ERCİYES ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
PROGRAMMING LABORATORY NOTLARI

11. HAFTA NOTU

(PYTHON İLE NETWORK TRACING UYGULAMASI)

Bu not dersin 11. haftasında yapacağınız uygulamaya hazırlanmanızı sağlayacak olup temel kavramları anlatan bilgileri içermektedir.

DERSİN ÖĞRETİM ÜYESİ

Prof. Dr. Bahriye AKAY

DENEYİ YAPTIRAN ÖĞRETİM ELEMANI

Arş. Gör. Fuat TÖRE

2026
KAYSERİ

1.NETWORK TRACING HAKKINDA TEMEL BİLGİLER

1.1. Network Tracing Nedir?

Network tracing, bir bilgisayardan hedef bir sunucuya gönderilen ağ paketlerinin hangi ara yönlendiricilerden geçtiğini inceleme işlemidir. Bu işlem sayesinde istemci ile hedef sistem arasındaki ağ yolu gözlemlenebilir.

Network tracing ile;

- hedefe kaç ara noktadan geçilerek ulaşıldığı,
- rota üzerinde hangi IP adreslerinin bulunduğu,
- her hop için yaklaşık gecikme değerinin ne olduğu,
- bazı yönlendiricilerin cevap verip vermediği

analiz edilebilir.

Bu deneyde network tracing işlemi doğrudan düşük seviyeli socket programlama ile değil, işletim sisteminin sağladığı `tracert` veya `traceroute` komutlarının Python üzerinden çalıştırılmasıyla gerçekleştirilecektir.

1.2. IP Adresi

IP adresi, ağ üzerindeki cihazların birbirini tanıması için kullanılan sayısal adrestir. Bilgisayarlar, yönlendiriciler, sunucular ve diğer ağ cihazları haberleşme sırasında IP adresleri aracılığıyla tanımlanır.

Örneğin:

```
192.168.1.1
8.8.8.8
193.255.88.116
```

Bu deneyde IP adresleri, `tracert/traceroute` çıktısından düzenli ifadeler yardımıyla ayrıştırılacaktır. IP adreslerinin doğru yakalanması, hedefe giden ağ yolunun yorumlanması açısından önemlidir.

1.3. Hop Kavramı

Hop, bir paketin hedefe ulaşırken geçtiği her ara yönlendiriciyi ifade eder.

Bir hedef sunucuya ulaşmak için paketler genellikle birden fazla ağ cihazından geçer.

Örnek:

```
1   192.168.1.1
2   100.120.128.1
3   81.212.72.151
```

Bu örnekte:

- hop: yerel ağ geçidi,
- hop: servis sağlayıcı tarafındaki ilk ağ noktası,
- hop: servis sağlayıcı omurgasındaki başka bir yönlendirici

olarak yorumlanabilir.

Hop sayısı, hedefe giden ağ yolunun uzunluğu hakkında fikir verir. Ancak hop sayısının fazla olması her zaman bağlantının kötü olduğu anlamına gelmez. Ağ topolojisi, yönlendirme politikaları, hedef sunucunun konumu ve güvenlik duvarı ayarları bu değeri etkileyebilir.

1.4. RTT (Round Trip Time)

RTT, bir paketin gönderici sistemden hedefe veya ara yönlendiriciye gidip geri dönmesi için geçen toplam süredir. Milisaniye cinsinden ifade edilir.

Örnek RTT değerleri:

```
1 ms
15 ms
42 ms
```

Düşük RTT değeri, ilgili ağ noktasına daha kısa sürede ulaşıldığını gösterir. Yüksek RTT değeri ise fiziksel uzaklık, ağ yoğunluğu, kuyruklama, yönlendirme farklılıkları veya ara ağ cihazlarının ICMP cevaplarını düşük öncelikte işlemesi gibi nedenlerden kaynaklanabilir.

Bu deneyde her hop için RTT değeri alınacak ve aynı satırda birden fazla RTT değeri varsa bu değerlerin ortalaması hesaplanacaktır.

1.5. Timeout ve * Gösterimi

Traceroute çıktısında bazı hop'lar için * işareti görülebilir. Bu durum genellikle ilgili yönlendiriciden belirlenen süre içinde cevap alınamadığını gösterir.

Ancak önemli nokta şudur:

```
* her zaman paket kaybı anlamına gelmez.
```

Bir yönlendirici güvenlik politikası nedeniyle ICMP cevaplarını engelleyebilir veya bu cevapları düşük

öncelikte işleyebilir. Buna rağmen paketler o yönlendiriciden geçip sonraki hop'lara ulaşabilir. Bu durum özellikle kurumsal ağlarda, üniversite ağlarında ve

güvenlik duvarı arkasındaki sistemlerde sık görülür.

