ACS-3911-050 Computer Network

Chapter 3
Transport Layer

ACS-3911-050 – Slides Used In The Course

THE UNIVERSITY OF WINNIPEG

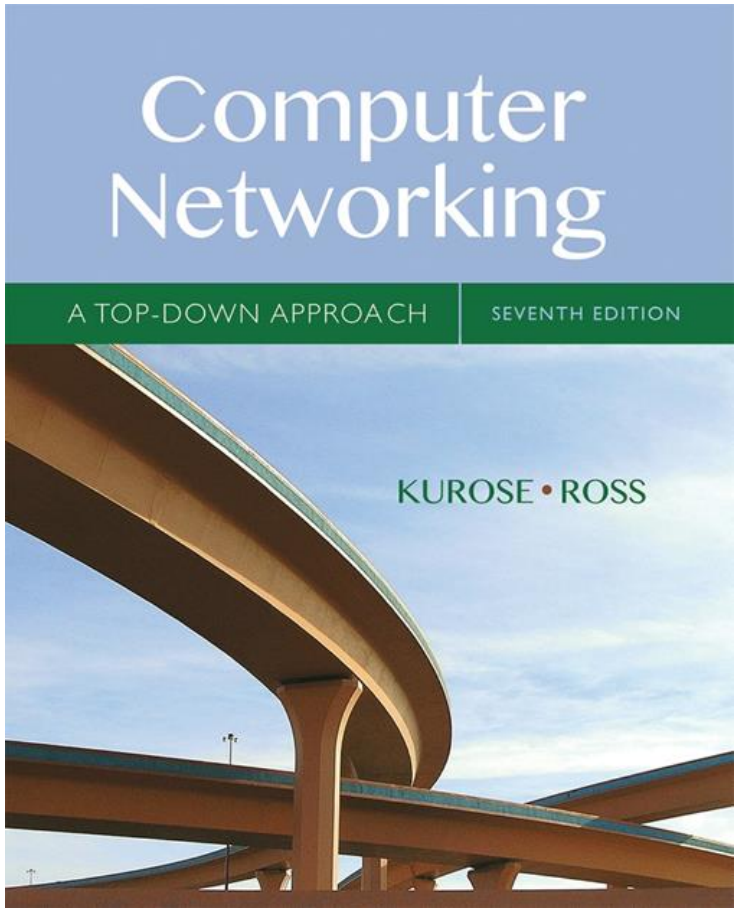
A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



Roadmap

THE UNIVERSITY OF WINNIPEG

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

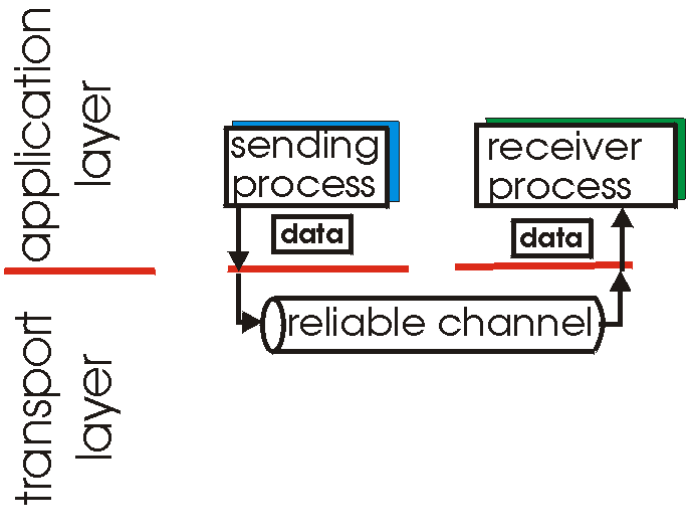
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control

3.7 TCP congestion control

Principle of Reliable Data Transfer

- important in application, transport, link layers
  - top-10 list of important networking topics!

