TCP Data Transfer - Complete Study Notes

Introduction

After TCP connection establishment through 3-way handshaking, the next phase is **data transfer**. This process involves how TCP manages the actual transmission of data between client and server.

Prerequisites - Resource Reservation

What happens during connection establishment:

- **Both client and server reserve their resources**
- Resources include:
  - **Buffers**: Memory spaces allocated for data storage
  - **Bandwidth**: Network capacity allocation
  - **Window size negotiation**: Agreement on how much data can be sent at once

Example Setup:

- **Window Size**: 10 bytes (for simplicity)
  - Client agrees to send 10 bytes at a time
  - Server agrees to send 10 bytes at a time
  - *Note: In reality, window size can be up to 16 bits (2^16 segments)*

Key Characteristics of TCP Data Transfer

1. Full Duplex Mode

- **Definition**: Data can flow in both directions simultaneously
- **Client to Server**: Client can send data to server
- **Server to Client**: Server can send data to client
- **Simultaneous Communication**: Both can send/receive at the same time

Data Transfer Process - Detailed Example

Initial Data Transfer Scenario

Step 1: Client Sends First Data

Client sends:

- **Data**: Bytes 21-30 (10 bytes total)
- **Sequence Number**: 21 (starting sequence number)

- **Acknowledgment Flag**: 1 (set)
- **Acknowledgment Number**: 71
  - *Assumption: Server previously sent data from 60-70*
  - *Client is now expecting data starting from 71*

Step 2: Server Receives and Responds

Server processing:

1. **Receives data**: Bytes 21-30 from client
2. **Sends to application layer**: Passes data up the protocol stack
3. **Prepares response**: Server now sends its own data

Server sends:

- **Data**: Bytes 71-80 (10 bytes total)
- **Sequence Number**: 71 (as expected by client)
- **Acknowledgment Flag**: 1 (set)
- **Acknowledgment Number**: 31
  - *Server received bytes 21-30, so expects next data from 31*

Step 3: Continued Data Transfer

Next exchanges:

- Client sends: Bytes 31-40
- Server sends: Bytes 81-90
- Pattern continues with proper sequencing

Acknowledgment Methods

1. Piggybacking

Definition: Sending data and acknowledgment together in the same packet

Advantages:

- **Reduces network traffic**: Fewer packets needed
- **Efficient bandwidth usage**: Single packet carries both data and ACK
- **Prevents network congestion**: Less packet overhead

Example from scenario:

- Client sends data (21-30) + ACK (71) together

- Server sends data (71-80) + ACK (31) together

2. Pure Acknowledgment

Definition: Sending acknowledgment alone without any data

When to use Pure Acknowledgment:

Scenario: No data available to send immediately

Problem without Pure ACK:

- Sender waits indefinitely for acknowledgment
- **Timeout concerns**: Sender doesn't know if data was received
- **Real-world analogy**:
  - Online payment (Paytm, Amazon)
  - After entering details and OTP
  - Waiting for transaction confirmation
  - Anxiety while page shows "buffering"
  - Relief when "Transaction Completed" message appears

Pure Acknowledgment Structure:

Client sends Pure ACK:

- **Sequence Number**: 31 (not consumed, just informational)
- **Acknowledgment Flag**: 1 (set)
- **Acknowledgment Number**: 81 (expecting server's next data)
- **Data**: None (empty payload)

Key Point: Sequence number is attached but not used/consumed

After Pure ACK:

When client later gets data to send:

- Uses the same sequence number (31) that was in pure ACK
- Sends actual data: Bytes 31-40
- Can include acknowledgment if needed

Important Exam Points

1. Terminology

- **Full Duplex**: Bidirectional simultaneous communication

- **Window Size**: Amount of data that can be sent before acknowledgment
- **Sequence Numbers**: Track data bytes for ordering
- **Piggybacking**: Data + ACK in same packet
- **Pure Acknowledgment**: ACK only, no data

2. Advantages of Piggybacking

- Reduces network traffic
- Prevents unnecessary packet multiplication
- Efficient resource utilization
- Lower network congestion

3. When Pure Acknowledgment is Used

- No immediate data to send
- Prevent sender timeout
- Maintain connection reliability
- Inform sender about successful data receipt

Additional TCP Features (Preview)

Topics for next lessons:

- **Push Flag**: Force immediate data delivery
- **Urgent Pointer/Flag**: Handle priority data
- **Connection Termination**: How to properly close TCP connections

Summary Flow

1. **Connection Establishment** → Resource reservation, window size negotiation
2. **Data Transfer Phase** → Full duplex communication with proper sequencing
3. **Acknowledgment Management** → Piggybacking or Pure ACK based on data availability
4. **Connection Termination** → Proper closure of established connection

Key Takeaways for Exams

- TCP data transfer is **full duplex**
- **Piggybacking** reduces network overhead
- **Pure acknowledgments** prevent timeout issues
- **Sequence numbers** ensure proper data ordering
- **Window size** controls flow of data

- Both methods (piggybacking and pure ACK) maintain connection reliability