Számítógépes Hálózatok

2. gyakorlat

JSON, subprocess

PYTHON ALAPOK II.

Subprocess hívások és shell parancsok

Ha nem érdekes az output:

```
import subprocess
subprocess.call(['df', '-h']) # új verziókban run(…) //linux
subprocess.call(['dir', '/B'], shell=True) # új verziókban run(…)  //windows
```

Ha érdekes az output:

```
import subprocess
p = subprocess.Popen(["echo", "hello world"], stdout=subprocess.PIPE)

print(p.communicate()) # eredménye egy tuple (stdout, stderr)

# ('hello world', None)
```

Néha a shell=True argumentum is kell, nézd meg a doksit!!! (pl. Windowson)

Hasznos segédletek:

https://docs.python.org/3/library/subprocess.html

https://www.pythonforbeginners.com/os/subprocess-for-system-administrators

subprocess – PIPE kezelés (linux)

Elvárt kimenet: dmesg | grep hda

```
from subprocess import PIPE, Popen

p1 = Popen(["dmesg"], stdout=PIPE)
p2 = Popen(["grep", "hda"], stdin=p1.stdout, stdout=PIPE)

p1.stdout.close()  # Allow p2 to receive a SIGPIPE if p1
exits.

output = p2.communicate()[0]
```

subprocess – várakozás a process végére

A process állapotának lekérdezése: poll

```
from subprocess import PIPE, Popen
import time

p1 = Popen(["ping", '-n', ,5', 'berkeley.edu'], stdout=PIPE)

while p1.poll()==None:
       print(" még fut " )
       time.sleep(1)
```

A process végének megvárása: wait – a communicate is megvárja a végét…

```
p1 = Popen(["ping", '-n', '20', 'berkeley.edu'], stdout=PIPE)

p1.wait() # várakozás a végére
```

subprocess - párhuzamosítás

```
from subprocess import PIPE, Popen
from time import sleep

cmd = ["timeout","5"]       #windows
process = []
for i in range(4):     #max active process
          p = Popen(cmd, stdout=PIPE, stderr=PIPE)
          process.append(p)
ures = False
while (not ures):
          fut = 0
          for p in process:
                    if p.poll() == None:
                              fut +=1
                    else:
                              if not p is None:
                                        print(p.communicate())
                                        p = None
          if (fut > 0):
                    print("Meg futnak")
                    sleep(1)
          else:
                    ures = True
```

Feladat

- Készítsünk egy olyan python kódot, ami csak annyi processzt használ, amennyi mag van a gépünkben. A program vizsgálja meg hogy Windowson vagy Linuxon fut. A subprocessek a következő parancsot futtassák: echo 'Windows' / echo 'Linux'

traceroute, ping

HÁLÓZATI ESZKÖZÖK I.

traceroute (linux) – tracert (windows)

Cél a hálózati útvonal meghatározása egy célállomás felé!

```
lakis@dpdk-pktgen:~$ traceroute berkeley.edu
traceroute to berkeley.edu (35.163.72.93), 30 hops max, 60 byte packets
 1  192.168.0.192 (192.168.0.192)  0.292 ms  0.344 ms  0.390 ms
 2  ikoktatok-gate.inf.elte.hu (157.181.167.254)  1.251 ms  1.250 ms  1.265 ms
 3  taurus.centaur-taurus.elte.hu (157.181.126.134)  5.180 ms  5.267 ms  5.325 ms
 4  fw1.firewall.elte.hu (157.181.141.145)  1.271 ms  1.358 ms  1.299 ms
 5  taurus.fw1.fw.backbone.elte.hu (192.153.18.146)  5.626 ms  5.356 ms  5.395 ms
 6  rtr.hbone-elte.elte.hu (157.181.141.9)  2.229 ms  1.245 ms  1.749 ms
 7  tg0-0-0-14.rtr2.vh.hbone.hu (195.111.100.47)  2.377 ms  2.415 ms  2.407 ms
 8  be1.rtr1.vh.hbone.hu (195.111.96.56)  1.945 ms  1.642 ms  1.877 ms
 9  bpt-b4-link.telia.net (80.239.195.56)  1.626 ms  1.581 ms  1.097 ms
10  win-bb2-link.telia.net (62.115.143.116)  196.574 ms win-bb2-link.telia.net (213.155.137.38)  196.993 ms
win-bb2-link.telia.net (213.155.135.222)  180.071 ms
11  ffm-bb4-link.telia.net (62.115.133.79)  199.425 ms  199.232 ms *
12  * * *
13  prs-bb3-link.telia.net (62.115.137.114)  180.494 ms  179.986 ms *
14  sjo-b21-link.telia.net (62.115.119.229)  197.252 ms  197.249 ms  197.264 ms
15  * a100row-ic-300117-sjo-b21.c.telia.net (213.248.87.118)  196.555 ms *
16  nyk-bb4-link.telia.net (62.115.142.222)  180.081 ms 54.240.242.148 (54.240.242.148)  200.986 ms
54.240.242.88 (54.240.242.88)  201.877 ms
17  54.240.242.161 (54.240.242.161)  200.935 ms * *
18  * * *
19  * * *
```