1.6. TTL (Time To Live)

TTL, IP paketlerinde bulunan ve paketin ağ üzerinde kaç yönlendiriciden geçebileceğini belirleyen bir alandır. Her yönlendirici, kendisine gelen paketin TTL değerini 1 azaltır. TTL değeri sıfıra ulaştığında paket düşürülür ve genellikle ICMP "Time Exceeded" cevabı gönderilir.

Traceroute mantığı bu mekanizmaya dayanır:

1. İlk paket TTL = 1 ile gönderilir.
2. Paket ilk yönlendiricide düşer ve cevap alınır.
3. Daha sonra TTL = 2 yapılır.
4. İkinci yönlendirici tespit edilir.
5. TTL değeri artırılarak işlem hedefe ulaşılana kadar devam eder.

Bu deneyde TTL değerini doğrudan Python kodu ile biz ayarlamıyoruz. Bu işlemi işletim sisteminin `tracert` veya `traceroute` komutu kendi içinde gerçekleştirmektedir.

1.7. ICMP

ICMP, ağ cihazlarının hata, uyarı ve kontrol mesajları göndermesi için kullanılan bir protokoldür. Traceroute mekanizmasında yönlendiriciler çoğunlukla ICMP cevapları üzerinden tespit edilir.

Örneğin:

- TTL bittiğinde: ICMP Time Exceeded
- Hedefe ulaşılamadığında: ICMP Destination Unreachable

Bazı ağlarda ICMP mesajları güvenlik nedeniyle kısıtlanabilir. Bu durumda traceroute çıktısında timeout veya * görülebilir.

1.8. DNS Çözümleme

DNS, alan adlarını IP adreslerine dönüştüren sistemdir.

Örneğin:
`google.com → 216.239.38.120`

Bu deneyde kullanılan komutlarda DNS çözümleme kapatılmaktadır.

Windows:
`Bashtracert -4-d google.com`

Linux/macOS:

`Bashtraceroute -4-n-q1 google.com`

Burada `-d` ve `-n` parametreleri DNS çözümlemeyi kapatır. Bunun amacı çıktının daha hızlı, sade ve düzenli alınmasını sağlamaktır. Böylece öğrenciler IP adreslerini daha kolay işleyebilir.

1.9. Logaritmik Grafik

Ağ gecikmelerinde bazı hop'lar 1–5 ms civarında olurken bazı hop'lar 20–100 ms aralığına çıkabilir. Bu değerler doğrusal ölçekte gösterildiğinde küçük gecikme farkları grafikte belirgin görünmeyebilir.

Bu nedenle deneyde RTT değerleri logaritmik ölçekte gösterilmektedir:
`plt.yscale("log")`

Logaritmik ölçek, küçük ve büyük RTT değerlerinin aynı grafik üzerinde daha anlamlı biçimde karşılaştırılmasını sağlar.

2. DENEYDE KULLANILACAK KÜTÜPHANELER VE FONKSİYONLAR HAKKINDA KISA BİLGİ

2.1. subprocess

`subprocess` kütüphanesi, Python programı içerisinden işletim sistemi komutlarını çalıştırmak için kullanılır. Bu deneyde `tracert` veya `traceroute` komutunun Python üzerinden çalıştırılması için kullanılacaktır.

Kullanılan temel yapı:
`subprocess.run(cmd, capture_output=True, text =True)`

Burada:

- `cmd`: Çalıştırılacak komut listesidir.
- `capture_output=True`: Komut çıktısının program içerisinde yakalanmasını sağlar.
- `text=True`: Çıktının metin olarak alınmasını sağlar.
- `.stdout`: Komutun standart çıktısını temsil eder.

2.2. re

`re` kütüphanesi, düzenli ifadeler kullanarak metin içerisinden belirli örüntüleri yakalamak için kullanılır. Bu deneyde hop satırlarını, IP adreslerini ve RTT değerlerini ayıklamak için kullanılacaktır.

Hop satırını seçmek için:
```
re.match(r"^\d+\s+", line.strip())
```

IPv4 adresini bulmak için:

```
re.search(r"\b\d{1,3}(?:\.\d{1,3}){3}
\b", line)
```

RTT değerlerini bulmak için:

```
re.findall(r"(\d+)\s*ms", line)
```

kullanılır.

