

UDP Socket projects

(단 모든 frame은 HDLC frame (flag 포함)을 기준으로 작성)

Project 1: UDP Echo Server & Client

목적: UDP Socket을 이용하여 패킷을 주고 받을 수 있는 기능 구현

내용: UDP socket을 이용하여 UDP 패킷을 보내고 그 응답 UDP 패킷을 받는 과제

- UDPMYEcho: Client Mode
- UDPMYEchoServer: Server Mode

할일(Mission):

1. Local IP 주소와 UDP 포트 번호를 찾아냄
2. Remote IP 주소와 remote UDP 포트 번호를 찾아냄
3. Multi-Thread UDP Echo Server/Client 구현
4. (Option) Timeout 기능 구현 (일정 시간내에 Echo가 오지 않으면 다시 전송하기 위한 timeout 설정과 재전송 동작)

결과: UDP Socket program을 통해 UDP 패킷의 송수신 프로그램 구현

참고: 주어진 파일

제출파일: 보고서, 모든 source, 실행파일(jar or Exe), 사용설명서(컴파일방법+동작방법)

제출기한: 2015년 5월 10일 자정

Project 2: UDP Chatting

목적: UDP Socket을 이용하여 메시지를 주고 받으며 화면을 통해 서로간에 통신이 가능한 기능 구현

내용: UDP socket을 이용하여 화면에서 keyboard를 입력 받아 상대방에게 UDP 패킷을 보내고 상대방으로부터 온 UDP 패킷을 화면에 표시하면서 UDP를 통해 메시지를 주고 받는 과제 (각자 client와 서버를 모두 포함하고 있음)

- UDPChatting: keyboard의 내용을 입력 받아 UDP 패킷을 만들고 만들어진 UDP 패킷을 IP 주소와 UDP port 번호로 정의된 상대방에게 보냄
- RcvThread: UDP 패킷이 오면 오는 즉시 패킷을 받고 화면에 표시해주는 기능

할일(Mission):

1. Multi_thread 기반 UDP Chatting 프로그램 완성
2. 송신 후 수신 (ACK: 상대방의 메시지 임) 기다리는 프로그램 완성 (단, 다음 메시지가 보내지거나 저장되지 말아야 함)
3. 일정 시간 내 응답(상대방의 메시지 임)이 없으면 timeout 에 의한 재전송 기능구현

4. (Option) Stop & Wait 기능 구현 (상대방의 메시지로 응답하는 것이 아니라 사용자에게는 보여지지 않는 ACK packet을 만들어 보내는 것임) 따라서 내가 메시지를 보내고 상대 프로그램이 ACK를 만들어 보내면 다음 메시지를 전송할 수 있음. 상대는 메시지를 지속적으로 보고만 있어도 됨)

결과: UDP Socket program을 통해 상대방과 채팅할 수 있어야 함. (UDP 패킷의 송신과 수신은 독립적인 Thread로 수행되는 프로그램 구현)

참고: 주어진 파일

제출파일: 보고서, 모든 source, 실행파일(jar or Exe), 사용설명서(컴파일방법+동작방법)

제출기한: 2015년 5월 17일 자정

Project 3: Reliable UDP Chatting (Stop-&-Wait ARQ)

목적: UDP Socket을 이용하여 HDLC 포맷에 따른 신뢰성 (Reliable) 있게 메시지를 주고 받으며 화면을 통해 서로간에 통신이 가능한 기능 구현 (교재에 제시된 ARQ 방식을 활용한 채팅 프로그램 완성,

내용: UDP socket을 이용하여 화면에서 keyboard를 입력 받아 상대방에게 UDP 패킷을 보내고 상대방으로부터 온 UDP 패킷을 화면에 표시하면서 UDP를 통해 메시지를 신뢰성 (Reliable) 있게 주고 받는 과제 (신뢰성 (Reliable) 있게란 다음과 같은 기능을 차례로 구현한다:)

- 프레임 구성 (HDLC 포맷에 따른)
- CRC32 구성 (CRC32 생성 및 체크)
- ACK 패킷의 구성 (패킷 수신 시 ACK 생성 후 전송)
- Timeout (ACK 가 없으면 일정 시간 후 재전송)
-

할일(Mission): Project 1과 2를 결합해서 ARQ 프로토콜 완성

1. ARQ Framing (주어진 프레임 구조 사용) (ACK:0x1, NAK:0x0)
2. 데이터 패킷과 별개의 ACK 패킷 생성 및 전송 (ACK:0x1, NAK:0x0) No data frame
3. 에러 검출 기능 구현 (CRC-32)
4. (Option) 송수신 데이터와 ACK의 각 필드와 절차를 모니터링
5. (Option) 상대의 패킷 전송 시 에러 검출, ACK 전송, timeout을 체크할 수 있는 conformance test 기능 구현

결과: UDP Socket program을 통해 상대방과 신뢰성 (Reliable) 있게 채팅할 수 있어야 함. (UDP 패킷의 Error를 검출하고 에러가 발생하여 전달되지 않은 경우 Timeout에 의해 재전송)