Linuxon

traceroute (linux) – tracert (windows)

Cél a hálózati útvonal meghatározása egy célállomás felé!

Windowson

```
C:\Users\laki>tracert berkeley.edu

Tracing route to berkeley.edu [35.163.72.93]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms  dlinkrouter [192.168.0.1]
  2    24 ms     6 ms    60 ms  10.0.0.85
  3    54 ms    18 ms    13 ms  fibhost-66-110-33.fibernet.hu [85.66.110.33]
  4    13 ms    14 ms    13 ms  ae0.info-c1.invitech.hu [213.163.54.245]
  5    13 ms    12 ms    17 ms  te0-0-2-3.nr11.b020698-1.bud01.atlas.cogentco.com [149.6.182.13]
  6    13 ms    13 ms    16 ms  te0-0-2-1.agr11.bud01.atlas.cogentco.com [154.25.3.237]
  7    15 ms    13 ms    12 ms  be3272.ccr31.bud01.atlas.cogentco.com [154.54.59.197]
  8    17 ms    16 ms    19 ms  be3263.ccr22.bts01.atlas.cogentco.com [154.54.59.177]
  9    22 ms    22 ms    21 ms  be3045.ccr21.prg01.atlas.cogentco.com [154.54.59.105]
 10    29 ms    30 ms    27 ms  be3027.ccr41.ham01.atlas.cogentco.com [130.117.1.205]
 11    41 ms    36 ms    41 ms  be2815.ccr41.ams03.atlas.cogentco.com [154.54.38.205]
 12   134 ms   136 ms   133 ms  be12194.ccr41.lon13.atlas.cogentco.com [154.54.56.93]
 13   133 ms   136 ms   132 ms  be2982.ccr31.bos01.atlas.cogentco.com [154.54.1.117]
 14   135 ms   134 ms   137 ms  be3599.ccr21.alb02.atlas.cogentco.com [66.28.4.237]
 15   134 ms   134 ms   135 ms  be2878.ccr21.cle04.atlas.cogentco.com [154.54.26.129]
 16   136 ms   136 ms   134 ms  be2717.ccr41.ord01.atlas.cogentco.com [154.54.6.221]
 17   148 ms   147 ms   146 ms  be2831.ccr21.mci01.atlas.cogentco.com [154.54.42.165]
 18   158 ms   159 ms   159 ms  be3035.ccr21.den01.atlas.cogentco.com [154.54.5.89]
 19   168 ms   169 ms   167 ms  be3037.ccr21.slc01.atlas.cogentco.com [154.54.41.145]
 20   183 ms   183 ms   183 ms  be3109.ccr21.sfo01.atlas.cogentco.com [154.54.44.137]
 21   186 ms   187 ms   184 ms  be3669.ccr41.sjc03.atlas.cogentco.com [154.54.43.10]
 22   184 ms   186 ms   185 ms  38.88.224.218
 23     *        *        *     Request timed out.
 24     *        *        *     Request timed out.
 25     *        *        *     Request timed out.
```

