Karadeniz Teknik Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Yazilim Mühendisliği
Mimari Tasarim Raporu

Dağıtık Dosya Eşitleme Düzeneği

Ahmet Emin KIZILKAN - 434340
Semih DOĞAN - 425470

İçindekiler

Kısım I
Giriş

Küçük çaplı kurum veya işletmelerde (*ör.* okullar) bilgisayar sayısı fazla olmasına rağmen yüksek maliyetlerden ve işletme zorluğundan ötürü bir merkezi bilgisayar bulunmamaktadır. Bu tür kurumlarda bilgisayarlar aynı ağda İnternet'e bağlanırken birbirleriyle neredeyse hiç iletişim kurmamaktadır. Özellikle veri aktarımı gibi işlemler bu tür kurumlarda uzak sunucular (ücretli ya da ücretsiz *SaaS*) üzerinden veya taşınılabilir aygıtlar ile yapılmakta. Uzak sunucuların kullanılması mahremiyet sorunu teşkil etmekte ve ek bir maliyete neden olabilmektedir. Taşınılabilir depolama aygıtlarının kullanılması ise pratik olmamaktadır. Bu iletişim güçlüğü yerel ağda çalışabilen *dağıtık mimariye* sahip bir düzenek ile çözülebilir.

0.1 Amaç

Bu proje, birçok kurumun sahip olduğu bilgisayar sistemlerini birbiriyle merkezi bir sistem olmadan (dağıtık biçimde);

- aynı ağ üzerinde **bağlantı kurmasını**,
- **veri aktarımını**,
- **eşleşebilmesini**,
- dosyaların **eşitlenebilmesini**

sağlamayı amaçlamaktadır.

0.2 Kapsam

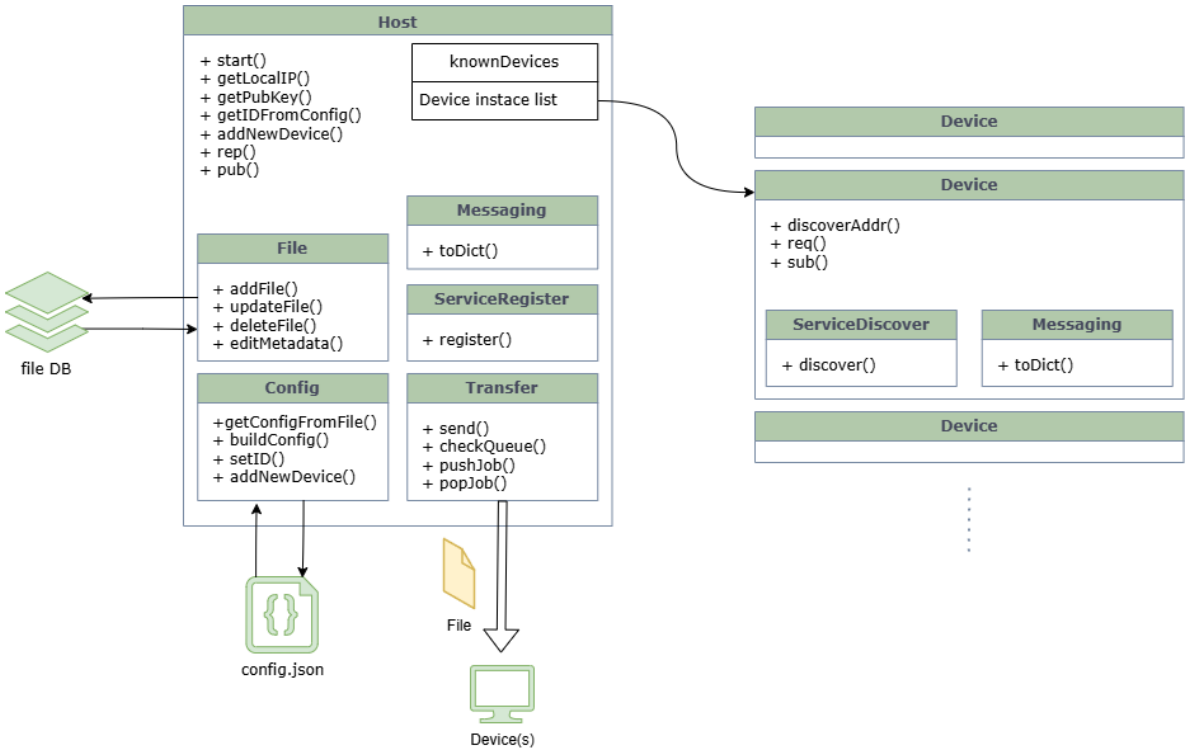
Aygıtlar arasında uçtan uca dosya aktarımı ve eşitlemesi sağlanacaktır. Kullanıcılar ile mesajlaşma gibi bir iletişim olmayacaktır. Dosyaların tamamı tüm bilgisayarlar ile eşitlenmeyecek seçilen dosyalar sadece kullanıcı tarafından belirtilen aygıtlar ile paylaşılacak ve eşitlenecektir. Bağlantılar yalnızca iki bilgisayar arasında yapılacak bir bilgisayar sadece eşitleyeceği dosyalara sahip olan bilgisayarlara bağlanacaktır. Aygıtlar yalnızca LAN üzerinde çalışacak İnternet üzerinden ya da bir *keşif sunucusu* (relay) aracılığı ile bir bağlantı gerçekleştirilmeyecektir.

