네트워크

127.0.0.1~255 는 대부분 loop back address 다. Host 라고도 한다.

프로토콜이란

첫번째 헤더에 쓰여있는 처리방법에 대한 명령어이다. 메모리로 따지자면 명령어이며 해당 명령어에 따라서 처리가 달라지며 계층또한 달라진다.

IPC ip 간의 커뮤니케이션

쓰레드란

프로세서에서 최소 1개 이상이고 실행의 단위이다. 즉 프로세스 위에 1개이상의 쓰레드가 있고, 쓰레드는 개별화된 문맥즉 컨텍스트와 전용 스택을 가지고 있다. 스택을 따로 가지고 전용 스택을 가진다. 모든 쓰레드는 힙공간과 data code 영역을 공유한다. Stack 만 따로 가진다.   
쓰레드는 프로세서가 만들어지면 메인쓰레드를 만들고, 해당 쓰레드에서 작업자 쓰레드를 만든다. 참고로 요즘은 코어가 따로 쓰레드를 실행할 수 있기 때문에 멀티 코어보다는 멀티쓰레드가 멀티코어도 지원한다라고 봐야한다.

진입제어(쓰레드 동기화)

동기화 문제를 해결하는 방법이다.

크리티컬 세션

이벤트 객체

텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

서스펜드는 런상태가 아니므로 아무것도 연산하지 않는다.

이벤트 객체 생성

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명

이름을 붙인다라는 것은 여러곳에서 사용하는 유명한거라고 생각하면 된다. 신호를 만드는거고 윈도우에서느 set 과 reset 이 있고, set 이 1 reset 이 0 이고, 감시하는 주체가 있는데 wait 가 감시한다.

Wiat 는 다른쓰레드에서 setEvent 를 하게 되면 다른쓰레드에서 waitForSingleObject 를 하고있는 쓰레드가 다른쓰레드가 wait 될때까지 기다렸다가 행동을 하게 할 수 있다. 즉 어떤 감시쓰레드를 만들고 해당 감시쓰레드가 해당 이벤트를 기다리는 동안 감시대상쓰레드가 setEvent 를 하게 되면 감시하고있는 쓰레드가 그 반응을 알수 있는것이다. 그런데 감시하는 동안에는 감시쓰레드는 wait 가 걸린것이나 마찬가지이다.

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

자동 생성된 설명

소켓프로그래밍

소켓의 본질은 무엇인가? os커널에 구현되어있는 프로토콜 요소에 대한 추상화 되어진 인터페이스. 장치파일의 일종이기도 하다. 일반 파일에 대한 개념이 대부분 적용된다.

프로세스가 대상체가 되는 파일에 open 이나 create 할 수 있다. 할 수 있는 일은 rwx 이고 r 은 read 고 w 는 write 이고 x 는 execute 이다. 기본적으로 tcp 도 file 이기 때문에 rwx 가 모두 지원되는데, w 는 send 라고하고 read 는 receive 라고한다. 물론 파일 입출력과 거의 똑같지만 뭐 socket 자체가 특수성을 가져서 생성하고 다른 함수를 호출해줘야하지만 그뒤에는 파일 io 와 같다.

File 이기 때문에 stream 이다. 즉 append 하면서 계속 쓰는것이다.

L4(tcp) 에서 데이터 단위는 segment L3(ip) 에서는 packet 이라한다.

라운드 트립 타임 == ping 과 비슷함. 데이터가 서버로 갔다 돌아오는걸 말함.

프로토콜 설계란 적절한 연결 적절한 연결끊기 같은 것을 설계하는게 프로토콜 설계이다.

반드시 클라이언트가 종료하자고 해야한다. 그리고 소켓을 닫는 close 는 shutdown 후에 한다

간단한 에코서버 시퀸스

1. fOpen
   1. Server
      1. socket() 일반적으로 fOpen 같을거다. 하지만 이걸 fopen 에 묶은 이유는 파일을 준비하는 단계이기 때문이다. 상태는 close 이다
      2. bind() file 을 열고나서 이제 tcp 니 udp 니 같은것에 대한 정보를 묶어준다라는 의미에서 bind 라고한다. Ip 주소와 포트번호가 이때 연결이 되는것이다. close 이다
      3. listen() 여기에서 상태가 변경이된다. 상태는 wait 이다. 다이어그램으로 보면 해당부분에서 accept 까지가 listen 이라고 보면된다.
      4. Accept() 여기도 listen 인데 중요한건 connect 가 들어올때까지 계속 대기한다는것이다. 그리고 connect 가 들어오면 socket 을 만들어서 반환한다. 이걸 서버의 통신 소켓이라 부른다.  
         즉 크게 보면 이렇다. 먼저 만드는 것을 서버 소켓 이라부르고 accept가 반환하는 소켓을 서버통신 소켓이라 부른다. 서버 소켓은 접속대기용이고, 서버통신소켓은 클라이언트와의 통신 소켓인것이다.  
         텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

         자동 생성된 설명
   2. Client
      1. socket() 이녀석은 서버에 연결하기 위한 소켓이다. 근데 그냥 클라이언트 소켓 이라부른다.
      2. Connect() 뭐 이녀석은 그냥 통신용 소켓이 따로 만들어지는게 아니다.

샌드와 리시브는 1:1 관계가 아니다. 네이글? 알고리즘이란게 있단다. 하여튼 딜레이에 의한 문제인것이다. 네트워크는 손실이 생길 수 있다.

TCP 샌드 옵션은 클라와 서버가 같은게 좋아보이기는하는데 사실 클라에서만 하고 서버에서는 사실 안한다.

SO\_REUSEADDR 옵션을 사용하면 바인딩할 때 주소를 명확하게 해줘야한다. Cq 코딩

Pingpong 쓰는 이유는 좀비 세션을 종료시키기 위해서이다.

