INF 443

Dağıtık Sistemler ve Uygulamaları

Proje Raporu

Ahmet Arca - 13401630

Duykan Berkin Sandal - 13401631

Yunus Emre Karazağ - 16401734

Furkan Karakoç - 16401750

Bayram Özatak - 16401754

Giriş

Bu proje kapsamında merkezcil olmayan (decentralized) mikroblog temelli bir sosyal ağ tasarlanmak ve gerçekleştirilmek istenmiştir. Bu sosyal ağda gönderilmek istenen her mesaj ve mikroblog şifrelenmiş olarak gönderilmektedir ve sadece ilgili alıcı tarafından çözümlenebilmektedir.

Yapım aşamasına ilk olarak protokol tasarımı ile başlanmıştır. Sonrasında bu tasarımın iki temel yapısı olan Aracı ve Yayıncı oluşturulmuştur. En sonunda da Aracı ve Yayıncı ikilisine loglama eklenmiştir.

Protokol

Protokoller genelde iki veya daha fazla taraf için yapılması gereken bir dizi işi belirtir. Özellikle veri iletişimi ve birden çok işin aynı anda yapıldığı multi-thread sistemlerde oldukça sık kullanılan bir terimdir.

İletişim protokolü olarak kullandığımız protokolü ortak bir çalışma sonuncunda belirledik. Belirlediğimiz bu protokol en az üç ya da daha fazla bilgisayar arasındaki iletişimi sağlamak amacıyla verileri düzenlemeye yarayan, standart olarak kabul ettiğimiz kurallardır.

Protokolümüz iki sistem arasında iletişim için kullanılan dili, yani mesajlaşma kurallarını belirtir. "Dil" yerine "protokol" kelimesinin seçilmiş olmasının sebebi, bu kelimenin programlama dili terimleri arasında, daha önceden yer alıyor olmasından kaynaklanır.

Bilgisayar üzerinden iletişim kurabilmek için belirli bir şablona ihtiyacımız vardı. Aracı ve yayıncı için ortak bir protokol belirledik. Söz dizimi ile karşı tarafa gidecek her kelimesi 4 harften oluşan ortak bir dil belirledik. Yani verinin içerisinde yer alan aracı ve yayıncının ortak dilini belirlemiş olduk. Verilerin söz dizimi oluşturulduktan sonra bu verinin nasıl gönderileceğini belirledik. Komutlarımız içsel ve dışsal komut olarak iki şekildedir. İçsel komutlar küçük harflerden dışsal komutlar büyük harflerden oluşur.

Dikkat ettiğimiz hususlar şu şekildedir;

*Protokolü kullanan tarafların tamamı için ortak bir protokol yaptık.
*Protokol unambigious oldu. Yani durum belirsizliğini ortadan kaldırdık.
*Protokol deterministic oldu. Yani her durumdan geçilebilecek bir sonraki durumu tanımladık.
*Protokole saldırıları önlemek amacıyla şifreli bir protokol kullandık.

Aslında sistemler iletişim için tek bir protokol kullanmazlar. Bunun yerine, protokol ailesi ya da diğer bir deyişle protokol takımı kullanırlar. Protokollerin birbirleriyle iletişiminin koordinasyonu için, konsept bir yapı gereklidir. Günümüzde birçok iletişim protokolü bulunmaktadır. Bunlardan popüler olanları; GSM, TCP/IP, HTTP vs.

Bu projede oluşturduğumuz protokol:

- **Şifreleme gerektirmeyen istekler ve cevapları:**

İstek	Parametre	Tanım	Cevap	Parametre	Tanım
ESCN	<UUID>, <adres>, <port>, <type>, <nickname>	Kayıt isteği	WAIT		UUID, adres, port üçlüsü kendi listesinde yoksa WAIT cevabını verir ve WHOU isteğinde bulunur
			ACCT		Kayıt isteğinin kabulü
			REJT		WHOU isteği başarısız olursa veya WHOU isteğinden gelen UUID farklı ise kayıt isteği reddedilir.
WHOU		UUID isteği ve bağlantı testi	MYID	<UUID>	UUID yollama
LIST	<Number> | “all”	Liste sorgulama. Belli bir sayıda kayıt veya tüm liste istenebilir.	LSIS	<UUID_i>, <Address_i>, <port_i>, <type_i>, <nickname_i>	Listedeki her satır arka arkaya yollanır.
KYRQ	<public key>	Public key isteği. Parametre olarak kendi public key’ini de yollar.	MYKY	<public key>	Public key yollama
SUBS		Abone olma isteği	BLOK		Bloklusun
			SUBA		Abonelik onayı
			ERKY		Public keyini bilmiyorum
UNSB		Abonelikten çıkma isteği	UNSA		Abonelikten çıkma onayı
UNBL		Engel kaldırma bildirimi	UNBA		Engel kaldırma onayı
			ERLO		ESCN ve WHOU dışındaki kayıt olmadan gönderilen isteklere verilen genel hata.
			ERSY		Syntax hatası bildirimi

- **Şifreleme gerektiren istekler ve cevapları:**

İstek	Parametre	Tanım	Cevap	Parametre	Tanım
KYCO		Public key kontrolü isteği	SIGN	<signed_text>, <text>	Private key ile imzalanmış metin ve açık metin parametre olarak yollanır.
MBRQ	<num>	Mikroblog isteği. <num>: istenilen mikroblog sayısı.	BLOK		Bloklusun
			MBLG	<mikroblog num> <mikroblog>	Mikrobloglar paket numarası ile birlikte yollanır.
			ERKY		Karşı tarafın publc key'i bilinmiyorsa bu hata kodu gönderilir.
CAST	<yayın>	Yayın gönderme	CSTA		Yayını alma bildirimi
MESG	<mesaj_içeriği>	Özel mesaj	MSGA		Özel mesaj onayı
			BLOK		Bloklusun
			ERKY		Karşı tarafın publc key'i bilinmiyorsa bu hata kodu gönderilir.

Aracı

Aracının genel çalışma mantığı şu şekildedir.

Bir peer olarak çalışan aracı sadece bağlantı bilgilerini aktarmak için kullanılır. Birbirlerini tanımayan peer'lar için herkesin tanıdığı bir peer görevi görür. Sisteme yeni dahil olan bir peer sadece aracının bilgilerine sahiptir. Aracı sayesinde diğer peer'ların bilgilerine ulaşır. Aracıdan liste bilgilerini alan yayıncılar arasında iletişim başlatılmaya hazır olur. Mesajlaşma ve mikroblog yayınları hakkında aracıların bilgisi yoktur. Bu nedenle ESCN, WHOU, LIST isteklerinin dışındaki istekleri tanımaz ve ERSY yazım hatası cevabını verir.

Aracının main kısmının sonunda bir sonsuz döngü bulunur. Bu döngü blocking bir şekilde accept() fonksiyonunda bekler. Yeni bir bağlantı geldiğinde o bağlantı serverThread'e aktarılır. Yani gelecek her bağlantıya bir tane serverThread açılır.

serverThread istemciden bir istek almak için soketi dinler. Soketten gelen veriler parser fonksiyonuna verilir. Parser'da veriler komut ve parametrelere ayrılır ve iki tane değişken döndürülür. Bunlardan ilki komut, ikincisi ise o verideki tüm parametreleri içeren bir listedir. Parser'dan gelen komuta göre varsa parametreler işlenir ve ilgili cevap aynı soket üzerinden istemciye gönderilir.