참고: 주어진 파일

제출파일: 보고서, 모든 source, 실행파일(jar or Exe), 사용설명서(컴파일방법+동작방법)

제출기한: 2015년 5월 24일 자정

Option: Project 3: Reliable UDP Chatting (Stop-&-Wait ARQ) over serial Interface

목적: **serial interface** 를 통해 UDP Socket을 이용하여 **HDLC 포맷에 따른 신뢰성 (Reliable)** 있게 메시지를 주고 받으며 화면을 통해 서로간에 통신이 가능한 기능 구현 (교재에 제시된 ARQ 방식을 활용한 채팅 프로그램 완성,

내용: UDP socket을 이용하여 화면에서 keyboard를 입력 받아 상대방에게 UDP 패킷을 보내고 상대방으로부터 온 UDP 패킷을 화면에 표시하면서 UDP를 통해 메시지를 **신뢰성 (Reliable)** 있게 주고 받는 과제 (**신뢰성 (Reliable)** 있게란 다음과 같은 기능을 차례로 구현한다:)

- 프레임 구성 (Seq No(1), Ack No(1), Flag(1), length(1), CRC(4), Data(~504) 단, 숫자의 표시는 바이트 단위
- CRC32 구성 (CRC32 생성 및 체크)
- ACK 패킷의 구성 (패킷 수신 시 ACK 생성 후 전송)
- Timeout (ACK 가 없으면 일정 시간 후 재전송)
-

할일(Mission): Project 1과 2를 결합해서 ARQ 프로토콜 완성

6. ARQ Framing (주어진 프레임 구조 사용) (ACK:0x1, NAK:0x0)
7. 데이터 패킷과 별개의 ACK 패킷 생성 및 전송 (ACK:0x1, NAK:0x0) No data frame
8. 에러 검출 기능 구현 (CRC-32)
9. (Option) 송수신 데이터와 ACK의 각 필드와 절차를 모니터링
10. (Option) 상대의 패킷 전송 시 에러 검출, ACK 전송, timeout을 체크할 수 있는 conformance test 기능 구현

결과: UDP Socket program을 통해 상대방과 **신뢰성 (Reliable)** 있게 채팅할 수 있어야 함. (UDP 패킷의 Error를 검출하고 에러가 발생하여 전달되지 않은 경우 Timeout에 의해 재전송)

참고: 주어진 파일

제출파일: 보고서, 모든 source, 실행파일(jar or Exe), 사용설명서(컴파일방법+동작방법)

제출기한: 2015년 5월 31일 자정

Option: Project 4: Reliable UDP Chatting (Go-B-N ARQ)

목적: UDP Socket을 이용하여 효율적이면서 **신뢰성 (Efficient & Reliable)** 있게 메시지를 주고 받으며 화면을 통해 서로간에 통신이 가능한 기능 구현

내용: UDP socket을 이용하여 화면에서 keyboard를 입력 받아 상대방에게 UDP 패킷을 보내고 상대방으로부터 온 UDP 패킷을 화면에 표시하면서 UDP를 통해 메시지를 효율적이면서 **신뢰성 (Efficient & Reliable)** 있게 주고 받는 과제 (**효율성 (Efficient)** 있게란 패킷 전송에 파이프라인 효과를 추가하는 것:)

- S-&-W 모든 기능 및
- 연속적 패킷 전송 (Go-Back-N Protocol)

- Cumulative Ack 관리
- 패킷 송신과 수신 병렬화

결과: UDP Socket program을 통해 상대방과 효율적이면서 신뢰성 (Efficient & Reliable) 있게 채팅할 수 있어야 함. (UDP 패킷의 Error를 검출하고 에러가 발생하여 전달되지 않은 경우 Timeout에 의해 재전송, 전송시 ACK가 바로 오지 않아도 연속적인 패킷을 전송할 수 있어야 하고 패킷이 에러가 나면 이를 극복할 수 있어야 함)

참고: 주어진 파일

제출파일: 보고서, 모든 source, 실행파일(jar or Exe), 사용설명서(컴파일방법+동작방법)

제출기한: 2015년 6월 7일 자정

Option: Project 5: Reliable UDP Chatting (Selevtive-Repeat ARQ)

목적: UDP Socket을 이용하여 효율적이면서 신뢰성 (Efficient & Reliable) 있게 메시지를 주고 받으며 화면을 통해 서로간에 통신이 가능한 기능 구현 (보다 효율적인 패킷 전송)

내용: UDP socket을 이용하여 화면에서 keyboard를 입력 받아 상대방에게 UDP 패킷을 보내고 상대방으로부터 온 UDP 패킷을 화면에 표시하면서 UDP를 통해 메시지를 효율적이면서 신뢰성 (Efficient & Reliable) 있게 주고 받는 과제 (효율성 (Efficient) 있게 란 패킷 전송에 파이프라인 효과를 추가하는 것임, 보다 효율적인 패킷 전송)

