:



Digital Security

MOD003264

Faculty: Science and Technology

Department: Computing and Technology

Student ID: 2179610

Word Count: 1948

Academic Year: 2023/24

Trimester: 2

**Section A**

In addressing the challenge faced by CommitCon Insurance Inc. of enhancing workflow efficiency and data management through networked communication, we've crafted a tailored solution to pioneer the company's foray into computer networking. This solution encompasses the design and implementation of a simplified network protocol suite, leveraging state of the art experimental technologies of the era. Central to this design is the establishment of a logical addressing scheme and packet based communication protocol, which together ensure reliable and secure data exchange between the company's newly acquired Commodore PET computers and a customer facing terminal. By integrating unique addressing for each device and defining a clear packet structure for communication, we've laid the groundwork for seamless file sharing and collaborative efforts across the network.

**Addressing Protocol**

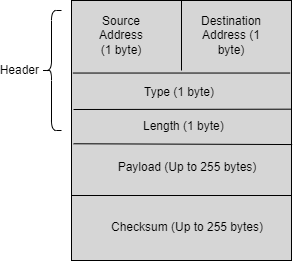
1. **Address Format -** Use a 1byte address format, allowing for up to 256 devices on the network. Given the small scale, this should suffice and offer room for network expansion.
2. **Address Assignment -**
   * **Customer Terminal -** **0x01**
   * **Alice's Computer -** **0x02**
   * **Bob's Computer -** **0x03**
   * **Dave's Computer -** **0x04**

The high speed switch doesn't require an address for this implementation but will facilitate the forwarding of packets based on destination addresses.

1. **Conflict Prevention -** Implement a simple protocol where a new device broadcasts a packet to every address upon joining the network. If it receives a response from any device, it knows the address is taken and selects another.

**Packet Structure**

1. **Header** 
   * **Source Address (1 byte) -** Identifies the sender.
   * **Destination Address (1 byte) -** Identifies the receiver.
   * **Type (1 byte) -** Indicates the packet type (e.g., data, acknowledgment).
   * **Length (1 byte) -** Indicates the length of the payload (max 255 bytes).
2. **Payload -** Contains the actual data being sent. Size can vary up to 255 bytes to accommodate different data needs.
3. **Checksum (1 byte) -** Provides a simple error checking mechanism. Can be a sum of all bytes in the packet modulo 255.



**Communication Regime**

* **Sending Packets -** When a device wants to send data, it constructs a packet with the appropriate header (including source and destination addresses), payload, and checksum. It then sends this packet to the switch.
* **Routing/Forwarding -** The switch reads the destination address and forwards the packet to the corresponding device. There's no need for routing tables due to the direct connections and static addresses (Shinder, Shinder and Todd, 2003).
* **Receiving Packets -** The receiving device checks the destination address, verifies the checksum, and if both are correct, processes the payload. If there's a checksum error, the packet is discarded (for simplicity, no retransmission request).

**Example - Sending a Message from Alice to Bob**

1. Alice constructs the packet -
   * Source Address - **0x02** (Alice)
   * Destination Address - **0x03** (Bob)
   * Type - **0x01** (data packet)
   * Length - **0x0A** (for example, 10 bytes of data)
   * Payload - Hello Bob!
   * Checksum - Calculated over the header and payload.
2. Alice sends the packet to the switch, which forwards it to Bob based on the destination address.
3. Bob's computer receives the packet, checks the checksum, sees it matches, and processes the payload**.**

This design provides a simple yet functional network communication framework with basic scalability and conflict prevention. It's tailored for the early stage, small scale network of CommitCon Insurance Inc., allowing for future expansions and modifications as needed.

**Section B**

To facilitate efficient communication within networks, it's crucial for devices to be able to locate each other effectively. This process is significantly streamlined by employing a combination of MAC addresses for physical device identification and logical addresses for network layer identification (Faircloth, 2014). Herein, we will design a Simplified Address Resolution Protocol (SARP) to bridge the gap between these two addressing schemes. Our focus will be on outlining the packet structure, the communication regime of the protocol, and providing a concrete example to illustrate how hosts can locate and communicate with one another within this framework.

**Simplified Address Resolution Protocol (ARP)**

**Assumptions**

* Each host on the network has a unique MAC address.
* The network uses a custom logical addressing scheme (similar to IP addresses) which is simpler and perhaps uses shorter addresses for easier handling in small networks.

**Packet Structure**

Each ARP request and reply packet will consist of the following fields:

1. Type (1 byte) - Indicates whether the packet is a **request** (0x01) or a **reply** (0x02).

2. Sender Logical Address (4 bytes) - The **logical address** of the sender.

3. Sender MAC Address (6 bytes) - The **MAC address** of the sender.

4. Target Logical Address (4 bytes) - The **logical address** of the intended recipient.

5. Target MAC Address (6 bytes, only in reply) - The **MAC address** of the recipient. This field is empty in request packets and filled in reply packets.

Total size for a request packet is 15 bytes. For a reply packet is 21 bytes.

