Embedded Applications on Intelligent Systems

Fang Yuan (yfang@nju.edu.cn)

School of Electronic Science and Engineering
Nanjing University
Nanjing 210046

1. 计算机网络

Python 的网络功能

完备的网络功能是 Linux 的特点之一. 基本 Linux 系统提供了多种联网手段. Python 是互联网编程语言, 支持多层次的网络编程功能.

本课程涉及的网络知识主要讨论以下两方面:

- Linux 系统的网络结构
- Python TCP/IP 通信 (套接字)

IP 地址

每一个网络设备都有一个唯一的物理设备编号, 称作 MAC 地址.

在计算机网络系统中, 联网的计算机 (主机) 都被分配了一个网络地址, 称为 IP 地址. 几乎所有的网络通信协议都建立在 IP 地址上.

- 第 4 版 IP 协议 (IPv4) 使用 32 bits 表示一个地址, 最多可支持 2^{32} (大约是 42 亿) 个网络设备.
- IPv4 地址在 2011 年已基本瓜分完毕, 其后仅在有限的范围内注销与再分配.
- 目前 IPv4 协议主要通过局域组网方式支持更多的设备, 即：多个子网 IP 地址共享一个公网地址
- 第 6 版 IP 协议 (IPv6) 使用 128bits 表示一个地址

本章主要讨论 IPv4

ifconfig 显示地址

Linux 常用 ifconfig命令显示和设置 IP 地址:

```
$ ifconfig
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
       inet 192.168.2.102
       netmask 255.255.255.0
       broadcast 192.168.2.255
       ether 30:10:b3:99:c3:10
```

以上显示了网络设备名 (wlan0)、IP 地址 (192.168.2.102)、掩码 (255.255.255.0)、广播地址 (192.168.2.255) 和 MAC 地址 (30:10:b3:99:c3:10)

ifconfig 设置地址

32bits IP 地址常写成 “点分十进制” 形式: 32bits 分成 4 个字节, 每个字节写成一个十进制, 中间用点分隔.

设置 IP 地址需要超级用户权限.

