CPSC 317 COMPUTER NETWORKING

2023W2: Transport – Day 2 – State Machines and Reliability

READING

- Reading: 3.4.1

LEARNING GOALS

Protocols and state machines

- Given a protocol, express it as a set of communicating finite state machines (FSMs)
- Given a set of communicating FSMs, trace the execution of the protocol for various event scenarios
- Given a set of communicating FSMs, analyze the execution of the state machines to determine whether or not the protocol operates correctly and propose ways to solve any problems

LEARNING GOALS

Alternating Bit Protocol

- Describe/build/execute/trace the various versions of the alternating bit protocol (stop and wait) protocol
- Explain how corruption in segments are detected and at what layer(s) in the protocol stack this is done
- Explain how corruption in segments is handled by a reliable delivery protocol

REMINDER: PROTOCOLS

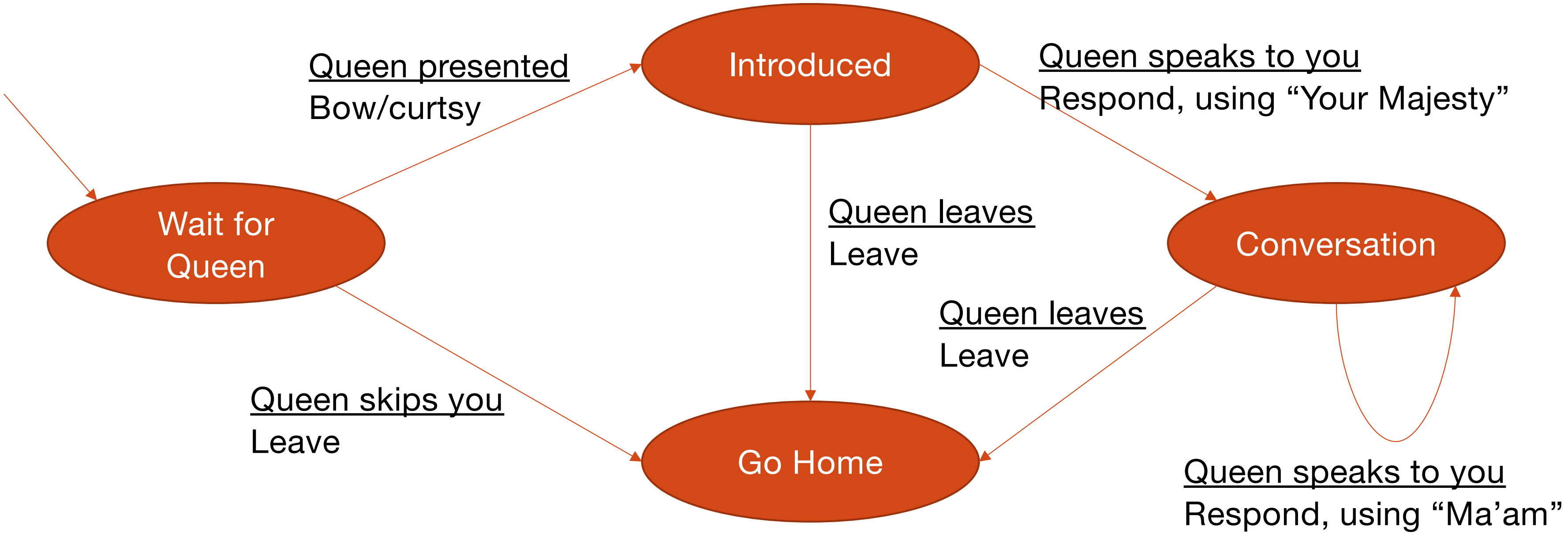
- From the textbook: A **protocol** defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event
- Key concepts:
  - Format and order of messages
  - Actions to be taken on events
- A fully-defined protocol must provide a proper action for any event in any state

A few years ago,
the Queen visited UBC

PROTOCOL EXAMPLE: GREETING THE QUEEN

- Don't speak until spoken to
- Upon presentation to the Queen
  - Men do a neck bow
  - Women a slight curtsy
- Queen speaks to you
  - First salutation is “Your Majesty”
  - Subsequently it is “Ma'am”

QUEEN GREETING PROTOCOL STATE MACHINE



Wait for Queen

Introduced

Conversation

Go Home

<u>Queen presented</u>
Bow/curtsy

<u>Queen speaks to you</u>
Respond, using “Your Majesty”

