EECE 4830-5830 Network Design, Dr. Vinod Vokkarane

Programming Project Phase 4 (Part 1):
Implement Go-Back-N protocol over an unreliable UDP channel

Project description: Project description: The TCP/IP stack has five layers, namely application, transport, network, link, and physical. In Phase 1, each group implemented the standard user datagram protocol (UDP) sockets. The intention was to transfer a file (say JPEG) between a UDP client process and a UDP server process. In Phase 2, Phase 3, and Phase 4, we provided reliable data transfer service over the unreliable UDP connection. In Phase 4, you will have to enhance your code to handle pipelined data transfer. The sender has to maintain a buffer as shown below.

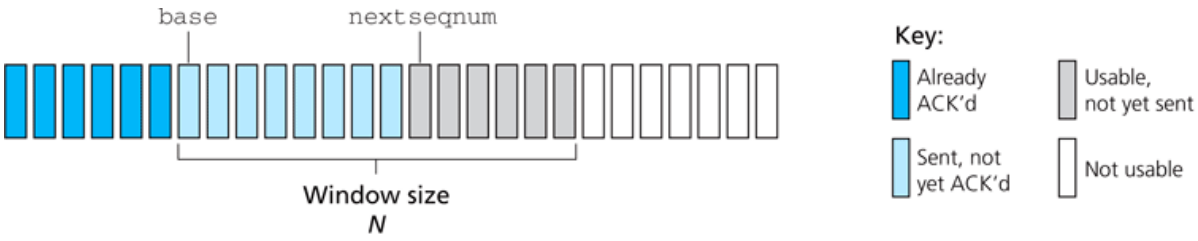

Figure 3.19 ♦ Sender's view of sequence numbers in Go-Back-N

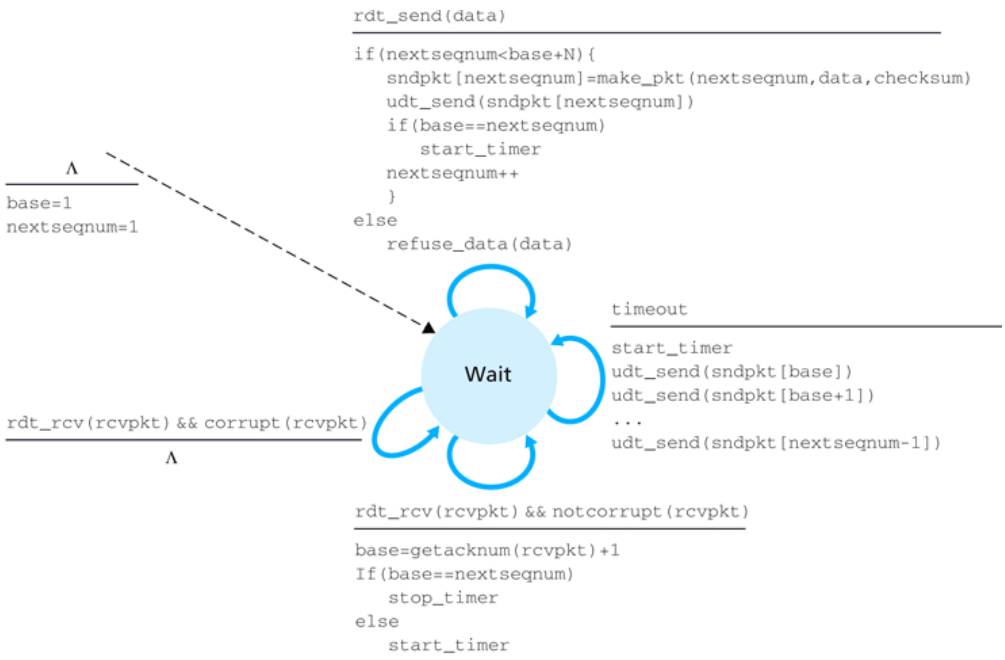

Figure 3.20 ♦ Extended FSM description of GBN sender

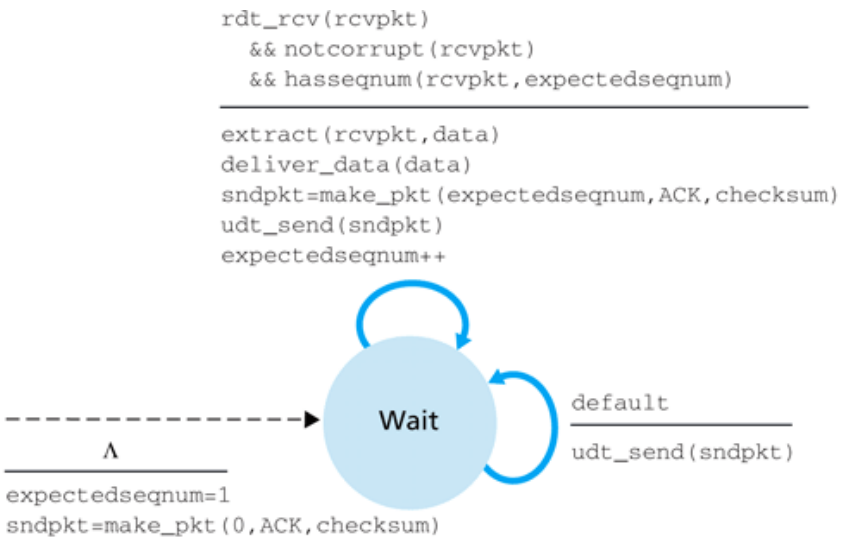

Figure 3.21 ♦ Extended FSM description of GBN receiver

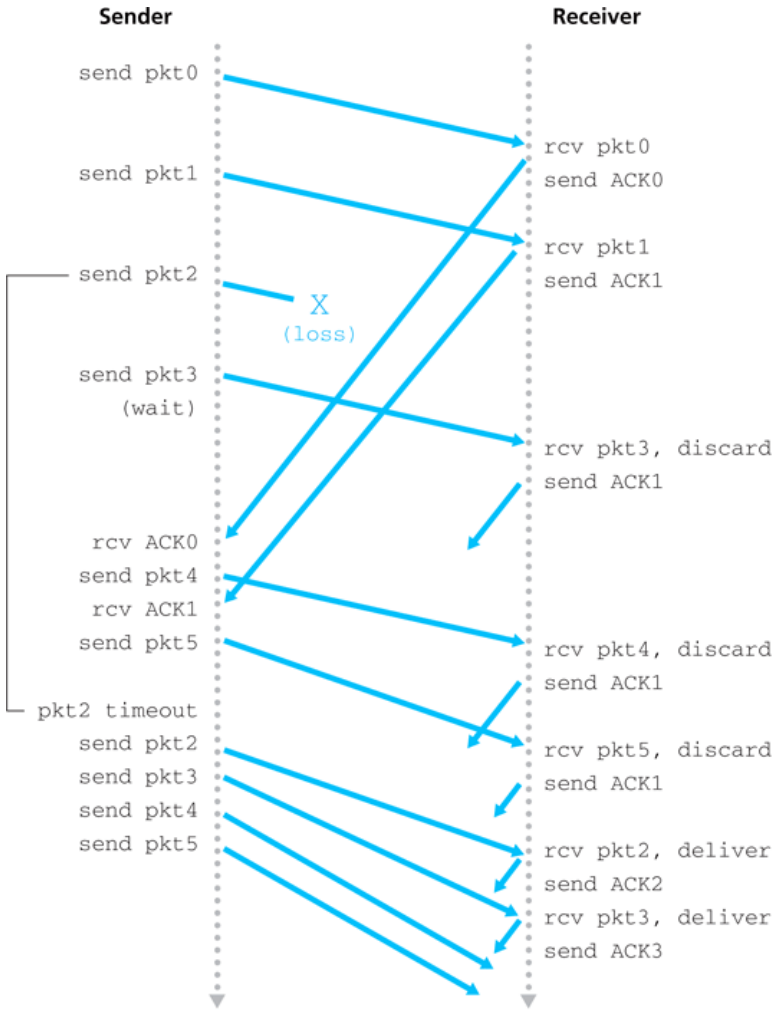

Figure 3.22 ♦ Go-Back-N in operation

The following are the basic implementation steps:

1. Pick a transfer file - BMP image file (recommended), easier identifying loss of packets in an image file (lost pixels).

2. *Make_packet* function - parses the image file and breaks it down to several packets (set a fixed packet size, say 1024 bytes).

3. Set a fixed window size (N, say 10).

4. Use the UDP sockets (Phase 1) to send and receive GBN packets. UDP sockets are unreliable, so they simulate the unreliable Internet IP (network) layer.

