

Telekommunikációs Hálózatok

2. gyakorlat

File átvitel

File bináris megnyitása

```
with open('input.txt', 'rb') as f:  
    f.read(128)
```

- `read(x)`:
 - `x` byte beolvasása (ha binárisként van megnyitva)
 - `x` karakter beolvasása (ha szöveggként van megnyitva)

“When size is omitted or negative, the entire contents of the file will be read and returned; it’s your problem if the file is twice as large as your machine’s memory.”

– python.org

Struktúraküldés

- Binárisra alakítjuk az adatot, majd vissza

```
import struct
values = (1, 'ab'.encode(), 2.7)
packer = struct.Struct('i 2s f') # Int, char[2], float
packed_data = packer.pack(*values)
print(packed_data)

unpacker = struct.Struct('i 2s f')
unpacked_data = unpacker.unpack(packed_data)
print(unpacked_data)
```

```
b'\x01\x00\x00\x00ab\x00\x00\xcd\xcc, @'
(1, b'ab', 2.700000047683716)
```

Note

`int` általában 4 byte, `char` általában 1 byte, tehát egy kis számot helytakarékosabb lehet stringként átvinni

Struktúra jellemzői

Az “Xs” (pl. “2s”) X db hosszú bytes objektumot jelent
(pl. `b'abc'`)

```
import struct
values = (1, 'ab', 2.7)
packer = struct.Struct('i 2s f')
packed_data = packer.pack(*values)
```

error: argument for 's' must be a bytes object

Helyesen

```
values = (1, 'ab'.encode(), 2.7) # vagy values = (1, b'ab', 2.7)
packed_data = packer.pack(*values)
print(packed_data)
```

`b'\x01\x00\x00\x00ab\x00\x00\xcd\xcc, @'`

Struktúra jellemzői

- A struktúra mérete byte-ban:

```
import struct
packer = struct.Struct('i 2s f')
print(struct.calcsize('i 2s f'))
print(packer.size)
```

12
12

- $i: \text{int} = 4, 2s: \text{char}[2] = 2, f: \text{float} = 4 \Rightarrow 4 + 2 + 4 \neq 12???$
- Az adattagokat úgy igazítja, hogy a kezdő pozíciójuk gépi szóhossz mérettel osztható legyen(itt most 4)
- [The Lost Art of Structure Packing](#)

Struct memory layout

0	1	2	3	4	5	6	7	8	9	10	11
int				char[2]		\0\0		float			

Struktúra

Character	Byte order
@	native
=	native
<	little-endian
>	big-endian
!	network(=big-endian)

Reference

Format	C Type	Python type	Size
x	pad byte	no value	
c	char	bytes of length 1	1
b	signed char	integer	1
B	unsigned char	integer	1
?	_Bool	bool	1
h	short	integer	2
H	unsigned short	integer	2
i	int	integer	4
l	unsigned int	integer	4
l	long	integer	4
L	unsigned long	integer	4
q	long long	integer	8
Q	unsigned long long	integer	8
n	ssize_t	integer	
N	size_t	integer	
e	(6)	float	2
f	float	float	4
d	double	float	8
s	char[]	bytes	
p	char[]	bytes	
P	void*	integer	

File kezelés

- `seek(offset, whence)`: pozícióváltás

```
seek.py
with open('alma.txt', 'r') as f:
    sor = f.readline()
    print('jelenlegi sor:', sor.strip()) # jelenlegi sor 1. sor

    sor = f.readline()
    print('jelenlegi sor:', sor.strip()) # jelenlegi sor 2. sor

    f.seek(0, 0)                # f.seek(offset, whence)

    sor = f.readline()
    print('jelenlegi sor:', sor.strip()) # jelenlegi sor 1. sor
```

- `offset`: olvasás/írás mutató pozíciója a file-ban
- `whence`: 0: abszolút, 1: relatív, 2: file végétől

File kezelés

- Seek bináris file-ban

```
bin_seek.py
import struct

packer = struct.Struct('i3si')

with open('dates.bin', 'wb') as f:
    for i in range(5):
        values = (2020 + i, b'jan', 10 + i)
        packed_data = packer.pack(*values)
        f.write(packed_data)

with open('dates.bin', 'rb') as f:
    f.seek(packer.size * 3)
    data = f.read(packer.size)
    print(packer.unpack(data))
```