<u>Queen leaves</u>
Leave

<u>Queen leaves</u>
Leave

<u>Queen skips you</u>
Leave

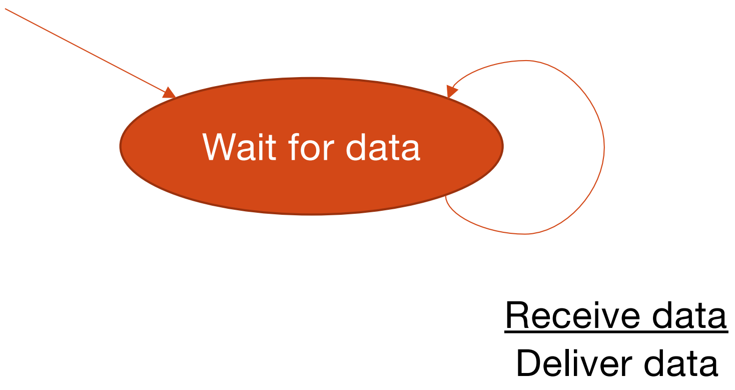
<u>Queen speaks to you</u>
Respond, using “Ma’am”

BUILDING A RELIABLE PROTOCOL (V.0)

- Let’s create a protocol for reliable delivery
  - Send only one segment at a time
  - Identify when sending is allowable action
  - Enumerate events and actions for both sender and receiver
  - Draw state machine
- Initial scenario assumptions
  - One sender, one receiver
  - Data to send comes from upper layer
  - Segments are never lost, never corrupted

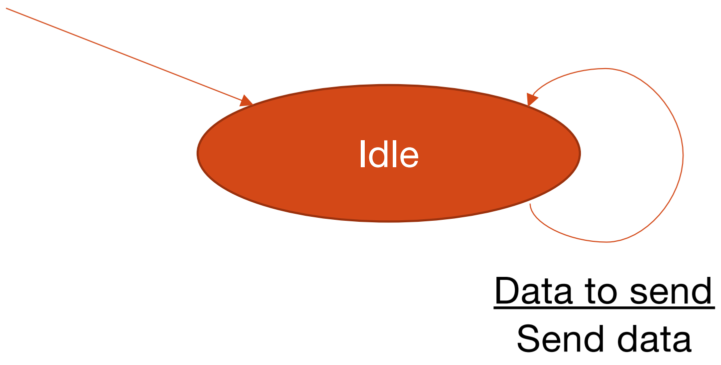
STATE MACHINE: RELIABLE CHANNEL

Receiver


Wait for data

Receive data
Deliver data

Sender


Idle

Data to send
Send data

WHAT COULD GO WRONG?

- We have assumed that nothing bad ever happens
- What bad things could happen in a real-world scenario?

BUILDING A RELIABLE PROTOCOL (V.1)

- Let’s create a protocol for reliable delivery
  - Send only one segment at a time
  - Identify when sending is allowable action
  - Identify when re-sending is required
  - Enumerate events and actions for both sender and receiver
  - Draw state machine
- Next scenario assumptions
  - One sender, one receiver
  - Data to send comes from upper layer
  - Segments are never lost, but may be corrupted

POSSIBLE EVENTS AND ACTIONS

Receiver

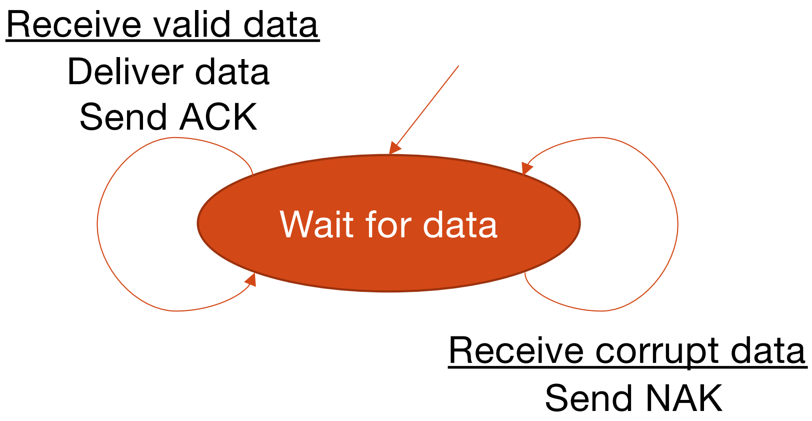
- Segment received without problems
  - Send ACK
- Segment received corrupted
  - Send NAK

Sender

- Data ready to send
  - Send data
- Receive ACK
  - Get ready to send more data
- Receive NAK
  - Re-send data