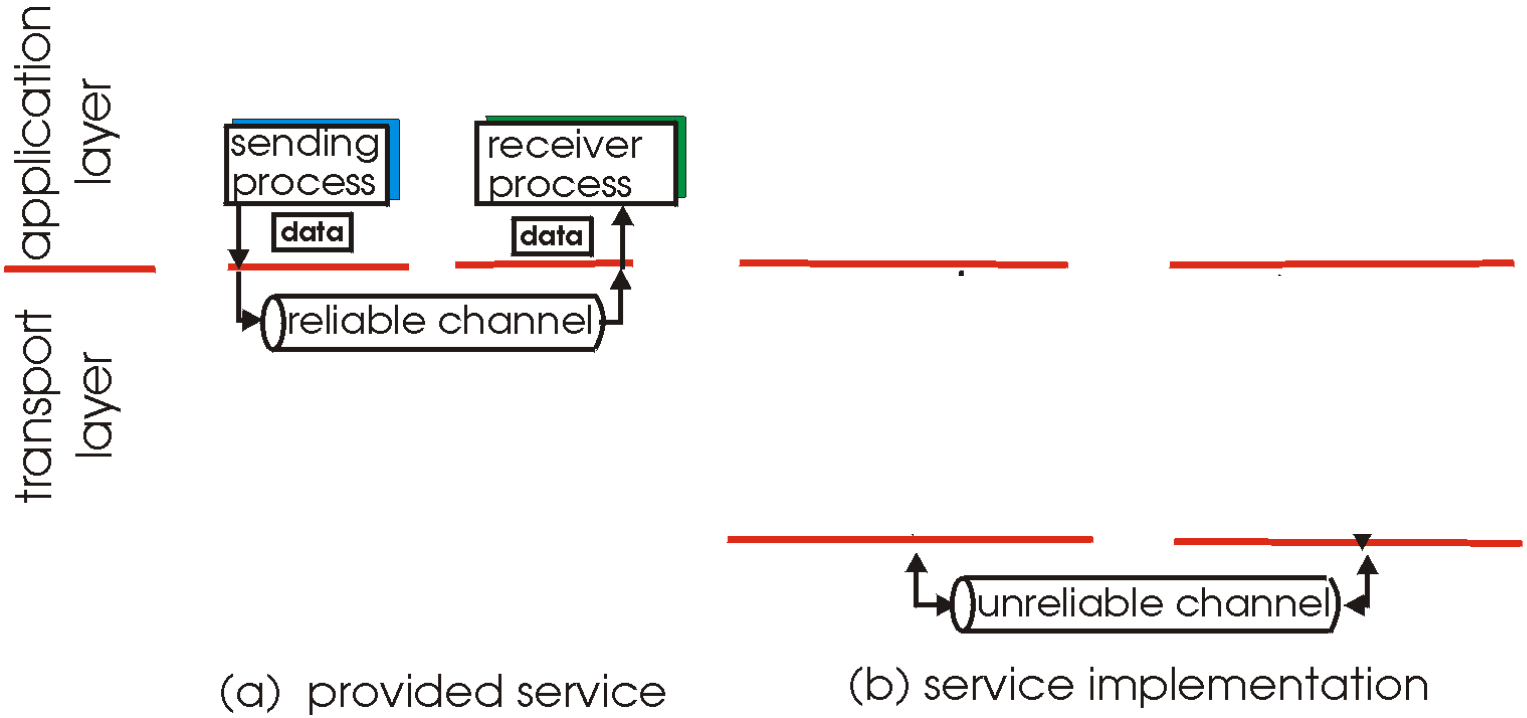


(a) provided service

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principle of Reliable Data Transfer

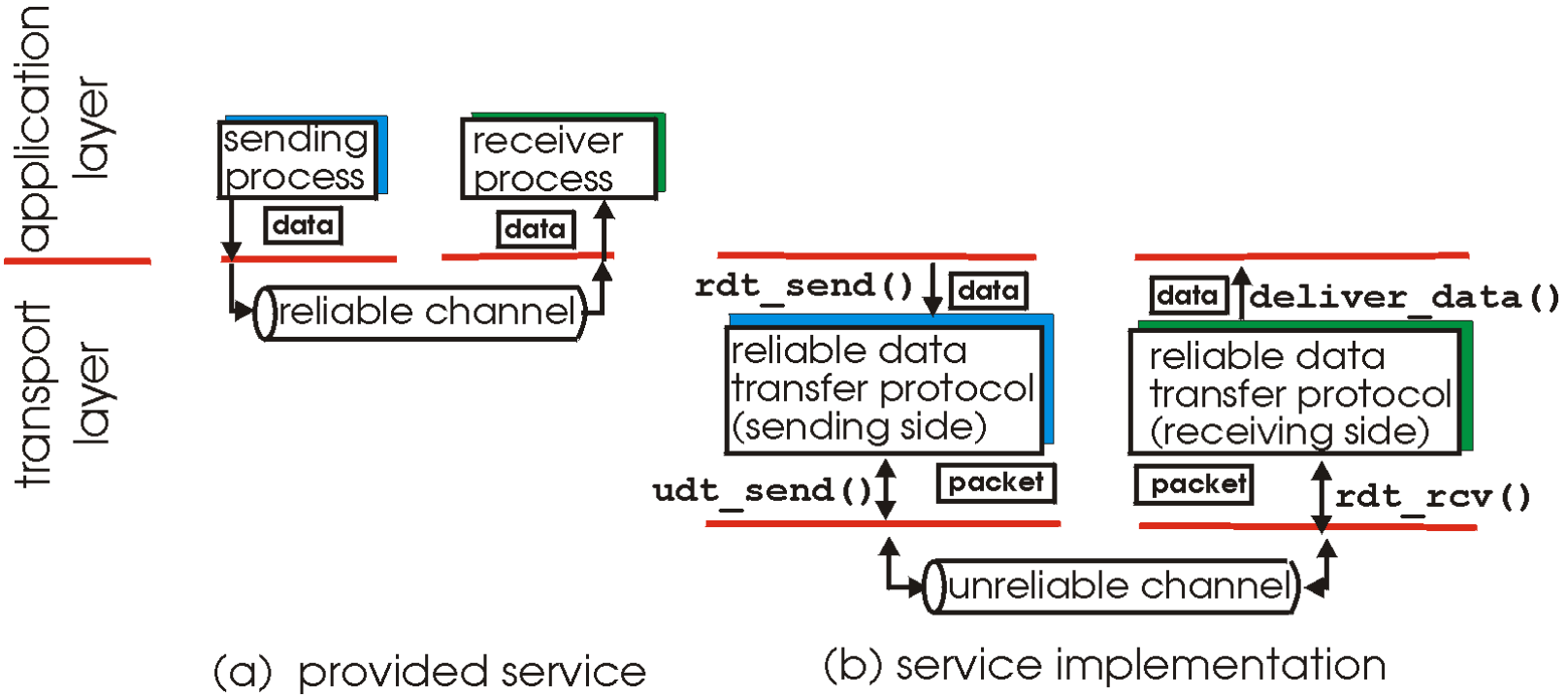
- important in application, transport, link layers
  - top-10 list of important networking topics!



- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principle of Reliable Data Transfer

- important in application, transport, link layers
  - top-10 list of important networking topics!

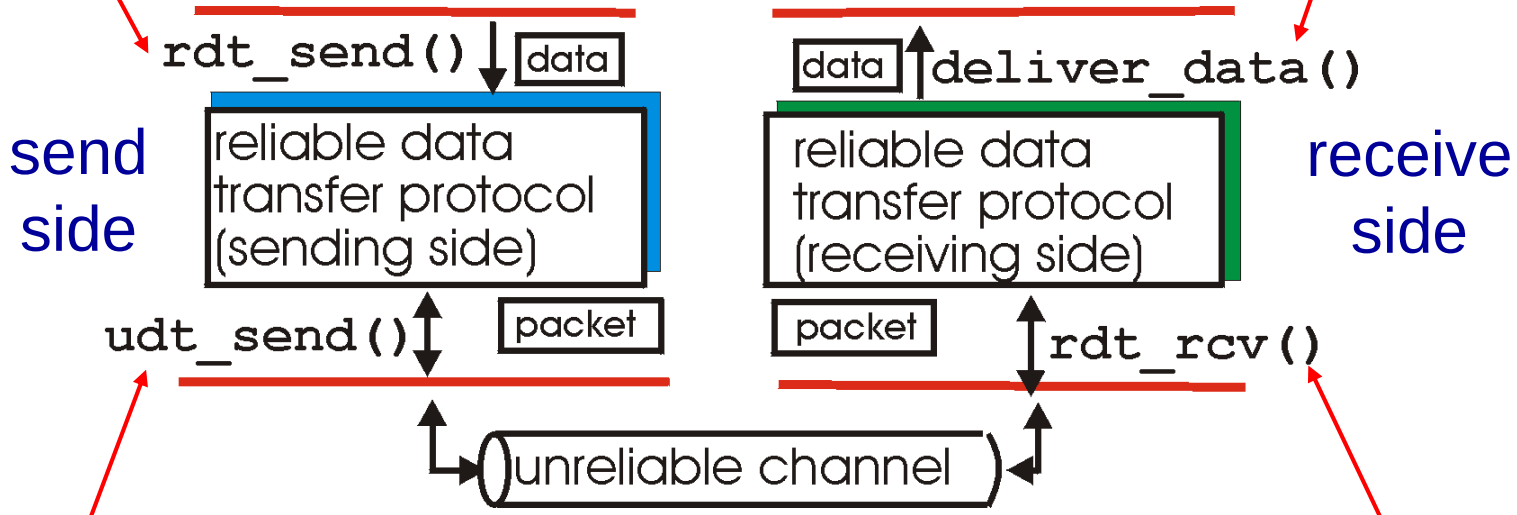


- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Reliable Data Transfer: Getting Started

`rdt_send()`: called from above, (e.g., by app.). Passed data to deliver to receiver upper layer

