Gebze Teknik Üniversitesi
Bilgisayar Mühendisliği

CSE 344 – Sistem Programlama

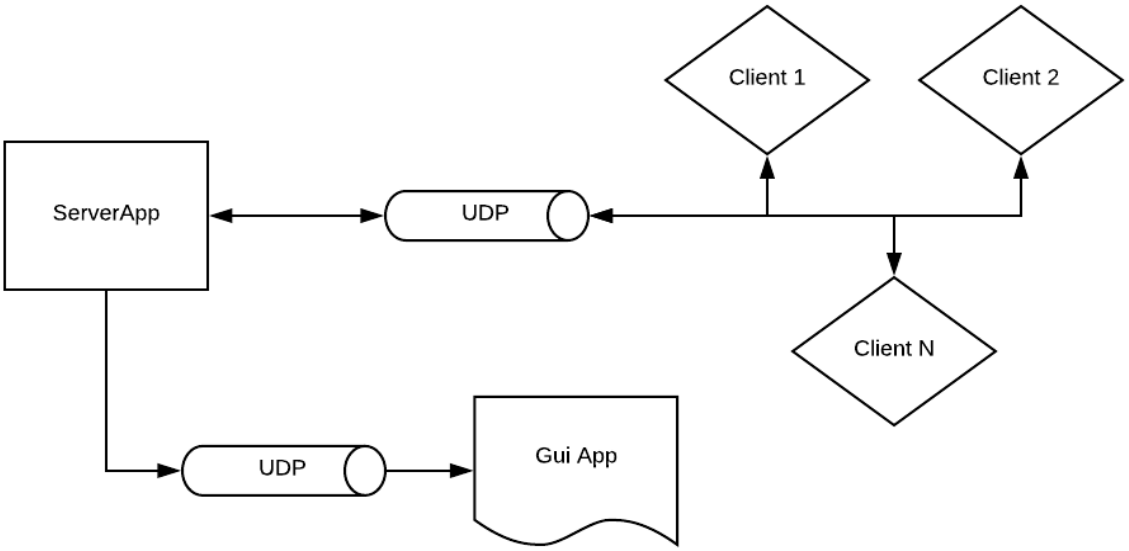
Tek Ders Projesi

Ali Kürşad ŞAHİN
091044038

Dersi Veren Öğretim Üyesi: Erkan Zergeroğlu

1 Projenin Tanımı

Bu projede bizden en temel haliyle multi-client – server bir uygulama yazmamız beklenmektedir. Client ve Server programları UDP soketleri üzerinden çift yönlü haberleşmeleri gerekmektedir. Bu yapıya ek olarak Server uygulaması, bir Gui uygulaması ile istenilen bazı çıktıları yazdırabilmek için tek yönlü haberleşmesi gerekmektedir.



Şekil 1 – Sistem Tasarımı

Şekil-1’de proje tasarımının genel şeması yer almaktadır.

Biraz daha detay vermemiz gerekirse, UDP'yi kullanan bir UDP Port sunucusu uygulaması yazmamız gerekiyor. Sunucu üzerinde çalıştığı bilgisayarda bulunan sevirslerin uygunluğuna dair istemciye bir sorgu sonucu döndürüyor.

Bir başka yazmamız gereken uygulama ise istemci uygulaması. İstemci UDP Port sunucusuna bir sorgu isteği gönderiyor ve bunun karşılığında bir cevap bekliyor. Cevap belirlenen timeout süresinde gelmez ise yeni bir istekte bulunuyor. Cevap gelirse cevabı ekrana bastırıp, tekrar bir istekte bulunuyor.

Bu iki uygulamaya ek olarak bir de 3. gui uygulaması yazmamız gerekiyor. Bu uygulama ise client’ın server’a yolladığı her bir sorgunun dökümünü göstermemize yarıyor. Bu uygulamanın diğerlerinden farkı Java ile yazılacak olması. İstemci ve sunucu uygulamaları ise C dilinde yazılacaktır.

2 Karşılaşılan Problemler

- Race Condition
- Client-Server UDP Connection
- Server-Gui UDP Connection
- Signal Handling
- Recv and Send Data
- Client Algorithm
- Server Algorithm