**Communication Regime**

1. **Broadcast Request** - When a host needs to communicate with another host whose MAC address it does not know, it sends a SARP request broadcast. This request includes the sender's logical and MAC addresses and the target's logical address.

2. **Unicast Reply** - The host with the matching logical address receives the request and sends a SARP reply directly (unicast) to the requester. This reply contains both the sender's and targets logical and MAC addresses.

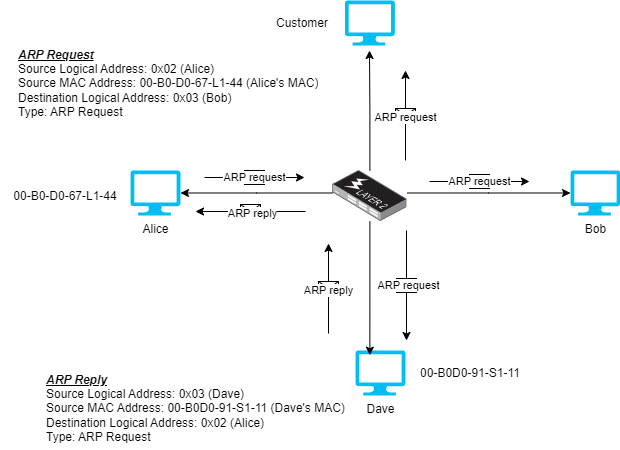
3. **Cache** - Each host maintains a small cache of recently resolved logical to MAC address mappings to reduce network traffic.

**Protocol Operation Example**

Let’s consider the below scenario in a network:

Alice - Logical Address = 0x02, MAC Address = 00B0D067L144

Dave - Logical Address = 0x03, MAC Address = 00B0D091S111



In this scenario, Alice wants to locate Bob.

Alice wants to send a packet to Dave but doesn't know Dave's MAC address

1. **Alice Broadcasts ARP Request -**
   * Source Logical Address - **0x02** (Alice)
   * Source MAC Address - **00B0D067L144** (Alice's MAC)
   * Destination Logical Address - **0x03** (Bob)
   * Type - ARP Request
2. **Bob Sends ARP Reply -**
   * Source Logical Address - **0x03** (Bob)
   * Source MAC Address - **00B0D091S111** (Bob's MAC)
   * Destination Logical Address - **0x02** (Alice)
   * Type - ARP Reply
   * Payload - **00B0D091S111** (Bob's MAC)

Bob receives the SARP Request and recognizes its logical address **00B0D067L144** in the Target Logical Address field

1. **Alice Receives and Updates ARP Table -**
   * Alice receives the SARP Reply, updates her ARP table, mapping Bob's logical address (**0x03**) to his MAC address (**00B0D091S111**), and can now send packets directly to Bob using its MAC address.

**Cache Example**

After the exchange, Alice’s cache contains -

Logical Address - **0x03**

MAC Address - 11 -22 -33 -44 -55 -66

This cache entry allows Alice to send future packets to Bob without needing to perform another SARP request, thus reducing network traffic and speeding up communications.

**Section C**

To establish secure remote access within the CommitCon Insurance Inc. network, we'll design a simplified version of the Secure Shell (SSH) protocol, focusing on initiating an encrypted channel between devices. This simplified protocol will incorporate both symmetric and asymmetric cryptography, along with hash functions, to ensure security through authentication, authorization, and confidentiality.

**Simplified Secure Access Protocol (SAP)**

**Key Components**

1. **Asymmetric Keys** - Each device generates a pair of keys (public and private). Public keys are shared, while private keys remain confidential.

2. **Symmetric Session Key** - A temporary key used for encrypting the communication during a session.

3. **Hash Function** - Used for verifying the integrity of the messages.

**Packet Structure**

*Header*

* Source Address (**1 byte**)
* Destination Address (**1 byte**)
* Type (**1 byte**) - Indicates the stage of the handshake or data transmission.

*Payload* - It will vary according to the type of packet but potentially includes public keys, symmetric session keys (encrypted under the public key), or data (encrypted under the symmetric key).

*Hash* - A hash of the payload to verify integrity.

**Handshake Procedure**

1. *Key Exchange*

The client sends a packet with its public key to the server.

The server responds with its public key, encrypted with the client's public key.

2. *Session Key Establishment*

The client generates a symmetric session key, encrypts it with the server's public key, and sends it to the server.

The server decrypts the session key using its private key.

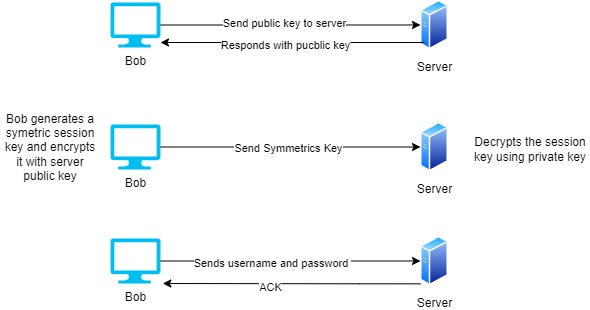
3. *Authentication*

The client sends its username and password, both encrypted with the session key, to the server.