`deliver_data()`: called by **rdt** to deliver data to upper



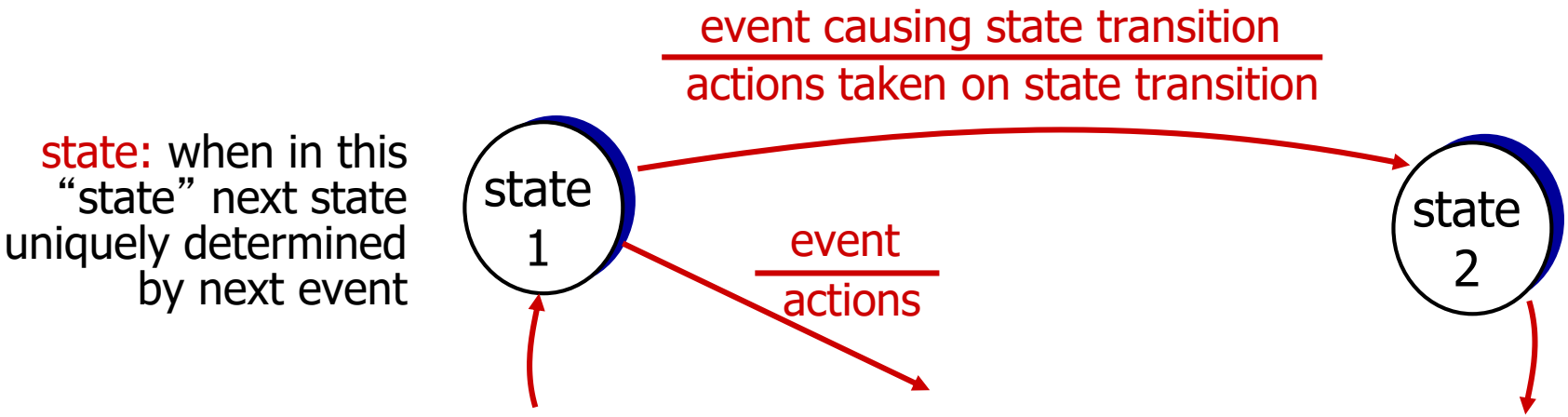
`udt_send()`: called by rdt, to transfer packet over unreliable channel to receiver

`rdt_rcv()`: called when packet arrives on rcv-side of channel

Reliable Data Transfer: Getting Started

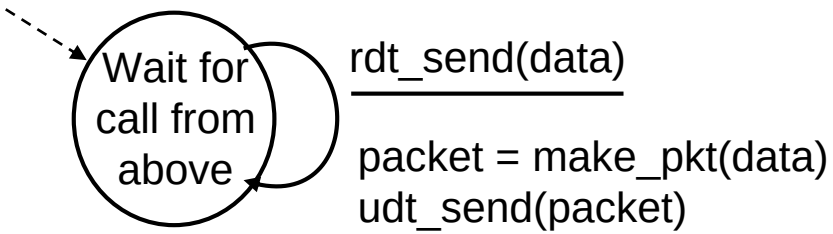
we'll:

- incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- consider only unidirectional data transfer
  - but control info will flow on both directions!
- use finite state machines (FSM) to specify sender, receiver

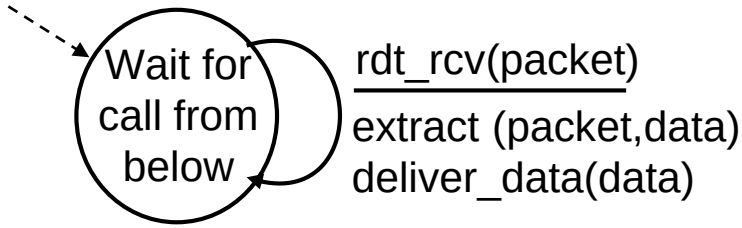


RDT 1.0: Reliable Transfer Over a Reliable Channel

- underlying channel perfectly reliable
  - no bit errors
  - no loss of packets
- separate FSMs for sender, receiver:
  - sender sends data into underlying channel
  - receiver reads data from underlying channel



