INF 443

Dağıtık Sistemler ve Uygulamaları

Proje Raporu

11401587 Arda İçmez

1-Projeyi yaparken geçtiğiniz aşamalar nelerdir?

Projeyi kodlamaya başlamadan önce, istenileni iyice anlayabilmek için birkaç kez proje tanımının bulunduğu pdf dosyasını birkaç kez okumam gerekti. Konuyu anladıktan sonra projeyi moduller halinde ayırmaya başladım. İlk ana modül Negotiator kodlamak, ikincisi Peer kodlamak, üçüncüsü ise gereken filtreleri kodlamak olduğuna karar verdim. Elimizde zaten çalışan filtreler hazır olarak bulunduğundan yeni filtreleri kodlamayı en sona bıraktım. İlk kodlanacak modülün Negotiator olması kaçınılmazdı, çünkü sistemi başlatan ve ayakta tutan elemanımız bu. İstenilenler doğrultusunda 3 farklı sub-modül ortaya çıktı:
- Peer-Client ile ilgilenecek kısım
- Peer-Server ile ilgilenecek kısım
- Bağlantıları düzenli bir şekilde kontrol eden Timer kısmı

Peer-Client kısmı için ilk düşündüğüm algoritma gereksiz bir şekilde komplike olduğundan bu parçayı 2 kez kodlamam gerekti. Peer-Client ve Peer-Server parçalarını kodlamayı bitirdikten sonra linuxtaki telnet komutuyla test ettim, gerekli hataları düzelttim. Timer ı kodlamayı daha sonraya, yani Negotiator – Peer bağlantısı kurulduktan sonraki aşamaya aldım. Sıra Peer kısmının kodlamasına geldiğinde çabucak anladım ki bu kısım, Negotiator kadar basit değil. Yine kağıt ve kalem yardımıyla karşılabileceğim zorlukları ve yapılması gerekenleri detaylı bir şekilde not aldım. İlk bir bakışta çıkan aşamalar :
- Peer-Client ve Peer-Server ı basit bir düzeyde kodlayarak Negotiator'a bağlamak
- Peer Neg-Client ile düzenli olarak Negotiator a request yollayan parçayı oluşturmak
- Birden fazla peer ile negotiator a bağlanmak ve CONNECT_POINT_LIST leri Negotiator'dan istemek
- Negotiator kısmında bahsettiğim Timer kısmını kodlayıp düşen peerları belirlemek
- Peer ları birbirine bağlamak (basitçe)
- Peer a updater thread ekleyip düşen bağlantıları bulmak
- Arayüz ile Peer veri alışverişini sağlamak
- Peer – Peer arasında arayüzden alınan paketlerin gönderildiği bağlantıyı sağlamak
- İşlenmiş PATCH leri tekrar Arayüze gönderip resmi güncellemek