STATE MACHINE: CORRUPTION SCENARIO

Receiver



Receive valid data

Deliver data

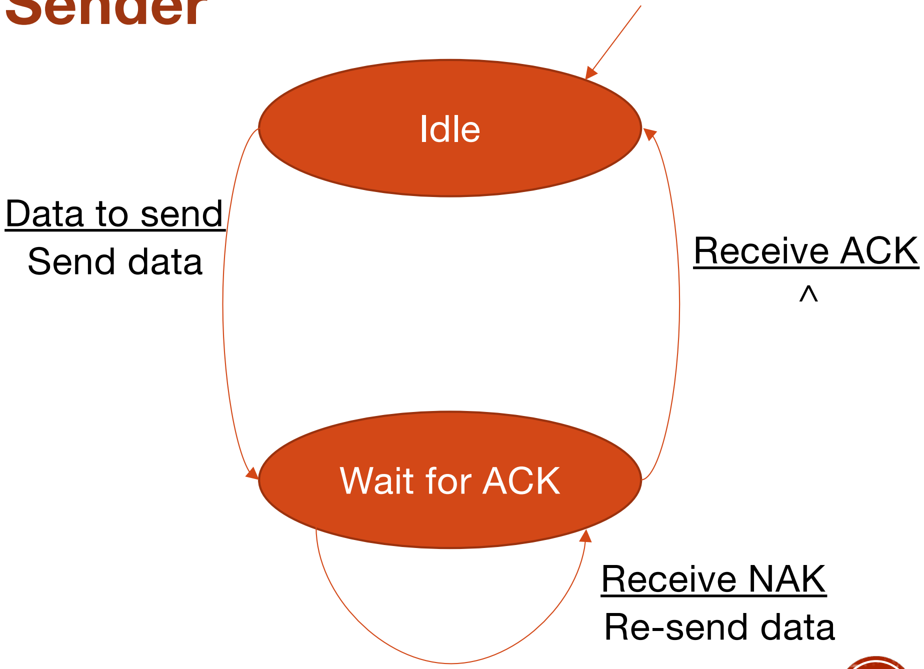
Send ACK

Wait for data

Receive corrupt data

Send NAK

Sender



Idle

Data to send

Send data

Receive ACK

^

Wait for ACK

Receive NAK

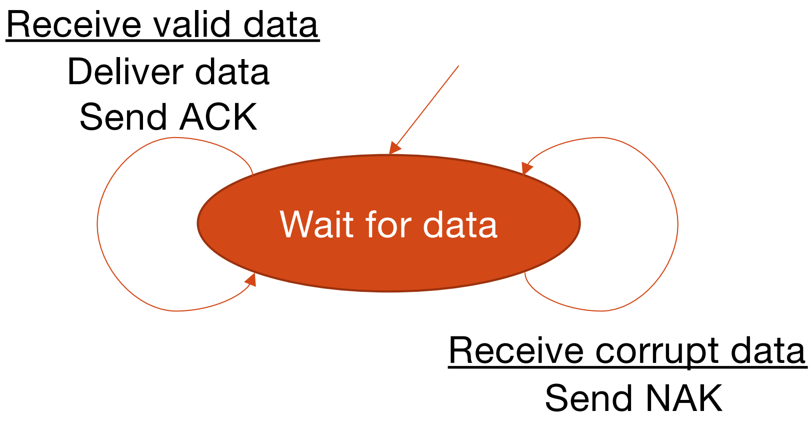
Re-send data

THE ACK/NAK MIGHT BE CORRUPTED

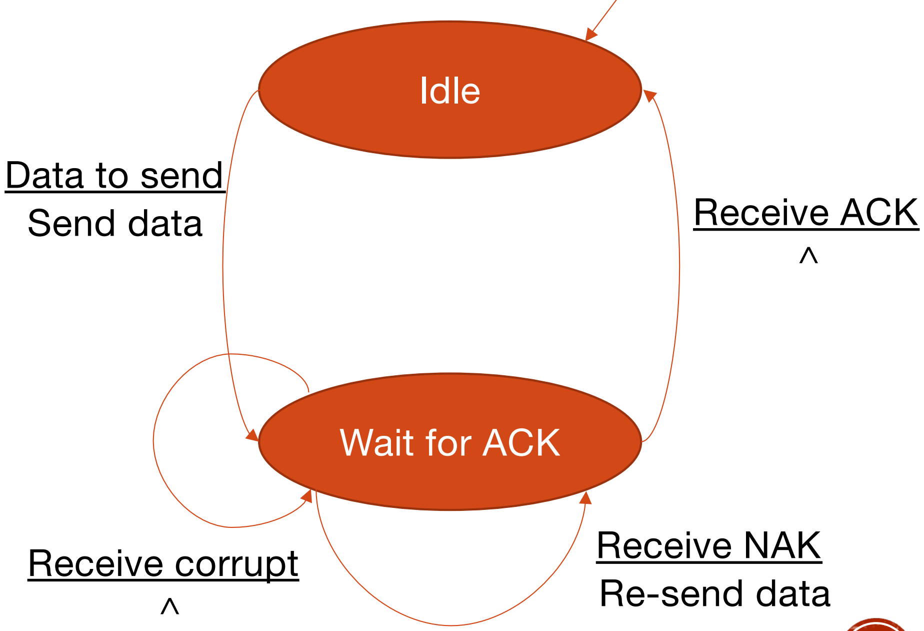
- What if the ACK/NAK is corrupted?
  - Sender cannot be sure if the receiver received the previous segment
- What can the sender do after receiving a corrupted ACK/NAK?
  - Ignore the corrupt ACK/NAK?
  - Treat it as an ACK?
  - Treat it as a NAK?