sender



receiver

RDT 2.0: Channel With Bit Errors

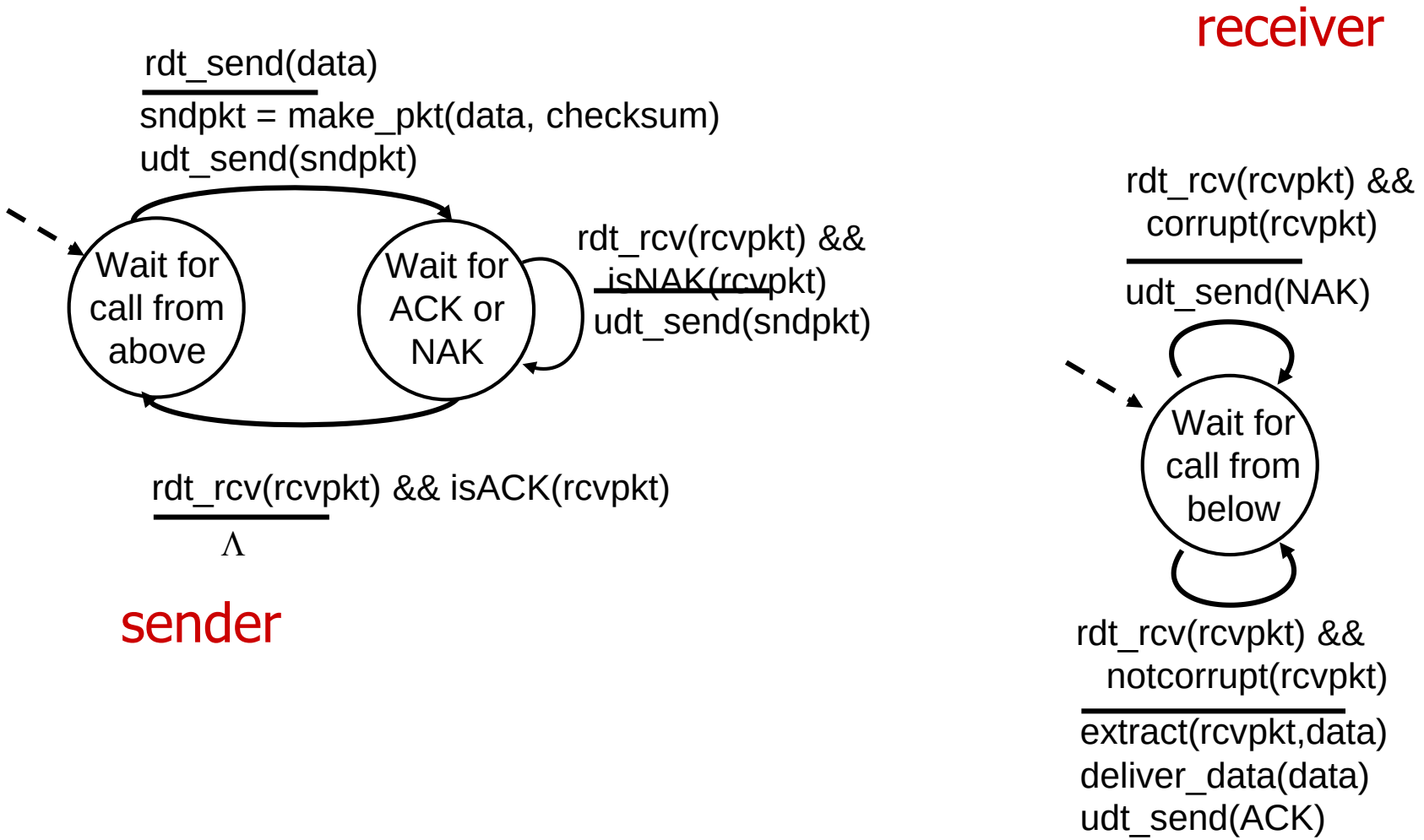
- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors:

How do humans recover from "errors" during conversation?