Yayıncı

Yayıncı proje kapsamında da istenildiği gibi bağlantı dinleyen bir sunucudan ve istenildiğinde bağlantı kuran bir istemciden oluşmaktadır. Temel olarak aracı kod alınmıştır, üstüne yayıncının gerçekleştirmesi istenen protokoller uygun biçimde implemente edilmiştir. Yayıncının doğası yani herkesle mesajlaşabilme ihtimali sebebiyle public key - private key çiftini yaratıp diğer peerlarla public key'ini paylaşabilmektedir. Authentication protokolü düzgün biçimde çalışmakta ve diğer peerlara kendisi olduğunu ispatlayabilmektedir. Sistemdeki herhangi 2 peer birbirlerinden belirli sayıda daha önce cast edilmiş mikroblokları sonrada talep edebilmektedir, gönderim asimetrik şifre ile yapılmıştır. Ayrıca bahsedilen cast işlemi de asimetrik şifrelemeyle cast yapan peerın bütün folowerlarına gönderilir. Subscription isteği herhangi 2 peerca yapılabilmektedir. Yayıncılar kullanıcı engelleyip engel kaldırabilir. Engelleme işlemi iç komutlarla gerçekleştirilir. Günümüzde bu tip platformlarda engelleme durumunda engellenen tarafın engellendiğine dair bir mesaj gönderilmesi gerekli görülmemektedir. Engellendiğine dair alabileceği mesaj, engelleyen peerdan bir istekte bulunursa alacağı mesajdır. Buna karşılık engelin kaldırılması karşı tarafı daha çok ilgilendireceği için engelleme kaldırıldığında buna dair bir protokol mesajı gönderilir.

Yayıncı tarafında gerekli dosyalar farklı olduğu için bunlarla ilgili dosyalar oluşturulur. Eğer daha önce oluşturulmuş olan gerekli dosyalar yoksa veya bir kısmı eksikse bunlar oluşturulur. İç işleyişte gerekli görülen yerlerde iç komutlar kullanılmıştır. Yapılan demoda az sayıda UUID kullanılsa da(demonun gereksinimlerinden ötürü), UUID'nin ethernet kartı üzerindeki mac adresinden alınması için gereken kod hazırdır. Yayıncının daha büyük ölçeklerde iş yapması gereken durumlarda implementation kolayca buna adapte edilebilir.

Asimetrik şifreleme ile ilgili anlatılan kısımlarda, anlatılan şey aslında şudur: Öncelikle bir yayıncı peer diğer bir yayıncı peer'ın publik keyini istemektedir ve aynı anda kendisi de public key'ini paylaşmaktadır. Sonrasında taraflar isterlerse herhangi bir zaman içerisinde authentication işlemini gerçekleştirebilirler. Bu işlemde öncelikle bir peer diğer bir peera kontrol isteği atar, sonrasında cevap olarak o peer imzalanmış metini ve orijinal metini atar. İmzalı metin kontrol isteği atan peerca kontrol edilir yani imzalı metinin sahibinin public key'i ile çözülür, orijinal metinle aynı olup olmadığı kontrol edilir. Şifreli mesajların paylaşımı ise öncelikle bir peer'ın paylaşımda bulunacağı peer'a, o peerın public key'i ile atacağı metni şifrelemesiyle başlar. Bu şifreli metin alan tarafın private key'i ile çözülür. Casting mesajları da asimetrik şifreleme ile yapıldığı için bu şekilde gerçekleşir. Tek fark mesajın döngü içerisinde bütün folowerlarına gönderilmesi, bu sürecin her biri için gerçekleşmesidir.

Loglama

Loglama genellikle ağ yöneticilerinin bir ağda neler olup bittiğini belirlemelerine yardımcı olmak için kullanılır. Yapılan her işin kaydını tuttuğu için birçok alanda kullanılır. Bu projede, arka planda yapılan işlemlerin gözlenebilmesi için kullanıldı. Ayrıca kod geliştirme aşamasında bir sorunla karşılaşıldığında programın hangi anda, niçin hata aldığının saptanmasında kullanıldı.

Loglama işi loggerThread isimli bir thread'e yaptırılmıştır. Bu thread üzerinde çalışmakta olduğu Aracı veya Yayıncı programının yaptığı tüm işlemlerin kaydını tutmaktan sorumludur. Tutulan kayıtlar zaman bilgisi de eklenerek "log.txt" dosyasına yazılmaktadır. Örneğin: "Fri Jan 11 13:42:02 2019 - ACCT alindi".

