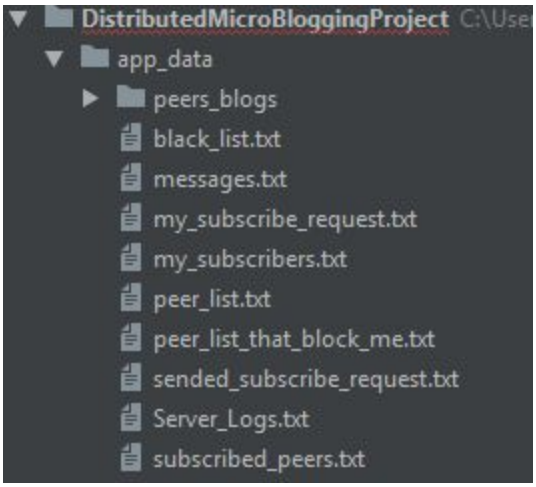
Katia Merdinoğlu, Mustafa Dağdelen,
Tuba Arat, Derya Turan

INF 443: Dağıtık Sistemler ve Uygulamalar Dönem Sonu Projesi Raporu

Dönem sonu projemiz olan dağıtık mikro blog tabanlı sosyal ağ tasarımımız için biz ilk öncelikle aracı yerine yayıncıyı yazmaya karar verdik. Yayıncı kodunun içerisinde tutmak istediğimiz bazı bilgilerin durduğu dosyalar mevcut. Bunlar :



Black list dosyası engellenen peer'lerin biligilerinin, messages dosyası kullanıcıya gelen özel mesajların tutulduğu, my_subscribe_request kullanıcıya gelen abonelik isteklerini, my_subscirbers kullanıcıya abone olan peer'lerin listesini, peer_list_that_block_me dosyası kullanıcı engelleyen peer'lerin listesini, sended_subscribe_request yollanan abonelik isteklerini, server_logs genel sistem loglarının tutulduğu ve subscribe_peers ise abone olunan peer'lerin listesinin tutulduğu dosya.

Sistemin genel yapı taşları:

İlk başta kullanıcının ip'si get_ip() fonksiyonuyla sistemden alınıyor daha sonra uuid olarak de mac adresini veriyoruz get_mac() fonksiyonuyla ve port belirliyoruz.

```
my_ip = get_ip()
print(my_ip)
my_port = 12344
my_username = str(get_mac())
```

Daha sonra public private key çiftleri create_rsa_pair fonksyonu ile RSA kullanılarak oluşturuluyor daha sonra soketten dinleme yapılıyor ve peer keys diye bir dosyanın içerisine anahtarların yazılacağı txt dosyaları oluşturuluyor.

Bağlantı geldiğinde sırasıyla logger_thread, new_peer_thread, server_thread ve qt_and_client çalışmaya başlıyor.

```
logger_thread = LoggerThread(logger_queue).start()

new_peer_thread = New_Peer_Thread(peer_list, my_ip, my_port, my_username, my_type, my_hash)
new_peer_thread.start()

server_thread = ServerThread(s, my_username, my_ip, my_port, my_hash, my_type, connections, logger_queue,
                             peer_list, my_subscribers, black_list,
                             my_subscribe_request, sended_subscribe_request, subscribed_peers,
                             peer_list_that_block_me, all_messages)
server_thread.start()

app = QtWidgets.QApplication(sys.argv)
qt_and_client = QtSideAndClient(connections, logger_queue, peer_list, my_ip, my_port,
                                my_username, my_type, my_subscribers, my_subscribe_request,
                                subscribed_peers, black_list,
                                sended_subscribe_request, peer_list_that_block_me, my_hash, message_list,
                                my_blogs, all_messages)
qt_and_client.run()
```