Peer-Client ve Peer-Server kısmını Negotiator ile haberleştirecek kadar kodlamak pek zamanımı almadı. Bundan sonra çoklu peer yapısını test edip Peer ların Negotiator dan CONNECT POINT LIST çekmesini başardım ve Negotiator-Timer kısmını implemente edip, programı kapanan peerları tespit edebilecek duruma getirdim. Bu aşamaya kadar olan kısmın benzerini zaten önceki ödevlerde yaptığımızdan beni o kadar zorladığını söylemem. Bundan sonraki aşama projenin kodlama kısmındaki en zaman alan aşaması olduğunu söyleyebilirim. Bize verilen iskelet kodu inceleyip Arayüzün resim PATCH lerini nasıl alıp dağıttığını anladıktan sonra Peer-Client da ilgili kısmı kodlamaya başladım. Her ne kadar buraya kadar olan kısım senkron olsa da burada asenkronluğun devreye girdiğini anladıktan sonra, görevler için farklı thread açmaya karar verdim. Bir thread, CONNECT POINT LIST deki diğer Peer lar ile bağlanmaya çalışacak, bağlantı sağlandığında iki farklı thread açılacak ve bir tanesi Arayüz den alınan işlenmemiş PATCH leri diğer Peer ların server kısmına atmaya çalışırken 2. thread ise bu serverları dinleyip işlenmiş PATCH mesajını arayüze iletmek ile görevlendirilmesine karar verdim. Bunları kodladıktan sonra Peer-Server kısmını kodladım, bu kısım Peer-Client kısmına göre biraz daha basitti diyebilirim, çünkü birçok thread açmama gerek yoktu. Bu aşamaya kadar olan kısımları bitirdikten sonra, Peerlar arası yapılan PATCH alışverişindeki hataları ve Arayüze aktarılırkenki olan format hatalarını düzelttim. Bundan sonra üzerimdeki yükün kalktığını farkettim, çünkü proje tam olarak düzgün çalışmasa da amacına ulaşmıştı, yani program, bize iskelet kodla verilen filtreler ile çalışır hale geldi ama data kayıpları olduğunu gördüm. Bunun için biraz daha kodlama yapıp Peer kısmını bitirdim. Birçok test ve hataları düzeltmenin ardından sistemin bize verilen 2 filtre ile düzgün bir şekilde teyit ettim ve diğer filtreleri kodlamaya başladım. İskelet koddaki SobelFilter daki hesaplarla küçük oynamalar yapmam, diğer filtreleri kodlamama yetti.

2- Dağıtık görüntü işleme senaryosu kendi projeniz çerçevesinde nasıl gerçekleşti?

2.1- Son kullanıcı

Son kullanıcı, diğer kullanıcılar gibi bir Arayüze sahip. İlk başta Negotiator'a bağlanıp CONNECT POINT LIST alıyor, ondan sonra düzenli bir şekilde Negotiator'dan listeyi isteyip bağlantı listesini güncelliyor. Ama eğer Peer'ın kendisinde bulunan bir bağlantı Negotiator'dan gelen listede yoksa o bağlantıyı direk silmiyor. Belirli aralıklarla kendi listesindeki bağlantıların canlılığını test ediyor.Bir resim koyduğunda ve işlenme isteği geldiğinde, son kullanıcı ilk önce sisteme bağlı olan diğer kullanıcılar ile bağlantı kurmaya çalışıyor, kurduktan sonra PATCH'i başka Peer ın Server ına göndermek ve o Server dan gelen veriyi dinlemek için birer Thread açıyor: SendWorkThread ve GetProcessedThread. Eğer PATCH Peer-Client SendWorkThread den Peer-Server a doğru bir şekilde gitmediyse Peer-Client bu PATCH'i tekrar gönderiyor. Eğer Peer-Server tarafından işlenen PATCH Peer-Client GetProcessedThread e doğru bir şekilde gelmediyse Peer-Client bunu Arayüze iletmiyor, ve tekrar işlemesi için workQueue ya yani iş kuyruğuna tekrar koyuyor. Eğer PATCH doğru bir şekilde geldiyse Peer-Client GetProcessedThread bu patch I Arayüzün kuyruğuna koyuyor ve Peer-Server ile arasındaki bağlantıyı kapatıyor. Yani her yeni bir PATCH'i göndermek için yeni bir bağlantı kurması gerekiyor.