Kısım II
Mimari Tasarım

1 Sistemler

Programın işleyişinde, programın çalıştığı makine (ana makine, `Host` veya kullanılan makine) ve eklenen *diğer* aygıtlar (`Device`) olmak üzere iki ana sınıf bulunuyor. `Host` sınıfı ile yeni aygıtların eklenmesi, yapılandırma arayüzü, dosyaların yönetilmesi ve aktarımı gibi programın çalıştığı aygıtla ilgili işlemler yürütülüyor. `Device` sınıfıyla ise eklenmiş aygıtlar ile iletişim işlemleri yapılıyor.

Dosya aktarımında `ssh` üzerinde çalışan `scp` protokolü kullanılacaktır. Yazıda bahsedilen anahtar çiftleri `ssh` anahtarlarıdır.



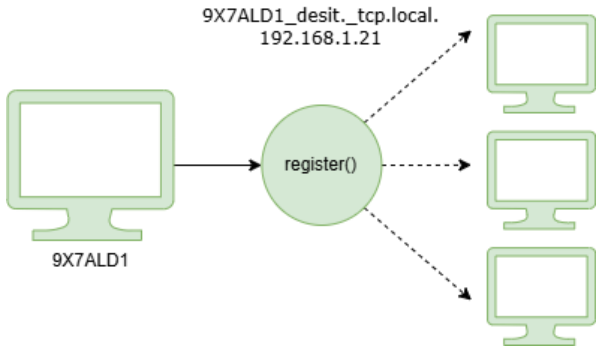
Şekil 1: Sistemin Genel Şeması

1.1 Keşif Arayüzü

Aygıtların birbirlerinin güncel IP adreslerine erişimi bu arayüz üzerinden gerçekleştirilecektir. Bu süreç üretilen anahtarlardan türetilen bir kimlik numarası (`ID`) ile gerçekleştirilecektir. Bu `ID`'ler ile yerel ağda her aygıt için bir `mDNS` kaydı oluşturulacak böylece her makine istediği aygıtın IP adresini ilgili ID ile yapacağı *DNS* sorgusu ile gerçekleştirebilecektir.