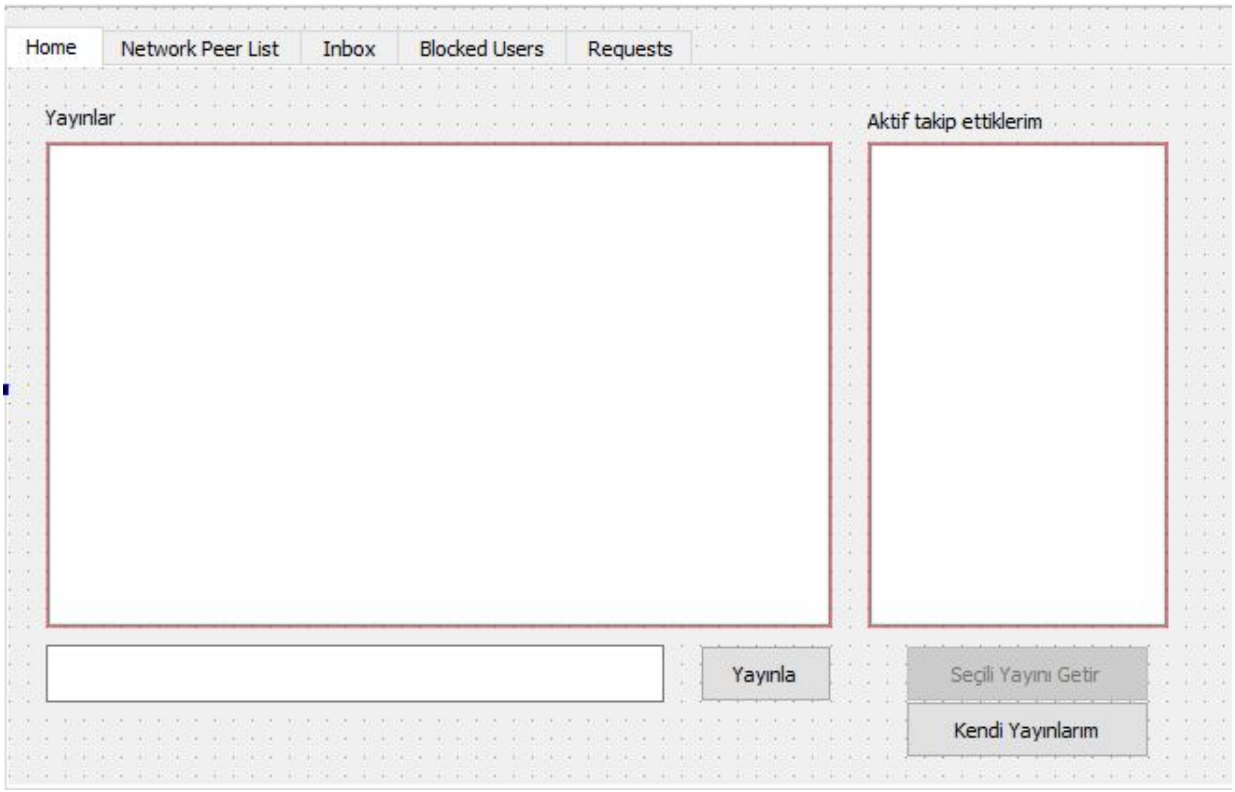
Qt and client sınıfı arayüz thread'ini oluşturuyor örneğin bir düğmenin ne yapması gerektiğini söyleyen fonksiyon bu thread tarafından çalıştırılıyor.

