네트워크 프로그래밍

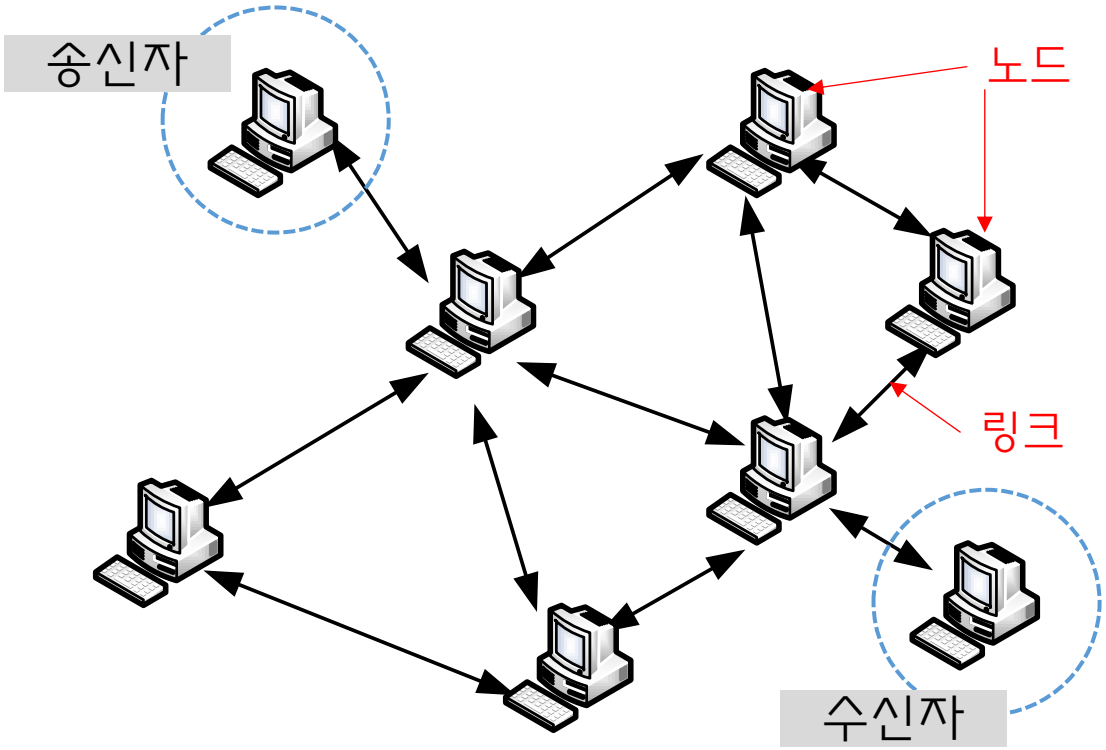
- 네트워크 개요 및 Python 네트워크 프로그래밍 모듈 -

순천향대학교 사물인터넷학과

네트워크

■ 네트워크란?