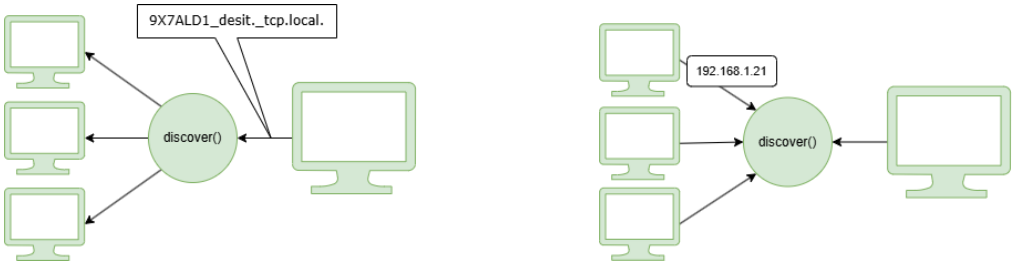
Bu sorgu işlemleri `Service` modülünün içindeki iki sınıf ile gerçekleştirilir. `ServiceRegister` sınıfı kullanılan makinenin ID numarasına bağlı olan mDNS'in kaydedilmesini sağlıyor. Bu sınıf gösterildiği gibi `Host` ana sınıfının bir üyesidir.

`ServiceRegister` sınıfı `register()` metodunu barındırır. Bu metot `Zeroconf` kütüphanesinden yararlanarak yerel ağ üzerinde bir servis kaydı (*mDNS* kaydı) oluşturur. Oluşturulan servis kaydı program kapanana kadar açık kalmaya devam eder. Program kapanırken bu kaydın silinmesini sağlar.



Şekil 2: mDNS servis kaydı

`ServiceDiscover` sınıfı `Device` sınıfının bir üyesidir. Bu sınıfın içerisinde `discover()` metodu çalıştırılır. Yeni bir `Device` sınıfı oluşturulduğunda ya da ilgili aygıtın *güncel* IP adresi bulunmak istediğinde bu metot ID ile türetilen bir *domain* sorgusu yapar.



(a) (b)

Şekil 3: mDNS servis sorgusu

1.2 Eşleşme Arayüzü

Eşleşme işlemi ile bir aygıt tarafından eklenen başka bir aygıtın bu ekleme işleminden haberdar edilir ve anahtar takası gerçekleştirilir. Eşleşme arayüzü için iletişim arayüzündeki *Reply-Request* (Bölüm 1.3.2) metotları kullanılacaktır.

Şekil 4: Anahtar Takası

1.3 İletişim Arayüzü

Aygıtların iletişimi `Host` ve `Device` sınıfı içerisindeki farklı metotlarla gerçekleştirilir. Bu metotlar `Messagging` modülüyle oluşturulan mesajları kullanır.

1.3.1 Mesajlaşma Arayüzü

Tüm iletileşme işlemleri genişletilebilir olmasından ötürü `json` biçimlendirmesinde olacaktır. Mesajların Python `dict` türünde oluşturulması için `Messaging` sınıfı kullanılacaktır. Bu sınıf ilgili anahtar değerlerini gönderici-alıcı makineye ve mesajın türüne uygun olarak `toDict()` metoduyla oluşturur.

```
{
  "TYPE" : "REQ::PUB_KEY",
  "TO" : "{DEVICE_ID}",      # or IDs [ "{DEV_ID-0}", "{DEV_ID-1}", ...]
  "FROM" : "{HOST_ID}",
  "HOSTNAME" : "{HOSTNAME}",
  "PUB_KEY": "ecdsa AvX...JmK lab@192...2" # HOST PUB_KEY
}

```

Listing 1: Mesaj şablonu

1.3.2 Request-Reply

Aygıtların arasında uçtan uca (*peer to peer*) bir iletişim gerektiğinde `Host` ve `Device` sınıflarından bulunan `rep()` ve `req()` metotları kullanılır.

`Host` sınıfı `rep()` metodunu barındırır. Programın başlangıcıyla birlikte bu metot çalıştırılır. Önceden belirtilen yapısal mesajlaşma şablonları ile gelen mesajın `TYPE` değerine göre gerekli fonksiyonun yürütülmesini sağlar.

Örneğin bağlanan bir aygıttan gelen *umumi anahtar* isteğini aygıtın `addNewDevice()` kaydedilmesi sağlayacak ve yerel makinenin umumi anahtarı ile yanıtlayacaktır.