3 Çözüm Önerileri

- Race Condition
  - Race condition ile karşılaştığımız yerlerde mutex kullanarak sorunu çözdüm. Aynı anda cevaplanacak request sayısına sınır getirdim. Eğer her hangi bir sınır olmazsa pthread_create() fonksiyonu memory'de yer kalmadığı için yeni thread açamıyordu.
- Client-Server UDP Connection
  - Client server arasında udp portu üzerinden haberleşme yapmamız gerekiyor. Udp üzerinden haberleşme yaparken, yanlış response datası client'a ulaşırsa, "sequence number" verisini karşılaştırıp o responese'u ekrana basarak bizden istenildiği gibi problemi handle ediyorum. Client programında ekrana "Sequence number mismatch" yazdırarak bu hatayı yakaladığımızı gösteriyorum.
- Server-Gui UDP Connection
  - UDP Port Sunucumuz tıpkı client ile iletişim kurabilmek için soket açtığı gibi gui ile de iletişim kurabilmek için soket açması gerekiyor. Ancak burada soket soket açan ve bu soketi "bind()" eden taraf UDP Port sunucusu olursa gui'ye veri iletemiyoruz. Bu sorunu Gui'yi server, UDP Port sunucusunu da client olarak düşünerek çözebildim. Yani hem gui hem sunucu tarafında haberleşmek için aynı port üzerinden soket açıyoruz ancak, dinleme yapması gereken taraf olan gui soketi "bind()" ediyor.
- Signal Handling
  - Client ve server uygulamarında programların normal akışlarında run time error olmadığı sürece programın sonlanması istenmiyor. Ancak programlar SIGINT-SIGTERM-SIGHUP sinyallerindne biri ile sonlandırılmak istenirse, pointerlar için allocate ettiğimiz yerleri ve soketleri program sonlandırmadan kapatmamız gerekiyor. Bu problemi signal hanler yazarak çözdüm.

- Recv and Send Data
  - Client, Server ve Gui arasında haberleşmeyaparken veri alışverişini yaparken, merge ve parsa işlemleri ile uğraşmamak için Şekil 2'de yer alan yapıları kullandım. Bu sayede soket üzerinden struct gönderip, struct alabildim.

```
typedef struct {
    request_t request;
    struct sockaddr_in cli_adr;
    size_t cli_adr_len;
} ext_request_t;

typedef struct {
    int client_id;
    int sequence_number;
    int port_number;
    char aliases[ALIAS_SIZE];
} response_t ;
```

Şekil 2 – Request & Response Struct

4 Testler

Kodun çalışıp çalışmadığını test etmek için çeşitli testler oluşturdum.

Örneğin Sunucu hiçbir delay süresi olmadan -sleep()- çalışmak isterse "pthread_create: Resource temporarily unavailable" hatası alıyoruz. Bu hata sistem kaynaklarının sonsuz birdöngüde aralıksız kullanımı durumunda yetersiz gelmesinden kaynaklanmaktadır. Bu sebeple sunucunun request aldıktan sonra geri response göndermesi için 1 saniyelik gecikme ekledim. Clientlar herhangi bir bekleme-gecikme yaşamadan çalıştırıldığında bazen timeout olabiliyor ve bir sonraki isteği yolluyor. Bu durum sürekli oluyor çünkü sunucu tarafındaki 1 saniyelik gecikme cevabı hiç beklemeyen client'ın 0 olan timeout süresinin her halikarda aşılmasına sebep oluyor.

İkinci testimizi yine 1 saniyelik sunucu gecikmesi ve 1 saniye yada biraz fazla yani (1-2sn) istemci timeout olması durumudur. Bu durumda sunucu cevabını 1 saniye gecikmeli verdiği için, bağlı istemciler istemci sayısına ve istemcilerin istek zamanlarına bağlı olarak kısmen rastlantısal olarak timeout durumuyla karşılaşmaktalar. Bu durumda client bazı cevapları gereken zamanda alamadığından olmaktadırç

Üçüncü testimiz yine 1 saniyelik sunucu gecikmesi ve client tarafında çok yüksek timeout olması durumudur. Bu durumda neredeyse hiç timeout olmuyor ancak çok çok nadir olarak timeout yakalayabiliyoruz. Bu da aynı anda fazla sayıda client ile aynı anda istek yolladığımız zaman oluyor. Ancak bizden istenilen proje isterlerinde, her bir request'in client'da kullanıcıdan input olarak alınması olduğu düşünülürse, bu durum da kolay kolay yaşanılmayacaktır.

Yukarıda yer alan testlerin tamamı istemcilerin aralıksız request göndermesi durumunda test edilmiştir. Yani normal kullanım koşullarında elle request bilgileri girildiğinden client uygulamasında timeout olma ihtimali bir hayli düşük olacaktır.

5 Kullanım

Uygulamaların rahatlıkla derlenip çalıştırılabilmesi için makefile yazılmıştır. Bu sebeple uygulamaları çalıştırabilmek için aşağıdaki adımlar yeterli olacaktır.

Dosyaların olduğu klasörde 3 adet terminal açılmalıdır.

Kodu derlemek için;
```
$ make
```

Gui çalıştırmak için termianal 1/3'e;
```
$ java -jar udp-server-monitor-1.0-SNAPSHOT.jar
```

Server çalıştırmak için terminal 2/3;
```
$ ./serverApp
```

Client çalıştırmak için terminal 3/3'e;
```
$ ./clientApp localhost 8081 5
```

yazmak yeterli olacaktır.

Not: Bu program aşağıdaki işletim sistemlerinde sorunsuzca çalışmıştır.

Ubuntu 18.04
Ubuntu 14.04
MacOS 10.13.6

Ders kapsamında kullanılan sanal makinede, yani "Debian 9.3 versiyonunda" ise openjdk'ya javafx paketleri dahil edilmediğinden Gui çalışmamaktadır. Ancak program çalıştırılmadan önce Oracle Java 1.8.0_181 kurulumu yapılırsa, JavaFx paketleri kurulacağı için çalışacaktır.