2.3. sys

`sys` kütüphanesi, programın çalıştığı işletim sistemi hakkında bilgi almak için kullanılır.

```
sys.platform.startswith("win")
```

Bu ifade Windows işletim sisteminde `True` döndürür. Böylece program, Windows için `tracert`, Linux/macOS için ise `traceroute` komutunu kullanır.

2.4. math

`math` kütüphanesi matematiksel işlemler için kullanılabilir. Bu deneyin temel kodunda minimum, ortalama ve maksimum değerler Python'un yerleşik fonksiyonları ile hesaplanmaktadır. Ancak standart sapma gibi ek istatistiksel analizler eklenmek istenirse `math` kütüphanesi kullanılabilir.

2.5. matplotlib

`matplotlib`, Python'da grafik çizimi için kullanılan yaygın bir kütüphanedir. Bu deneyde hop numarasına göre RTT değerlerinin grafiksel olarak gösterilmesi için kullanılacaktır.

Programda `matplotlib` opsiyonel olarak ele alınmıştır:
```
try:
importmatplotlib.pyplotasplt
HAS_MPL=True
except:
HAS_MPL=False
```

Böylece `matplotlib` kurulu değilse program tamamen durmaz; yalnızca grafik çizimi yapılmaz.

3. DENEYDE KULLANILACAK PROGRAMLAMA YAPILARI

3.1. Değişkenler

Programda hedef adres, çalıştırılacak komut, komut çıktısı, hop bilgileri ve RTT değerleri değişkenler içinde tutulur.

Örnek:
```
TARGET=input("Target (Enter=google.com): ").s
trip() or"google.com"
```

Bu satırda kullanıcıdan hedef adres alınır. Kullanıcı herhangi bir giriş yapmadan Enter tuşuna basarsa varsayılan hedef olarak `google.com` seçilir.

3.2. Koşul Yapıları

Koşul yapıları, programın farklı durumlara göre karar vermesini sağlar.

Örnek:

```
if sys.platform.startswith("win"):
```

Bu yapı, işletim sistemine göre farklı traceroute komutunun seçilmesini sağlar.

3.3. Döngüler

Döngüler, traceroute çıktısındaki satırları tek tek incelemek için kullanılır.

```
for line in output.splitlines():
```

Bu döngü, komut çıktısını satır satır dolaşır.

3.4. Listeler

Her hop'a ait bilgiler `results` listesinde saklanır.

```
results.append((hop, ip, rtt))
```

Bu yapı sayesinde tüm hop bilgileri daha sonra tablo, analiz ve grafik işlemleri için kullanılabilir.

3.5. Tuple

Her hop kaydı üç elemanlı bir tuple olarak tutulur:

```
(hop, ip, rtt)
```

Burada:

- `hop`: Hop numarası
- `ip`: IP adresi
- `rtt`: Ortalama gecikme değeri

bilgisini temsil eder.

4. UYGULAMA İÇİN TEMEL KOMUTLAR

4.1. Windows İçin

```
tracert -4 -d google.com
```

Açıklama:

- `tracert`: Windows üzerinde rota izleme komutudur.
- `-4`: IPv4 kullanılmasını sağlar.
- `-d`: DNS çözümlemeyi kapatır.
- `google.com`: Hedef adresi belirtir.

4.2. Linux/macOS İçin

```
traceroute -4 -n -q 1 google.com
```

Açıklama:

- `traceroute`: Linux/macOS üzerinde rota izleme komutudur.
- `-4`: IPv4 kullanılmasını sağlar.
- `-n`: DNS çözümlemeyi kapatır.
- `-q 1`: Her hop için bir ölçüm yapılmasını sağlar.
- `google.com`: Hedef adresi belirtir.

5. DERSİN ÖĞRENME ÇIKTILARI

Bu deney sonunda öğrencilerin aşağıdaki kazanımları elde etmesi beklenmektedir:

- Network tracing kavramını açıklayabilmek,
- `tracert` ve `traceroute` komutlarının kullanım amacını kavrayabilmek,
- IP adresi, hop, RTT, TTL, ICMP ve timeout kavramlarını açıklayabilmek,
- Python ile işletim sistemi komutu çalıştırabilmek,
- Komut çıktısını metin olarak alıp işleyebilmek,
- Düzenli ifadeler ile IP adresi ve RTT değerlerini ayıklayabilmek,
- Liste ve tuple veri yapılarını kullanarak ölçüm sonuçlarını saklayabilmek,
- Temel istatistiksel analiz yapabilmek,
- Logaritmik ölçekte grafik oluşturabilmek,
- Elde edilen ağ çıktısını teknik olarak yorumlayabilmek.