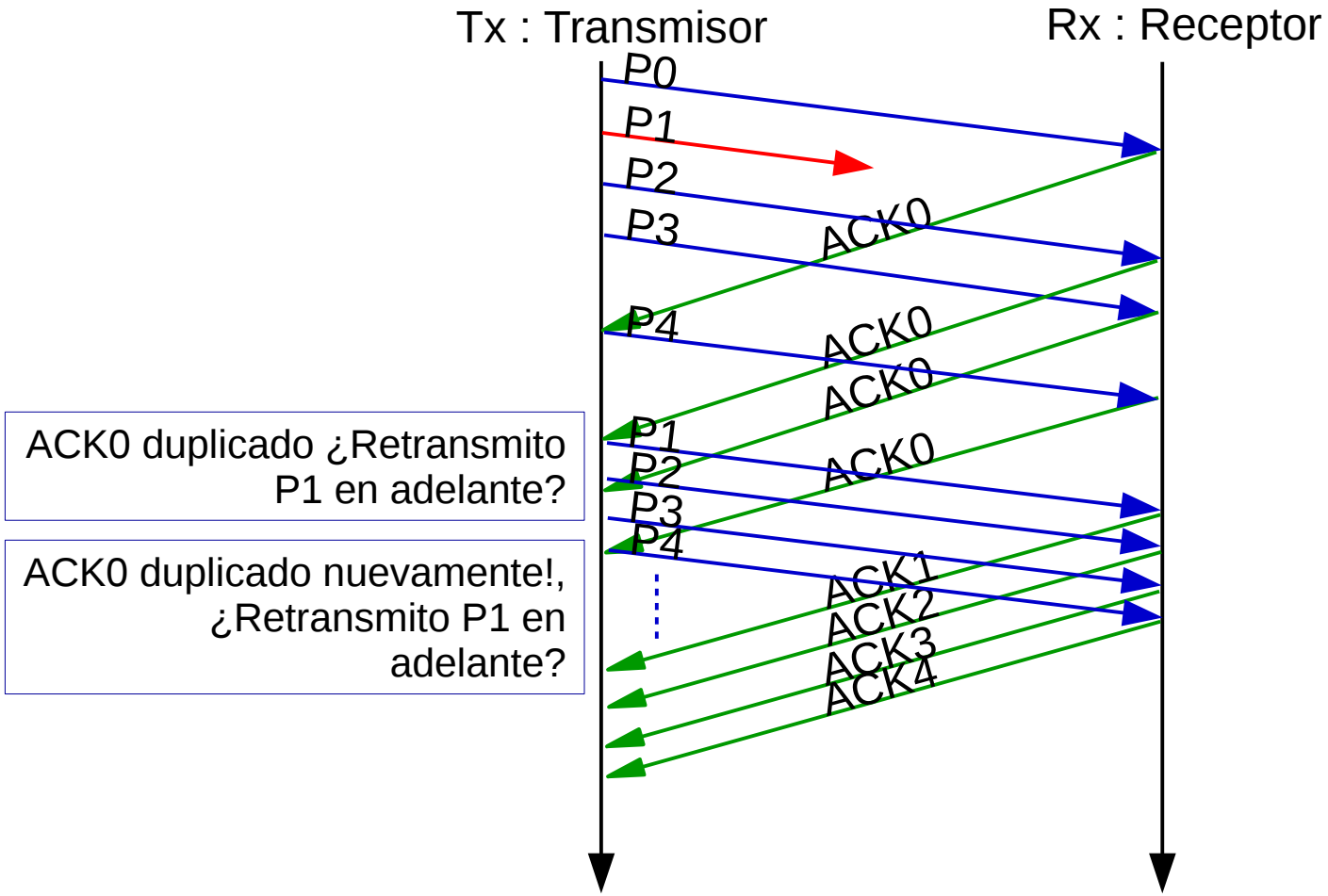


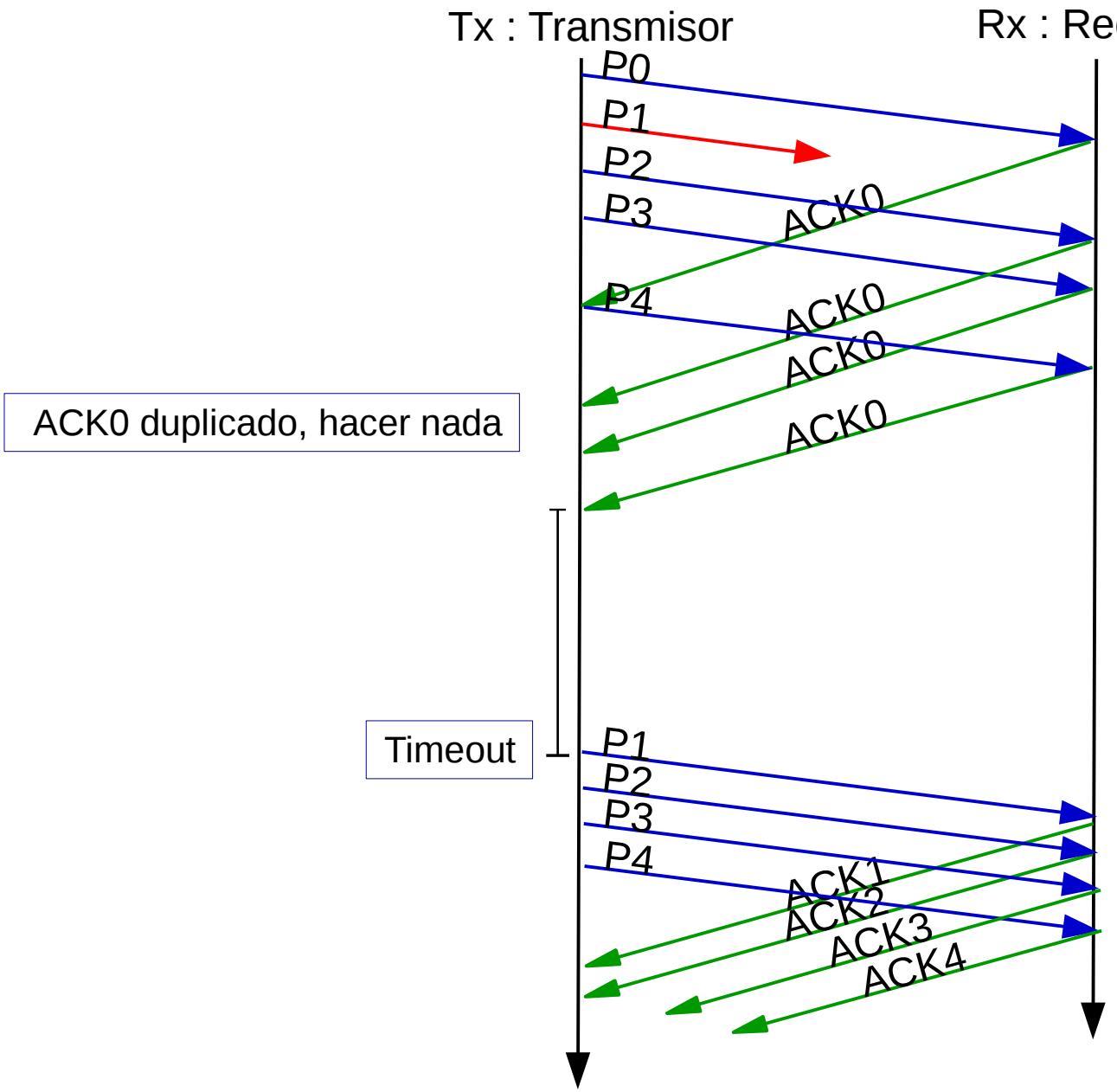
Figure 3.20 ♦ Extended FSM description of GBN sender

Analizar el caso ACK duplicado



- Como los ACK se pueden perder, cuando llega un duplicado al Rx, éste debe reenviar el ACK. No tiene otra opción.
- Si Tx reenvía el paquete cuando llega un ACK duplicado, terminará enviando varias veces varios paquetes.

