**Lab5. Network lab**

컴퓨터공학부 2022-19510 오정윤

1. TCP/IP network environment

우선 서버에서는 start\_server()에서 listen socket을 만들고 client의 request를 기다린다. 이를 위해 getsocklist()에 AF\_INET, SOCK\_STREAM 변수를 인자로 넘겨주어 IPv4, TCP 환경에서의 listen용 socket을 찾고 while문을 통해 bind()와 listen()이 성공하는 listen socket을 만든다. 또한 while문을 통해 계속 listen socket으로 오는 client request를 받아 최대 고객 수를 넘지 않는지 “if (server\_ctx.total\_queueing >= CUSTOMER\_MAX)”를 통해 검사하고, 넘는다면 while문을 continue한다. 넘지 않는다면 total\_queueing을 증가시키고 serve\_client thread를 만든다. 이때 인자로 넘겨주는 thread\_client\_fd는 malloc()을 통해 고정한다.

동시에 클라이언트에서는 main()에서 미리 n개의 client task thread를 만든다. 이 thread\_task는 서버와 마찬가지로 getsocklist를 통해 리스트를 가져오고 connect()가 성공하는 socket을 찾는다.

serve\_client thread와 thread\_task thread는 put\_line(), get\_line()을 통해 메시지를 전달한다. 특히 serve\_client에서는 thread\_task에서 오는 request를 get\_line()하여 여러 개의 order로 나눈다. 이렇게 나눈 order를 issue\_orders()를 통해 enqueue한 후 main()에 의해 작동되고 있는 kitchen\_task thread에 의해 dequeue되도록 한다.

1. Atomicity between threads

같은 ordre\_list에 속한 order들이 remain\_count, cond, cond\_mutex를 공유하고 있기 때문에 pthread\_mutex\_lock()을 통해 일관성을 보장해야 했다. skeleton에서는 issue\_orders()를 실행할 때 노드를 enqueue하면서 서버 차원의 뮤텍스를 사용하여 링크 도중의 데이터 일관성을 보장하고 있다. 또한 order\_list의 remain\_count가 전부 동일하기 때문에 이를 보호하는 것이 중요했다. 이를 위해 serve\_client thread에서는 첫번째 order의 remain\_count가 양수일 때까지, 즉 order가 남아있을 때까지 pthread\_cond\_wait()을 통해 serve\_client thread 작동을 기다렸다. 또한 kitchen\_task thread에서는 make\_burger()를 실행하고 remain\_count를 감소시키는 과정 전후로 mutex를 추가하여 remain\_count의 값을 보호했다. 이뿐만 아니라 주어진 skeleton에서는 get\_order()와 order\_left(), 즉 dequeue할 때와 남은 order 수를 살필 때에도 server context의 mutex를 사용하여 데이터를 보호하고 있다. 또한 통계를 위한 total\_burgers[]를 증가시킬 때, customer ID를 정하기 위해 total\_customer를 증가시킬 때, 주문 완료 후 total\_queueing을 감소시킬 때 데이터를 보호하고 있다.

1. optimization

앞서 언급했던 kitchen\_task에서의 mutex 활용에서, global variable인 kitchen\_mutex를 사용하면 kitchen thread 간의 지연이 발생하여 최적화를 위해서는 이를 사용하지 않아야 했다. 따라서 make\_burger()과 remain\_count 감소하는 과정을 order\_list들이 공유하는 order->cond\_mutex를 사용하여 lock, unlock을 했다. remain\_count를 보호하는 본래 목적이 같은 order\_list 내에서 remain\_count를 공유하기 때문에 이를 이용하는 것이 더 효율적이다. 실제로 kitchen\_mutex를 사용하였을 때는 reference와 비슷한 성능을 보였는데, cond\_mutex를 사용한 결과 시간을 절약할 수 있었다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 1. Client 5일 때 최적화 코드의 실행 시간

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 2. Client 5일 때 reference 코드의 실행 시간

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 3. Client 10일 때 최적화 코드의 실행 시간

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 4. Client 10일 때 reference 코드의 실행 시간