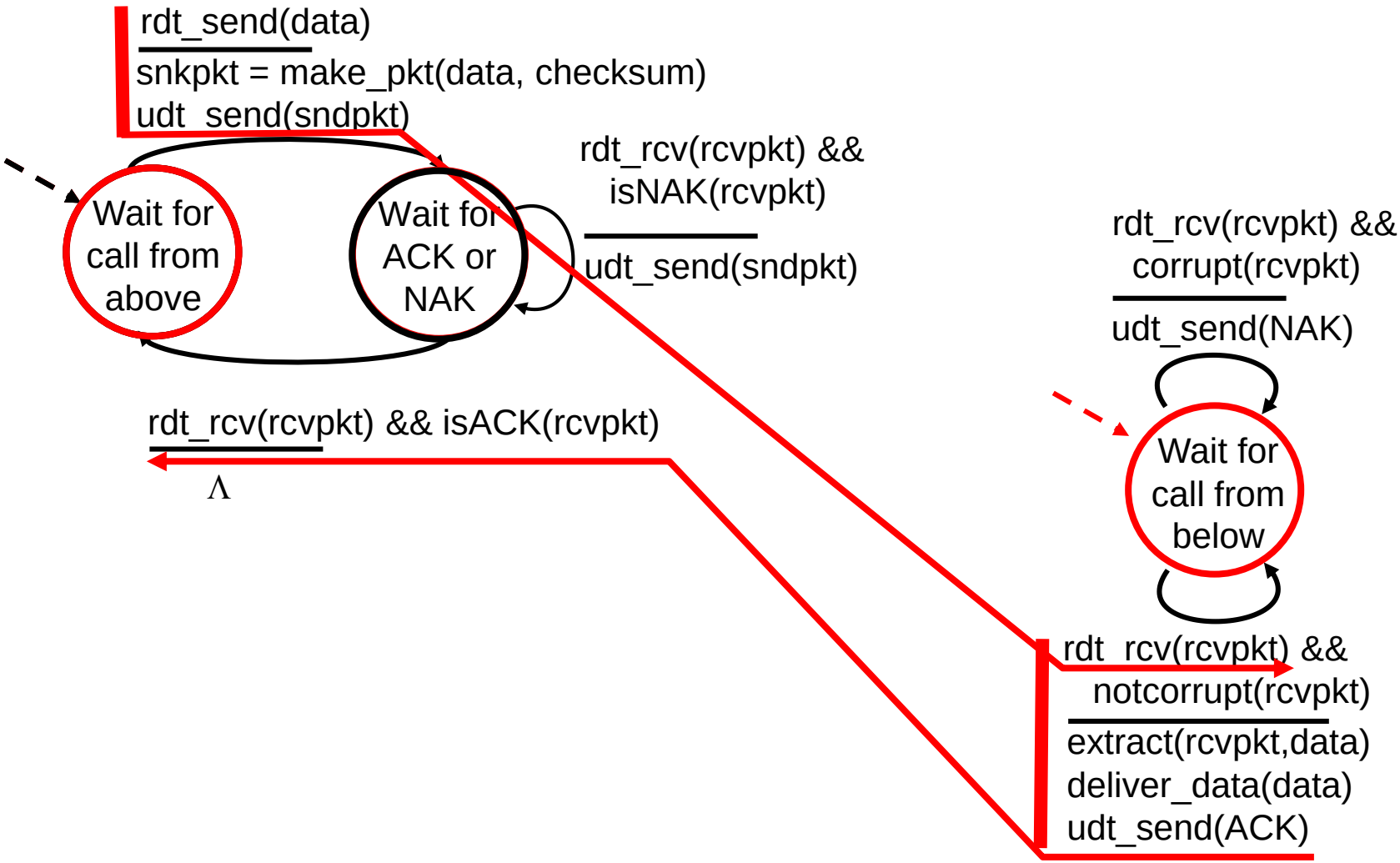
RDT 2.0: Channel With Bit Errors

- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors:
  - *acknowledgements (ACKs):* receiver explicitly tells sender that packet received OK
  - *negative acknowledgements (NAKs):* receiver explicitly tells sender that packet had errors
  - sender retransmits packet on receipt of NAK
- new mechanisms in **`rdt2.0`** (beyond **`rdt1.0`**):
  - error detection
  - receiver feedback: control messages (ACK,NAK) receiver->sender