내가 원하는게 IO멀티플렉싱 서버이다. 여기에서 가장 중요한건 모든 입출력은 프로세스가 아니라 os 가 주도한다는것이다. 그도 그럴것이 소켓은 os 에서 한다. Os 한테 부탁하는거다. 개발자 관점에서는 송신과 수신이 동시에 발생하는것이다.   
근데 MFC 나 os 에서 지원해주는게 있긴하다. 그게 FDRead 니 뭐 그렇게 부른다. FD 는 64개 밖에 안되며 해당 녀석은 변화가 감지되어지면 뭐 FD 소켓들을 돌면서 뭐하는거다. 진짜로 그냥 os 가 알려줌! 그리고 너무 옛날거라고 한다. 싱글쓰레드에서만 돌아가고 os 가 감시를 해준다. 마치 비동기적으로 작동을 하고 컨텍스트 스위칭 비용이 적다는 것이 장점임.

EventSelect 서버는 이벤트(os 가 감시하는 기능)가 발생하면 콜백하는거다. 이게 좀더 나은방식임

Event AsyncSelet 방식은 MFC 에서 사용하는 방식이다. 이벤트 방식이라 싱글은 맞는데, 이거쓰면 에러남. 구형에서만 사용이 가능함.

서버 진화 순서 : 멀티플렉싱 -> eventSelects -> iocp || epoll -> Rio

동기와 비동기는 만약 wait 가 걸리면 동기 아니면 비동기이고, 큐에 넣은 이벤트를 기다리는걸 pending 되었다고 한다.

장애유형

TCP 의 장애유형

일단 알아야할 것이있다. 인터넷 환경은 카오스 즉 이마테리움이다.

인터넷 인프라의 문제

해당 문제는 네트워크 관리자가 고민해야할 문제이다. 물론 내가 ㅋ

PacketLoss 패킷 손실(L3 문제 및 L2 L1 문제일수도 있음)

TCP Out of Order 패킷 순서가 꼬이는 것 이게 재선송이 일어날 수 있음(L4~L3 문제)

Retransmission 과 Dup ACK : 재전송시에 애크날리즘이라고 전송완료 데이터인데 이게 두번내지 n 번오는것을 말한다.

어플리케이션 관리자가 해야할일(L5 수준에서 고민하는것)

Zero Window : 수신측 버퍼에 여유공간이 없는경우. 네트워크 속도보다 처리속도가 느릴 때 발생할 수도 있다. 물론 os 단이 L4나L3 혹은 그 이하에서 나면 인프라 문제라 싸우면 되기는 하는데, 그래도 혹시나 말이다 내가 처리를 못했을경우, 나으이 알고리즘이 절었을 경우에는 내가 혼나야지 ㅋ그냥 뭐 heartBeat 하면 된다. 이거 효율 높이는걸 해야한다. 게임쪽이 가장 고성능에 절절맨다

TCP 에서 데이터를 말할 때 segmant 라고 한다.운영체제에서 ACK 가 1초미만이 정상이고 해당 부분을 넘어간다면 재전송을 한다.

운영체제에서 수신을 관리하는 버퍼가 따로있긴하다.

프로토콜 정의

기본헤더와 확장헤더로 나눠서 정의

Send 와 recv 와 1:1은 아니다. Send 하고나면 recv 는 여러 번 끊어내어 읽어내는것이다. 보내는건 stream 이라 생각하고 recv 즉 수신측에서는 끊어 낸다. View 와 desc 와 같은 관계이다.

기본헤더는 이어지는 확장 헤더를 결정하기위한 코드 값 포함

프로토콜 정의란 보내고 받는 규약을 정하는것이다. 보통 헤더에는 코드+사이즈 가 좋다. Ex 는 추가적인 부분이다. 내가 웹에서 만든게 이런것과 동일하다.

코드 값에 따른 switch-case

Lookup 테이블 방식으로 최적화를 한다.

각 경우에 대한 함수 처리 및 해석 방법적용

파일 송수신 프로토콜 정의 – 헤더

명령코드와 데이터 바이트크기를 정해서 넣는게 편한데, 언사인드가 좀 좋긴하다. 보통 문서화하고 그냥 내가 웹설계해서 사용할때가 프로토콜 정의이며 extend 를 잘정의하는게 좋다. 이걸 내가 정보도 없이 한게 대단하다! 하여튼 헤더는 구조체로 정하는게 좋긴하다.

프로토콜 설계란 결국 내가 해석하는 것을 정하고 만드는것이다. 어려울것이 없다.룩업테이블에 함수포인터 때려넣으면 더 좋다.

이벤트방식

프로세서에서 io 처리해달라고 os 에게 요청하고 io처리가 끝나면 io overlaped 구조체 안에 이벤트 핸들을 os 가 set 해준다. 그럼 wait 하고있던 프로세서는 wiat 에서 완료되었다는 것을 반환한다.

콜백방식

콜백방식은 이벤트처럼 무언가를 set 해주는게 아니라 그냥 함수를 실행시킨다. Call 은 os 가 알아서 해주므로 우리는 볼 수 없다. 이때 완료가 되면 해당 쓰레드는 sleepEX 가 걸리고 그뒤에 callback 이 이루어지며 그다음 complete 가된다.  
쓰레드는 Running 과 suspended 가 기본인데, sleepEX 가 호출되면 Alertable Wait 라는 상태가 추가된다. 해당 상태는 callBack 이 완료되면 wait 시간과 관계없이 상태가 변화한다.

Iocp 는 속도도빠르고 스레드풀도 관리할필요없지만 대신 규칙을 지켜야한다.  
텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명