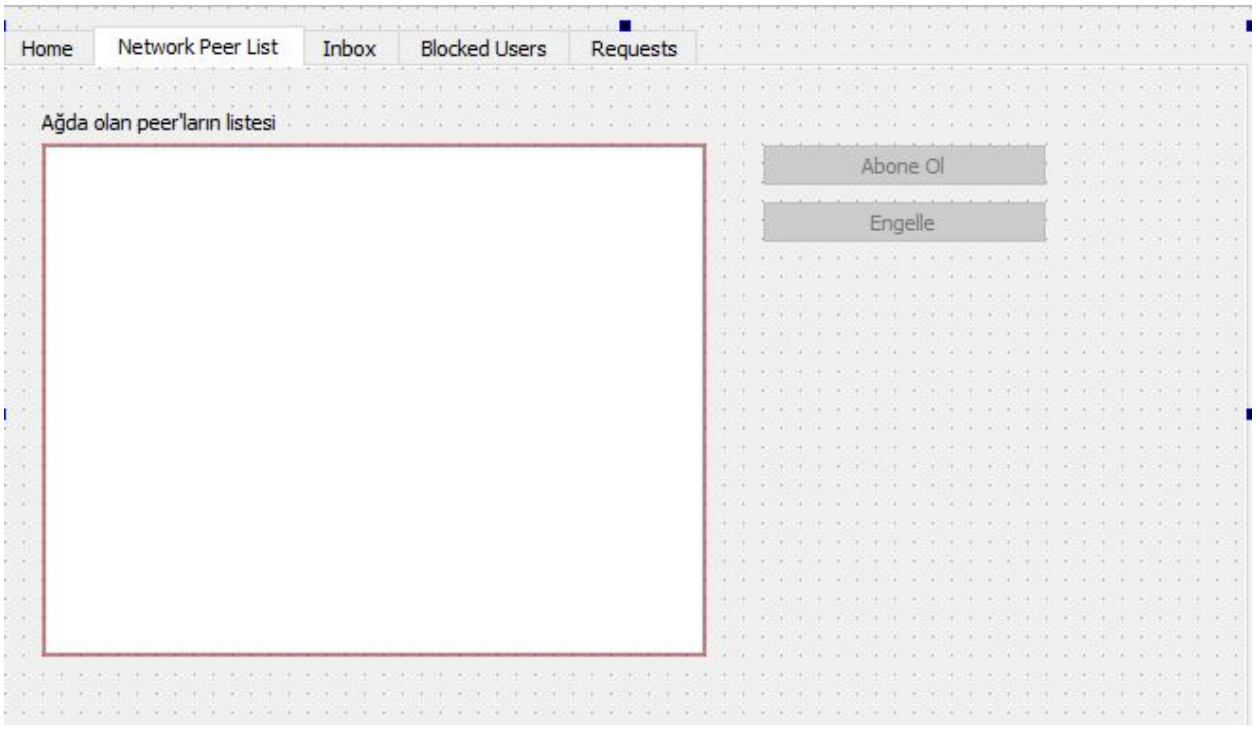
Bunlar dışında ayrıca reader ve writer thread sınıfları da mevcut, reader sokete yazılan string'i okuyup parse edip gelen protokol mesajı neyse ona göre gerekli kontrolleri yapıp gelen protokol mesajına tekabül eden protokol cevabını veriyor, writer thread ise cevabı sokete yazıyor.

