VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2023-24
School Of Computer Science and Engineering (SCOPE)

Digital Assignment - II

Computer Network Lab

Apurva Mishra, 22BCE2791

11 September 2024

1. Exercise

Problem 1.1.
Implement the Stop and Wait Protocol ARQ for noisy channel Sender has to send frames: 1234567

Codes

server.c

```c
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

typedef struct packet {
  char data[1024];
} Packet;

typedef struct frame {
  int frame_kind; // ACK:0, SEQ:1 FIN:2
  int sq_no;
  int ack;
  Packet packet;
} Frame;

int main() {
  int port = 8080;
  int sockfd;
  struct sockaddr_in serverAddr, newAddr;
  char buffer[1024];
  socklen_t addr_size;

  bool frame_received[10] = {false};

  int frame_id = 0;
  Frame frame_recv;
  Frame frame_send;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);

  memset(&serverAddr, '\0', sizeof(serverAddr));
  serverAddr.sin_family = AF_INET;
```

```
  serverAddr.sin_port = htons(port);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

  bind(sockfd, (struct sockaddr *)&serverAddr, sizeof(serverAddr));
  addr_size = sizeof(newAddr);

  // infinite loop
  while (1) {
    int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(Frame), 0,
                               (struct sockaddr *)&newAddr, &addr_size);

    if (frame_received[atoi(frame_recv.packet.data)] == true) {
      printf("[+]Duplicate frame discarded: %s\n", frame_recv.packet.data);
      continue;
    }
    frame_received[atoi(frame_recv.packet.data)] = true;

    // Send ack for received frames
    if (f_recv_size > 0 && frame_recv.frame_kind == 1 &&
        frame_recv.sq_no == frame_id) {
      printf("[+]Frame Received: %s\n", frame_recv.packet.data);

      frame_send.sq_no = 0;
      frame_send.frame_kind = 0;
      frame_send.ack = frame_recv.sq_no + 1;
      sendto(sockfd, &frame_send, sizeof(frame_send), 0,
             (struct sockaddr *)&newAddr, addr_size);
      printf("[+]Ack Send\n");
    } else {
      printf("[+]Frame Not Received\n");
    }
    frame_id++;
  }

  close(sockfd);
  return 0;
}
```

client.c

```
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

typedef struct packet {
  char data[1024];
```

```
} Packet;

typedef struct frame {
  int frame_kind; // ACK:0, SEQ:1 FIN:2
  int sq_no;
  int ack;
  Packet packet;
} Frame;

int main() {

  int port = 8080;
  int sockfd;
  struct sockaddr_in serverAddr;
  char buffer[1024];
  socklen_t addr_size;

  int frame_id = 0;
  Frame frame_send;
  Frame frame_recv;
  int ack_recv = 1;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);

  // Create server address
  memset(&serverAddr, '\0', sizeof(serverAddr));
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_port = htons(port);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

  while (1) {

    if (ack_recv == 1) {
      frame_send.sq_no = frame_id;
      frame_send.frame_kind = 1;
      frame_send.ack = 0;

      printf("Enter Data: ");
      scanf("%s", buffer);
      strcpy(frame_send.packet.data, buffer);
      sendto(sockfd, &frame_send, sizeof(Frame), 0,
             (struct sockaddr *)&serverAddr, sizeof(serverAddr));
      printf("[+]Frame Send\n");
    }
    int addr_size = sizeof(serverAddr);
    int f_recv_size = recvfrom(sockfd, &frame_recv, sizeof(frame_recv), 0,
                               (struct sockaddr *)&serverAddr, &addr_size);

    // Send ack/nack message
```

```c
    if (f_recv_size > 0 && frame_recv.sq_no == 0 &&
        frame_recv.ack == frame_id + 1) {
      printf("[+]Ack Received\n");
      ack_recv = 1;
    } else {
      printf("[-]Ack Not Received\n");
      ack_recv = 0;
    }

    frame_id++;
  }

  close(sockfd);
  return 0;
}
```