2.2 - Son kullanıcının kullandığı eş

Son kullanıcıda da olduğu gibi ilk başta Negotiator'a bağlanıp CONNECT POINT LIST alıyor, ondan sonra düzenli bir şekilde Negotiator'dan listeyi isteyip bağlantı listesini güncelliyor. Ama eğer Peer'ın kendisinde bulunan bir bağlantı Negotiator'dan gelen listede yoksa o bağlantıyı direk silmiyor. Belirli aralıklarla kendi listesindeki bağlantıların canlılığını test ediyor Kullanılan eş, başka bir Peer dan bağlantı geldiğinde, Negotiator dan gelen bağlantı gibi bir ServerThread açıyor, ve bu ServerThread genel olarak bütün sorguları karşılamaya yetiyor. FUNRQ REGME gibi gelen isteklerin hepsine başka bir isteğe cevap verir gibi ilgili cevabı veriyor. Eğer gelen istek EXERQ ise o zaman Peer-Client SendWorkThread den gelen mesajın eksik gelip gelmediğine bakıyor (Gelen mesaj ın size ı zaten bu soruya cevap veriyor.). Eğer düzgün gelmişse, ServerThread bu PATCH'i işlemek için bir WorkerThread açıyor ve normal socketini dinlemeye devam ediyor. WorkerThread gelen PATCH parçasına uygun filtre uygulayıp Peer-Client GetProcessedThread e gönderiyor. Server kısmının aynı anda en fazla 4 PATCH işleyeceğini bildiğimiz için bunun kontrolunu de yapıyor. Gönderme işlemi bittiğinde WorkerThread kendi kendini kapatıyor, ilgili Server ise ancak CLOSE mesajı geldiğinde kendini kapatıyor.

2.3 – Fonksiyon sağlayan eş

Son kullanıcının kullandığı eş, fonksiyon sağlayan eş olduğundan bkz 2.2

2.4- Arabulucu

Sistem ilk olarak Negotiator'u başlattığımızda başlıyor. Bağlanmak isteyen Peer'ların Peer-Server kısmını test ettikten sonra listeye ekliyor. Düzenli bir şekilde listesindeki Peer'ların online olup olmadığını belirli aralıklarla test ediyor. Negotiator'un görevi zaten sadece Peer'ların kullanarak birbirine bağlanabileceği listeyi göndermek olduğundan bunun dışında başka birşey yapmıyor. Eğer bağlantı testi esnasında bir Peer offline olmuş ise onu listesinden siliyor.

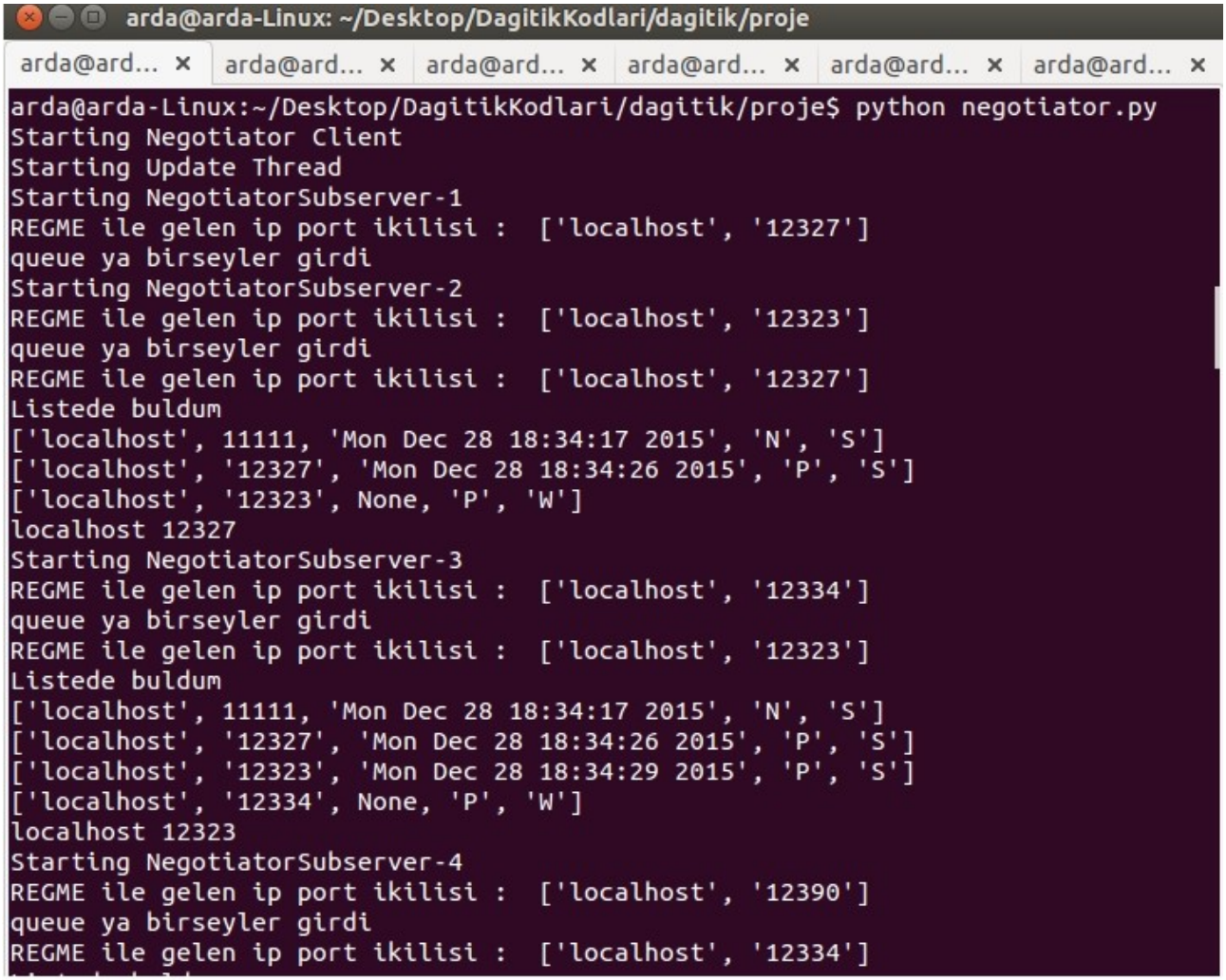
3 – İki tane çözüm bulunmamış problem nedir? Bunları çözmek için ne yapmak gerekirdi?