STATE MACHINE: CORRUPTION OPTION I

Receiver



Sender

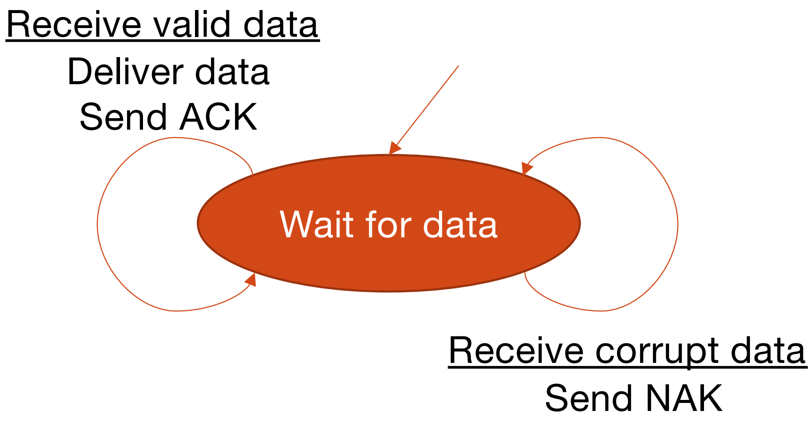


THE ACK/NAK MIGHT BE CORRUPTED

- What if the ACK/NAK is corrupted?
  - Sender cannot be sure if the receiver received the previous segment
- What can the sender do after receiving a corrupted ACK/NAK?
  - Ignore the corrupt ACK/NAK?
    - Everything stops
  - Treat it as an ACK?
  - Treat it as a NAK?