Byte sorozat vs string

- String → Byte sorozat

```
import struct
str = 'hello'
print(str.encode())
print(struct.pack('8s', str.encode() ))
```

b'hello'

b'hello\x00\x00\x00'

- Byte sorozat → String

```
import struct
d = struct.pack('8s', str.encode())
print(d)
print(d.decode().strip('\x00'))
```

b'hello\x00\x00\x00'

hello

Python socket, host név feloldás

- Socket csomag használata

```
import socket
```

- `gethostname()`

```
hostname = socket.gethostname()
```

- `gethostbyname()`

```
hostip = socket.gethostbyname('www.example.org')
```

- `gethostbyname_ex()`

```
hostname, aliases, addresses = socket.gethostbyname_ex(host)
```

- `gethostbyaddr()`

```
hostname, aliases, addrs = socket.gethostbyaddr('157.181.161.79')
```

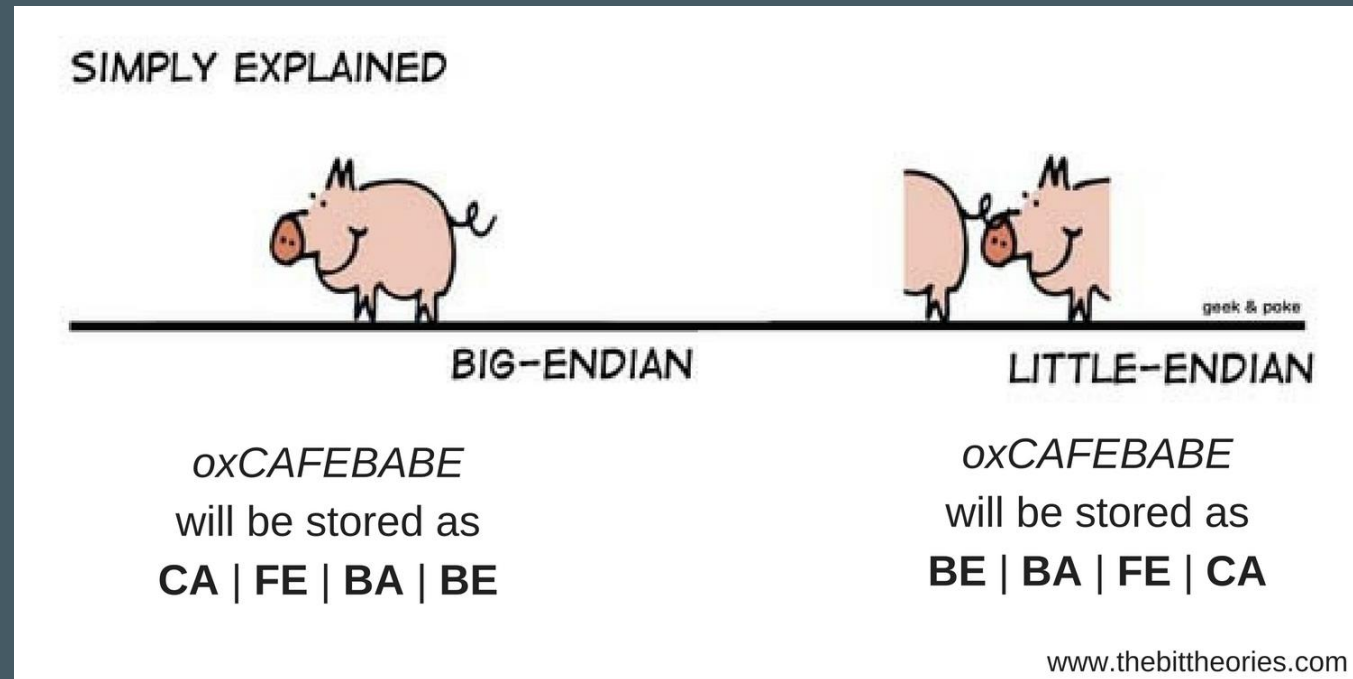
Port számok

- Bizonyos protokollokhoz tartoznak fix portszámok, konstansok (szállítási protokollok)!
- `getservbyport()`

```
import socket  
print(socket.getservbyport(22, 'tcp'))
```

- Írassuk ki a 1-100-ig a portokat és a hozzájuk tartozó protokollokat!

