RAPOR

Projenin başında nasıl bir yol izlenmesi gerektiği konusunda düşünüldü ve öncelikle ip.py kısmında grayscale(self, header, patch, threshold) fonksiyonu yazılarak kod yazma aşaması start edildi. Daha sonra negotiator_server, main kısmında socket oluşturularak bağlantı kuruldu ardından pcfunctionList = {} ve pcQueue tanımlandı ve akabinde pcCreateThread(n), pcPrint_time fonksiyonları yazıldı bu fonksiyonlarla öncelikle Thread üretimi ve Lock mekanizmaları sağlandı pcCreateThread(n) fonksiyonuyla aynı zamanda ip ve port numaraları random olarak oluşturuldu. Bu fonksiyonlar her çalıştırıldığında THREADNUM değişkeninin sayısı kadar ReedThread ve WriteThread oluşturuludu başlangıçta thread sayısı 4 olarak atandı.

Aynı zamanda CONNECT_POINT_LIST = {} yapısıyla kayıt olmuş olan bütün peerların bilgilerinin tutulduğu yapı oluşturuldu. Negotiator_server kısmında rThread (threading.Thread) threadi üzerinden protokolde istenen aşamalar adım adım takip edilerek yazıldı.

DataList = [] adında global bir dizi tanımlandı daha sonra ReedThread içerisinde protokol kurallarınca tanımlanan parser adlı fonksiyonda gelen dataların ilk beş terimine göre bu listeye eleman eklendi ve daha sonra bu liste WriteThread içerisinde kullanılarak gelen isteğe cevap verilmesi gereken durumlarda cevap dönüldü.

Negotiator_client yapısı oluşturulurken yine ReadThread, WriteThread, Parser ve Queue yapıları kullanılarak protokolde istenen şartlar ışığında ReadThread içerisinde data parserden geçirilerek eğer bir cevap verilmesi gerekiyorsa bu cevap queue'ya yazdırılıp sonrada WriteThread üzerinden bu bilgiler queue'den okunarak cevap olarak gönderildi. Örneğin peer negotiator dan kayıtlı olan eş listesini istediğinde CONNECT_POINT_LIST = {} üzerinden işlem yapılarak liste protokolde istendiği gibi cevap olarak geriveriliyor. Bu işlem peer_server ve peer_client tarafında da aynı şekilde gelen datalar protokolde verilen bilgiler ışığında parçalanıp tanımlı değişkenlere atanarak daha sonra kendi içerisinde tanımlı function ve kişi listelerine eklenerek sonlandırıldı ve yine burada da ReadThread, WriteThread, Queue ve Tupl lar kullanılarak işlemler negotiator daki genel işleyiş tarzında gerçekleştirildi.

Dağıtık görüntü işleme senaryosu son kullanıcı açısından ve son kullanıcının kullandığı eş açısından bakılacak olursa son kullanıcı protokolde belirtilen adımlar ışığında istek gondererek kayıt olup login olduktan sonra gondermek istediği fotoyu karşıya gondermeden hangi fonksiyonu kullanacağını sorgulayarak var olup olmadığını öğrenir ve daha sonra foto parçasını yollar sonunda bu parçayı değiştirilmiş, işlenmiş olarak geri alır.

Fonksiyonu sağlayan eş açısından, bir foksiyon sorgusu yapıldığında fonksiyonun olup olmadığı fonksiyon listesine bakılarak öğrenilir ve istekte bulunan eşe cevap verilir bir parça gönderildiğinde bu parça fonkiyonda işlenir fonksiyon sonucu bir değişkene atanarak geri cevap olarak dönülür.

Arabulucu bir eş kayıt olmak için istekte bulunduğunda bu isteği işleme alır ve işlem bitiminde protokolde belirtildiği gibi eş bilgilendirilir ve eş bilgileri tupl da tutulur.

Çözüm bulamadığım problemler test aşaması ve bonus olarak verilen kısımlar, çünkü projeyi düzgün test edemedim.

Uygulama protokolünde değişiklik yapmak isteseydim negotiator_istemcinin yaptığı Hello ve Close isteklerini kaldırıp peer_istemcinin bunu negotiator_sunucuya istek olarak göndermesini sağlardım.

SADDAM TANRIVERDİ-11401593