Çözüm bulamadığım ilk problem, program Arayüzde STOP tuşuna bastığımızda direk durmuyor, birkaç saniye alıyor durması. Yani Arayüz thread'inin workQueue ya koyduğu tüm parçalar işlendikten sonra duruyor.(STOP a bastıktan sonra 3-4 tane daha PATCH işleniyor, resimde güncellenip yerine konuyor ve bundan sonra duruyor.) Arayüz kodundaki STOP un ilişkilendirildiği kısıma bir tane Queue koyup, bu Queue'ya STOP basıldığına bir eleman ekleyip, Peer-Client GetProcessedThread de bu Queue'yu düzenli okuyup STOP tuşuna basılır basılmaz anlayıp işlenen PATCH I resime koymayı kesebilirdim, ama düzenli bir şekilde Queue'yu dinlemem bana başka bir thread yazmamı gerektirdiğini düşündüğümden bunu implemente edemedim.
Çözüm bulamadığım ikinci problem ise CONNECT POINT LIST'den aldığım Peerların Time kısmını string olarak okuduktan sonra tekrar asctime formatına dönüştürmekti. Python documentation da yapılan her şeyi doğru yapmama rağmen, Arayüzü çalıştırmazken işleyen conversion fonksiyonu **time.asctime(time.strptime(actTime, "%a %b %d %H:%M:%S %Y"))** , Arayüz çalıştığında format hatası vermeye başlıyordu, ve bu da benim time ları karşılaştırırken gereksiz zorlanmama neden olacaktı. O yüzden her bağlantının listedeki zamanının belirli bir zaman geçip geçmediğini anlamak yerine Peer'a da liste bağlantılarını düzenlikli aralıklarla test eden bir thread yazdım. Yani her Peer, belirlenen zaman aralığıyla CONNECT POINT listesindeki kayıtlı peerlara HELLO atıp karşılığında SALUT bekliyor hale geldi.

Uygulama protokolüne değişiklik yapmak isteseydiniz ne eklerdiniz?