Supongamos Tx hace nada....



¿Por qué reiniciar el timer ante la llegada de cada ACK?

Propuesta de modificación de Go-Back-N, me apoyan?

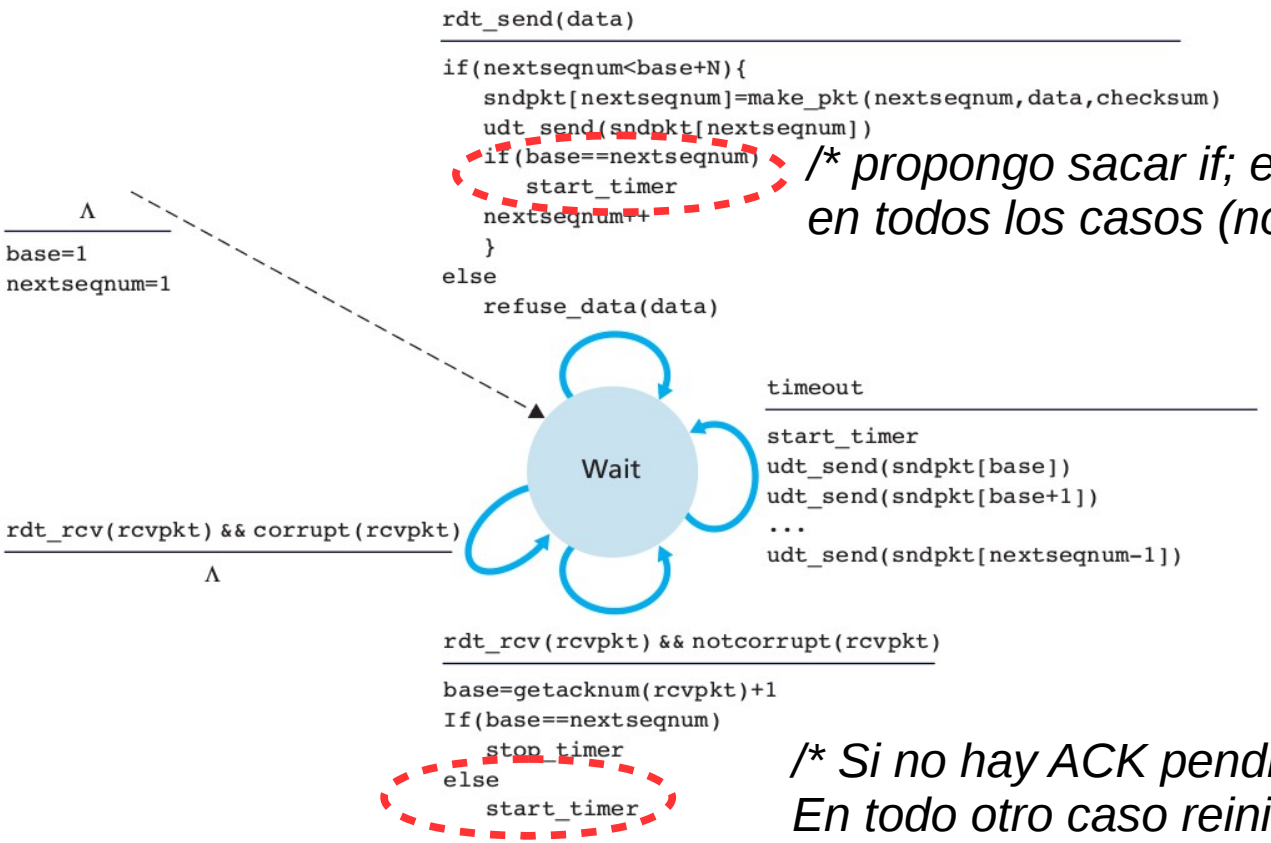
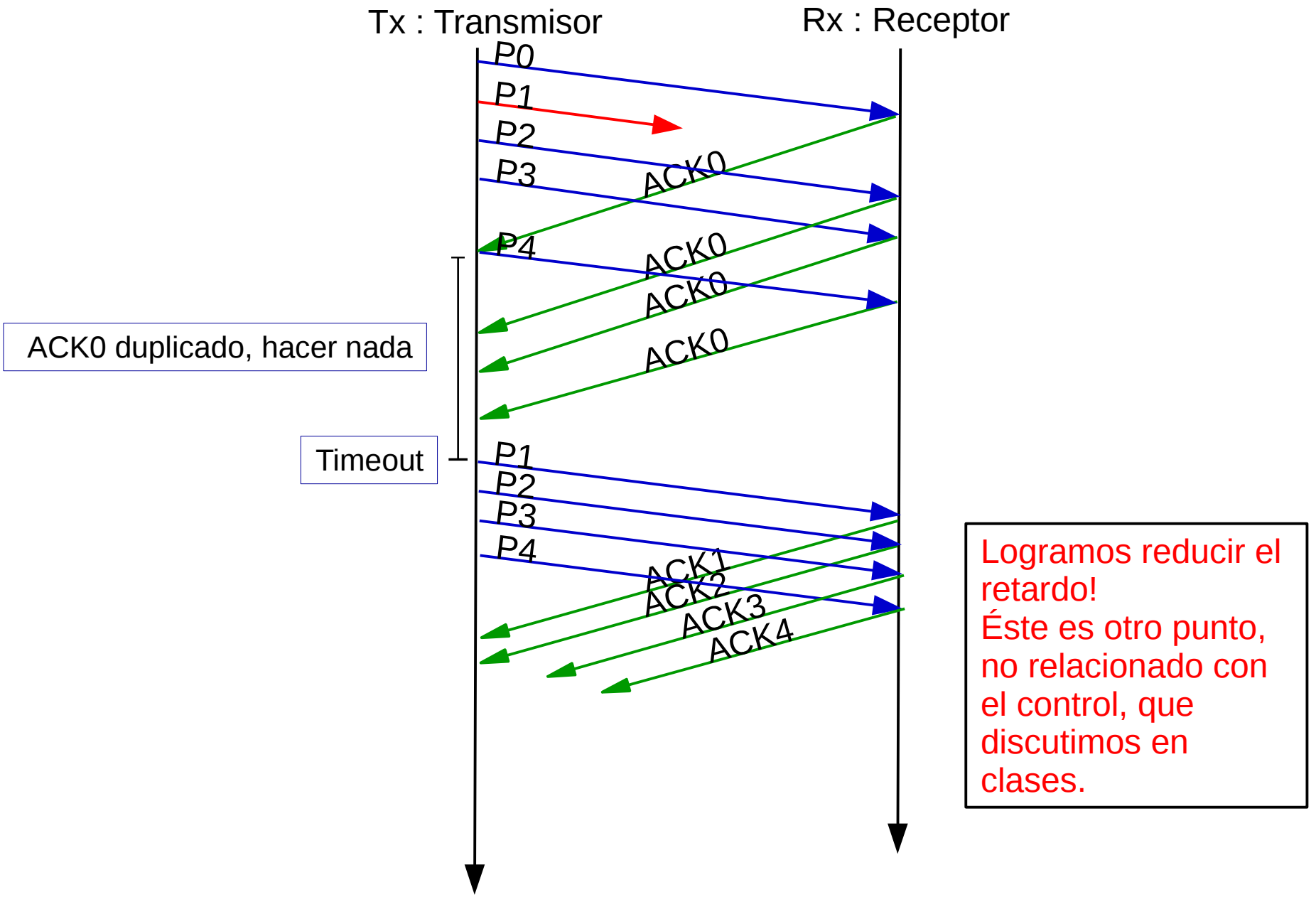


Figure 3.20 ♦ Extended FSM description of GBN sender

Por qué? Propongo eliminarlo e iniciar el timer cada vez que se envía un nuevo paquete. ¿Apoyarían que le escriba a los autores?

Éste sería el diagrama....



Logramos reducir el retardo!
Éste es otro punto, no relacionado con el control, que discutimos en clases.