PROJE ÖN TASARIMI:

PROTOKOL GENEL KURALLARI:

- Protokol genel olarak http ve IRC standartlarından esinlenmiştir.
- Sistemde iki çeşit bağlantı vardır: PEER-PEER ve NEGOTIATOR-PEER.
- PEER-PEER bağlantısı sayesinde eşler veri paylaşımında bulunabilecekler.
- NEGOTIATOR-PEER bağlantısının amacı ise PEER buluşturma olacaktır.
- Uç noktalar arası aktarım request-response paradigmasına uymalıdır. Yani, bir uç nokta istekte (request) bulunacak ve bu isteğe karşılık öteki uç nokta bu isteğe olumlu veya olumsuz cevap verecektir.
- Bir uç nokta talepte bulunduğu zaman ona cevabın gelmesini beklemek zorundadır. Cevap ona gelene kadar da bloke olacaktır. Arka arkaya mesaj yollayamaz.
- Merkez sunucunun (NEGOTIATOR) çevrimiçi, çevrimdışı, busy gibi status leri yoktur. Ama eşlerin status leri vardır.
- Protokol P2Pdir. Yani eşler üzerinde hem sunucu hem de istemci mevcut olacaktır.
- Taleplerin doğru response larla eşleştiğini garantilemek için opr_id (işlem id) isimli bir kod mesajların body sine eklenecektir.
- IP adresleri genel olarak sistemin elemanları arasında paylaşılmamaktadır. Sadece istemciler kendi contactlarının IP'sini paylaşırlar P2P iletişime geçebilmek için.
- Her eş büyük sisteme dâhil olabilmek için öncelikle NEGOTIATOR'a bağlanacaktır.
- NEGOTIATOR üzerinden görüntü verisi alışverişi yapılmayacaktır.

1. NEGOTIATOR - PEER PROTOCOL:

NEGOTIATOR-PEER protokolü sayesinde eşler kendilerini sunucuya kayıt ederler. Yine bu protokol sayesinde eşler kendi kimlik doğrulamalarını yaparlar (authentication). Bunun dışında bu protokol sayesinde eşler kendilerine bağlı olan eşlerin bağlantı adresleri bilgisini elde ederler. NEGOTIATOR veritabanının üzerinden bilgileri ancak bu protokol sayesinde edinebilirler. Bir eş başka bir eş ile bağlantı kurmak istediği zaman yine bu işlem NEGOTIATOR üzerinden gerçekleşmektedir. Eşler ayrıca herhangi bir bilgi değişikliği için yine NEGOTIATOR'a başvururlar. NEGOTIATOR-PEER protokol kesinlikle soru-cevap seklindedir (senkron). Hem eşler NEGOTIATOR a istekte bulunabilir, hem de NEGOTIATOR eşe sorguda bulunabilir. Yalnız sorgu yapmadığı sürece NEGOTIATOR bir şey söylemeyecektir. Yapılan eş isteklerine göre de NEGOTIATOR cevap verir. Eşlerin NEGOTIATOR'dan asenkron cevaplar beklemesine gerek yoktur.

SUNUCU-EŞ BAĞLANTISINDA EŞ SORGULARI: (*Peer Requests in Negotiator-Peer connection*)

PRPULSE: (*Peer Pulse*)

Peer Pulse dediğimiz bu mesaj PEER_CLIENT 'tan NEGOCIATOR a yollanan bir bildirimdir. Bu bildirimin amacı bu mesajı yollayan eşin hâlihazırda çevrimiçi olduğunu ve işlem görmek istediğini iletir. Belirli bir süre zarfı içerisinde eşlerimizin her birinden bu mesajın sunucuya gelmesi beklenmektedir. (bu süre 120 saniyedir). Eğer 5 dakika (300 saniye) boyunca bu bildirim sunucuya gelmezse sunucu bu eşi çevrimdışı kabul eder ve listesinden bu eşi çıkarır. Her PRPULSE mesajı için sunucu NEGOK (ok mesajı) veya NEGERR (hata mesajı) cevabı vermektedir.