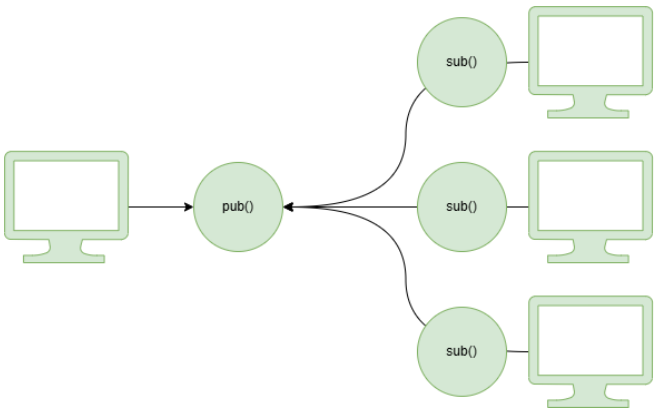
`Device` sınıfı `req()` metodunu barındırır. Bu metoda verilen ilgili istek argümanına göre nesnenin bağlı olduğu aygıta mesaj gönderilir. Yanıtı geri döndürür ya da istek ile ilgili fonksiyonu çağırır.

Örneğin bir aygıtın *umumi anahtarına* ihtiyaç olduğunda `req()` metodu ile mesaj gönderilecek ve yanıta göre işlem gerçekleştirilecektir.

1.3.3 Publisher-Subscriber

Her aygıtın dosya değişiklikleri, bağlantı durumları gibi bilgileri yayınladığı bir yayın kanalı (`PUB`) vardır. Bu yayın kanalları `pub()` metodu ile oluşturulur ve yönetilir. Programın çalışma süresince açık kalacaktır. Eşleşilen aygıtların yayın kanallarına `sub()` metodu ile bağlanılır.

`Host` sınıfı dosya değişiklikleri ve bağlantı durumu gibi işlemleri yayınlayacak olan `pub()` metodunu barındırır. Programın başlangıcıyla birlikte `pub()` metodu çalışmaya başlar. Bu metot programın icra süresinde gerçekleşen dosya ekleme/güncelleme değişikliklerini ve bağlantının kapanması durumunu kendisine bağlanan (`SUB` olan) aygıtlara bu kanal üzerinden iletir.



Şekil 5: PUB-SUB Şeması

Örneğin programın kullanıcı tarafından kapatılması durumunda `pub()` metodu ile yayınlanan bir `disconnect` mesajı ile diğer aygıtlar bu durumdan haberdar edilir. Eklenmiş dosya ile ilgili bilgiler bu dosyanın paylaşıldığı aygıtlara bu metot ile paylaşılır.

`Device` sınıfı eşleşilen aygıtların yayın kanalına bağlanılmasını sağlayan `sub()` metodunu barındırır. Gelen iletileri değerlendirir ve buna göre işlemin gerçekleştirilmesine karar verir.

1.4 Aktarım Arayüzü

Yeni eklenen ya da değişiklik yapılan dosyaların aktarılması `Transfer` sınıfındaki metotlar ile yapılır. Yapılacak işlemden önce `Host` sınıfındaki `jobQueue` denetlenir ve bundan sonra dosyaya sahip olan diğer aygıtlara ilgili dosyanın gönderilmesi sağlanır. Dosya gönderiminde `Paramiko` kütüphanesindeki `scp` işlevlerinden yararlanılacaktır.

2 Alt Sistemler

2.1 Yapılandırma Sistemi

Sistem ile ilgili tüm bilgiler bir yapılandırma dosyasına kaydedilir. Bu işlemler `Config` sınıfı aracılığıyla gerçekleştirilir. Daha önce eşleşilmiş aygıtların kimlik numaraları (`ID`) ve *umumi anahtarları* gibi bilgiler yapılandırma dosyasının içine kaydedilir. Program başlangıcında bu dosyanın varlığı yoklanır. Kaydedilmiş bilgiler `Config` arayüzü ile dosyadan okunur. Program çalışırken eklenen bilgiler yine bu arayüz ile yapılandırma dosyasına kaydedilir.

