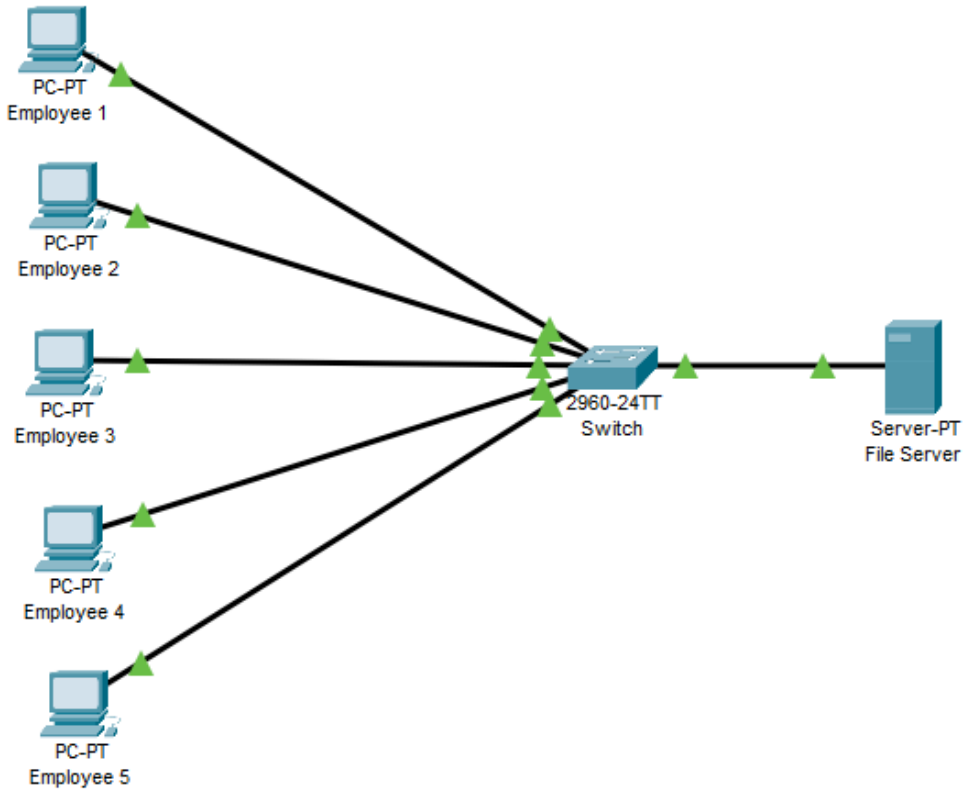