Girilen komut: PRPULSE

Format (PEER_CLIENT tarafı) : PRPULSE

Sunucu tarafından gönderilen cevap: NEGOK veya NEGERR. NEGERR bağlantının başarısız olması durumunda yollanacaktır. NEGOK mesajında ise NEGOK ’un yanında request yapan eşin ismi (ID), IP si ve durumu (status) verilecektir. (<komut> , <user_name> , <user_IP>, <status>).

Örnek cevap formatı:

- (NEGOK, Şakir, 212.36.37.38, online) Bu bir tuple şeklinde gönderilebilir.
- (NEGERR , <hata_sebebi>)

(*PRCONN da da bu sorguyu kullanıyoruz*)

PRREGIS : (*Peer Register*)

PRREGIS eşin büyük sisteme dâhil olabilmek için NEGOTIATOR’a istekte bulunduğu bir komuttur. Bu request sayesinde aday olan eş sisteme girebilir veya giremeyebilir. Sunucu eşi kabul etmeden önce mesajın formatına bakar. Mesaj formatı şöyle olmalıdır:

(<PRREGIS>, <user_name>, <user_IP>). Sunucu kendi CONNECT_POINT_LIST'ine bakacaktır ve eğer kayıt olmak isteyen eş bu listede mevcut değilse ve yollanan mesaj doğru formatta atılmışsa sunucu bu yeni eşi sisteme dâhil edecektir.

Karşılığında yine NEGOK veya NEGERR mesajı yollayacaktır. NEGOK mesajı döndürdüğünde, NEGOK un yanına Merhaba ve kullanıcı ismini de yazacaktır. NEGERR durumunda ise neden eşin kabul edilmediğini sisteme yazacaktır NEGERR in yanında.

Girilen komut: PRREGIS

Format (PEER_CLIENT tarafı) : (PRREGIS, <user_name>, <ip_no>)

Sunucu tarafından gönderilen cevap: (<NEGOK> , <Merhaba> , <user_name>) veya (<NEGERR>, <reason>)

Örnek cevap formatları:

- (NEGOK, Merhaba, veli_34)
- (NEGERR, böyle bir kullanıcı zaten sistemde mevcut)
- (NEGERR, Yanlış format)

PRCLOSE: (*Client End Connection Message)*

PRCLOSE ile eş sistemden çıkmak istediğini belirtir. Bu mesajı alan sunucu yine diğer durumlarda olduğu gibi ya NEGOK ya da NEGERR ile cevap verecektir. Mesaj yanlış formatta atıldıysa, sunucu buna NEGERR ile cevap verecektir aksi halde NEGOK ile cevap verecektir.

Girilen komut: PRCLOSE

Format (PEER_CLIENT tarafı): PRCLOSE

Sunucu tarafından gönderilen cevap: (<NEGOK> , <Hoşçakal>, <user_name>) veya (<NEGERR> , <reason>)

Örnek cevap formatları:

- (NEGOK, Hosçakal, ahmet_dursun)
- (NEGERR, yanlış format)

PREDIT : (*Client Profile Edit)*

PREDIT sayesinde ilgili eş sunucuya profil bilgilerini değiştirmek istediğini iletir. PREDIT ile istekte bulunan eş içinde değişiklik yapılmış mesajı sunucuya yollar. Sunucu NEGOK veya NEGERR ile yine cevap verecektir bu durumda. Mesela yukarıda örneğini verdiğimiz 212.36.37.38 IP'li Şakir adlı kullanıcı örneğini verelim. Mesaj aynen şöyle gelecek:

---- (PREDIT, hakki_92, 212.36.37.38) ---- gördüğünüz gibi burada Şakir ismi geçmiyor, eş direkt olarak değişikliği yapıyor ve öyle sunucuya yolluyor. Sunucu IP'yi kontrol eder kendi listesi üzerinde ve eğer böyle bir IP varsa değişikliği onaylar ve NEGOK mesajını yollar.