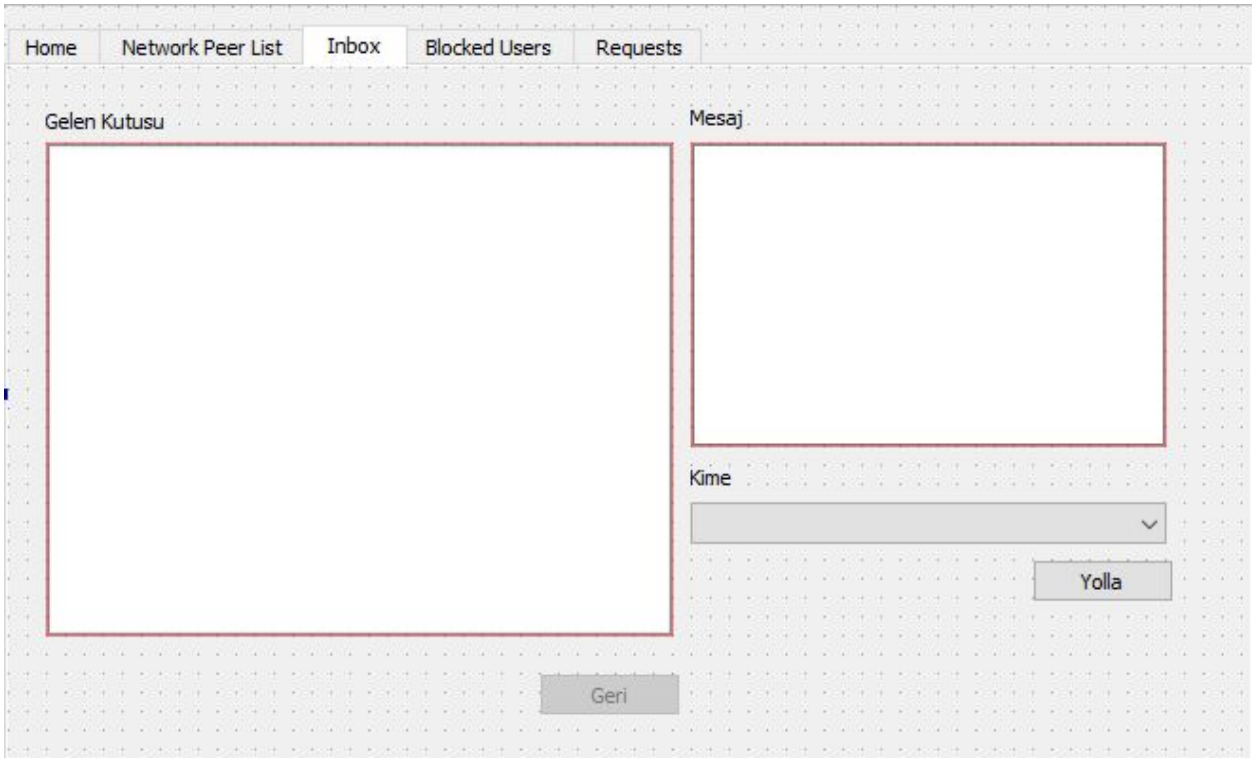
Projemizde mesajlaşma, ağdaki peer listesini görüntüleme buna göre abonelik isteğinde bulunma, yayın yapabilme ve diğer kullanıcıların yayınlarını alabilme hatta sistemden çıkış yapıldığında geri giriş yapılana kadar abone olunan peer'lerin yayınladığı blog yazılarını getirme (pending blogs gibi), abonelik isteklerini kabul ya da reddetme gibi işlemler hatasız çalışmakta.

Bütün bunları Mustafa'nın evde ki sunucusuna bağlanıp test etme şansına sahiptik.

