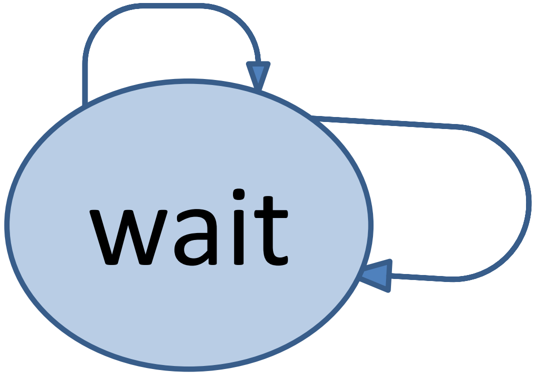
rdt_rcv(rcvpkt)&& notcorrupt(rcvpkt)

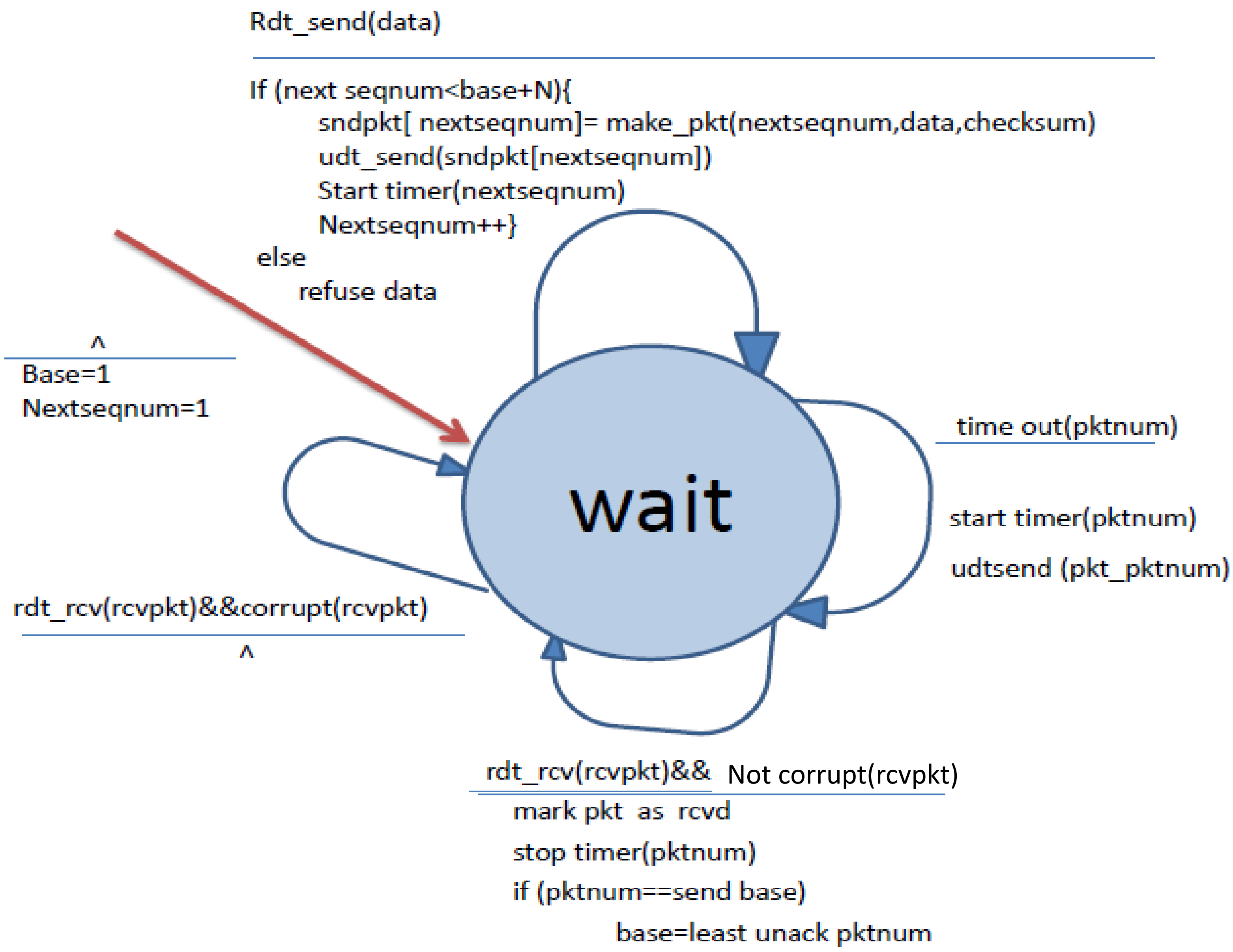
```
If (rcv_base<=rcv pktnum< =rcv_base+N-1) {
          if (rcvdpktnum== rcv_base)
           {
             for(i=rcv_base,i<leastunackpktnum,i++){
                extract(rcvpkt(i),data)
                deliver_data(data)
                }
                sndpkt=make_pkt(rcv_base,ACK,checksum)
               udt_send(sndpkt)
                rcv_base= least unack pktnum
            }
           elseif (rcv_base not acked yet)
            {
                 buffer(rcvpkt)
                  sndpkt=make_pkt(pktnum,ACK,checksum)
                 udt_send(sndpkt)
             }
elseif(rcv_base-N<=rcv pktnum<= rcv_base-1)
 {         sndpkt=make_pkt(rcvpktnum,ACK,checksum)
           udt_send(sndpkt)
 }
```

Λ

Expected seqnum=1

snd_pkt=make_pkt(0,ack,checksum)





```
Rdt_send(data)
_____________________________________________
If (next seqnum<base+N){
        sndpkt[ nextseqnum]= make_pkt(nextseqnum,data,checksum)
        udt_send(sndpkt[nextseqnum])
        Start timer(nextseqnum)
        Nextseqnum++}
 else
      refuse data
```

```
     ^
______________
Base=1
Nextseqnum=1
```

```
 time out(pktnum)
_________________
start timer(pktnum)

udtsend (pkt_pktnum)
```

```
rdt_rcv(rcvpkt)&&corrupt(rcvpkt)
________________________________
              ^
```

```
rdt_rcv(rcvpkt)&&  Not corrupt(rcvpkt)
______________________________________
    mark pkt  as  rcvd
    stop timer(pktnum)
    if (pktnum==send base)
              base=least unack pktnum
```