Girilen komut: PREDIT

Format (PEER_CLIENT tarafı): (PREDIT , <new_user_name> , <IP_no>)

Sunucu tarafından gönderilen cevap: (<NEGOK> , <user_name_modified>) veya (<NEGERR> , <reason>)

Örnek cevap formatları:

- (NEGOK, hakki_92)
- (NEGERR, 212.26.37.38 ipli bir kullanıcı mevcut değildir).

PRFIND : (*Client Find User)*

Client bu sorguyu sunucuya gönderdiği zaman, sunucu client a kayıtlı kullanıcıların listesini döndürür. Client burada PRFIND komutunun yanında anahtar kelime ekler. Sunucu yine NEGOK komutunu döndürür, NEGOK komutunun yanında o keyword ü içeren bütün kullanıcıların listesini de gönderir. Listede o kullanıcıların bilgilerini de gönderir.

Girilen komut: PRFIND

Format (PEER_CLIENT tarafı): (PRFIND , <keyword>)

Sunucu tarafından gönderilen cevap: (<NEGOK> , <user_name_list_with_information>) veya (<NEGERR> , <reason>)

Örnek request - cevap formatları:

- (PRFIND , "met")
- (NEGOK ; Ahmet, information about Ahmet ; Samet, information about Samet ; Mehmet, information about Mehmet)

PRCONN: (*Client Contact Request*)

Bu komut ile bir eş başka bir eşe bağlanmak istediğini sunucuya iletir. Server bu isteği görünce kendi veritabanında (burada contact_list oluyor) bir tarama yapar ve bağlanılmak istenilen kullanıcıyı bulur. Hedef kullanıcıyı bulduğu zaman NEGOK cevabını bu mesajı gönderen verici kullanıcıya gönderir (yani ilk kullanıcı). Hedef kullanıcı eğer PRPULSE sorgusu atarsa sunucuya, o zaman sunucu NEGOK komutunun yanında Contact requesti de atar. Bunun karşılığında hedef kullanıcı bu requesti kabul eder veya etmez. Hedef kullanıcının bu kabul veya ret cevabı PRRESP dediğimiz dediğimiz komutla olacaktır. (altta)

Girilen komut: PRCONN

Format (PEER_CLIENT tarafı): (PRCONN , <bağlantı_kurulmak_istenen_user>)

Sunucu tarafından gönderilen cevap: (<NEGOK> , <kullanıcı bulunmuştur>) veya (<NEGERR> , <reason>) (bu bağlantı kurmak isteyen kullanıcıya gönderilir)

*** Eğer hedef kullanıcı PRPULSE atmışsa → (<NEGOK> , <bağlantı_talebi> , <bağlantıyı_talep_eden_kullanıcı>)

Örnek request - cevap formatları:

- (PRCONN , Ahmet) (burada mesela Şakir adlı kullanıcının bu bağlantıyı talep ettiğini varsayalım)
- (NEGOK , bağlantı talebi , Şakir)
- (NEGERR , hata)

PRRESP: (*Client Contact Request – Response*):

Bir kullanıcı sunucu üzerinden bir contact request mesajı alırsa bunun karşılığında PRRESP mesajı yayınlar. PRRESP mesajının içinde PRRESP komutu ve bu komutun yanında ACCEPT veya DENY durumlarını ekler. Sunucu bunun karşılığında yine NEGOK veya NEGERR mesajını döner kullanıcıya.

Girilen komut: PRRESP

Format (PEER_CLIENT tarafı): (PRRESP , <ACCEPT> or <DENY>)

Sunucu tarafından gönderilen cevap: (<NEGOK> , <bağlantı kurulmuştur>) veya (<NEGERR> , <reason>) (Bu cevap hem bağlantıyı kabul eden eşe hem de bağlantı kurmaya çalışan eşe gönderilmektedir).