5. Implement: Sequence Numbers (to identify duplicates), Checksum (implement your own, similar to UDP), ACKs, and a countdown timer (to handle packet loss).

6. The GBN receiver should assemble packets in order and deliver the entire transfer file to the application.

In your implementation, use binary variables for representing bits (instead of strings).

In this phase, you will need to implement the following data transfer scenarios

Option 1 - No loss/bit-errors.

Option 2 - ACK packet bit-error: you need to intentionally change the data bits of the received ACK packet at the sender (some error %) and implement suitable recovery mechanism.

Option 3 - Data packet bit-error: you need to intentionally change the data bits of the received DATA packet at the receiver (some error %) and implement suitable recovery mechanism.

Option 4 - ACK packet loss: you need to intentionally drop the received ACK packet at the sender (some loss %) and implement suitable recovery mechanism.

Option 5 - Data packet loss: you need to intentionally drop the received DATA packet at the receiver (some loss %) and implement suitable recovery mechanism.

IMPORTANT: You should be able to demo Phase 4 with and without loss recovery.

Performance Graphs

Note:

- Use at least a 500KB file for transmission.
- Don't set your timeout to be too high (30-50ms should be enough).
- Each point on the plot can be an average of multiple runs at a given loss/error rate.
- For accuracy remove print commands and debug statements to the output console statements while measuring the completion time for the chart.