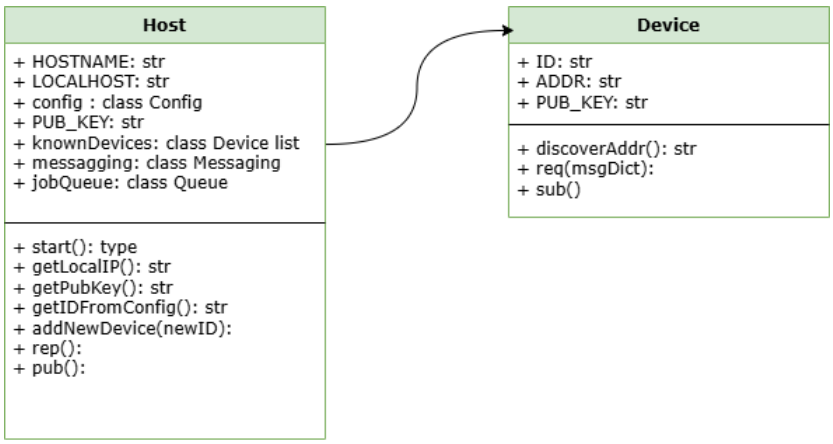
2.2 Dosya Arayüzü

Dosyaları veritabanına ekleme, güncelleme, düzenleme gibi işlemler `File` sınıfı içindeki metotlar ile gerçekleştirilir. Dosyaların hash değerlerinin hesaplanması ve *dağıtık hash tablosunun* düzenlenmesi yine bu arayüzle yapılır.

3 Sınıf Tanımları

3.1 Host

Programın çalıştırıldığı makineyle ilgili gerekli tüm bilgiler ve metotlar bu sınıf altında bulunmaktadır.



Şekil 6: Host&Device sınıfı UML Şeması

`getLocalIP()` Kullanılan makinenin yerel IP adresini geri döndürür.

`getPubKey()` Varsayılan bir dizinde kullanılan makinenin *umumi anahtarını* yoklar. Eğer dosya yoksa dosyayı bu dizinde oluşturur.

`getIDFromConfig()` Yapılandırma arayüzünü kullanarak yapılandırma dosyasından ID değerini okur ve geri döndürür.

`addNewDevice(newID)` Yeni bir aygıt eklenmesi için gerekli işlem ardışıklığını yürütür. Bir `Device` nesnesi oluşturur. Bu aygıtın aynı zamanda yapılandırma arayüzüyle dosyaya kaydedilmesini sağlar.

3.1.1 Config

Host sınıfının yapılandırma dosyası üzerinde yapacağı değişiklikler ve bu bilgilerin okunması için kullandığı arayüz sınıfıdır.

`getConfigFromFile()` Yapılandırma dosyasını varsayılan bir dizin içinde arar. Dosya bulunursa bu dosyayı *Python* `dict` türünde bir nesneye dönüştürerek `_config` üye değişkenine atar. Dosya dizinde yoksa ya da dizin yoksa duruma göre dizini ve dosyayı oluşturur. Dosyayı oluşturmak için `buildConfig()` metodunu çağırır.

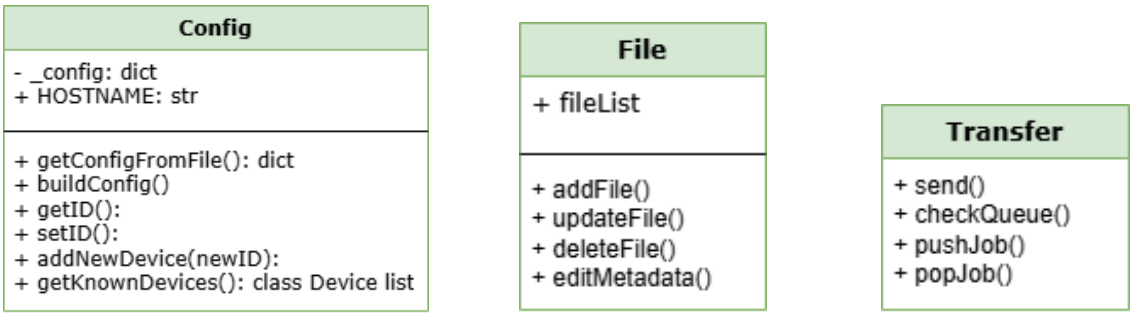
`buildConfig()` Yapılandırma dosyası yoksa bir `dict` nesnesi oluşturarak içini kullanılan aygıtın bilgileri ile doldurur ve yapılandırma dosyasının içerisine yazar.