Protokol değişikliğinden önce belirtmek isterim ki socket.recv() fonksiyonunda içerisine yazılacak byte sayısı çok önemli. Ben genel olarak 1024 kullandığımdan(PATCH iletilmesi gibi durumlar hariç), 1024 byte ı aşan veri alışverişinde sıkıntı çıkabilir kodda.(Aynı anda 20-30 peer bağlandığında GETNL prokolüne giden cevap 1024 byte dan daha büyük olacak, 1024 sayısını değiştirmek gerekecek) Bu sayı proje dokumanında belirtilebilirdi.
İlk olarak FUNLI protokolünü kaldırırdım, çünkü Proje dokumanından anladığım kadarıyla hiçbir Peer bu fonksiyonu kullanmıyor, çünkü zaten 1 Peer kendi arayüzünde aynı anda 1 resimi 1 filtre ile işleyeceğinden, FUNRQ prokolünü kullanmak yetiyor.
İkincisi, proje dokümanında sanki NLIST BEGIN bir stream olarak aktarılıyor ama satır araları \n ile biter denildiğinden ben bunu sanki bütün listeyi 1 send ile göndericeğimizi anladım ve öyle yaptım. Socket in 1 Send inde NLIST BEGIN <PEER LISTESI> NLIST END olarak gönderdim. Projede dokümanı bu konuda hangi türde gönderileceği hakkında daha açık olabilirdi.(her satır... cümlesi işleri karıştırmaya yetti)
Üçüncüsü, Negotiator kısmında Peer listesinin sonuna “W” veya “S” eklemek çok mantıklı geliyor, ama bir Peer kendi listesi için Negotiator'dan liste istediğinde gelen elemanları nasıl etiketleyeceği, bu durumu Peer için mantıksız bir hale getirdi. Yani Negotiator dan aldığı her Peer ı tek tek test mi etmesi lazım? Yoksa direk “S” olarak mı ekleyecek tam anlayamadım. Ben kodumda hem “S” olarak yazıyorum, hem de belirli aralıklarla listedeki Peerları test ediyorum.
Son olarak EXERQ protokolündeki PATCH gönderme kısmı değiştirilebilir, çünkü zaten biz o protokolü gönderirken 65000 gibi bir byte boyutu olan bir dosya gönderiyoruz. Bunun yerine bir stream gibi EXERQ gönderilip sonra 1024 veya 2048 byte lık paketler ile resim dosyası gönderilebilir ve sonuna da PCFIN gibi bir patch gönderimi bitti protokolü gönderebilirdik. Ben direk o büyük boyuttaki dosyayı gönderip mesajın doğru ulaşıp ulaşmadığının kontrolünü yapıyorum kodumda.

NEGOTIATOR BEGIN



PEER'LAR KAPATILDIĞINDA:

```
['localhost', 11111, 'Sun Dec 27 23:52:35 2015', 'N', 'S']
['localhost', '12320', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12321', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12323', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12334', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12390', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
localhost 12320
Starting NegotiatorSubserver-60
Server  NegotiatorSubserver-60 closing
['localhost', 11111, 'Sun Dec 27 23:52:35 2015', 'N', 'S']
['localhost', '12320', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12321', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12323', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12334', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
['localhost', '12390', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
localhost 12390
siliyorum sunu  ['localhost', '12321', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
siliyorum sunu  ['localhost', '12323', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
siliyorum sunu  ['localhost', '12334', 'Sun Dec 27 23:56:50 2015', 'P', 'S']
siliyorum sunu  ['localhost', '12320', 'Sun Dec 27 23:57:00 2015', 'P', 'S']
siliyorum sunu  ['localhost', '12390', 'Sun Dec 27 23:57:00 2015', 'P', 'S']
```