STATE MACHINE: CORRUPTION OPTION II

Receiver

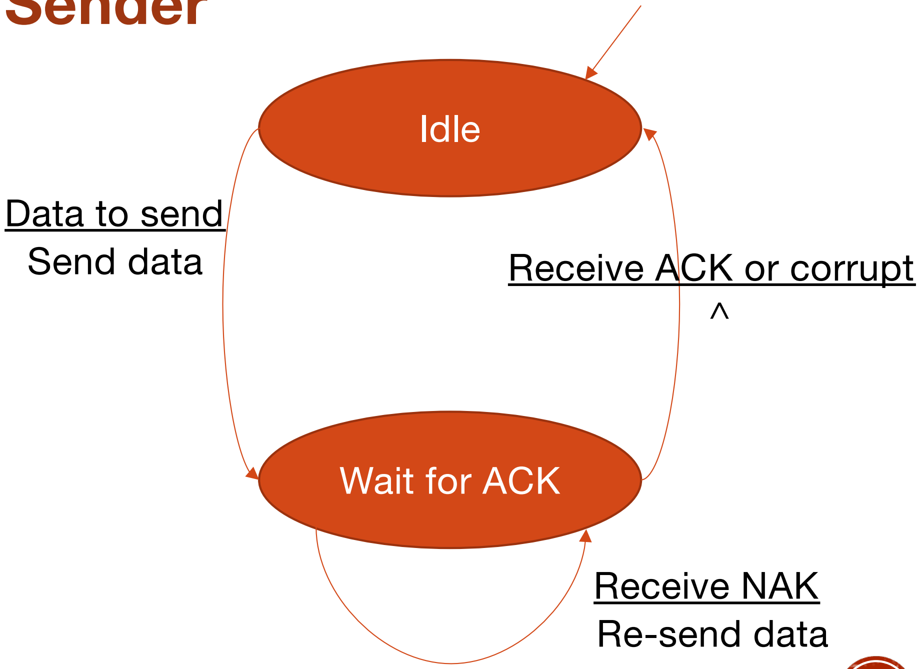


Receive valid data
Deliver data
Send ACK

Wait for data

Receive corrupt data
Send NAK

Sender



Idle

Data to send
Send data

Receive ACK or corrupt
^

Wait for ACK

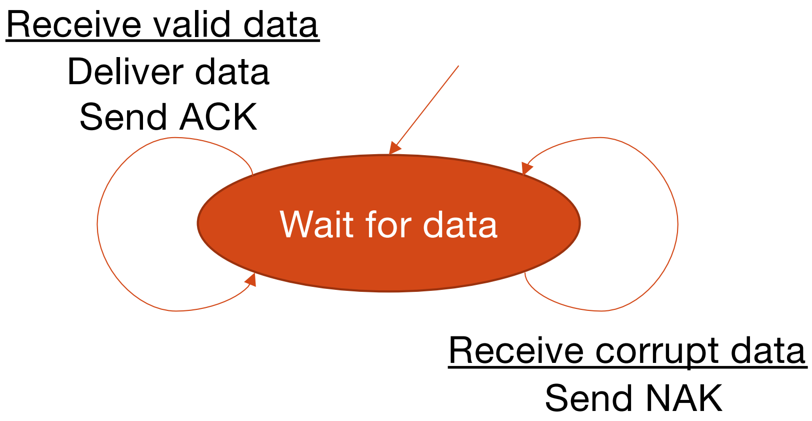
Receive NAK
Re-send data

THE ACK/NAK MIGHT BE CORRUPTED

- What if the ACK/NAK is corrupted?
  - Sender cannot be sure if the receiver received the previous segment
- What can the sender do after receiving a corrupted ACK/NAK?
  - Ignore the corrupt ACK/NAK?
    - Everything stops
  - Treat it as an ACK?
    - If it was a NAK, the receiver never receives the data
  - Treat it as a NAK?

STATE MACHINE: CORRUPTION OPTION III

Receiver

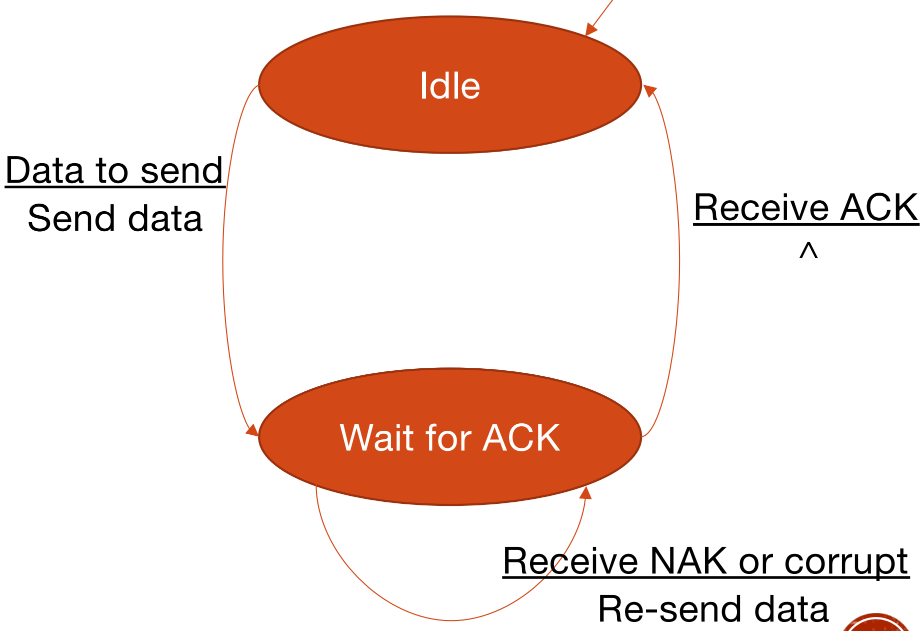


Receive valid data
Deliver data
Send ACK

Wait for data

Receive corrupt data
Send NAK

Sender



Idle

Data to send
Send data

Receive ACK
^

Wait for ACK

Receive NAK or corrupt
Re-send data

THE ACK/NAK MIGHT BE CORRUPTED

- What if the ACK/NAK is corrupted?
  - Sender cannot be sure if the receiver received the previous segment
- What can the sender do after receiving a corrupted ACK/NAK?
  - Ignore the corrupt ACK/NAK?
    - Everything stops
  - Treat it as an ACK?
    - If it was a NAK, the receiver never receives the data
  - Treat it as a NAK?
    - If it was an ACK, the receiver receives the data twice

DEALING WITH DUPLICATE TRANSFER

- How can the receiver deal with a duplicate segment?
- What information needs to be included to allow receiver to identify segment as duplicate?
- Reminder: scenario does not lose data, only corrupts

