ANSWERS

A12.2 Topologies and wireless networks

Networks and the web · A level · OCR H446 1.3.3, AQA 7517 4.9.2.1, Eduqas A500QS 2.1 · about 25 min

BugBotLab

What this lesson is about

Physical and logical bus and star; Wi-Fi, SSIDs, WPA2 and CSMA/CA with RTS/CTS.

- Bus and star, physically
- Physical star, logical bus
- Wireless networking
- CSMA/CA

Questions

6 marks in all

1. Devices are cabled to a central hub, which repeats every frame out of every port. What is the network's topology? [1 mark]

- ☐ **A** Physical star, logical bus
- ☐ **B** Physical bus, logical star
- ☐ **C** Physical star, logical star
- ☐ **D** Physical bus, logical bus

> **Answer:** A. It is wired as a star, but every device receives every frame over what is in effect one shared channel, as on a bus.

2. Which of these are advantages of a physical star over a physical bus? Choose all that apply. [1 mark]

Tick every answer that is true.

- ☐ **A** A single cable failure only affects one device
- ☐ **B** It uses less cable
- ☐ **C** Performance holds up better when the network is busy
- ☐ **D** There is no central device that could fail

> **Answer:** A, C. Each device has its own cable, and a switch keeps traffic apart. But a star uses more cable and the central device is a single point of failure.

3. Why does Wi-Fi use collision avoidance rather than detecting collisions? [1 mark]

- ☐ **A** A transmitting wireless device cannot hear other signals over its own, so it cannot detect a collision
- ☐ **B** Wireless signals never collide
- ☐ **C** Collision detection needs a hub
- ☐ **D** RTS/CTS is only possible on cables

> **Answer:** A. A device's own transmission drowns out others at its antenna, so it must avoid collisions by sensing the channel and backing off.

4. Put the steps of CSMA/CA with RTS/CTS in order. [1 mark]

Number the lines 1 to 6 to put them in the right order.

___ `Send a request to send (RTS)`

___ `Receive clear to send (CTS) from the access point`

___ `Transmit the data frame`

___ `If it is busy, wait a random back-off time and listen again`

___ `Wait for an acknowledgement (ACK)`

___ `Listen to check whether the channel is idle`

> **Answer:**
>
> ```
> Listen to check whether the channel is idle
> If it is busy, wait a random back-off time and listen again
> Send a request to send (RTS)
> Receive clear to send (CTS) from the access point
> Transmit the data frame
> Wait for an acknowledgement (ACK)
> ```
>
> The device only sends RTS once the channel is idle; the CTS silences hidden nodes before the data is sent, and the ACK confirms it arrived.

5. What problem does RTS/CTS solve? [1 mark]

☐ **A** Two devices that cannot hear each other both transmit to the access point at once (the hidden node problem)

☐ **B** A device forgetting the SSID

☐ **C** Devices with spoofed MAC addresses

☐ **D** Weak encryption keys

> **Answer:** A. Every device in range of the access point hears its CTS, including ones out of range of the sender, and stays quiet.

6. Why is a MAC address allow list, on its own, weak protection for a wireless network? [1 mark]

☐ **A** MAC addresses can be read from captured frames and copied (spoofed)

☐ **B** MAC addresses change every time a device joins

☐ **C** Access points cannot read MAC addresses

☐ **D** It stops the SSID being broadcast

> **Answer:** A. An attacker can see allowed MAC addresses in traffic and set their own device to use one; strong encryption such as WPA2 is still needed.

The task: wait for a quiet channel

The Neighbour is sending frames for the first couple of seconds. The AccessPoint replies `CTS bot` when it hears `RTS`, and `ACK bot` when it hears a message starting `DATA`. Both replies come about 0.2 s later. Carry out CSMA/CA with RTS/CTS: 1. Call `messages()` once to clear anything old. Then listen for a window of 0.4 s: `wait(0.4)`, then `messages()`. 2. If anything arrived in the window, print `busy, backing off`, wait a random time from 0.1 to 0.5 seconds (`random.uniform(0.1, 0.5)`), and listen for another window. Repeat until a window is silent. 3. Print `channel clear` and send `RTS`. 4. Wait for a message starting `CTS`, checking `messages()` every 0.1 s for up to 2 seconds. When it comes, print `got CTS` and send `DATA hello`. 5. Wait the same way for a message starting `ACK`, then print `got ACK`. The robot does not drive.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
import random

send("RTS")
wait(1)
print(messages())
```

The hint students can ask for: Listen for a short window. If anything arrived, the channel is busy: say so, wait a random time and listen again. Only when a whole window is silent do you ask to send, and only after the access point answers do you send the data and wait for its acknowledgement.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import random

def heard_in(seconds):
    wait(seconds)
    return len(messages()) > 0

def wait_for(word, timeout):
    waited = 0
    while waited < timeout:
        wait(0.1)
        waited = waited + 0.1
        for sender, text in messages():
            if text.startswith(word):
                return True
    return False

messages()
while heard_in(0.4):
    print("busy, backing off")
    wait(random.uniform(0.1, 0.5))
print("channel clear")
send("RTS")
if wait_for("CTS", 2):
    print("got CTS")
    send("DATA hello")
```

```
    if wait_for("ACK", 2):
        print("got ACK")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.