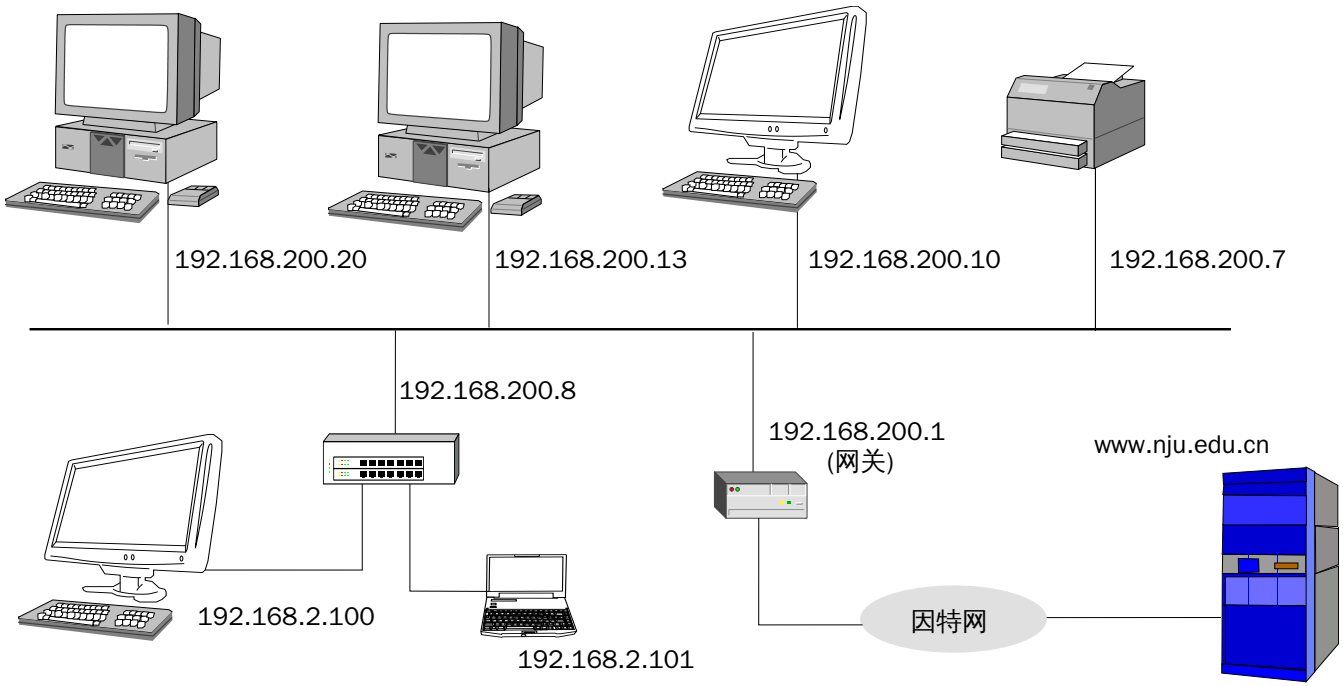
```
# ifconfig wlan0 192.168.2.100
```

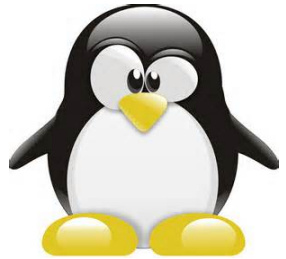
以上设置方法被称作 “静态分配”.

局域网中通常会有一个 DHCP (动态主机配置协议) 服务器, 可以为接入的主机自动分配 (又称 “动态分配”) 一个 IP 地址. 相比于静态分配, 动态分配可以避免地址冲突.

1. 计算机网络

网络拓扑结构





Linux 远程联网

Linux 提供了多种远程联网功能：

- telnet
- ssh
- vncviewer
- ...

利用 Linux 的网络功能, 可以在本地 PC 端远程登录树莓派系统进行操作. (这里的 “远程” 不一定是地理距离)

1. 计算机网络

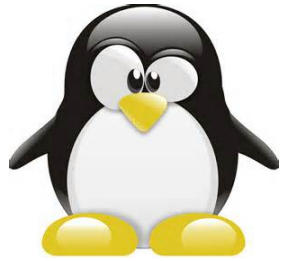
TCP/IP 协议

同一个计算机有多种网络服务需求, 不同的服务通过 “端口” 加以区分. 例如, HTTP 使用 80 端口, TELNET 使用 23 端口, SSH 使用 22 端口, VNC 使用 5900 端口.....

在 TCP/IP 网络通信中, 套接字代表了一个网络接口端点, 套接字由 IP 地址和端口号组成.

套接字的性质与文件描述符相同, 从而对网络的操作就变成了对文件的操作.

Python 的套接字模块是 socket



创建服务器

创建服务器的流程:

1. 创建套接字: socket() 函数
2. 绑定固定的端口: bind() 函数
3. (TCP 协议) 设置监听队列最大值: listen() 函数
4. 等待连接请求: accept() 函数. 该函数返回连接描述符, 用于和客户端进行数据交换.
5. 数据读写: read()/write()
6. 通信结束, 关闭连接描述符和套接字: close() 函数

简化的服务器程序

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.bind(('', 9999,))          # server doesn't need to specify
s.listen(5)                  # set back log
c, addr = s.accept()         # wait a client to connect
print ('connection␣request␣from␣', addr)
c.send('hello,␣client!')
c.close()                    # close this connection
```

客户端

客户端流程:

1. 创建套接字: socket() 函数
2. 客户端无需绑定端口, 系统会自动选择一个
3. 向服务器 (IP 地址 + 端口号) 发出连接请求: connect() 函数. 连接成功后, 客户端通过套接字描述符和服务器通信
4. 数据读写: read()/write()
5. 通信结束, 关闭套接字: close() 函数

简化的客户端程序

```
import socket

# create IPv4, TCP socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.connect(('192.168.88.1', 9999))
msg = s.recv(1024)          # read 1024 byte from socket
print(msg)
s.close()
```