POSSIBLE EVENTS

Receiver

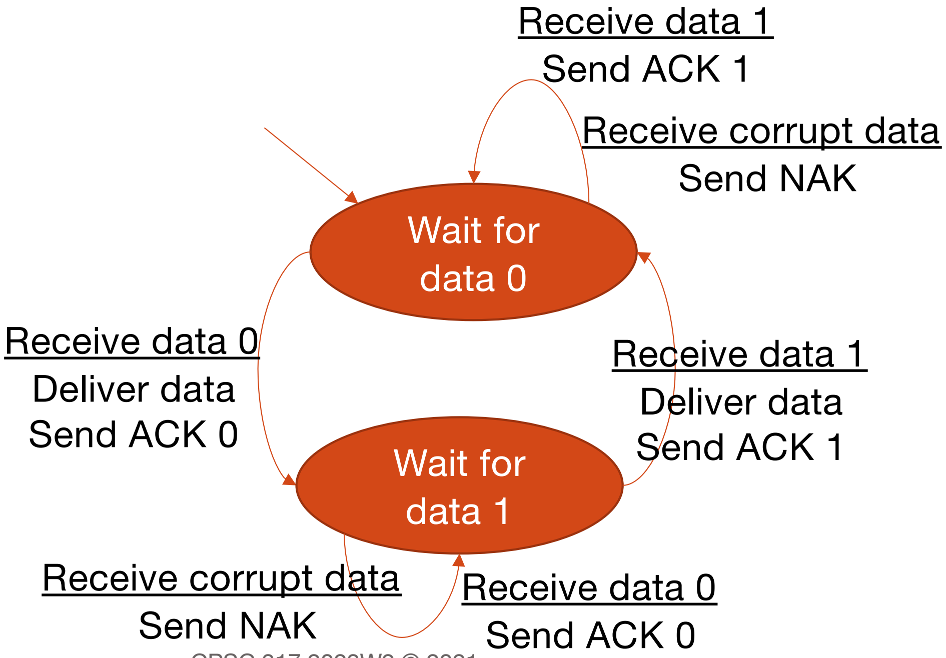
- Segment 0 received without problems
- Segment 1 received without problems
- Segment received corrupted

Sender

- Data ready to send
- Receive ACK 0
- Receive ACK 1
- Receive NAK
- Receive corrupt response

