



Figure 3.14 *rdt2.2* receiver

is initiated. The approach thus adopted in practice is for the sender to judiciously choose a time value such that packet loss is likely, although not guaranteed, to have happened. If an ACK is not received within this time, the packet is retransmitted. Note that if a packet experiences a particularly large delay, the sender may retransmit the packet even though neither the data packet nor its ACK have been lost. This introduces the possibility of **duplicate data packets** in the sender-to-receiver channel. Happily, protocol *rdt2.2* already has enough functionality (that is, sequence numbers) to handle the case of duplicate packets.

From the sender's viewpoint, retransmission is a panacea. The sender does not know whether a data packet was lost, an ACK was lost, or if the packet or ACK was simply overly delayed. In all cases, the action is the same: retransmit. Implementing a time-based retransmission mechanism requires a **countdown timer** that can interrupt the sender after a given amount of time has expired. The sender will thus need to be able to (1) start the timer each time a packet (either a first-time packet or a retransmission) is sent, (2) respond to a timer interrupt (taking appropriate actions), and (3) stop the timer.

Figure 3.15 shows the sender FSM for *rdt3.0*, a protocol that reliably transfers data over a channel that can corrupt or lose packets; in the homework problems, you'll be asked to provide the receiver FSM for *rdt3.0*. **Figure 3.16** shows how the protocol operates with no lost or delayed packets and how it handles lost data packets. In **Figure 3.16**, time moves forward from the top of the diagram toward the bottom of the