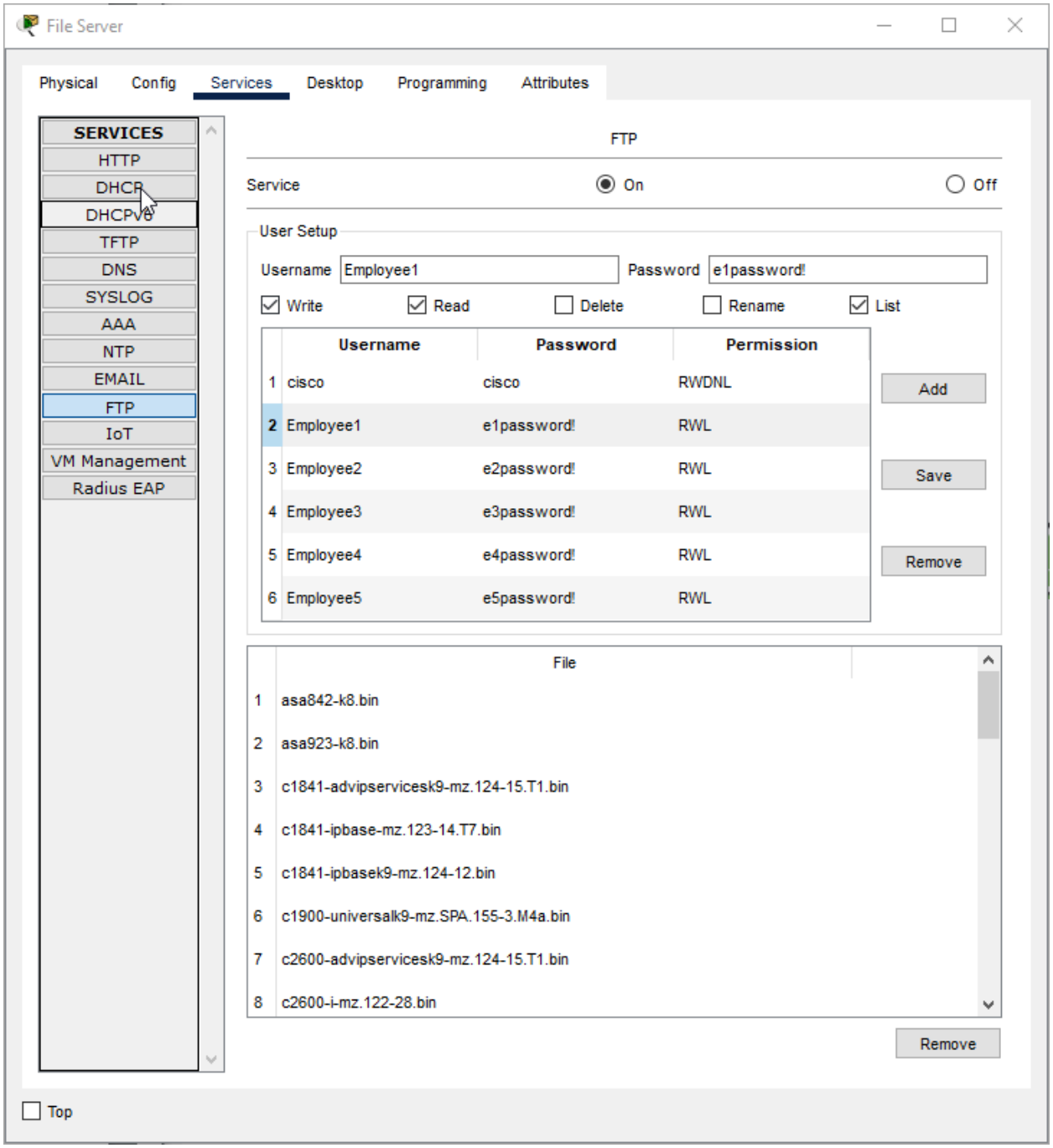
Assignment 2