ALTERNATING BIT PROTOCOL

Receiver



Receive data 1
Send ACK 1

Receive corrupt data
Send NAK

Wait for data 0

Receive data 0
Deliver data
Send ACK 0

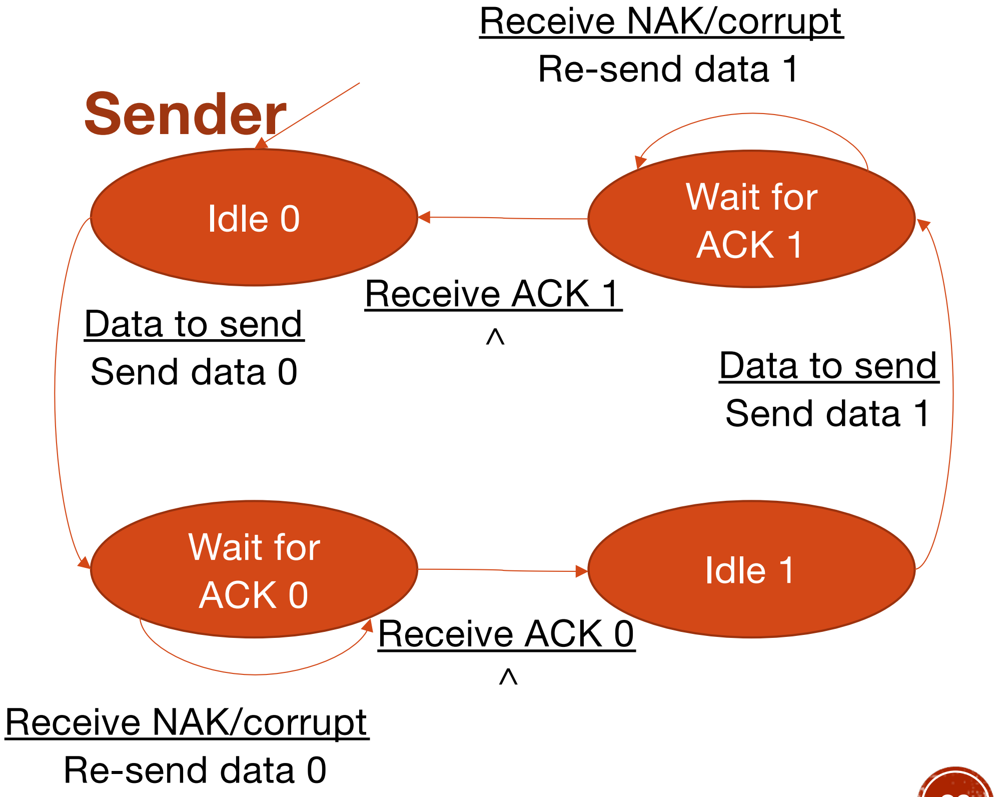
Receive data 1
Deliver data
Send ACK 1

Wait for data 1

Receive corrupt data
Send NAK

Receive data 0
Send ACK 0

Sender



Receive NAK/corrupt
Re-send data 1

Idle 0

Wait for ACK 1

Receive ACK 1
^

Data to send
Send data 0

Data to send
Send data 1

Wait for ACK 0

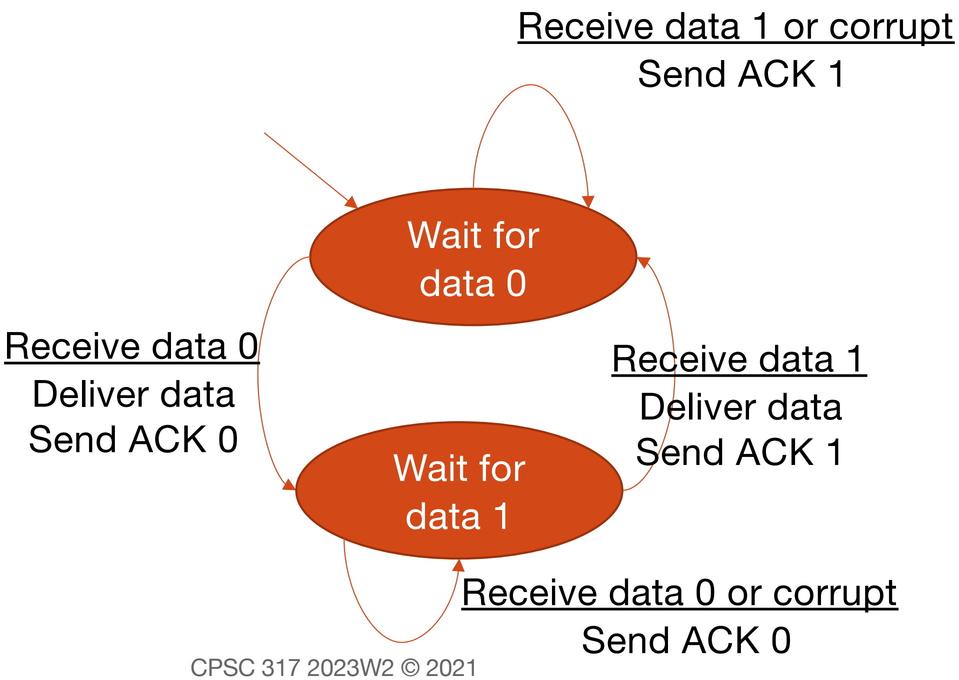
Idle 1

Receive ACK 0
^

Receive NAK/corrupt
Re-send data 0

ALTERNATING BIT PROTOCOL (NO NAK)