Örnek request - cevap formatları:

- (PRCONN , Ahmet)
- (NEGOK , bağlantı kurulmuştur)
- (NEGERR , hata)

SUNUCU-EŞ BAĞLANTISINDA SUNUCU CEVAPLARI : (*Server Responses in Negotiator-Peer connection*)

Yukarıda da görmüş olduğunuz gibi NEGOTIATOR iki türlü cevap vermektedir: NEGOK ve NEGERR. Bunun yanında bir de NEGCON dediğimiz bir komutumuz vardır. Bu komutu da aşağıda anlatacağım. NEGOK ve NEGERR'ler yapılan request lere göre NEGOK ve NEGERR'in yanında ekstra bilgiler de içerebilir. Bunların örneklerini aşağıda verelim:

- **PRPULSE REQUEST:**
  Burada sunucu NEGOK un yanında requesti yapan kullanıcının ismini, ipsini ve statusunu de gönderir.
  Daha önceden bir PRCONN requesti olmuşsa PRPULSE yapan kullanıcıya o zaman NEGOK un yanında bağlantı talebi stringini ve bağlantıyı talep eden kullanıcının ismini de gönderir.
- **PRREGIS REQUEST:**
  Burada sunucu NEGOK'un yanında Merhaba stringini ve kayıt olan kullanıcının ismini de gönderir.
- **PRCLOSE REQUEST:**
  Burada sunucu NEGOK'un yanında Hoşçakal stringini ve sistemden çıkmak isteyen kullanıcının ismini de gönderir.
- **PREDIT REQUEST:**
  Burada sunucu NEGOK'un yanında yenilenmiş kullanıcı ismini de gönderir.
- **PRFIND REQUEST:**
  Burada sunucu NEGOK'un yanında kayıtlı bütün kullanıcıların ismini ve bilgilerini gösterir.
- **PRCONN REQUEST:**
  Burada sunucu NEGOK'un yanında "kullanıcı bulunmuştur" mesajını da gönderir.
- **PRRESP REQUEST:**
  Burada sunucu NEGOK'un yanında "bağlantı kurulmuştur" mesajını da gönderir.

NEGERR komutuyla beraber de hata çıktığını veya ne hatası olduğu bilgisini de ekler.

NEGCON REQUEST:

Sunucumuzun tek yaptığı sorgu budur. Bu sorgu ile sunucu kayıt olan eşe eş gibi bağlanmaya çalışıp [PEER_IP, PEER_PORT] ikilisinde bir problem olup olmadığını kontrol eder.

Girilen komut: NEGCON

Format (NEGOTIATOR tarafı): (NEGCON, PEER_IP, PEER_PORT)

Eş tarafından gönderilen cevap: (<PROKK> , <bağlantı kurulmuştur>) veya (<PRERR> , <reason>)

Örnek request - cevap formatları:

- (NEGCON , 234.45.45, 8081)
- (PROKK, bağlantı kurulmuştur)
- (PRERR , hata_sebebi)

*** PRERR ve PROKK eş tarafında merkez sunucuya gönderilen tek cevaplardır.

2. PEER - PEER PROTOCOL:

Yukarıda da belirttiğimiz gibi bütün görüntü işlemleri ve görüntü alışverişleri bu protokol aracılığıyla yapılacaktır. Her eşin hem sunucusu hem de istemcisi olacaktır. Bu yüzden P2P request-response bir protokol diyebiliriz bu protokol için. Her eşin bir fonksiyon havuzu olacaktır (FUNCTIONS). Bir eş bir diğer eş üzerindeki fonksiyonları sorgulayabilecektir. Gönderilmek istenen görüntüler parça parça gönderilecektir (CHUNCK). Eşler FUNCTIONS listesi sorgusu yapabilecekler. Eşler PARAMETERS sorgusu yapabilecekler. Eşler CONNECT_POINT_LIST sorgusu yapabileceklerdir. Yani eş listesi bilgisi sadece sunucudan değil başka eşlerden de sağlanabilecektir. Eşler EXECUTE isteği yapabileceklerdir.