Task 1.1 - Device Topology

Screenshot 1:

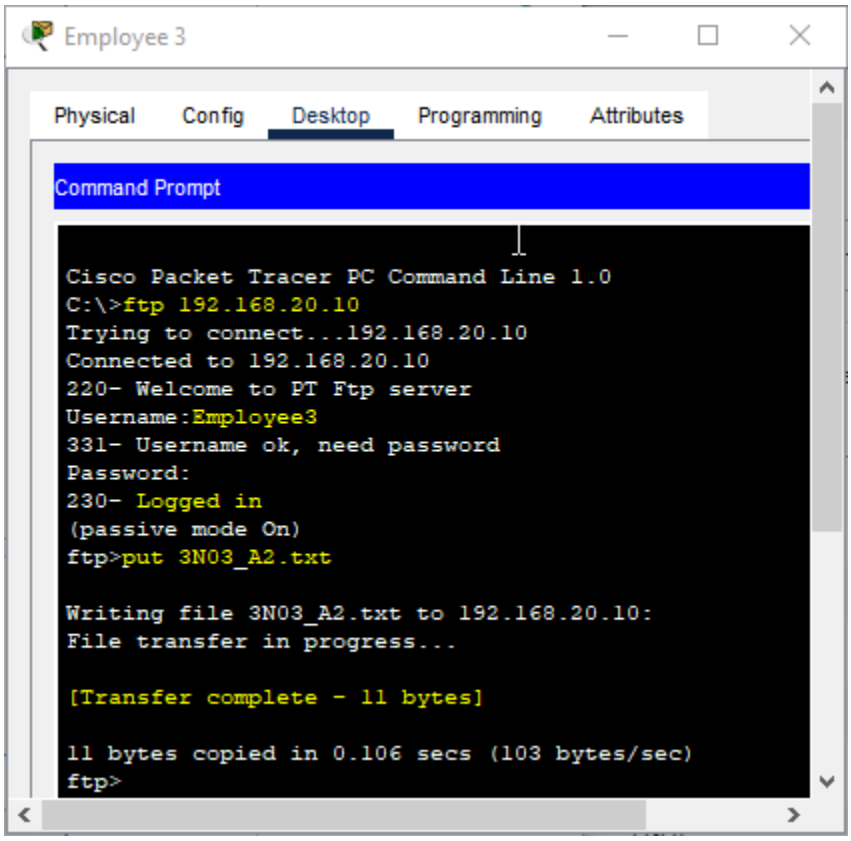


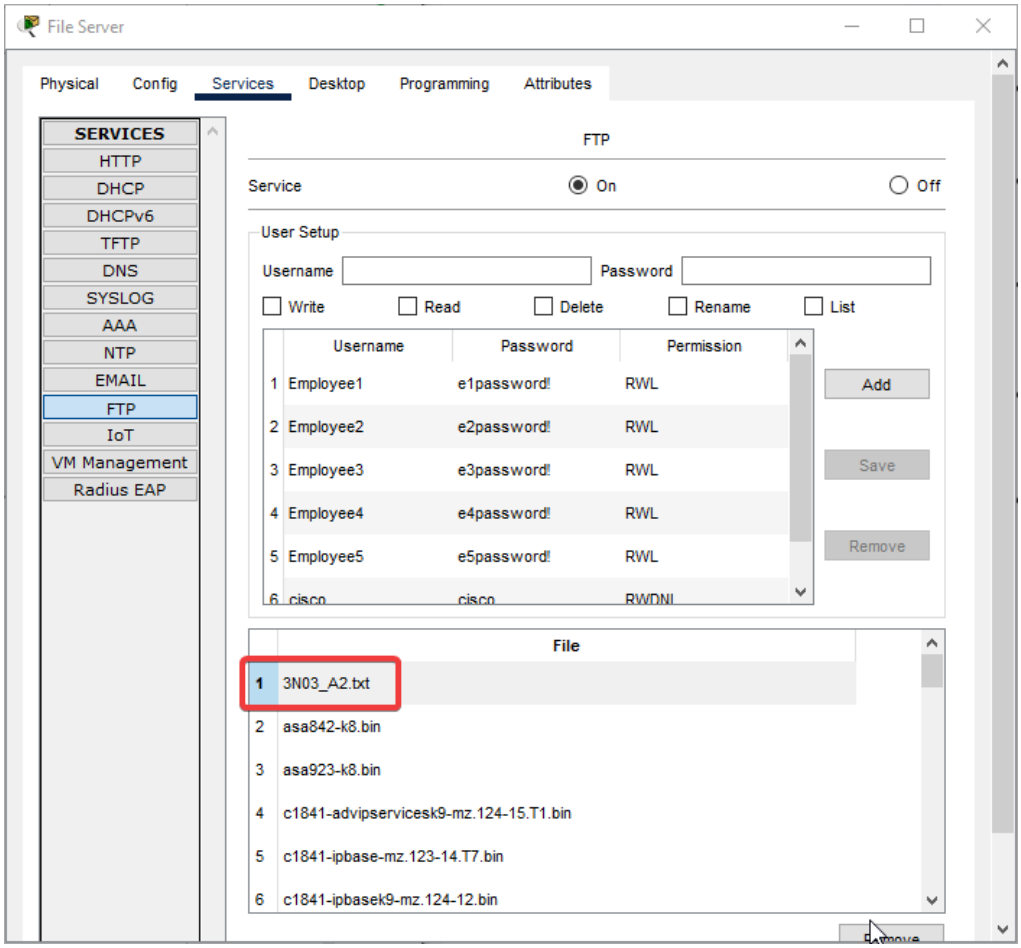
Task 1.2 - Device Configurations
Screenshot 2:



Task 1.3 - Create and Send File

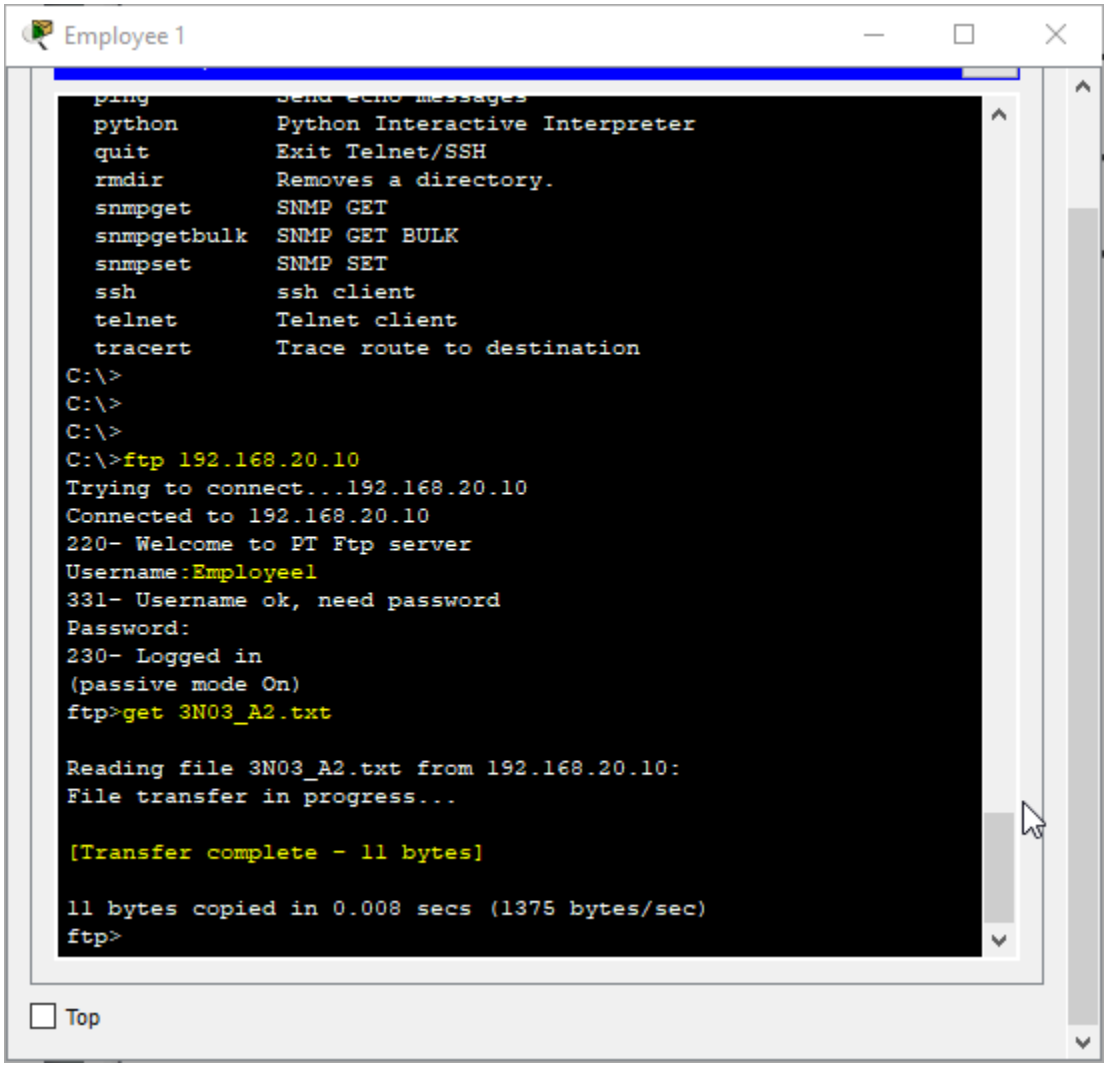
Screenshot 3:




File Server
Physical
Config
Services
Desktop
Programming
Attributes
SERVICES
HTTP
DHCP
DHCPv6
TFTP
DNS
SYSLOG
AAA
NTP
EMAIL
FTP
IoT
VM Management
Radius EAP
FTP
Service
On
Off
User Setup
Username
Password
Write
Read
Delete
Rename
List
Username
Password
Permission
1 Employee1 e1password! RWL
2 Employee2 e2password! RWL
3 Employee3 e3password! RWL
4 Employee4 e4password! RWL
5 Employee5 e5password! RWL
6 cisco cisco RWDNL
Add
Save
Remove
File
1 3N03_A2.txt
2 asa842-k8.bin
3 asa923-k8.bin
4 c1841-advipservicesk9-mz.124-15.T1.bin
5 c1841-ipbase-mz.123-14.T7.bin
6 c1841-ipbasek9-mz.124-12.bin