`getID()` Yapılandırma dosyasının `dict` biçiminde kopyası olan `_config` üye değişkeninden ID'nin alınmasını sağlar.

`setID()` `_config` üye değişkeni içindeki ID değerini değiştirir ve bu değişikliğin yapılandırma dosyasına yazar.

`addNewDevice(newID)` Eklenen aygıtların yapılandırma dosyası içine eklenmesini sağlar.

`getKnownDevices()` Yapılandırma dosyasındaki eklenmiş aygıtların için birer `Device` *instance*'ı oluşturur. Bu nesnelerin hepsini bir listeye ekler ve bu listeyi geri döndürür.



(a) Config sınıfı (b) File sınıfı (c) Transfer sınıfı

Şekil 7: UML şemaları

3.1.2 File

Dosyaların güncellenmesi, eklenmesi, *metaverilerinin* okunması gibi işlemleri gerçekleştirir. Bunun için `addFile()`, `updateFile()`, `deleteFile()` ve `editMetadata()` metotlarını kullanır.

3.1.3 Transfer

Eklenen dosyaların ilişkili olduğu aygıtlara aktarılmasını sağlar. Aktarımı yapmadan önce `Host` sınıfının içindeki `jobQueue` kuyruğunu yoklar. Aktarım için çakışan herhangi bir iş yoksa eylem gerçekleştirilir. Bunun için `send()`, `checkQueue()`, `pushJob()` ve `popJob()` metotlarını kullanır.

3.2 Device

Her eklenen aygıtın bilgileri bir `Device` *instance*'ı içerisinde saklanır. `Device` sınıfı aygıtlar ile iletişim için gerekli metotları barındırır.

`discoverAddr()` Güncel IP adresini bulmak için kullanılır. `ServiceDiscover` arayüzünü kullanarak ID üzerinden bir DNS sorgusu yapılmasını sağlar. Eğer aygıt çevrim içi ise güncel IP adresi geri döner.

`req(msg)` Argüman verilen iletiyi (`msg` parametresi) *request* olarak aygıta gönderir. Yanıtı geri döndürür. Bu metot gönderimden önce `discoverAddr()` metodunu çağırarak IP adresini günceller.

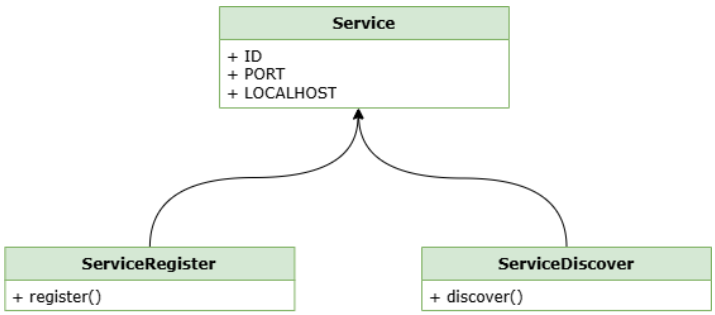
`sub()` Aygıtın yayın kanalına (`PUB`) bağlanılmasını sağlar. Gelen mesajların türüne göre gerekli fonksiyonları çağırır. Programın işleyişinde her aygıt için eş zamanlı olarak bir `sub()` işlevi yürütülür.

3.3 Service

Service modülü iki ayrı sınıf tanımından oluşmaktadır. Bu iki sınıf `Service` adlı sınıftan türetilmiştir.

`ServiceRegister` Programın çalıştığı makinenin ID'si, adresi, portu ve adı (*hostname*) gibi bilgileri içerir. `register()` metodunu kullanarak yerel ağda bir *mDNS* kaydı oluşturur. Program durdurulunca bu kaydın silinmesini sağlar. Bu sınıf `Host` sınıfının bir üyesidir.