Output

```
ass2/q1/src via C v15.0.0-clang
❯ just sr
zig cc  ./src/server.c -std=c23 -o ./bin/server
./bin/server
[+]Frame Received: 1
[+]Ack Send
[+]Frame Received: 2
[+]Ack Send
[+]Frame Received: 3
[+]Ack Send
[+]Frame Received: 4
[+]Ack Send
[+]Frame Received: 5
[+]Ack Send
[+]Frame Received: 6
[+]Ack Send
[+]Frame Received: 7
[+]Ack Send
[+]Duplicate frame discarded: 7
^C

ass2/q1/src via C v15.0.0-clang took 13s
❯
```

```
ass2/q1/src via C v15.0.0-clang
❯ just cr
zig cc  ./src/client.c -std=c23 -o ./bin/client
./bin/client
Enter Data: 1
[+]Frame Send
[+]Ack Received
Enter Data: 2
[+]Frame Send
[+]Ack Received
Enter Data: 3
[+]Frame Send
[+]Ack Received
Enter Data: 4
[+]Frame Send
[+]Ack Received
Enter Data: 5
[+]Frame Send
[+]Ack Received
Enter Data: 6
[+]Frame Send
[+]Ack Received
Enter Data: 7
[+]Frame Send
[+]Ack Received
Enter Data: 7
[+]Frame Send
^C

ass2/q1/src via C v15.0.0-clang took 7s
❯
```

Problem 1.2.
Implement the Selective Repeat ARQ

Codes

server.c

```c
#include <arpa/inet.h>
#include <complex.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```c
#define PORT 8080
#define WINDOW_SIZE 4
#define BUFFER_SIZE 1024
#define TOTAL_FRAMES 8

int main() {
  int skip_frame_1 = true;
  int server_fd, new_socket;
  struct sockaddr_in address;
  int opt = 1;
  int addrlen = sizeof(address);
  char buffer[BUFFER_SIZE] = {0};

  if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    perror("socket failed");
    exit(EXIT_FAILURE);
  }

  address.sin_family = AF_INET;
  address.sin_addr.s_addr = INADDR_ANY;
  address.sin_port = htons(PORT);

  if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
    perror("bind failed");
    exit(EXIT_FAILURE);
  }

  if (listen(server_fd, 3) < 0) {
    perror("listen");
    exit(EXIT_FAILURE);
  }

  printf("Server listening on port %d\n", PORT);

  if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                           (socklen_t *)&addrlen)) < 0) {
    perror("accept");
    exit(EXIT_FAILURE);
  }

  int base = 0;
  int next_seq_num = 0;
  int expected_seq_num = 0;
  bool received_frames[TOTAL_FRAMES] = {false};
  bool negative_frames[TOTAL_FRAMES] = {false};

  int prev = -1;

  // infinite loop
```

```
  while (1) {
    // Receive data from client
    int valread = read(new_socket, buffer, BUFFER_SIZE);
    if (valread <= 0) {
      break;
    }

    // parse recieved input
    for (int i = 0; i < BUFFER_SIZE; i++) {
      if (buffer[i] == 0) {
        break;
      } else {
        int current_frame = (int)(buffer[i] - '0');
        if (current_frame == 0 || current_frame % 5 == 4) {
          char nack[BUFFER_SIZE];
          sprintf(nack, "ACK %d", current_frame);
          send(new_socket, nack, strlen(nack), 0);
          printf("Sent ACK for frame %d\n", current_frame);
        }
        if (prev < 0) {
          prev = current_frame;
        } else {
          prev = current_frame - 1;
        }

        if (current_frame == 1 && skip_frame_1) {
          skip_frame_1 = false;
          continue;
        }

        received_frames[current_frame] = true;

        // slide the window
        if (i == base) {
          base++;
          next_seq_num = base + WINDOW_SIZE;
        }
        printf("Received Frames: ");
        for (int i = 0; i < TOTAL_FRAMES; i++) {
          if (received_frames[i]) {
            printf(" %d ", i);
          }
        }
        printf("\n");

        // check nack if prev frame in sliding window missing
        for (int j = base; j <= current_frame; j++) {
          if (received_frames[j] == false && negative_frames[j] == false) {
            char nack[BUFFER_SIZE];
```

```
            sprintf(nack, "ACK %d", -j);
            send(new_socket, nack, strlen(nack), 0);
            printf("Sent NACK for frame %d\n", -j);
            negative_frames[j] = true;
            break;
          }
        }
      }
    }
  }

  close(new_socket);
  close(server_fd);
  return 0;
}
```