- 정보와 자원 공유를 위해 링크로 연결된 노드(컴퓨터, 스마트폰, IoT 디바이스 등)들의 집합

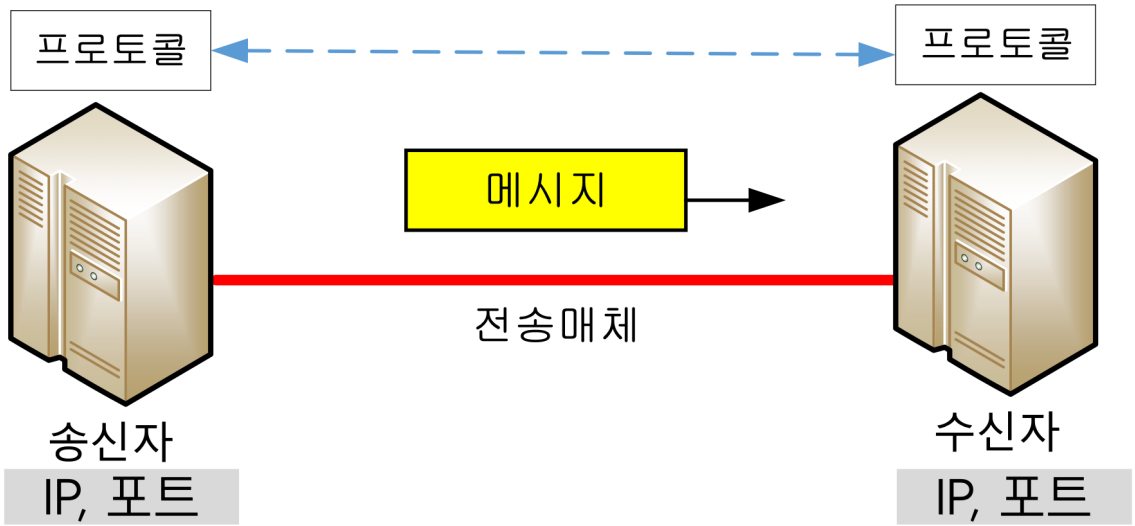


송신자-수신자의 입장

네트워크는 정보(bit)를 보내고 받게 해주는 서비스 인프라

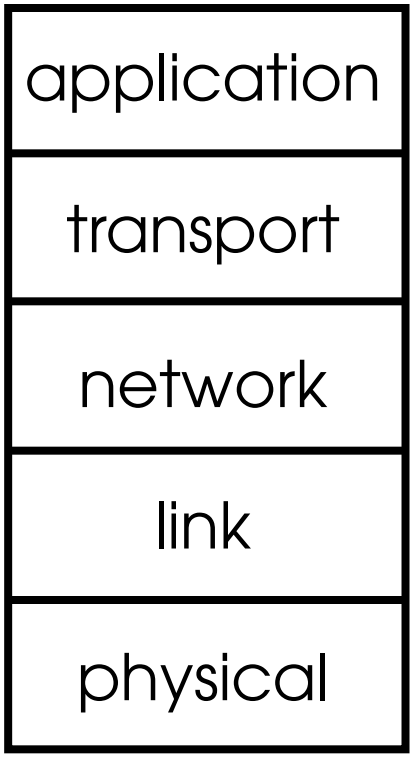
네트워크의 3가지 중요 요소

- 주소
  - 데이터를 주고 받을 원격 컴퓨터와 서비스를 지정할 수 있어야 함
- 연결
  - 상호 간에 전송매체(유선, 무선)를 통해 연결되어 있어야 함
- 프로토콜
  - 적절한 순서로 상호 간에 이해할 수 있게 데이터를 송수신하여야 함



인터넷 프로토콜 스택 Internet Protocol Stack

- 애플리케이션(응용) 계층Application layer
  - 네트워크 애플리케이션을 지원
  - HTTP, SMTP, IMAP
- 트랜스포트(전송) 계층Transport layer
  - 프로세스 간 데이터 전송
  - TCP, UDP
- 네트워크 계층Network layer
  - 발신지에서 수신지까지 데이터그램을 라우팅
  - IP, 라우팅 프로토콜
- 링크 계층link layer
  - 경로 상의 인접한 노드 간 데이터 전송
  - Ethernet, WiFi
- 물리 계층Physical layer
  - 물리 매체를 통해 비트를 전송