Little endian, big endian



- 16 és 32 bites pozitív számok kódolása
 - `htons()`, `htonl()` – host to network short / long
 - `ntohs()`, `ntohl()` – network to host short / long

Gyakorlás I.

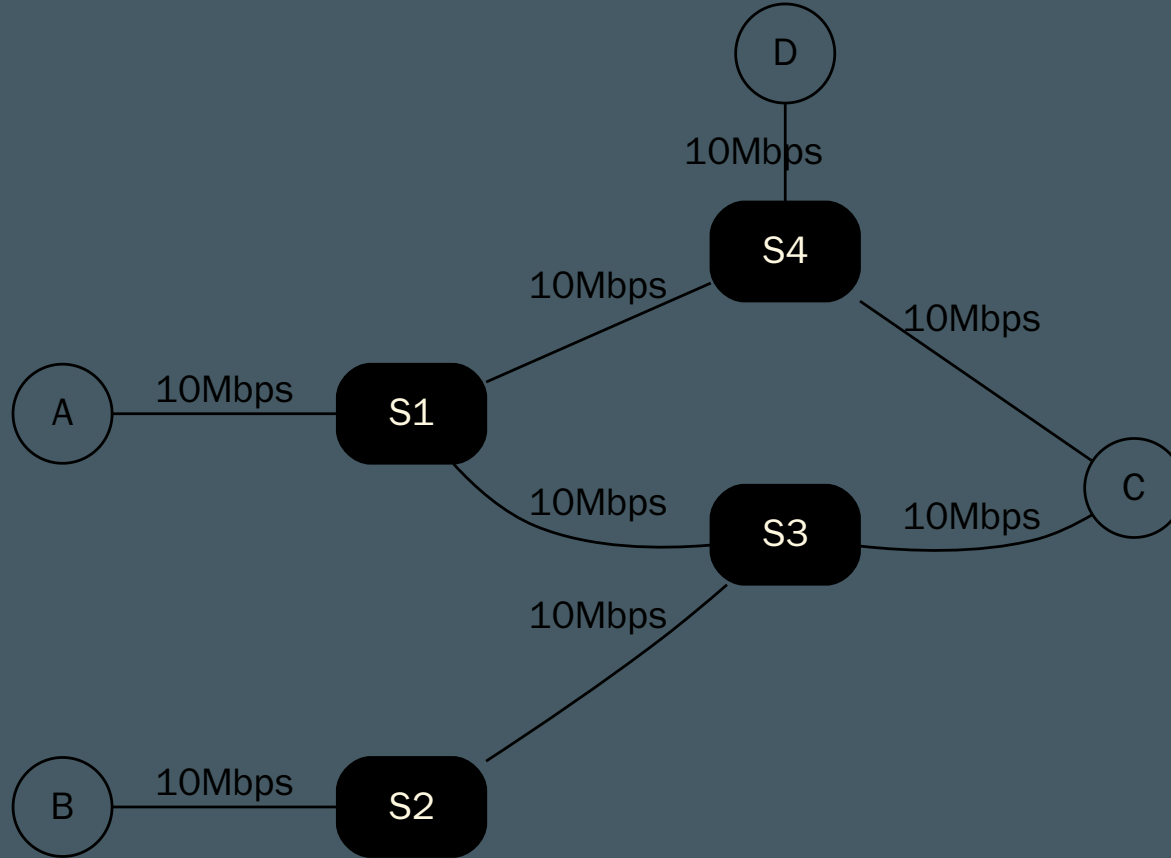
- Írjunk python scriptet, ami a következő struktúrájú bináris fájlokat képes létrehozni:
 - Domain (20s), port (i)
- Írjunk python scriptet, aminek paramétere egy az előző scripttel gyártott file, illetve:
 - `port < sor >`: paraméterül adott sor portjáról megmondja milyen service tartozik hozzá
 - `domain < sor >`: paraméterül adott sorból kiveszi a domaint és lekérdezi az ip címét