Bu işlemleri yapabilmek için logQueue isimli bir kuyruk kullanılmıştır. Programın yaptığı her işlem (programın açılması, soketten veri gönderme, soketten veri alma vs.) sonrasında, yapılan işlem bu kuyruğa eklenmektedir. loggerThread ise sürekli olarak bu kuyrukta veri olup olmadığını kontrol etmektedir. Eğer kuyruk boşsa beklemeye devam eder. Eğer kuyruğa bir veri gelmişse başına zaman bilgisi eklenir ve "log.txt" dosyasına yazılır, kuyruğa yeni veri gelmesi beklenmeye devam eder. Kuyruktan "QUIT" mesajı alınırsa programın kapandığı log dosyasına eklenir ve loggerThread sonlanır.

Karşılaşılan zorluklar

Her yazılım projesinde olduğu gibi bu projede de birçok zorlukla karşılaşılmıştır. Bu zorluklardan başlıcaları:

Senkron haberleşmeyi bozan etkenler:

Senkron haberleşme mantığı gereği istemci bir istekte bulunup cevap bekler, bir istek alan sunucu ise karşı tarafa cevap yollar. Cevap alan istemci tekrar bir istekte bulunur veya bağlantıyı kapatır. Bu proje kapsamında senkron haberleşme ile tasarım yapmak uygun gözükse de senkronluğu bozan bazı durumlarla karşılaştık. Örneğin liste paylaşımı sırasında sunucu tarafı arka arkaya veri göndermektedir. Ancak istemci tarafı bir veri aldıktan sonra onu işleyip kullanıcıdan input beklemeye geçer, input sonrası sunucudan tekrar bir istekte bulunur. Bu durumda bir uyumsuzluk ortaya çıkmaktadır. Bu sorunu çözmek için "LSIS" komutunu alan istemci sonsuz döngüde recv() ile soketi dinlemeye geçer. Eğer parametresiz bir "LSIS" alırsa listenin bittiğini anlar ve sonsuz döngüden çıkar.

Şifrelenmiş metnin yollanması ve geri çözümlenmesi:

Oluşturduğumuz protokol gereği soket üzerinden gönderilen her veri 4 harflik bir komut ve bu komutun ilgili parametrelerini içermektedir. Örneğin şifreli mesaj göndermek için soketten *"MESG <şifreli mesaj>"* gönderilir.

Python3'te RSA ile şifrelenmiş bir metin, string formatından çıkıp tuple formatına dönüşmektedir. Üstelik bu tuple'ın elemanı da binary formatındadır. "MESG" string'inin devamına RSA şifreli metninden gelen binary değişkenini eklemek

istediğimizde program string ile binary ikilisini birleştirememektedir. Binary değişkenini string'e çevirmek istediğimizde ise, RSA'nın oluşturduğu şifreli metin içerisinde '\' karakterinin bolca bulunması yüzünden string'e çevrilen binary değişkeni bozulmaktadır. Bu sebepler yüzünden *"MESG <şifreli mesaj>"* formatında bir veri soket üzerinden yollanamamıştır. Bunun yerine komut ve parametre iki seferde yollanmıştır. Bu işlem sayesinde binary değişkeni soket üzerinden, bozulma olmadan doğrudan gönderilmiştir. "MESG" komutunu alan sunucu tarafı ise send() aşamasına geçmeden tekrar bir recv() yaparak veri beklemek zorunda kalmıştır.

Sonuç

Yaptığımız bu projede merkezcil olmayan (decentralized) mikroblog temelli bir sosyal ağ tasarlanması geliştirildi. Bu sosyal ağda gönderilmek istenen her mesaj ve mikroblog şifrelenmiş olarak gönderilip sadece ilgili alıcı tarafından çözümlenebiliyor. İlgili protokoller yukarıda tabloda sekil üzerine gösterilmiş olup, gerekli tasarlanması proje kodu üzerinde gösterilmiştir. Tasarlanan aracı ve yayıncı thread'leri kodlama kısmında olması gerektiği şekilde birbirine bağlatılmıştır. Ayrıca yaptığımız demoda aracı ve yayıncı arasında kurulan bağlantılar sağlıklı bir şekilde sonuçlar vermiş olup, yayıncı olarak sosyal medyaya giren bir kişi rahat ve güvenilir bir şekilde özel mesaj atma, twit atma, takip etme vs. isteklerini yerine getirebiliyor.