파이썬 네트워킹

■ 파이썬 네트워킹

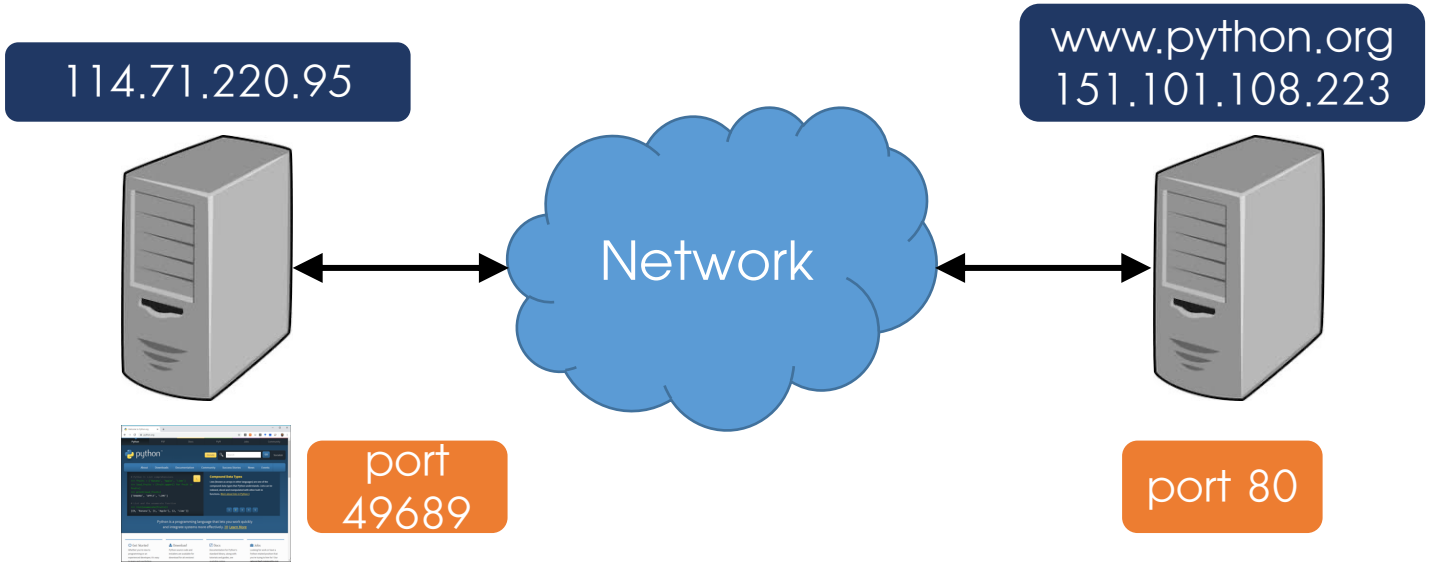
- 네트워크 프로그래밍은 Python의 주된 용도 중 하나
- 파이썬 표준 라이브러리는 네트워크 프로토콜, 데이터 인코딩/디코딩 등 네트워크 서비스를 위한 다양한 네트워크 관련 표준 모듈을 제공
- 파이썬으로 네트워크 프로그램을 작성하는 것이 C/C++보다 훨씬 간단함

■ 파이썬 네트워킹 레벨

- 파이썬은 두 가지 레벨의 네트워크 서비스를 제공함
- Low level: 소켓socket 기반 서비스 제공
  - ✓ connection-oriented: TCP
  - ✓ connectionless: UDP
- High level: 다양한 애플리케이션 계층 프로토콜을 위한 라이브러리 제공
  - ✓ HTTP, HTTPS, SSH, SMTP, POP3, FTP, ...

네트워크 주소 network address

- 네트워크 주소
  - 컴퓨터는 호스트네임hostname과 IP 주소를 가짐
  - 프로그램/서비스는 포트 번호port number를 가짐



포트 번호 port numbers

- 널리 쓰이는 서비스의 서버 쪽 포트는 미리 정해져 있음
  - 알려진 포트 번호well-known ports라고 부름
  - 클라이언트 쪽 포트는 OS가 임의로 할당함

포트	서비스
21	FTP
22	SSH
23	Telnet
25	SMTP (Mail)
80	HTTP (Web)
110	POP3 (Mail)
443	HTTPS (Web)