Ping a hoszt elérhetőségének ellenőrzésére és a Round Trip Time (RTT) méréséhez

Linuxon

```
lakis@dpdk-pktgen:~$ ping -c 3 berkeley.edu
PING berkeley.edu (35.163.72.93) 56(84) bytes of data.
64 bytes from ec2-35-163-72-93.us-west-2.compute.amazonaws.com (35.163.72.93): icmp_seq=1 ttl=23 time=194 ms
64 bytes from ec2-35-163-72-93.us-west-2.compute.amazonaws.com (35.163.72.93): icmp_seq=2 ttl=23 time=194 ms
64 bytes from ec2-35-163-72-93.us-west-2.compute.amazonaws.com (35.163.72.93): icmp_seq=3 ttl=23 time=193 ms

--- berkeley.edu ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 193.093/193.937/194.428/0.786 ms
```

Ping a hoszt elérhetőségének ellenőrzésére és a Round Trip Time (RTT) méréséhez

Windowson

```
C:\Users\laki>ping -n 3 berkeley.edu

Pinging berkeley.edu [35.163.72.93] with 32 bytes of data:
Reply from 35.163.72.93: bytes=32 time=200ms TTL=39
Reply from 35.163.72.93: bytes=32 time=201ms TTL=39
Reply from 35.163.72.93: bytes=32 time=200ms TTL=39

Ping statistics for 35.163.72.93:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 200ms, Maximum = 201ms, Average = 200ms
```

HÁZI FELADAT I. (1 PONT)

Alexa-top-1M

Az Alexa-top-1M adathalmaz tartalmazza a legnépszerűbb 1 millió website domain nevét népszerűségi sorrendben:

http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

Válasszuk ki az első és utolsó 10 nevet a listából, írjunk egy python programot, ami végig megy a leszűkített 20 elemű listán és minden címre lefuttatja a traceroute és ping toolokat, majd az eredményeket rendezett formában két fájlba írja! Ld. subprocess!!! Lehetőség szerint ne az egyetemi hálózaton futassuk az adatbegyűjtést!

Script paraméterezése: python3 program.py <top-1m.csv>

Traceroute paraméterek: max. 30 hopot vizsgáljunk

Ping paraméterek: 10 próba legyen

Kimeneti fájlok (ld. következő dia):

traceroute.json

ping.json

A párhuzamos futtatás esetén vigyázzunk és limitáljuk a processek maximális számát!!!

Leadás: A program leadása a BE-AD rendszeren .zip formátumban, amiben egy client.py szerepeljen!

Határidő: BEAD-ban

traceroute.json

```
traceroute.json:
        {
                "date" : "20180916",
                "system" : "windows",
                "traces" : [
                        {
                          "target" : "www.valami.com",
                          "output" : "Tracing route to www. . . "
                        },
                        …
                ]
        }
```