PEER sisteme katıldığında:

```
arda@arda-Linux:~/Desktop/DagitikKodlari/dagitik/proje$ python peer.py localhost
 23452
Starting MainThread
Starting Client
Starting TesterThread
Starting UpdateThread
starting  GuiListenerPeer server side waiting connection

Starting Server Thread
###Servera gelen dataPeer server side waiting connection
 HELLO $$$
###Servera gelen data CLOSE $$$
Kapaniyor  Server Thread
Starting Server Thread Peer server side waiting connection

###Servera gelen data HELLO $$$
###Servera gelen data CLOSE $$$
Kapaniyor  Server Thread
REGOK GELDI
CPL e eklenen :  localhost:11111:Mon Dec 28 18:48:33 2015:N
CPL e eklenen :  ['localhost', '11111', 'Mon Dec 28 18:48:33 2015', 'N', 'S']
CPL e eklenen :  ['localhost', '12345', 'Mon Dec 28 18:51:33 2015', 'P', 'S']
Starting Server Thread
 Peer server side waiting connection###Servera gelen data
HELLO $$$
###Servera gelen data CLOSE $$$
```

Peer PATCH işletmek için diğer Peer'ları arayıp bağlantı kurup gönderdiğinde:

```
Image loaded: /home/arda/Desktop/sAGBIj5.jpg
Ekran flag koydu
Is geldi
Baglanacak peer buldum ['localhost', '55434', 'Mon Dec 28 18:41:57 2015', 'P', '
S']
baglanti sagladim
Sender Thread calisiyor
calisacagim data buyuklugu 16384
Starting Getter Thread
 son calistigim data:  108
gonderilecek mesaj basi EXERQ GrayScale:128:384:640:103,108,117,112,104,10 buyuk
: 55474
REGME gonderiyorum
PeerServer - PeerClient gelen data : REGOK Mon Dec 2
Baglanacak peer buldum ['localhost', '12345', 'Mon Dec 28 18:41:57 2015', 'P', '
S']
baglanti sagladim
Sender Thread calisiyor
calisacagim data buyuklugu 16384
Starting Getter Thread
son calistigim data:  129
gonderilecek mesaj basi EXERQ GrayScale:128:1280:128:114,117,117,117,114,1 buyuk
: 65562
REGME gonderiyorum
PeerServer - PeerClient gelen data : REGOK Mon Dec 2
FUNRQ gonderiyorum
Baglanacak peer buldum ['localhost', '12345', 'Mon Dec 28 18:41:57 2015', 'P', '
S']
 PeerServer - PeerClient gelen data : FUNYS GrayScale
baglanti sagladim
```

Resim işlemek için bağlantı alan peer:

```
Starting Server Thread
 Peer server side waiting connection
###Servera gelen data REGME localhost:23456 $$$
REGME ile gelen ip port ikilisi :  ['localhost', '23456']
Listede buldum
###Servera gelen data FUNRQ GrayScale $$$
['localhost', '11111', 'Mon Dec 28 18:41:54 2015', 'N', 'S']
['localhost', '12345', 'Mon Dec 28 18:41:54 2015', 'P', 'S']
['localhost', '55434', 'Mon Dec 28 18:41:18 2015', 'P', 'S']
['localhost', '23456', 'Mon Dec 28 18:41:57 2015', 'P', 'S']
localhost 23456
Aradigim func :  GrayScale elimdeki func :  ['GrayScale']
patch parcasi geliyo 55474
EXERQ ile gelen datanin buyuklugu :  55446
worker thread aciliyor
Server Worker Thread0: Starting.
Peer server side waiting connection
Starting Server Thread
Worker thread isi bitti, kapaniyor
###Servera gelen data REGME localhost:23456 $$$
REGME ile gelen ip port ikilisi :  ['localhost', '23456']
Listede buldum
###Servera gelen data PATYS 640:384 $$$
PATYS geldi
Peer server side waiting connection
Starting Server Thread
###Servera gelen data REGME localhost:23456 $$$
REGME ile gelen ip port ikilisi :  ['localhost', '23456']
Listede buldum
###Servera gelen data FUNRQ GrayScale $$$
```