- netstat
  - 네트워크 연결을 보여주는 명령어
  - '-a' 등 다양한 옵션 제공

```
dhkim@workstation:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 workstation:domain      *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 localhost:6010          *:*                     LISTEN
tcp        0      0 localhost:6011          *:*                     LISTEN
tcp        0      0 localhost:6012          *:*                     LISTEN
tcp        0      0 114.71.220.46:ssh       114.71.220.95:59664     ESTABLISHED
```

ssh 서버(114.71.220.46:ssh)에
ssh 클라이언트(114.71.220.95:59664)가 접속하였음

네트워크 연결 network connections

■ 네트워크 연결

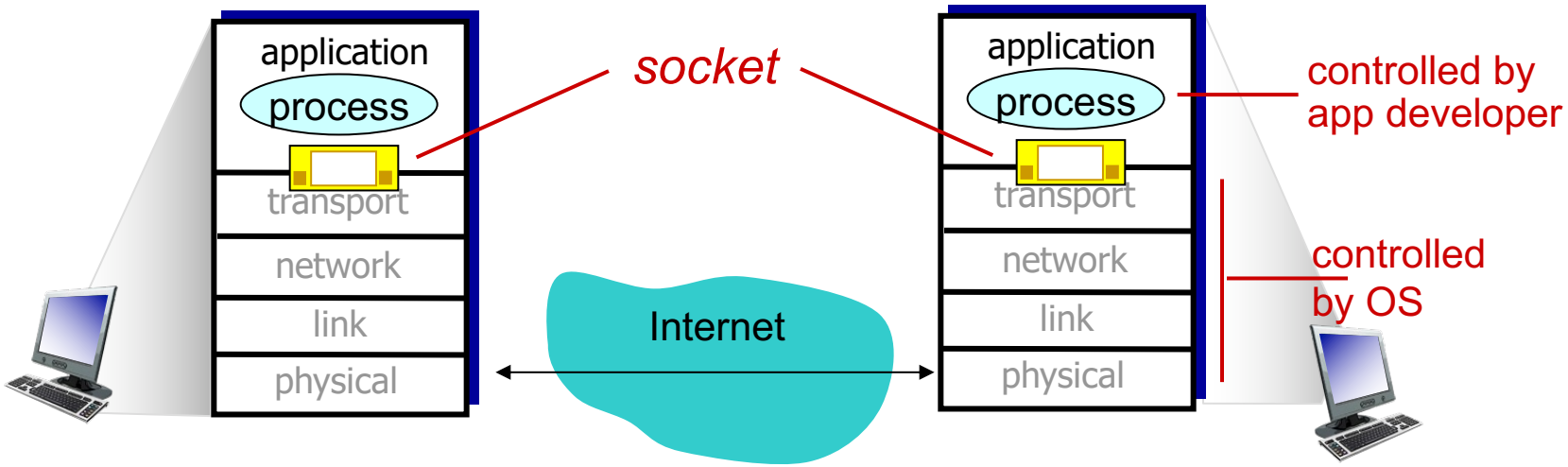
- 네트워크를 통해 연결된 양 끝단 노드의 주소로 표현
- 양 끝단 노드의 주소는 (host, port) 형태로 표현함
- 예1) 송신노드: (www.python.org, 80)
- 예2) 수신노드: (114.71.220.95, 59664)
- 네트워크 연결은 노드의 쌍 "(114.71.220.95, 59664), (www.python.org, 80)" 으로 나타낼 수 있음
  - ✓ 즉, 1개의 네트워크 연결은 송신 노드의 IP, 송신 노드의 포트 번호, 수신 노드의 IP, 수신 노드의 포트 번호를 이용해서 구별 가능

Q. 아래 3개의 연결은 동일한 연결인가?

- 예1) (114.71.220.95, 59664), (www.python.org, 80)
- 예2) (114.71.220.95, 59665), (www.python.org, 80)
- 예3) (114.71.220.96, 59664), (www.python.org, 80)

소켓 socket

- 프로세스는 소켓을 통해 메시지를 보내고 받음
  - 소켓은 호스트의 애플리케이션 계층과 트랜스포트 계층 간의 인터페이스
  - 프로세스는 "집", 소켓은 "출입구"에 비유
    - ✓ 송신 프로세스는 출입구(소켓) 바깥 네트워크로 메시지를 밀어냄
    - ✓ 송신 프로세스는 수신 프로세스에 있는 소켓으로 메시지를 전달하기 위해 하위 계층의 서비스에 의존
  - 소켓은 애플리케이션과 네트워크 사이의 API Application Programming Interface