- S-&-W 및 G-B-N 모든 기능 및
- 연속적 패킷 수신
- Selective Ack 관리
- 선별적 패킷 재 구성

결과: UDP Socket program을 통해 상대방과 효율적이면서 신뢰성 (Efficient & Reliable) 있게 채팅할 수 있어야 함. (UDP 패킷의 Error를 검출하고 에러가 발생하여 전달되지 않은 경우 Timeout에 의해 재전송, 전송시 ACK가 바로 오지 않아도 연속적인 패킷을 전송할 수 있어야 하고 패킷이 에러가 나면 이를 극복할 수 있어야 함 또한 수신된 패킷이 에러가 없을 경우 순서에 상관없이 재전송을 하지 않고 사용 가능하도록 함. (보다 효율적인 패킷 전송))

참고: 주어진 파일

제출파일: 보고서, 모든 source, 실행파일(jar or Exe), 사용설명서(컴파일방법+동작방법)

제출기한: 2015년 6월 7일 자정

상호 운영성 테스트 (option on demand)

친구끼리 미리 자신이 작성한 코드를 기반으로 상호 연동되는지 확인 (포맷, CRC 포함 내용 범위, 위치, sequence 번호 설정 및 방법 등)

서로 확인 중 나온 여러 가지 문제점을 나열하고 개선방법을 기술해서 보고 (서로간의 의견차에

의한 이견이나 문제를 제시하고 해결하는 방법)

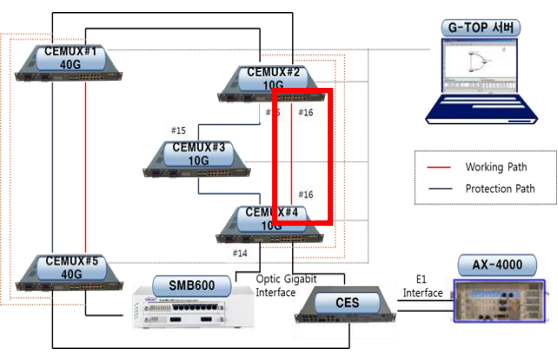
메시지 포맷이나 타임아웃 값 등의 변화를 시도해서 에러가 있는지를 확인 하고 수정 (상호 운영 성 테스트는 모르는 사람들이 짧은 시간에 자신들의 작업에 대한 의견을 조율하는 것으로 각자 자신의 방향에 따라 기존 문서를 기반으로 만들어야 합니다. 서로 동작하면서 이상한 점이나 동작의 시행 방법을 다양하게 변화하면서 테스트 해야 하는 것이죠. 시험절차서라는 방법을 동원하여 문서화 해 놓기도 합니다.)

기타 서로 운영하면서 발생할 수 있는 에러를 미리 감지해서 수정 후 보고

단, 강의 진도에 따라 다소 변경될 수 있음.

참조 자료

1. 항목 시험 절차

항목 번호	1-1	시험일자	2013.05.09	시험자	신용섭
대항목	G-TOP Broker 관리 기능 검증	중항목	Test1	소항목	Switch Hub 를 Working Path 에 연결 후 LSP 설정 변경
목적	LSP 설정 과정 중에 Cemux 간의 패킷 흐름 및 G-TOP Server 와의 패킷 흐름 확인				
판정기준	Wireshark 캡처				
시험 절차 (시험 절차 또는 방법 작성)		시험 구성 (시험 구성도 및 관련 명령어 작성)			
<ol style="list-style-type: none"> 1. Cemux 가 모두 살아있는 상태에서 시작한다.. 2. G-TOP 서버를 새로 시작을 시킨다. 3. 기존에 설정되어 있는 T-LSP 와 S-LSP 설정을 삭제한다. 4. Smart Bit 의 패킷 흐름을 확인한다. 5. Auto S-LSP 를 사용하여 T-LSP 와 S-LSP 를 설정한다. 6. Smart Bit 의 패킷 흐름을 확인한다. 					
시험 결과 (시험 절차 에 대한 결과 작성)					
<ol style="list-style-type: none"> 1. 기존에 설정된 T-LSP 와 S-LSP 를 삭제하는 과정에 대한 패킷 캡처 2. 기존에 설정된 T-LSP 와 S-LSP 를 삭제하는 과정에서 CEMUX 간의 패킷 캡처 3. 새로 T-LSP 와 S-LSP 를 설정하는 과정에 대한 패킷 캡처 4. 새로 T-LSP 와 S-LSP 를 설정하는 과정에서 CEMUX 간의 패킷 캡처 					
판정	<ol style="list-style-type: none"> 1. Wireshark 를 이용해서 해당하는 패킷을 캡처 하도록 한다. 2. 각각의 상태에서 Smart Bit 에서 패킷이 흐르는지 확인한다. 			비고	<ol style="list-style-type: none"> 1. G-TOP Server 는 1.0 버전을 사용하여서 측정하였다. 2. Switch Hub 의 경우 위의 시험구성도의 빨간색 상자 부분에 연결을 하여서 측정하였다. 3. 패킷 흐름은 Smart Bit 를 사용해서 확인하였다.