Task 1.4 - Get File by other host

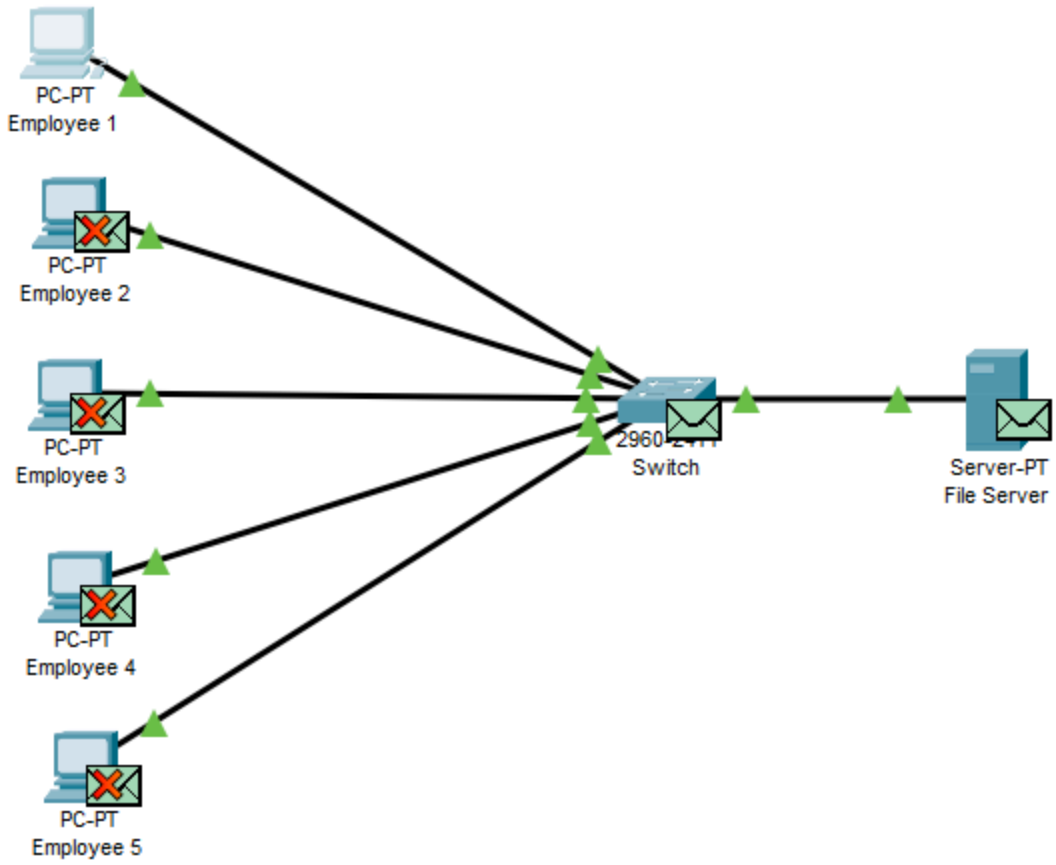
Screenshot 4:



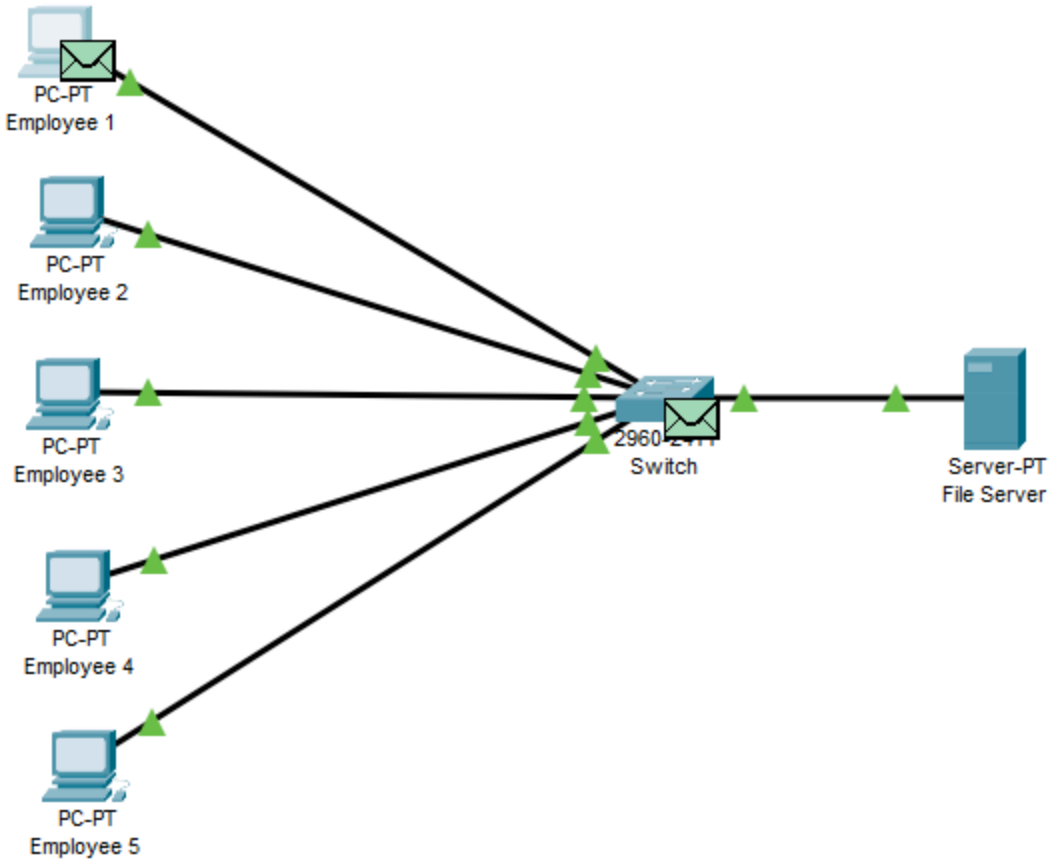
Task 1.5 - 3-way Handshake

Screenshot 5: Packet outgoing from Employee 1's Device

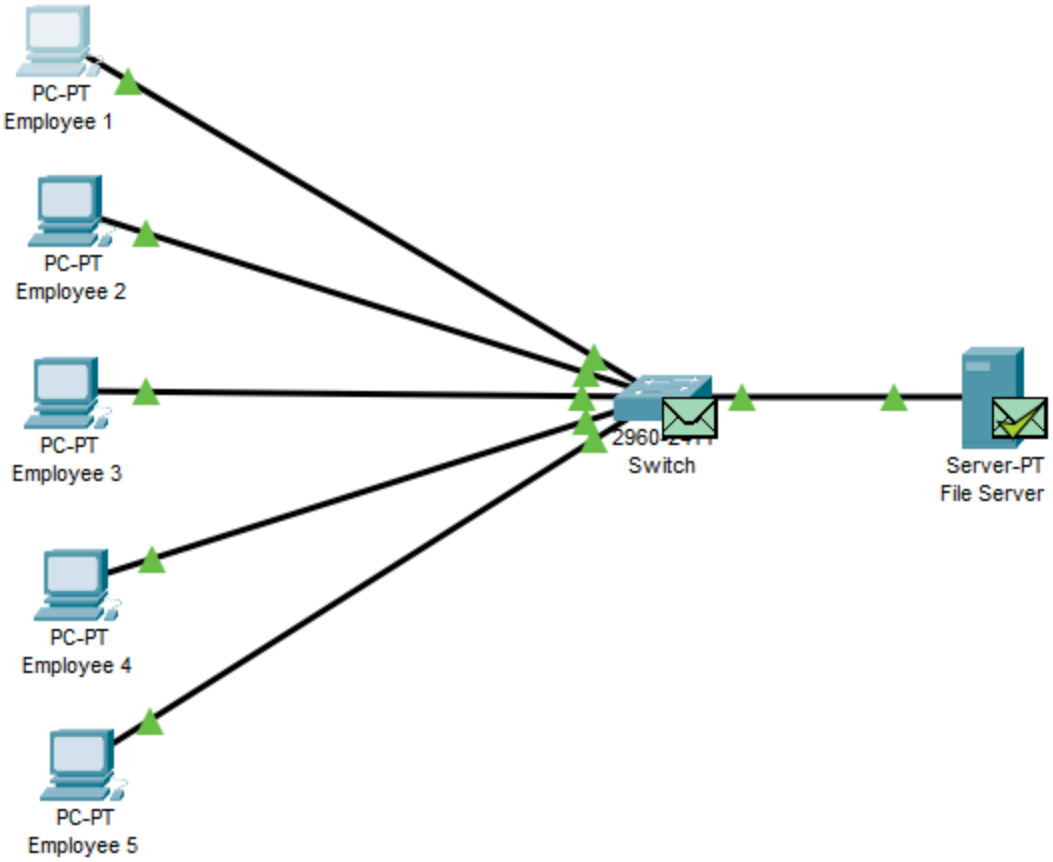
Employee 1 initiates connection with **SYN (synchronize)** request.