소켓

■ 소켓을 이용한 2가지 기본 통신 방법

- 신뢰성 있는 데이터 전송 (TCP)
  - ✓ 클라이언트-서버 간에 연결을 설정한 이후, 데이터 송수신
- 비신뢰적 데이터 전송 (UDP)
  - ✓ 연결 설정 없이 데이터 송수신
- 소켓 생성 시 TCP/UDP 중 어느 것을 사용할 지 설정할 수 있음

■ Python 소켓 지원

- `socket` 모듈 제공
- `socket()`: 소켓 객체 생성 함수
- `gethostbyname(), gethostbyaddr()` 등 다양한 통신 관련 함수 제공
- `send()/recv()` 등의 데이터 송수신 함수 제공

네트워크 관련 파이썬 표준 모듈

- 파이썬은 TCP/IP 통신과 네트워크 서비스를 위한 다양한 표준 모듈을 제공

모듈	내용
ipaddress	IP 주소 관련 작업 모듈
socket	데이터 송수신을 위한 소켓 통신 모듈
select	입출력을 효율적으로 처리하기 위한 모듈
selectors	입출력 다중화 모듈
socketserver	네트워크 서버를 작성하기 위한 모듈
asyncio	비동기 입출력 모듈
urllib	URL 관련 프로그래밍 모듈
http	HTTP 프로토콜을 이용한 프로그래밍 모듈

ipaddress 모듈

ipaddress 모듈

- 파이썬에서 IP 주소를 표현하고 처리하기 위해 사용하는 모듈
- ip_address()
  - ✓ IP 주소 객체를 생성하는 함수
  - ✓ IPv4 또는 IPv6를 자동 인식

```
>>> import ipaddress
>>> addr4 = ipaddress.ip_address('192.0.2.1')
>>> addr4
IPv4Address('192.0.2.1')

>>> addr6 = ipaddress.ip_address('2001:A8::1')
>>> addr6
IPv6Address('2001:a8::1')

>>> addr4.version
4
>>> addr6.version
6
```

ipaddress 모듈

- ip_network()
  - ✓ 네트워크 주소 객체를 생성하는 함수
    - 네트워크 주소: 특정 노드의 주소가 아니라 네트워크 전체에 대한 주소
    - 예) 114.71.220.0/24

```
>>> import ipaddress
>>> net = ipaddress.ip_network('114.71.220.0/24')
>>> net
IPv4Network('114.71.220.0/24')

>>> net.with_netmask
'114.71.220.0/255.255.255.0'

>>> net.num_addresses
256

>>> net.netmask
IPv4Address('255.255.255.0')

>>> net.hostmask
IPv4Address('0.0.0.255')
```

ipaddress 모듈

- 네트워크에서 사용 가능한 호스트 주소 알아보기

```
>>> net = ipaddress.ip_network('114.71.220.0/24')
>>> for x in net.hosts():
        print(x)

114.71.220.1
114.71.220.2
...
114.71.220.254
```

- in 연산자를 이용하여 호스트 주소가 네트워크에 속하는지 알아보기

```
>>> net = ipaddress.ip_network('114.71.220.0/24')
>>> addr = ipaddress.ip_address('114.71.220.95')
>>> addr in net
True

>>> addr = ipaddress.ip_address('192.168.0.1')
>>> addr in net
False
```

socket 모듈

■ socket이란?