I. Beadandó

Áramkörkapcsolt hálózatok

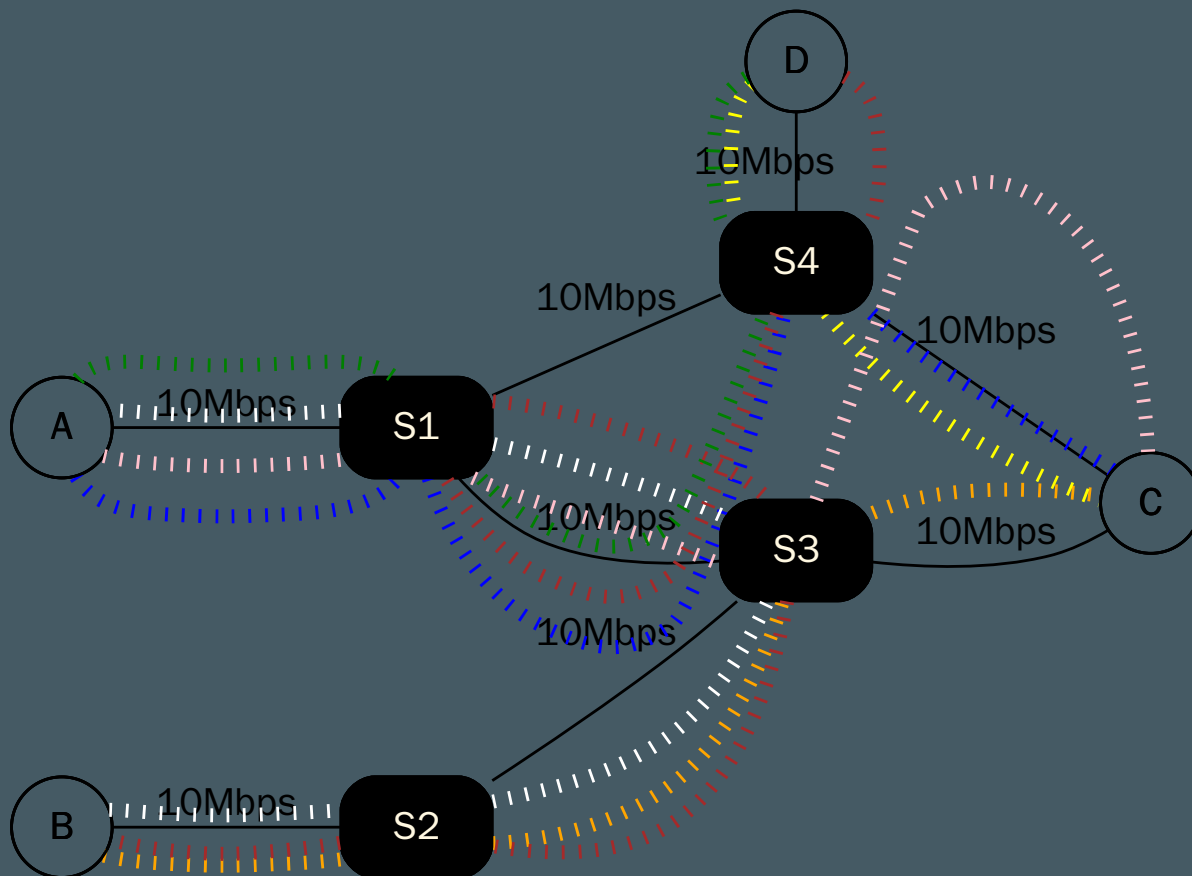
- Leírás
- Teszt szerver státusz

Topológia - cs1.json



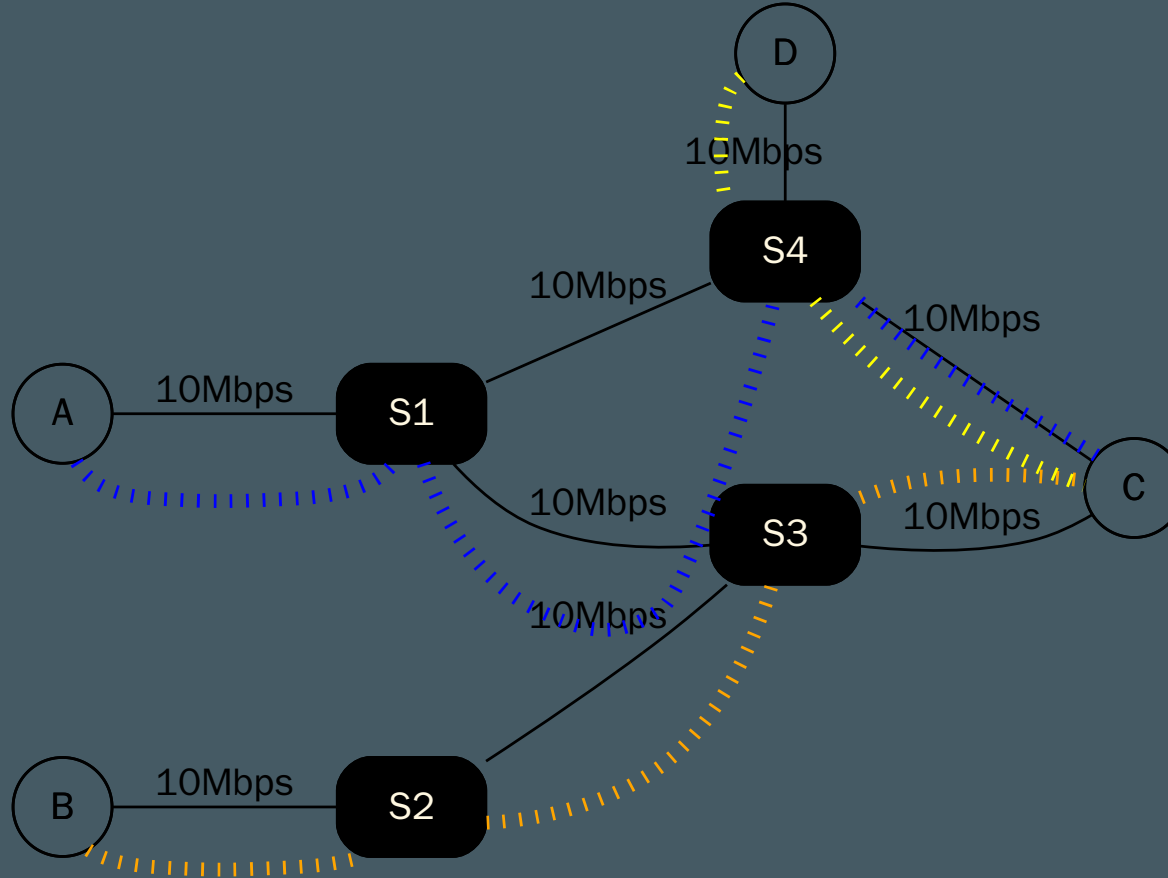
```
"end-points": [ "A", "B", "C", "D" ],  
"switches": [ "S1", "S2", "S3", "S4" ],  
"links" : [  
  {  
    "points" : [ "A", "S1" ],  
    "capacity" : 10.0  
  },  
  {  
    "points" : [ "B", "S2" ],  
    "capacity" : 10.0  
  },  
  {  
    "points" : [ "D", "S4" ],  
    "capacity" : 10.0  
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

Lehetséges áramkörök - cs1.json



```
"possible-circuits": [  
  ["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"]  
]
```


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  
    },  
    {  
      "start-time" : 7,  
      "end-time" : 10,  
      "end-points" : ["A", "C"],  
      "demand" : 10.0  
    }  
  ]  
}
```

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 foglalo protokollt használunk. Feltesszük, hogy csak a linkek megosztandó és szűk erőforrások. A json tartalmazza a kialakítható lehetséges útvonalakat (`possible-circuits`), 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!

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

Paraméterezés:

```
python3 program.py <cs1.json>
```

Kimenet:

```
<esemény sorszám>. <esemény név>: <node1><-><node2> st:<szimulációs idő> [- (sikeres|sikertelen)]
```

Leadás: A program leadása a TMS rendszeren .zip formátumban, amiben egy [client.py](#) szerepeljen!

Határidő: TMS-ben