communication networks laboratory

Computer Networks

Sándor Laki

ELTE-Ericsson Communication Networks Laboratory

ELTE FI – Dept. Of Information Systems

lakis@elte.hu

http://lakis.web.elte.hu

Eötvös Loránd University

Python basics

Socket programming

Pyton socket, name resolution

- We use the socket package

```
import socket
```

- gethostname()

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

- gethostbyname()

```
hostname = socket.gethostbyname(‚www.example.org’)
```

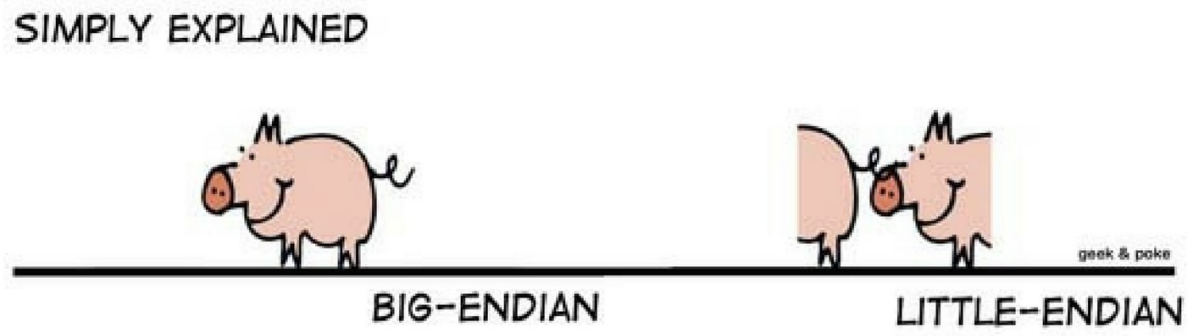
- gethostbyname_ex()

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

- gethostbyaddr()

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

Little endian, big endian



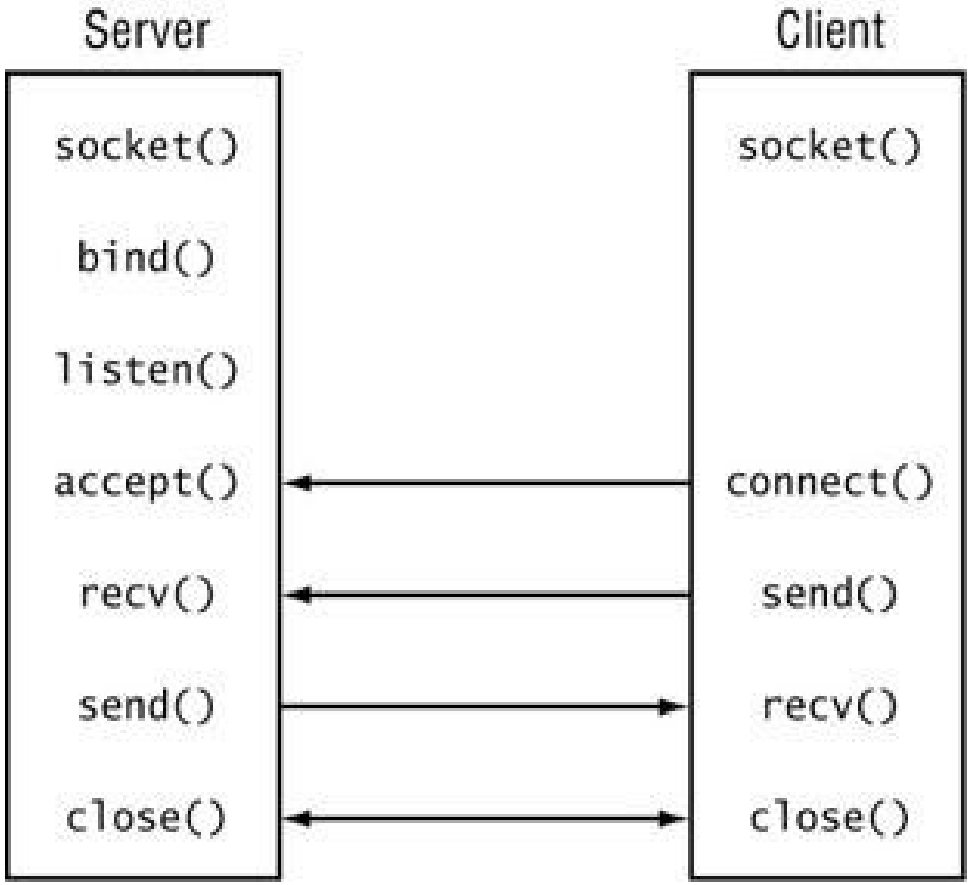
oxCAFEBABE
will be stored as
CA | FE | BA | BE

oxCAFEBABE
will be stored as
BE | BA | FE | CA

www.thebittheories.com

- Encoding 16 and 32 bit unsigned integers
  - htons(), htonl() – host to network short / long
  - ntohs(), ntohl() – network to host short / long

TCP – Connection oriented, reliable



TCP

- socket()

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- bind()

```
server_address = ('localhost', 10000)
sock.bind(server_address)
```

- listen()

```
sock.listen(1)
```

- accept()

```
connection, client_address = sock.accept()
```

TCP

- send(), sendall()

```
connection.sendall(data)
```

- recv()

```
data = connection.recv(16)
```

- close()

```
connection.close()
```

- connect()

```
server_address = ('localhost', 10000)
sock.connect(server_address)
```

Exercise – 1

- Write a client and a server applications.
- The client sends a „Hello server” message to the server and the server replies with a „Hello client” message.
- Then the client closes the connection and terminates.

Sending binary data

- Transforming data into binary

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

- Converting back to data

```
import struct
unpacker = struct.Struct('I 2s f')
unpacked_data = unpacker.unpack(data)
```

- Note: I integer, f float, Xs string of X characters

Excercise – 2

- Create a client-server application where the client sends two numbers and an operator. The server does the computation and sends the result back to the client. The message should be transmitted ina binary format.

Select

- setblocking() or settimeout()

```
connection.setblocking(0)      # or connection.settimeout(1.0)
```

- select()

```
inputs = [ server ]
outputs = [ ]
timeout=1
readable, writable, exceptional = select.select(inputs, outputs, inputs, timeout)
…
for s in readable:
        if s is server:    #new client connect
                connection, client_address = server.accept()
                print('New client from %s:%d' % client_address)
                inputs.append( connection )
        else:
                ….          #handle client
```

Excercise - 3

- Extend the previous calculator to support the communication with multiple clients in parallel.