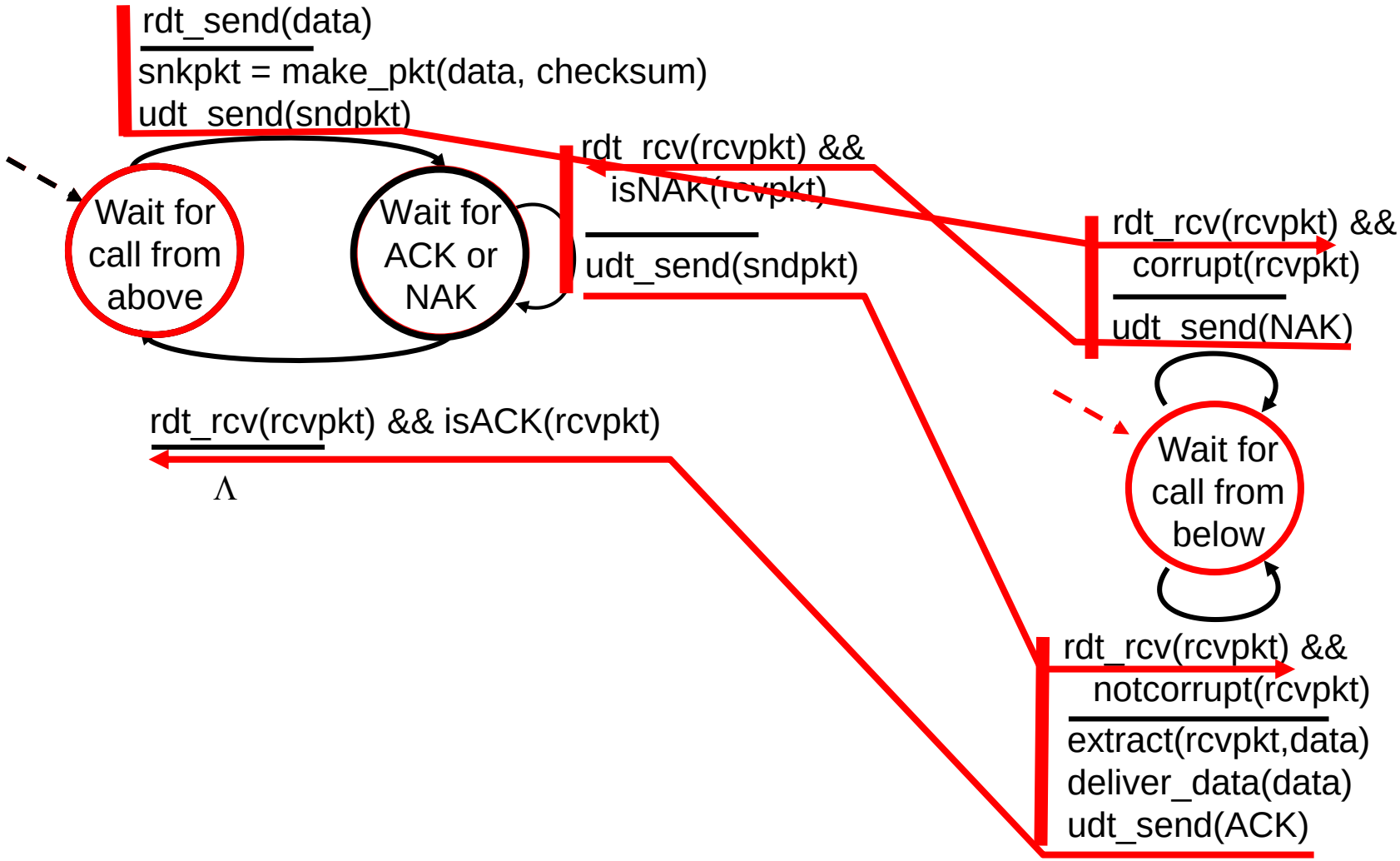
RDT 2.0: FSM Specification



RDT 2.0: Operation With No Errors



RDT 2.0: Error Scenario



RDT 2.0 has a Fatal Flaw!

THE UNIVERSITY OF WINNIPEG

what happens if ACK/NAK corrupted?

- sender doesn't know what happened at receiver!
- Can't just retransmit: possible duplicate

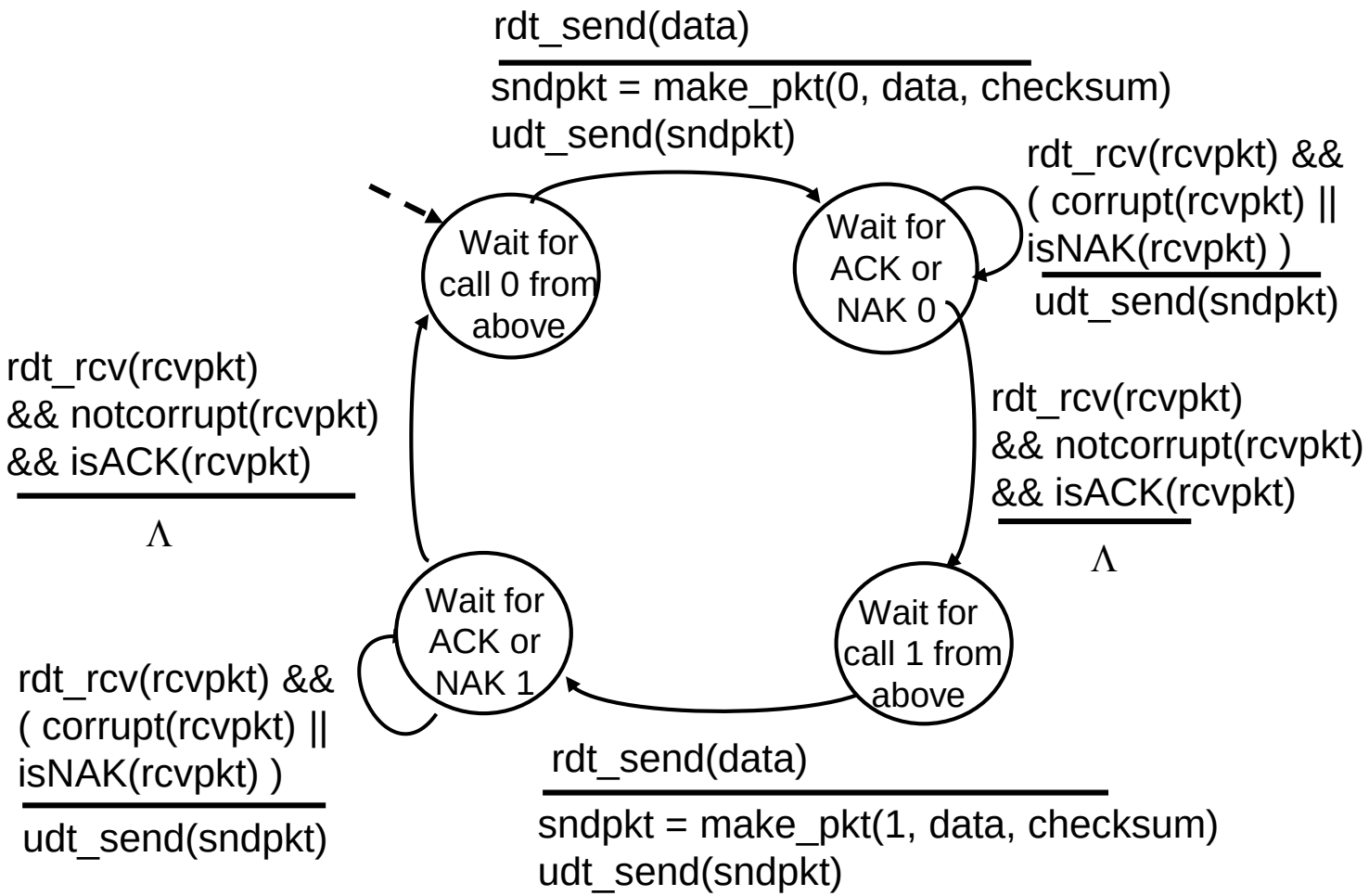
handling duplicates:

- sender retransmits current packet if ACK/NAK corrupted
- sender adds *sequence number* to each packet
- receiver discards (doesn't deliver up) duplicate packet

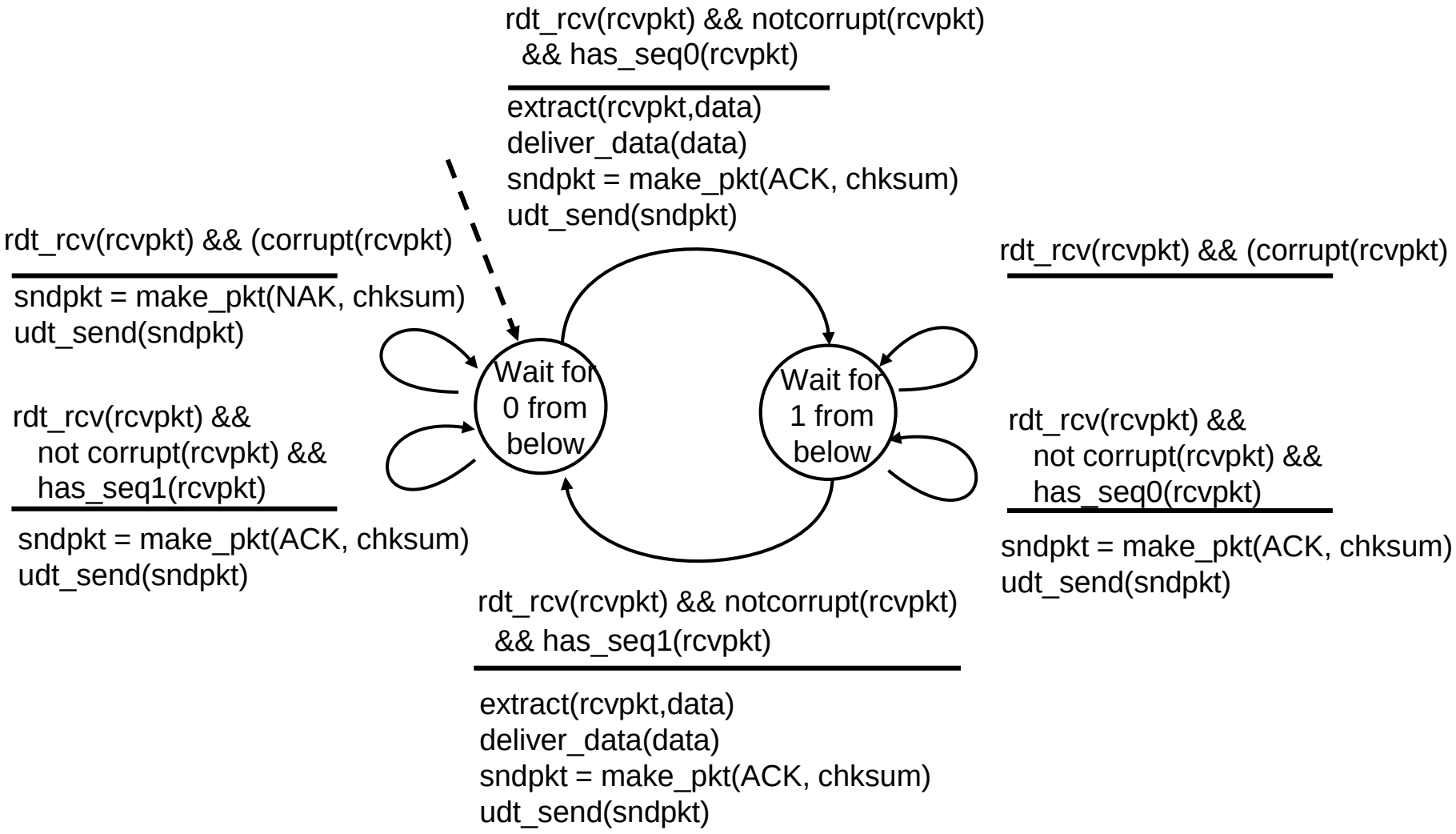
stop and wait

sender sends one packet, then waits for receiver response

RDT 2.1: Sender, Handles Garbled ACK/NAKs



RDT 2.1: Receiver, Handles Garbled ACK/NAKs



RDT 2.1: Discussion

THE UNIVERSITY OF WINNIPEG

sender:

- seq # added to pkt
- two seq. #' s (0,1) will suffice. Why?
- must check if received ACK/NAK corrupted
- twice as many states
  - state must "remember" whether "expected" pkt should have seq # of 0 or 1

receiver:

- must check if received packet is duplicate
  - state indicates whether 0 or 1 is expected pkt seq #
- note: receiver can *not* know if its last ACK/NAK received OK at sender

Questions?