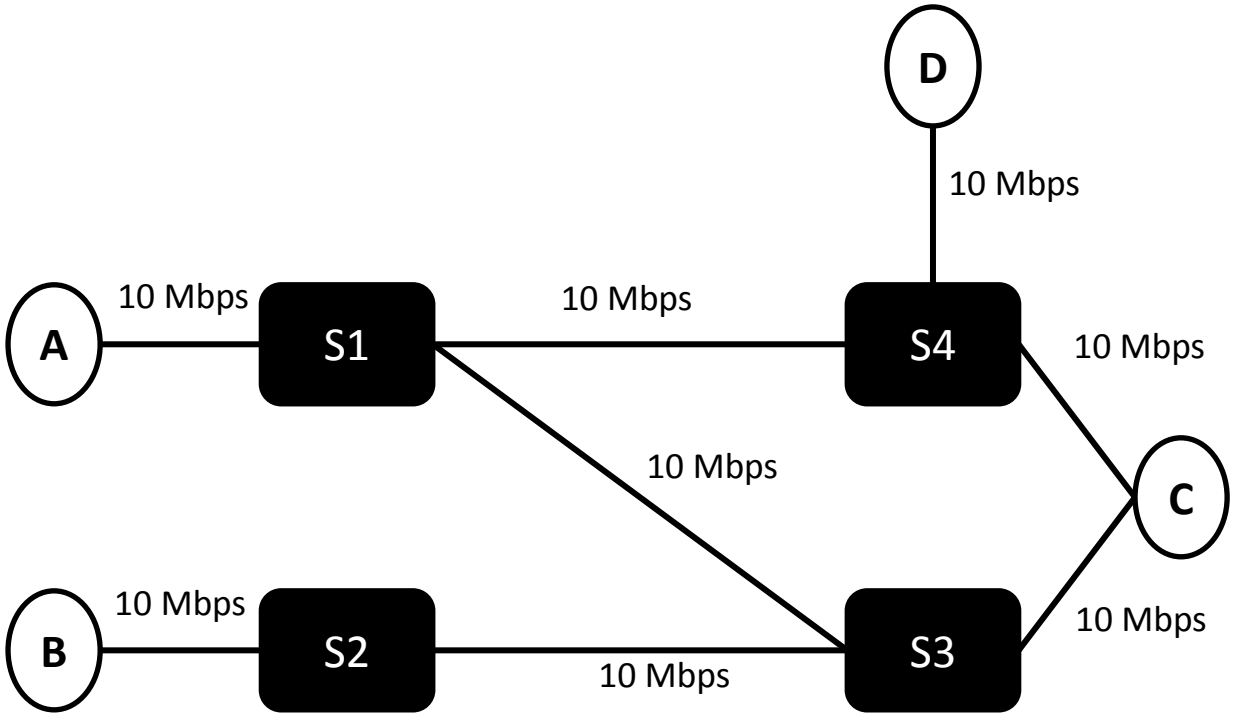
ping.json

```
ping.json:
        {
                "date" : "20180916",
                "system" : "linux",
                "pings" : [
                        {
                          "target" : "www.valami.com",
                          "output" : „Pinging www. . . "
                        },
                        …
                ]
        }
```

Áramkörkapcsolt hálózatok

HÁZI FELADAT II. (1 PONT)