Receiver



Receive data 1 or corrupt
Send ACK 1

Wait for data 0

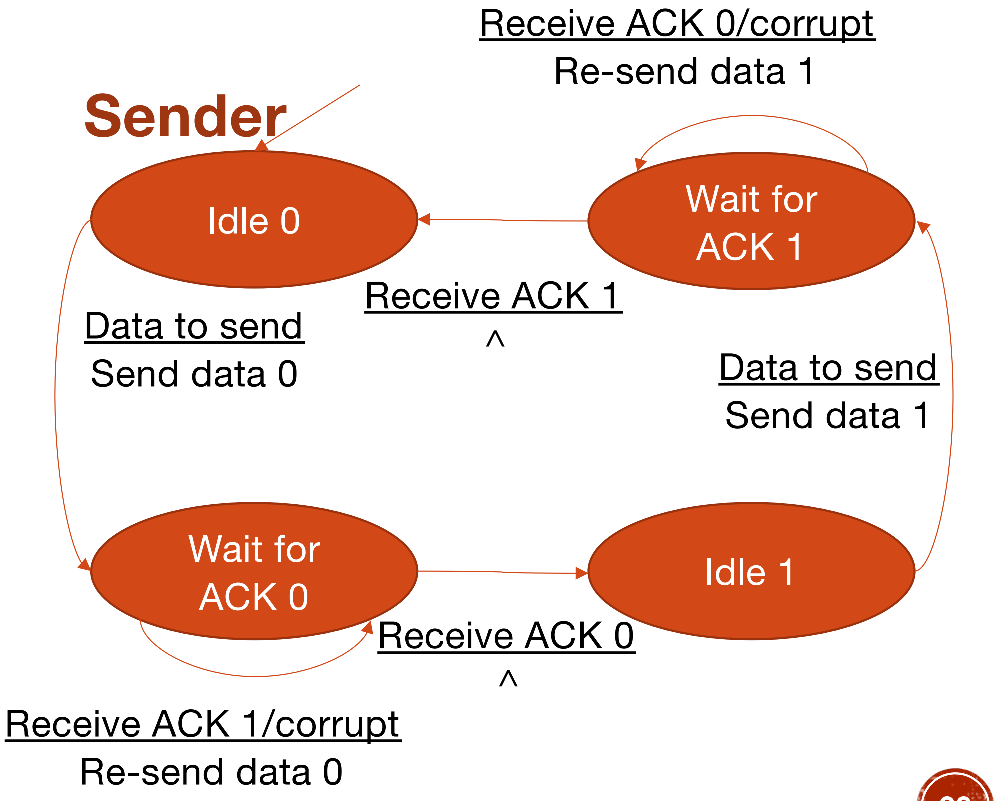
Receive data 0
Deliver data
Send ACK 0

Receive data 1
Deliver data
Send ACK 1

Wait for data 1

Receive data 0 or corrupt
Send ACK 0

Sender



Receive ACK 0/corrupt
Re-send data 1

Idle 0

Wait for ACK 1

Receive ACK 1
^

Data to send
Send data 0

Data to send
Send data 1

Wait for ACK 0

Idle 1

Receive ACK 0
^

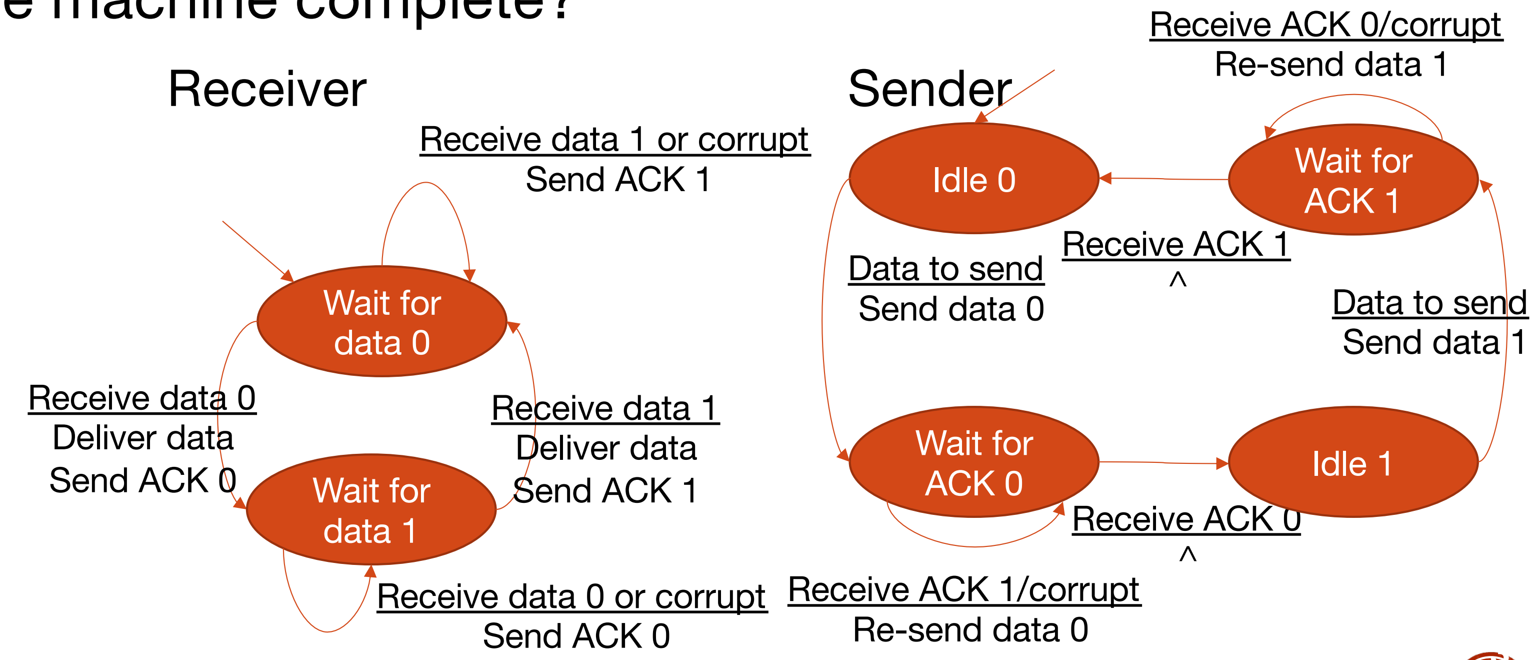
Receive ACK 1/corrupt
Re-send data 0

CLICKER QUESTION

Is this state machine complete?

A. Yes

B. No



Receiver

Sender

Wait for data 0

Wait for data 1

Receive data 0
Deliver data
Send ACK 0

Receive data 1
Deliver data
Send ACK 1

Receive data 1 or corrupt
Send ACK 1

Receive data 0 or corrupt
Send ACK 0

Idle 0

Wait for ACK 1

Wait for ACK 0

Idle 1

Receive ACK 0/corrupt
Re-send data 1

Receive ACK 1
^

Data to send
Send data 0

Data to send
Send data 1

Receive ACK 0
^

Receive ACK 1/corrupt
Re-send data 0

IN-CLASS ACTIVITY

- ICA42
- Create FSMs for DNS servers