Plot charts for completion time for all your 5 data transfer scenarios for transferring the same file at 0% loss/error to 70% loss/error in increments of 5%.

Chart 1: Phase 4 performance

- X-axis: Intentional loss probability (0% - 70% range)
- Y-axis: File Transfer Completion Time

Chart 2: **optimal timeout value** - Phase 4 performance with a fixed loss probability (20% data loss/error; ACK loss/error)

- X-axis: Retransmission Timeout value (10ms – 100ms range)
- Y-axis: File Transfer Completion Time

Chart 3: **optimal window size -** Phase 4 performance with a fixed 20% loss probability

- X-axis: Window Size (1, 2, 5, 10, 20, 30, 40, and 50)
- Y-axis: File Transfer Completion Time

Chart 4: Performance comparison of the different phases at a fixed loss probability (say 20%) or a fixed bit-error probability (say 20%) using the same transfer image.

- X-axis: Phase 2, Phase 3, Phase 4, (Selective Repeat and UDP - optional)
- Y-axis: File Transfer Completion Time

Observe and conclude if there is an **optimal window size** and an **optimal timeout value** for your project.

Programming Project Phase 4 (Part 2 – Extra Credit):

1. **Selective Repeat (SR) Comparison of Window-Based Protocols (50%)**
    1. **GBN** – A window-based protocol that retransmits only the lost or erroneous packets instead of the entire window.
    2. **Stop-and-Wait (SW)** – A simple protocol where the sender transmits one packet at a time and waits for an acknowledgment before sending the next.
2. **Implement a multi-threaded version of all window protocols to optimize sender and receiver operations (25%)**
    1. Use separate threads for sending, receiving, and managing the retransmission timer.
    2. Ensure thread synchronization to avoid race conditions in shared resources
3. **Implement an applet/GUI to show real-time data transfer with sender and receiver FSM (25%)**

Expectations: In this phase of the project, you will learn about the basic principles used to provide pipelined reliable data transfer over a data channel with bit errors and packet losses.

Programming language: C, C++, Python or Java (your choice).

Deliverables:

1. **ReadMe File**: Name of the team members, list the names of files submitted and their purpose, and explain steps required to set up and execute your program. Also, how to check for the different scenarios (1 - no loss/error; 2 - ACK bit error; 3 - Data packet error, 4 – ACK packet loss, and 5 – Data packet loss).
2. **Design Documents**: This document should contain 4 sections: Introduction, flow chart, code description, and snapshot of results.
    1. Introduction: Should provide the TA a clear picture of your project design.
    2. Flowchart: Flowchart should demonstrate the author's logic. This section is optional. Ref: https://creately.com/blog/diagrams/flowchart-guide-flowchart-tutorial/.
    3. Code Description: Explain the logic of your code based on functional blocks. Please do not just copy and paste commented code into this document.
    4. Execution Screenshots: Include the screenshots of your executions of the 5 difference scenarios (Option 1-5) in this section.
    5. Performance Plots: Plot charts for completion time for all the 5 data transfer options for the same file starting from 0% loss/error to 60% loss/error in increments of 5%.
3. **Sender and Receiver source files**: well commented source code; mention ALL references for reuse of source code (if any).
4. **Deliverables for Extra Credit:**
    1. **Implementation of Additional Protocols**: Code for at least one additional window-based protocol alongside Go-Back-N.

2. **Performance Analysis**: Comparison of Stop-and-Wait, Go-Back-N, and Selective Repeat in different network conditions (0%–70% loss/error).
3. **Multi-Threaded Implementation**: Provide a multi-threaded sender and receiver version of Go-Back-N and compare performance against a single-threaded implementation.
4. **All Protocols Comparison**: Measure and compare the file transfer completion time between UDP and Go-Back-N/Selective Repeat.
5. **Discussion & Observations**: Include a report discussing which protocol performs best under various conditions and the benefits of multi-threading.
6. **Performance Graphs**: Plot results showing completion time variations across different loss rates, window sizes, and multi-threaded vs. single-threaded implementations.

5. **Individual Contribution (contribution.txt/doc):**
    1. Bulleted list of tasks implemented by each team member and
    2. Teammate rating between 1 (not good) – 5 (excellent) on
        1. Project time commitment
        2. Design contribution
        3. Coding contribution
        4. Debugging contribution, and
        5. Report preparation (confidential submission for each member).

One team member can submit all your documents (except extra credit items) in a single compressed file with the name: **TeamNumber_Phase4.zip.**

References:

1. Socket Programming
2. Python: Socket Programming (Textbook Sec 2.7)
    a. JAVA: Socket Programming (PDF attached with Phase 1 assignment)
    b. Socket Programming by InfoWorld
    c. C/C++:

    d. Linux Gazette's Socket Programming
    e. Beej's Guide
3. UDP: RFC768
4. Principles of Reliable Data Transfer, Section 3.4.1, Page 206 -207, Kurose/Ross (7th Ed).