EŞ-EŞ SORGULARI: (*Peer-Peer Requests*)

PRMESS : (Peer Message Request)

Bu komut sayesinde bir eş başka bir eşe mesaj yollayabilir. Bu komutun kullanılabilmesi için ancak ve ancak mesaj yollamak isteyen eşin daha önceden merkez sunucuya PRCONN ile request yapmış olup bağlantı kurmak istediği kullanıcının da bunu kabul etmiş olması gerekir. Mesaj iletilen eş bu mesaj karşılığında PRACK (Peer Message Acknowledge) veya PRREJ (Peer Message Rejection) mesajı yollar duruma göre.

Girilen komut: PRMESS

Format (PEER_CLIENT tarafı): (PRMESS , <mesaj_içeriği>, <eşin_ismi>)

Öteki eş tarafından gönderilen cevap: (<PRACK> , <mesaja_cevap>) veya (<NEGERR> , <reason>)

Örnek request - cevap formatları:

- (PRMESS , <Nasılsın ?> , <Ahmet>)
- (PRACK , iyiyim)
- (PRREJ , hata)

PRFUNC : (Peer Function Request)

Bu komut sayesinde bir eş kendi havuzunda olmayan bir fonksiyonu başka bir eşte sorgulayabilir. Eğer varsa, bu fonksiyonu talep edebilir. Bunun karşılığında istekte bulunulan eş eğer fonksiyon kendi

havuzunda mevcutsa isteyen eşe fonksiyonu gönderir PRACK ile. Eğer bu fonksiyon kendi havuzunda mevcut değilse, PRREJ ile böyle bir fonksiyonun kendisinde bulunmadığını belirtir.

Girilen komut: PRFUNC

Format (PEER_CLIENT tarafı): (PRFUNC , <fonksiyonun_ismi>, <sorgulanan_eşin_ismi>)

Öteki eş tarafından gönderilen cevap: (<PRACK> , <fonksiyon>) veya (<PRREJ> , <reason>)

Örnek request - cevap formatları:

- (PRMESS , <concatenate()> , <Erdem>)
- (PRACK , <concatenate()>)
- (PRREJ , böyle bir fonksiyon bulunmamaktadır)

PRFUNL : (Peer Function List Request)

Bu komut sayesinde bir eş başka bir eşin fonksiyon listesine bakabilir. PRFUNL yazıp eşe gönderildiği zaman eş eğer bir hata yoksa, kendi fonksiyon listesini sorguyu atan eşe gönderir.

Girilen komut: PRFUNL

Format (PEER_CLIENT tarafı): (PRFUNL , <sorgulanan_eşin_ismi>)

Öteki eş tarafından gönderilen cevap: (<PRACK> , <fonksiyon_list>) veya (<PRREJ> , <reason>)

Örnek request - cevap formatları:

- (PRMESS , <Erdem>)
- (PRACK , <concatenate()>;<range()>;<palindrome()>….)
- (PRREJ , hata var)

PRPARA : (Peer Parameters Request)

Bu komut sayesinde bir eş başka bir eşten parametre talep edebilir. PRPARA yazıp eşe gönderildiği zaman eş eğer bir hata yoksa, istenen parametreyi sorguyu atan eşe gönderir.

Girilen komut: PRPARA

Format (PEER_CLIENT tarafı): (PRPARA , <sorgulanan_eşin_ismi> , <istenen_parametre>)

Öteki eş tarafından gönderilen cevap: (<PRACK> , <istenen_parametre>) veya (<PRREJ> , <reason>)

Örnek request - cevap formatları:

- (PRPARA , <Erdem>, <parametre>)
- (PRACK , <parametre>)
- (PRREJ, hata var)