client.c

```
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define PORT 8080
#define WINDOW_SIZE 4
#define BUFFER_SIZE 1024
#define TOTAL_FRAMES 8

int main() {
  bool skip_1 = true;
  int sock = 0;
  struct sockaddr_in serv_addr;
  char buffer[BUFFER_SIZE] = {0};
  bool frame_ack[10] = {false};

  // Create socket file descriptor
  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("\n Socket creation error \n");
    return -1;
  }

  serv_addr.sin_family = AF_INET;
  serv_addr.sin_port = htons(PORT);

  if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
    printf("\nInvalid address/ Address not supported \n");
    return -1;
```

```
  }

  if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    printf("\nConnection Failed \n");
    return -1;
  }

  int base = 0;
  int next_seq_num = 0;

  while (base < TOTAL_FRAMES) {
    while (next_seq_num < base + WINDOW_SIZE && next_seq_num < TOTAL_FRAMES)
{
      if (skip_1 && next_seq_num == 1) {
        skip_1 = false;
        next_seq_num += 1;
        continue;
      }
      char frame[BUFFER_SIZE];
      sprintf(frame, "%d", next_seq_num);
      send(sock, frame, strlen(frame), 0);
      printf("Sent frame %d\n", next_seq_num);
      next_seq_num++;
      if (next_seq_num == 1) {
        break;
      }
    }
    fd_set readfds;
    struct timeval tv;
    tv.tv_sec = 1;
    tv.tv_usec = 0;

    FD_ZERO(&readfds);
    FD_SET(sock, &readfds);

    int activity = select(sock + 1, &readfds, NULL, NULL, &tv);

    if (activity > 0) {
      int valread = read(sock, buffer, BUFFER_SIZE);
      if (valread > 0) {
        int acked_frame;
        sscanf(buffer, "ACK %d", &acked_frame);
        if (acked_frame < 0) {
          acked_frame = (-acked_frame);
          printf("Received negative acknowledge: %d\n", acked_frame);

          char frame[BUFFER_SIZE];
          sprintf(frame, "%d", acked_frame);
          send(sock, frame, strlen(frame), 0);
```

```
          printf("Sent frame %d\n", acked_frame);
          continue;
        }

        if (acked_frame > TOTAL_FRAMES) {
          acked_frame /= 10;
        }

        printf("Received ACK for frame %d\n", acked_frame);
        frame_ack[acked_frame] = true;

        if (acked_frame == base) {
          base++;
        }
      }
    } else {
      if (skip_1) {
        skip_1 = false;
        continue;
      }
      printf("Timeout occured\n");
      next_seq_num = base;
    }
  }

  close(sock);
  return 0;
}
```

Output

```
ass2/q2/src via C v15.0.0-clang
❯ just sr
zig cc  ./src/server.c -std=c23 -o ./bin/server
./bin/server
Server listening on port 8080
Sent ACK for frame 0
Received Frames:  0
Received Frames:  0  2
Sent NACK for frame -1
Received Frames:  0  2  3
Sent ACK for frame 4
Received Frames:  0  2  3  4
Received Frames:  0  2  3  4
Received Frames:  0  2  3  4
Received Frames:  0  1  2  3  4
```

```
da/ass2/q2
❱ just cr
zig cc  ./src/client.c -std=c23 -o ./bin/client
./bin/client
Sent frame 0
Received ACK for frame 0
Sent frame 2
Sent frame 3
Sent frame 4
Received negative acknowledge: 1
Sent frame 1
Received ACK for frame 4
```