The server validates the credentials and responds with an acknowledgment (ACK), encrypted with the session key.

**Example Establishing a Connection**

Bob wants to securely authenticate to the server, the handshake process will be as depicted



1. **Bob** Sends Public Key to Server

* Type - Key Exchange
* Payload - Bob's public key
* Hash - Hash of the payload

2. **Server** Responds with Public Key -

* Type - Key Exchange
* Payload - Server's public key, encrypted with Bob's public key
* Hash - Hash of the payload

3. **Bob** Sends Session Key -

* Type - Session Key
* Payload - Symmetric session key, encrypted with the server's public key
* Hash - Hash of the payload

4. **Authentication**

* Type - Authentication
* Payload - "Bob" and "123456", encrypted with the symmetric session key
* Hash - Hash of the payload

5. **Server** Sends ACK -

* Type - ACK
* Payload - Encrypted with the session key
* Hash - Hash of the payload

In essence, this is a simplistic protocol that stipulates securely encrypted communication between devices in a network using both asymmetric and symmetric encryption, accompanied by hash functions for support.

**Section D**

In the same way that TCP pairs with TLS, or alternatively operates in a VPN, we'll have to be able to come up with a lean communication protocol that, nonetheless, is able to guarantee security in the exchange of messages. The protocol will use symmetric and asymmetric cryptography along with the password hash functions to ensure the messages sent across the network are secure.

Simplified Encryption Communication Protocol (ECP)

**Key Components**

**Asymmetric key pair** - It is securely exchanged during the handshake and used for the exchange of another symmetric session key. Every participant has a public and private key.

**Symmetric Session Key** - Used for encrypting and decrypting the messages sent during the session.

**Hash Function** - Utilized to ensure the integrity of the messages and the handshake process.

**Packet Structure**

*Header*

* Source Address (**1 byte**)
* Destination Address (**1 byte**)
* Type (**1 byte**) - Distinguishes the stage of communication or handshake.

*Payload* - Depending on the type of packet received, the payload will be either the public keys generated in the handshake or the symmetric session key encrypted with the public key.

*Hash* - A hash of the payload to ensure integrity.

**Handshake Procedure**

1. Public Key Exchange

* The initiator sends a packet with their public key to the recipient.
* The recipient responds with their public key.

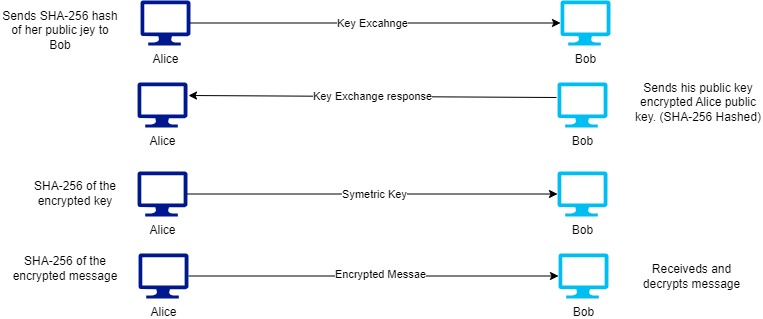
2. Symmetric Session Key Exchange

* The initiator generates a symmetric session key, encrypts it with the recipient's public key, and sends it over.
* The recipient decrypts the session key using their private key.

3. Secure Message Transmission - Messages are encrypted with the symmetric session key and sent over the network with a hash to verify integrity.

**Secure Message Exchange example**

Consider a where Alice wishes to send an encrypted message to Bob. The message is Alice's PASSWORD.



1. Alice Initiates Contact with Bob:

* Packet Type - Key Exchange
* Payload - Alice's public key
* Hash - SHA256 hash of Alice's public key

2. Bob Responds:

* Packet Type - Key Exchange Response
* Payload - Bob's public key, encrypted with Alice's public key for confidentiality
* Hash - SHA256 hash of the payload

3. Alice Sends Symmetric Key:

* Packet Type - Session Key Exchange
* Payload - Symmetric session key, encrypted with Bob's public key
* Hash - SHA256 hash of the encrypted key

4. Alice Sends Encrypted Message:

* Packet Type - Secure Message
* Payload - "PASSWORD\_Alice", encrypted with the symmetric session key
* Hash - SHA256 hash of the encrypted message

5. Bob Receives and Decrypts the Message:

* Bob uses the symmetric session key to decrypt the message.
* Bob verifies the message integrity using the provided hash.

This simple protocol would ensure that Alice and Bob would be able to exchange messages with confidentiality, integrity, and authentication. The simplified basic concept of how TLS and VPN works into basic form is suitable in understanding secure communication with network fundamentals.

Reference

Shinder, D.L., Shinder, T.W. and Todd, C. (2003) 'MCSA/MCSE 70-291: Reviewing TCP/IP Basics,' in *Elsevier eBooks*, pp. 1–96. https://doi.org/10.1016/b978-193183692-0/50007-x.

Faircloth, J. (2014) 'Networks,' in *Elsevier eBooks*, pp. 27–79. https://doi.org/10.1016/b978-0-12-407773-7.00002-8.