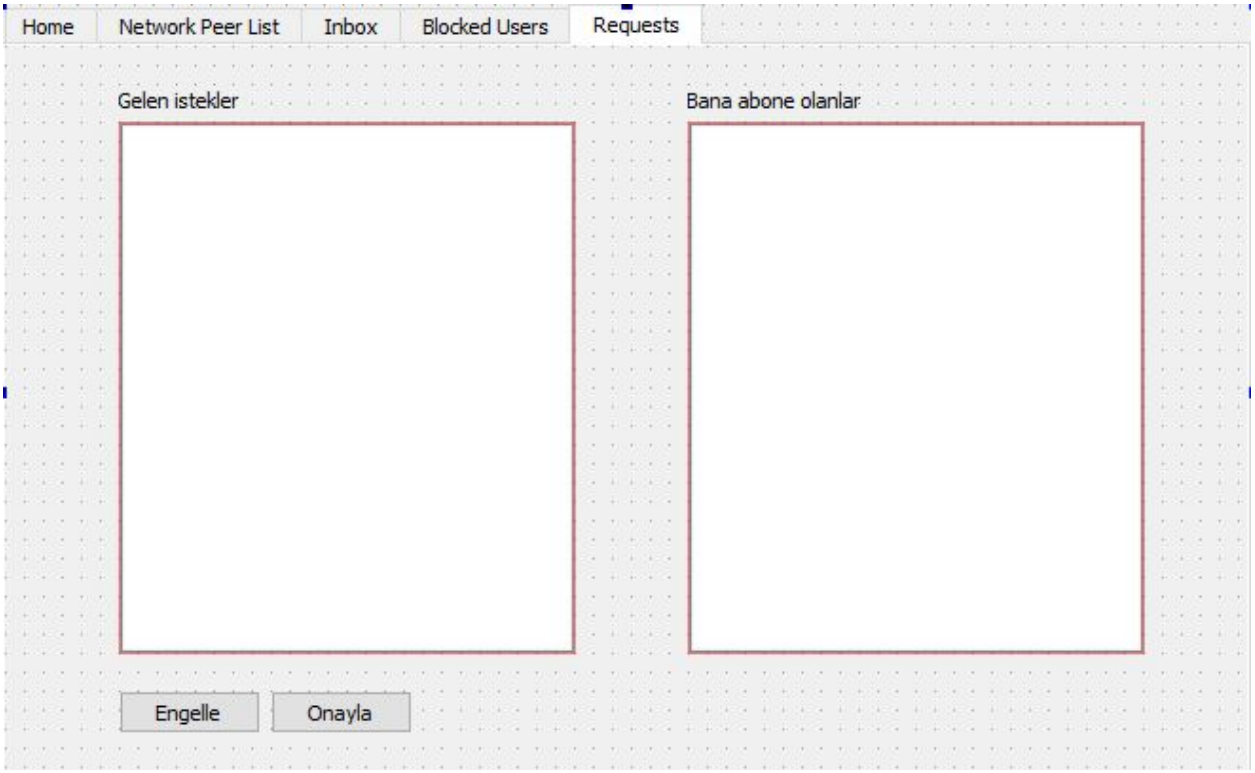
Arayüz tasarımımız:
Arayüzü tasarlarken sayfaları birbirlerine bağlamakla uğraşmamak için tab mantığı kullanmaya karar verdik.






Home
Network Peer List
Inbox
Blocked Users
Requests
Gelen Kutusu
Mesaj
Kime
Yolla
Geri


Home
Network Peer List
Inbox
Blocked Users
Requests
Engellenmiş Peer Listesi
Engelli Kaldır



Karşılaştığımız problemler:

Aracı kısmını yazmaya çalışırken bir problemle karşılaştık. Bu problem ağ katılmak isteyen yeni bir peer'in doğrulanma aşamasında oldu, yollanılan USR mesajı sonsuz döngüye giriyordu. İlk olarak USR mesaj sayısını bir sayaç ile tutup bununla kontrol yapıp zorla sonsuz döngüden çıkarmaya çalıştık fakat olmadı. Daha sonra bu çözüm işe yaramadı ve ayrı bir tane LGN diye bir protokol mesajı eklemeyi denedik fakat bu da işe yaramadı ve sonuç olarak bu sorunu halledemediğimiz için birincisi peer ekleme kısmını manuel olarak peer_list.txt içerisine yazarak yaptık, ikincisi ise public key değişimini yapamadık ve dolayısıyla sistem şifreli mesajlaşma yapamadı. Dosyanın içine peer bilgilerini bu formatda yazdık :

202544476549901: ['192.168.43.139','12344', '151515263213','Y','Fri Jan 11 11:02:19 2019','ON']

Mac adresi (uuid): ip adresi, port numarası, user name, tipi yayıncı mı aracı mı, ne zaman ağa katıldığının biligisi ve aktif olma durumu

Protokol detayları:

İstek	Parametre	Boyut	Cevap	Parametre	Boyut
USR	<uuid : ip : port : type>	<16 byte | 4 byte|2 byte|1 bit >	HEL		
			REJ		
PBK			PBO	<uuid : public_key>	<16 byte | 256 byte>
LSQ			LSA	<user_dictionary>	<10 byte>
			BLC		
MSG	<uuid : message>	<16 byte | uzunluk kısıtlaması yok>	MOK		
			MNO		
			BLC		
SBS	<uuid>	<16 byte>	SBO	<uuid>	<16 byte>
			SNO	<uuid>	<16 byte>
			BLC		
			UNK		
			ERR		
USB	<uuid>	<16 byte>	USO		
			BLC		
			UNK		
			ERR		
CHK			ACK	<uuid>	<16 byte>
BLU	<uuid>	<16 byte>	BLO		
			BLC		
			UNK		
			ERR		
UBL	<uuid>	<16 byte>	UBO		
PSH	<uuid : micro_blog_hash : microblogs>	<16 byte ||mikroblog uzunluğu>	PSO		

			BLC		
			UNK		
			ERR		
			PNO		
SRC	<uuid : micro_blog_hash >	<16 byte | >	SRO	<uuid>	<16 byte>
			SRN		
			BLC		
			UNK		
			ERR		
GVI	<uuid : micro_blog_hash >	<16 byte|>	GVO	<uuid : micro_blog_hash : microblogs>	<16 byte | | mikroblog sayısı*mikroblog uzulugu>
			GVN		
			BLC		
			UNK		
			ERR		
HSH	<uuid(Peer B) : text : hash'li text>	<16 byte | uzunluk yok | >	HSO		
			HSE		
MBR	<uuid(Peer B) : microblogs>	<16 byte | istenilen mikroblog sayısı*mikroblog uzunlugu>	MBO		
			BLC		
			UNK		
			ERR		

- Protokol komutları 3 byte'lıkdır
- Sistemde kullanılan dictionary yapıları:

    user_dictionary[uuid] = <ip, port, type, active_time, status, micro_blog_hash>
    user_public_keys[uuid] = <user_public_key>
    user_blocked[uuid] = <uuid>

Yeni Bağlantı Kurma

Yeni bağlantı kurmak isteyen kullanıcı bağlantı kuracağı eş yada aracının IP adresi ve port numarasını bilmelidir.
Bağlantı kurmak isteyen kullanıcı karşı tarafa kendi uuid'si,IP numarasını, port numarasını ve bir aracı mı, eş mi olduğunu belirten tipini belirtmelidir.

Örnek: peer A -> peer B
USR <uuid, IP, port,
type>
peer B -> peer A
HEL

Bağlantı Listesi İsteme

Eşlerin bağlantı listesinin güncellenmesi şu şekilde tasarlanmıştır. Sisteme kayıt olan yeni kullanıcıLSQilebağlandığıaracıyadayayıncıdankullanıcılistesiniister.Bağlanılanaracıyadaya yıncı ise yeni gelen kullanıcı bilgilerini de user_dictionary'e ekler ve LSA ile tümaktif kullanıcılara yeni kullanıcı listesini bildirir.

Örnek: peer A -> peer B
LSQ
peer B -> Tüm
Sistem LSA
<user_dictionary
>

Özel Mesaj

Özel mesaj gizli statüde olan bir servistir.Dolayısıyla her mesajdan önce mesaj gönderilmek istenen kullanıcın public_key'i, user_public_keys dictionary'isinden alınır.Eğer burada public_key bulunamazsa mesaj protokolünden önce public_key isteme protokolü çalıştırılır.

Örnek: peer B -> peer A
PBK
peer A -> peer B
PBO <uuid(peer B) : publicKey>

public_key'i Alan kullanıcı kendi user_public_keys dictionary'sini günceller.Bu kontrol işlemi gizli statüdeki her işlem için geçerlidir.

Mesaj göndermek isteyen kullanıcı karşı tarafın uuid'sini ve mesajın karşı tarafın public_key'i ile şifrelenmiş halini parametre olarak kullanır.

Örnek: peer A -> peer B

MSG <uuid : message>

Mesajı alan karşı taraf onaylamak için protokol mesajı gönderir

Örnek: peer B ->peer A

MOK

Bağlantı Kontrol

Bu işlem eşlerin diğer eşlere kontrol mesajı atarak o anki aktiflik durumunu kontrol etmesi için yapılmaktadır. Ayrı bir thread her 2 dakikada bir user_dictionary'de bulunan tüm eşlere CHK mesajı göderir ve ACKalır.Bu ACK ile user_dictionary içerisinde bulunan activate_time değerini günceller.Eğer eş ACK alamaz ise user_dictionary içerisindeki status alanını False olarak set eder.