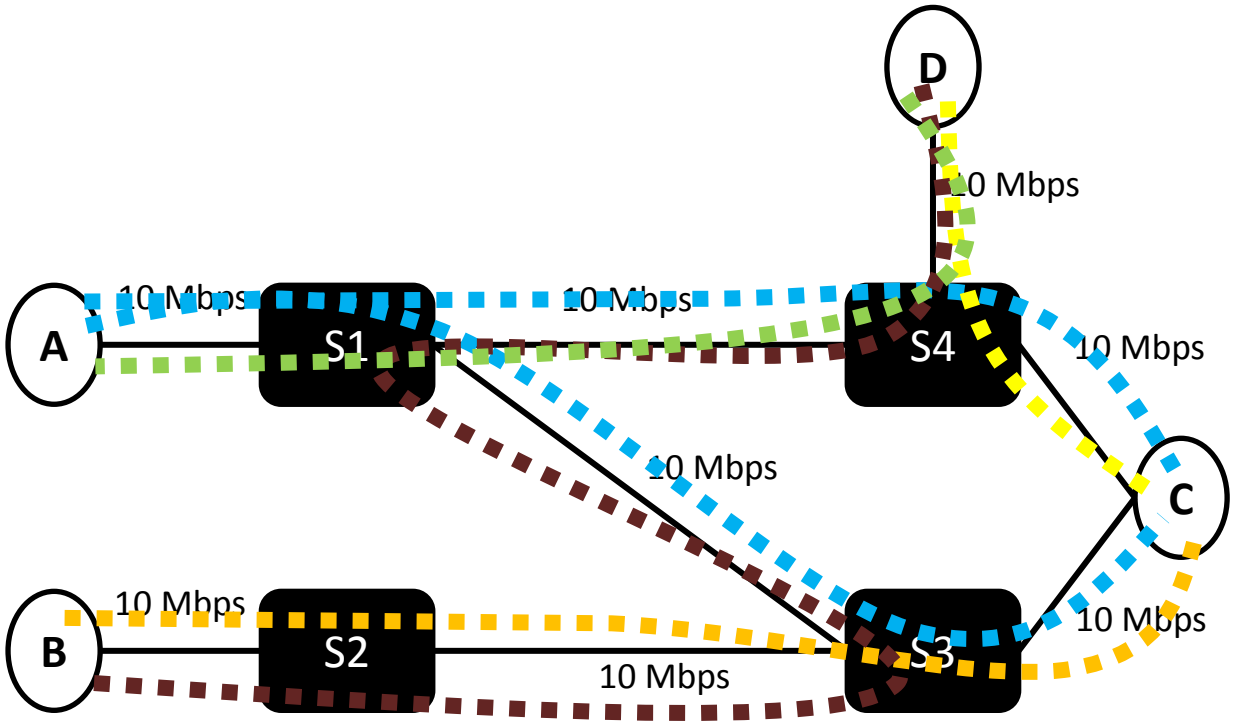
Topológia – cs1.json



Irányítatlan legyen a gráf!!!

```
"end-points": [ "A", "B", "C", "D" ],
"switches": [ "S1", "S2", "S3", "S4" ],
"links" : [
        {
                "points" : [ "A", "S1" ],
                "capacity" : 10.0
        },
        {
                "points" : [ "B", "S2" ],
                "capacity" : 10.0
        },
        {
                "points" : [ "D", "S4" ],
                "capacity" : 10.0
        },
        {
                "points" : [ "S1", "S4" ],
                "capacity" : 10.0
        },
        {
                "points" : [ "S1", "S3" ],
                "capacity" : 10.0
        },
        {
                "points" : [ "S2", "S3" ],
                "capacity" : 10.0
        },
        {
                "points" : [ "S4", "C" ],
                "capacity" : 10.0
        },
```

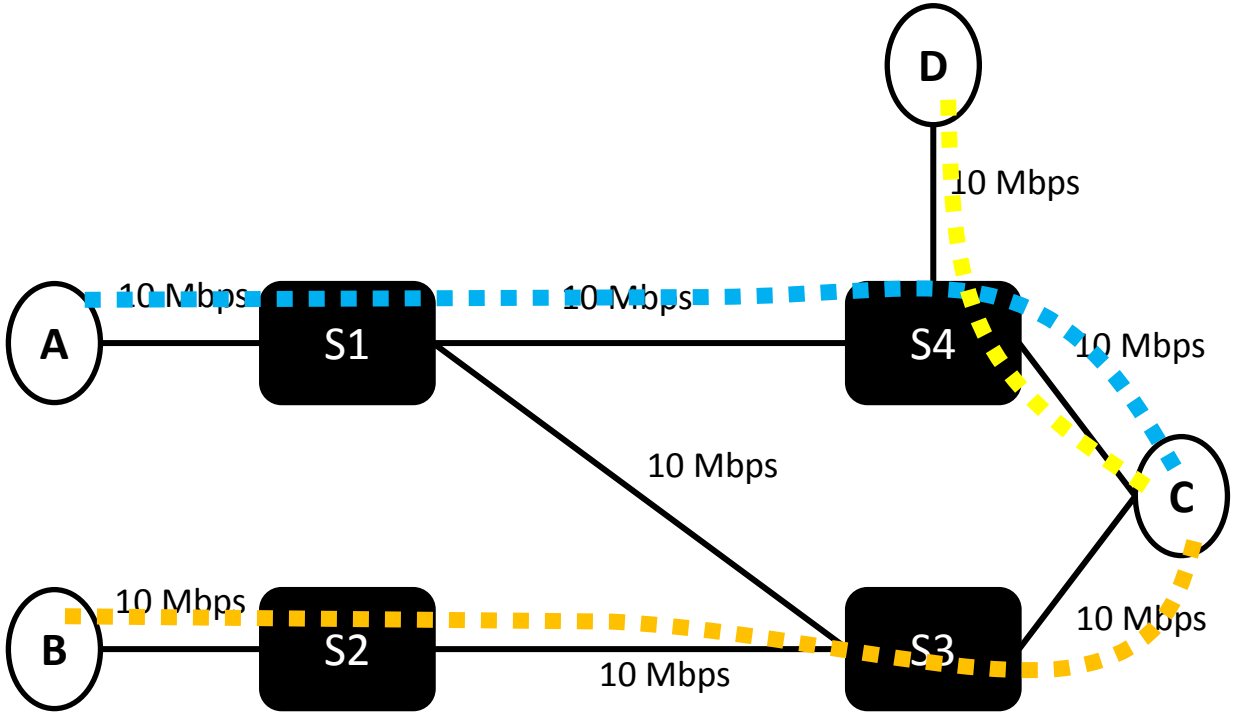
Lehetséges áramkörök – cs1.json




```
"possible-cicuits" : [
        ["D", "S4", "C"],
        ["C", "S4", "D"],
        ["A", "S1", "S4", "C"],
        ["A", "S1", "S3", "C"],
        ["C", "S4", "S1", "A"],
        ["C", "S3", "S1", "A"],
        ["B", "S2", "S3", "C"],
        ["C", "S3", "S2", "B"],
        ["B", "S2", "S3", "S1", "A"],
        ["A", "S1", "S3", "S2", "B"],
        ["D", "S4", "S1", "S3", "S2", "B"],
        ["B", "S2", "S3", "S1", "S4", "D"],
        ["A", "S1", "S4", "D"],
        ["D", "S4", "S1", "A"]
],
```