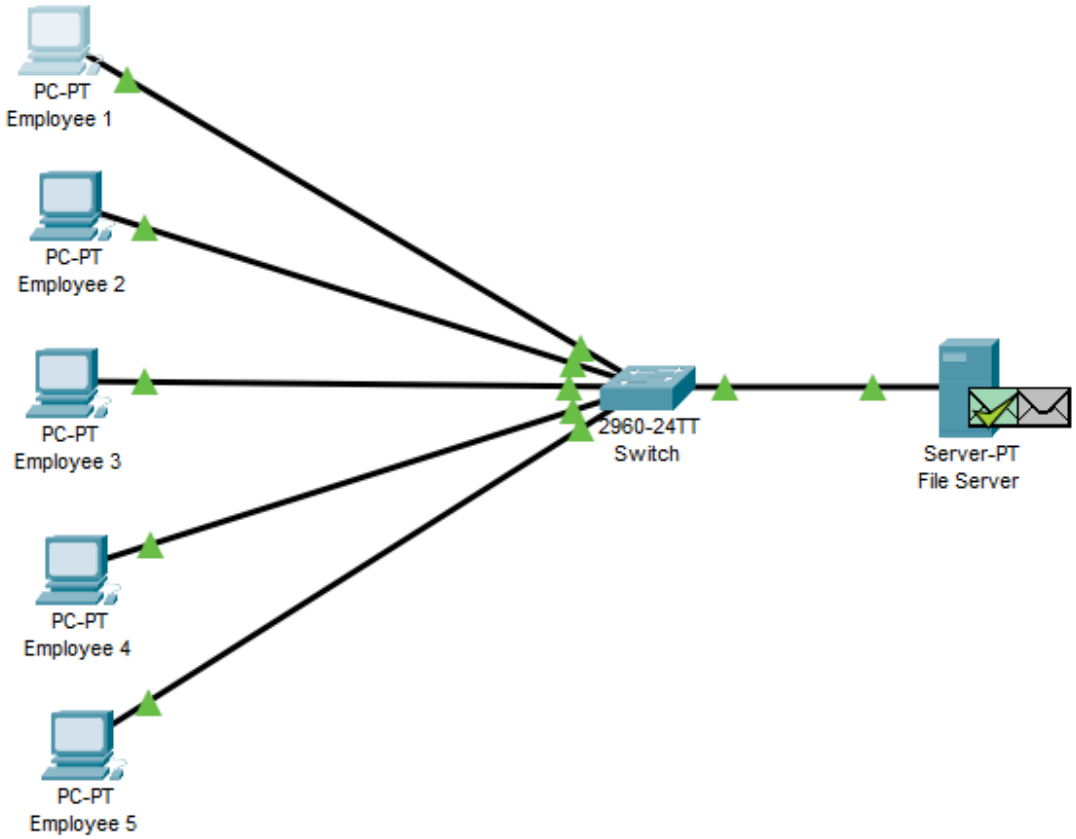
Screenshot 6: Outgoing **SYN-ACK** packet from the Server.
Server responds with **SYN-ACK** (acknowledgment + sync).



Screenshot 7: Packet outgoing from the Employee 1's Device
Employee 1 confirms with **ACK**, completing the handshake.



Screenshot 8: Packet incoming to the Server (Screenshot 8)
Server receives **ACK**, confirming the session is established.



Task 1.6 - Reflect

- TCP doesn't use a **2-way handshake** because it can't confirm if the receiver is ready.
- A **3-way handshake** ensures both sender and receiver are synchronized before data transfer.
- The first SYN ensures the sender initiates, the SYN-ACK ensures the receiver is available, and the final ACK confirms both sides are ready.
- This prevents packet loss, retransmissions, and ensures **reliable communication.**

Question 2.1

The **ACK number acknowledges the next expected byte** from the sender.

Formula for ACK Number: Acknowledgment Number = Sequence Number + Segment Size

ACK Number = 2512+356 = 2868

Question 2.2

The sequence number of the next segment from the sender would be 2868. This is because the next sequence number is equal to the acknowledgment number of the current segment.

Question 2.3
Need to verify.

Question 2.4
If the **checksum** was wrong at the receiver, the packet would be **discarded**. UDP uses checksums for error **detection**, not error correction. The receiver would not attempt to recover or request retransmission of the corrupted packet. Since UDP is a connectionless protocol that prioritizes speed over reliability, the sender remains unaware of the loss unless an application-level mechanism detects it.