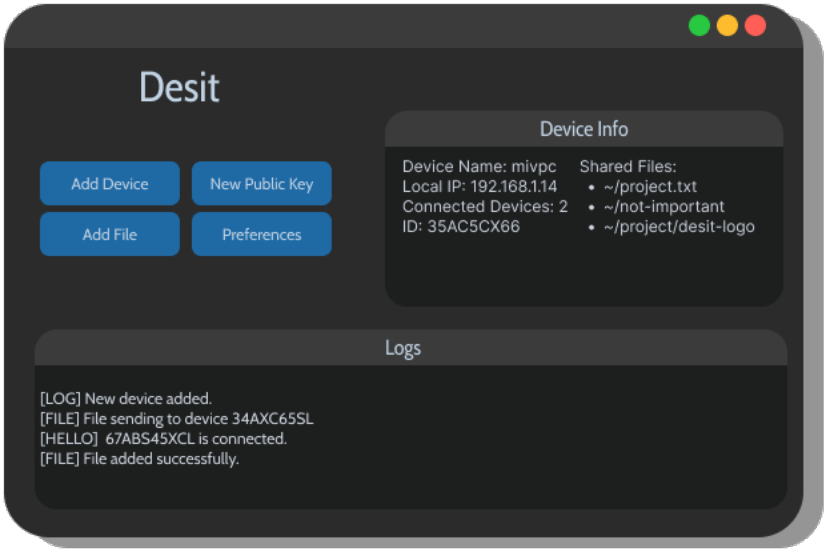
`ServiceDiscover` Verilen aygıt ID'sine göre bir mDNS sorgusu yapan `discover()` metodunu barındır. Bu metot ağdaki tüm servisleri tarar. Verilen ID ile eşleşen IP adresini geri döndürür.



Şekil 8: Service sınıfı UML Şeması

4 Rutin ve Arayüz Tanımları

Program tüm kullanıcı işlevlerini ve kayıtları gösteren tek bir pencere üzerinden yönetilir. Kullanılan makine hakkındaki bilgiler ve kayıtlar burada gösterilir. Yeni aygıt ve dosya ekleme işlemleri bu pencere üzerindeki düğmeler ile yürütülür.



Şekil 9: Kullanıcı Arayüzü Prototipi

Yeni aygıt ekleme düğmesi ile kullanıcıya eklenecek aygıtın ID'sini girebileceği bir istem kutucuğu açılır. Girilen ID ile daha önce bahsedilen `addNewDevice()` işlevi çağırılır.

Yeni dosya ekleme düğmesi ile kullanıcı için bir dosya penceresi (*file dialog*) açılır. Dosya seçildikten sonra eklenmiş aygıtlardan hangileriyle dosyanın paylaşılacağı kullanıcıya sorulur. Seçilen dosyanın yolu ve seçilen aygıtlar `File` sınıfının `addNewFile()` metoduna argüman olarak verilir.

5 Kütüphaneler ve Standartlar

- Program Python ile yazılacaktır.
- Programın uçtan uca (*peer to peer*) iletileşme işlemlerinde `ZeroMQ`[1] kütüphanesinden yararlanılacaktır.
- Dosya aktarımı için `ssh` protokolü altında çalışan `scp` protkolü kullanılacaktır.

  Bu protkolü kullanmayı sağlayan `Paramiko`[2] kütüphanesinden yararlanılacaktır.
- PEP8 Python Kodlama Standardı[3] kullanılacaktır.
- Sabit üye değişkenler büyük harfle adlandırılacaktır.
- Her sınıf modüllere ayrılacaktır.
- Fonksiyonlar adlarına uygun olarak küçük görevlere bölünecektir.

6 Test Yaklaşımları

- Birim Testleri: Kullanıcı kaydı, giriş ve dosya yükleme modülleri için `pytest`[4] ve `unittest`[5] kullanılacaktır.
- Dosya ve aygıt ekleme özellikleri elle test edilecektir.

[1] zeromq.org
[2] paramiko.org
[3] peps.python.org
[4] pytest.org
[5] docs.python.org#unittest