Irányítatlan legyen a gráf!!!

Igények – cs1.json




```
"simulation" : {
        "duration" : 11,
        "demands" : [
                {
                        "start-time" : 1,
                        "end-time" : 5,
                        "end-points" : ["A", "C"],
                        "demand" : 10.0
                },
                {
                        "start-time" : 2,
                        "end-time" : 10,
                        "end-points" : ["B", "C"],
                        "demand" : 10.0
                },
                {
                        "start-time" : 6,
                        "end-time" : 10,
                        "end-points" : ["D", "C"],
                        "demand" : 10.0
                }
        ]
}
```

Irányítatlan legyen a gráf!!!

Feladat

Adott a cs1.json, ami tartalmazza egy irányítatlan gráf leírását. A gráf végpont (end-points) és switch (switches) csomópontokat tartalmaz. Az élek (links) kapacitással rendelkeznek (valós szám). Tegyük fel, hogy egy áramkörkapcsolt hálózatban vagyunk és valamilyen RRP-szerű erőforrás foglaló protokollt használunk. Feltesszük, hogy csak a linkek megosztandó és szűk erőforrások. A json tartalmazza a kialakítható lehetséges útvonalakat (possible-cicuits), továbbá a rendszerbe beérkező, két végpontot összekötő áramkörigényeket kezdő és vég időponttal. A szimuláció a t=1 időpillanatban kezdődik és t=duration időpillanatban ér véget.

Készíts programot, ami leszimulálja az erőforrások lefoglalását és felszabadítását a JSON fájlban megadott topológia, kapacitások és igények alapján!

Script paraméterezése: python3 program.py <cs1.json>

A program kimenete:

<esemény sorszám. <esemény név>: <node1><-><node2> st:<szimuálciós idő> [- <sikeres/sikertelen>]

Pl.:

1. igény foglalás: A<->C st:1 – sikeres
2. igény foglalás: B<->C st:2 – sikeres
3. igény felszabadítás: A<->C st:5
4. igény foglalás: D<->C st:6 – sikeres
5. igény foglalás: A<->C st:7 – sikertelen

…

Leadás: A program leadása a BE-AD rendszeren .zip formátumban, amiben egy client.py szerepeljen!

Határidő: BEAD-ban

VÉGE
KÖSZÖNÖM A FIGYELMET!