Örnek: peer A -> Tüm Sistem

CHK

Tüm Sistem ->

Peer A ACK

<uuid>

Karşı Eşin Microbloglarına Üyelik İsteği Gönderme

Eşler birbirleri ile mesajlaşmanın yanı sıra birbirlerine de üye olabilirler. Üye olunan bir eşin microbloglarını görebilir hale geliriz. Karşı eşe üyelik isteği şu şekilde gerçekleştirilir.

Örnek:peer A -> peer B

SBS <uuid(peer B)>

peer B -> peer

A SBO <uuid>

SBO protokol mesajı ile uuid'nin tekrar dönmesinin nedeni kullanıcı üyelik isteğinin ilerleyen bir zamanda kabul edilmesi durumunda, üyelik işleminin hangi eş için tamamlandığını anlamak içindir.

Üyelikten çıkma işlemi de aynı şekilde işlemektedir.

Peer B tarafından diğer durumlarda gönderilecek protokol mesajları:
UNK (Peer B'nin Peer A'yı tanımadığı durumda gönderilecek mesaj.)

BLC (Peer B'nin Peer A tarafından engellenmiş olması durumunda gönderilir.)
ERR (Hatalı protokol mesajı gönderilmesi durumunda gönderilir.)

Engelleme ve Engeli Kaldırma

Eşler diğer eşlerden herhangi birini engelleyebilir ve bu sayede mesajlaşma ve microblog paylaşma işlemlerini durdurabilir.

Örnek: peer A -> peer B
BLU <uuid>
peer B ->
peer A
BLO

Peer B mesaj göndermek yada microblogları almak için önce bloklanmış olup olmadığını kontrol etmelidir.
Engeli kaldırma işlemi yine peer A tarafından peer B'ye bildirilmelidir. Örnek:

peer A -> peer B
UBL <uuid>
peer B->peer A
UBO

Peer B tarafından diğer durumlarda gönderilecek protokol mesajları:
UNK (Peer B'nin Peer A'yı tanımadığı durumda gönderilecek mesaj.)
BLC (Peer B'nin Peer A tarafından engellenmiş olması durumunda gönderilir.)
ERR (Hatalı protokol mesajı gönderilmesi durumunda gönderilir.)

Karşı Eşe Yayın Gönderme

Bir eş yayınladığı bir microblogu sistemde kendisine kayıtlı olan diğer eşlere bildirir. Yayınlanan microbloğun eşlerden birinin sistemde olmamasından dolayı alınamaması durumunda daha sonradan microblogu alamayan eşin bir karşılaştırma yapması açısından bir hash parametresi oluşturulur. Her eş kayıtlı olduğu diğer eşlerin hashlerini bir dictionary içinde tutar. Örneğin peer A yayın yapsın ancak peer B, peer A'ya üye olmasına rağmen sistemde online olmadığından peer A'nın son paylaşımını alamamış olsun. Peer A sisteme giriş yaptığında ilk yapacağı şey LSQ ile sistemde olan kullanıcıları(online yada offline) sorgulamak olur dolayısıyla sistemde olan kullanıcıların herhangi bir hash değişikliğini anlaya bilir. Çünkü hash user_dictionary içerisinde bulunmaktadır. Hash de bir farklılık olduğunu anlarsa SRC ile yeni hash'e ait microblog sorgusunu gönderir ve o hash'e sahip online bir kullanıcıdan yeni blogları alır.

Örnek: peer A -> peer A'ya abone olan peerlardan Online olanları

PSH <uuid : micro_blog_hash : microblogs>
peer A'ya abone olan peerlardan Online olanları-> peer A
PSO

Online olmayan peerlar ise sonradan girdiğinde LSQ sonucu hashlerin değiştiğini fark eder ve SRC ile sisteme hash sorgusu yapar

peer A'ya abone olan peerlardan sonradan online olanlar -> Tüm Sistem

SRC <micro_blog_hash>
Tüm Sistemde <micro_blog_hash> dosyasına sahip olan peerlar -> peer A'ya abone olan peerlardan sonradan online olanlar
SRO <uuid>

Peer A'ya abone olan peerlardan sonradan online olanlar, SRO cevabı ile kimlerde istediği dosyanın olduğu bilgisine sahip olur ve rastgele birinden dosyayı alır.

peer A'ya abone olan peerlardan sonradan online olanlar -> Rastgele seçilen peer

GVI <uuid : micro_blog_hash>
Rastgele seçilen peer -> peer A'ya abone olan peerlardan sonradan online olanlar
GVO <uuid : micro_blog_hash : microblogs>

Peer B tarafından diğer durumlarda gönderilecek protokol mesajları:
UNK (Peer B'nin Peer A'yı tanımadığı durumda gönderilecek mesaj.)
BLC (Peer B'nin Peer A tarafından engellenmiş olması durumunda gönderilir.)
ERR (Hatalı protokol mesajı gönderilmesi durumunda gönderilir.)

Üyelikten Çıkma İşlemi(Unsubscribe)

Peer A Peer B nin microblog üyeliğinden çıkmak istiyorsa; Peer A Peer B'ye USB <Peer A'nın UUID'si> şeklinde bir protokol mesajı gönderir. Bu mesajı alan Peer B Peer A'ya mesajı aldığını ve Peer A'ya artık microblog yayını yapmayacağını söylemek için USO mesajını gönderir.
Örnek;
Peer A->Peer B

USB <UUIDPeerA>
Peer B->Peer A
USO

Peer B tarafından diğer durumlarda gönderilecek protokol mesajları:
UNK (Peer B'nin Peer A'yı tanımadığı durumda gönderilecek mesaj.)
BLC (Peer B'nin Peer A tarafından engellenmiş olması durumunda gönderilir.)
ERR (Hatalı protokol mesajı gönderilmesi durumunda gönderilir.)

Onay İçin HASH Gönderme

İlk Public key paylaşımı yapıldıktan sonra Peer B'ye kayıt olmak isteyen Peer A, Peer B'nin UUID'sini, hash kontrolü test edilecek text'i ve hash'lenmiş text'i HSH protokol mesajına parametre vererek gönderir. HSH protokol mesajını alan Peer B hash'i çözerek gönderilen text'i elde ederse Peer A'ya HSO mesajı gönderir. Eğer Peer A'nın gönderdiği hash'lenmiş text ile kontrol text'i eşleşmezse bu durumda Peer B Peer A'ya HSE mesajı gönderir.

Peer A-> Peer B
HSH <UUID(Peer B) : text : hash'li text>

Peer B->Peer A
HSO ya da HSE

Bağlanılan peer'ın belirli sayıdaki son mikroblog yayınlarını isteme

Peer A daha önceden Peer B'nin mikrobloglarına üye olduğu durumda Peer B'nin belirli sayıdaki son mikroblog yayınlarını istemek için MBR protokol mesajını göndermelidir.Peer A; Peer B'nin UUID'sini ve kaç adet mikroblog yayını istediğini bu protokol mesajına parametre olarak vermelidir.

Örnek;
Peer A->Peer B
MBR <UUID(Peer B) : istenilen microblog adedi>

Peer B->Peer A
MBO <hash'lenmiş microbloglar>

Peer B tarafından diğer durumlarda gönderilecek protokol mesajları:
UNK (Peer B'nin Peer A'yı tanımadığı durumda gönderilecek mesaj.)
BLC (Peer B'nin Peer A tarafından engellenmiş olması durumunda gönderilir.)
ERR (Hatalı protokol mesajı gönderilmesi durumunda gönderilir.)