- 통신 채널의 종단점
- 통신을 하기 위해서는 프로그램이 상대편 socket에 접속해야 함

■ socket 모듈

- BSD 소켓 인터페이스에 대한 액세스를 제공
- 모든 현대 유닉스 시스템(리눅스 포함), 윈도우, MacOS에서 사용 가능
- 호스트에 관한 정보를 알아내고, 소켓 접속 및 데이터 송수신 함수를 제공

호스트 정보 알아내기

■ 호스트 정보 관련 함수

모듈	내용
socket.gethostname()	실행 중인 호스트 이름을 반환
socket.gethostbyname(*hostname*)	*hostname*을 IPv4 주소로 변환
socket.gethostbyname_ex(*hostname*)	*hostname*을 IPv4 주소로 변환하고 추가 정보를 제공 (hostname, aliaslist, ipaddrlist)
socket.gethostbyaddr(*ip_address*)	*ip_address*에 대한 호스트 정보 반환 (hostname, aliaslist, ipaddrlist)
socket.getfqdn(*name*)	*name*에 대한 정규화된 도메인 이름을 반환

```
>>> import socket
>>> name = socket.gethostname()
>>> name
'DESKTOP-C119NVN'

>>> socket.gethostbyname(name)
'192.168.56.1'
```

호스트 정보 알아내기

```
>>> socket.gethostbyname('homepage.sch.ac.kr')
'220.69.189.98'

>>> socket.gethostbyname_ex('homepage.sch.ac.kr')
('homepage.sch.ac.kr', [], ['220.69.189.98'])

>>> socket.gethostbyaddr('220.69.189.98')
('homepage.sch.ac.kr', [], ['220.69.189.98'])

>>> socket.getfqdn('220.69.189.98')
'homepage.sch.ac.kr'

>>> socket.getfqdn('www.daum.net')
'www.daum.net'

>>> socket.getfqdn('www.google.com')
'del03s01-in-f4.1e100.net'
```

호스트 정보 알아내기

■ 여러 사이트의 IP 주소를 확인하는 프로그램

```
import socket

HOSTS = [
    'www.sch.ac.kr',
    'homepage.sch.ac.kr',
    'www.daum.net',
    'www.google.com',
    'iot'
]

for host in HOSTS:
    try:
        print('{} : {}'.format(host, socket.gethostbyname(host)))
    except socket.error as msg:
        print('{} : {}'.format(host, msg))
```

실행결과

```
www.sch.ac.kr : 220.69.189.98
homepage.sch.ac.kr : 220.69.189.98
www.daum.net : 121.53.224.15
www.google.com : 142.250.207.68
iot : [Errno 8] nodename nor servname provided,
or not known
```

인터넷 서비스 정보 알아내기

- 서비스service
  - 응용 계층 프로토콜 (예: http, smtp, ssh 등)
  - 각 서비스는 하위 트랜스포트 계층 프로토콜을 사용하고, 포트 번호가 할당됨

- socket.getservbyname(servicename[,protocolname])
  - servicename: http, ssh 등과 같은 애플리케이션 계층 프로토콜
  - protocolname: tcp, udp와 같은 트랜스포트 계층 프로토콜
  - 인터넷 서비스 이름에 대한 포트번호 반환

```
>>> socket.getservbyname('http')
80
>>> socket.getservbyname('ssh')
22
>>> socket.getservbyname('https')
443
>>> socket.getservbyname('ftp')
21
```

인터넷 서비스 정보 알아내기

- socket.getservbyport(port)
  - 포트 번호에 대한 서비스 이름을 반환

```
>>> socket.getservbyport(80)
'http'
>>> socket.getservbyport(25)
'smtp'
```

  - 예제

```
import socket
for port in [80, 443, 21, 25, 143, 993, 110, 995]:
    url ='{}://example.co.kr/'.format(socket.getservbyport(port))
    print('{:4d}'.format(port), url)
```

실행결과

```
  80 http://example.co.kr/
 443 https://example.co.kr/
  21 ftp://example.co.kr/
  25 smtp://example.co.kr/
 143 imap://example.co.kr/
 993 imaps://example.co.kr/
 110 pop3://example.co.kr/
 995 pop3s://example.co.kr/
```

IP 주소 변환

IPv4 주소 표현

- 일반적으로 114.71.220.95와 같이 8비트씩 끊어서 표현
  - ✓ 파이썬에서는 문자열로 처리 ('114.71.220.95')
  - ✓ 사람이 알아보기 쉬우나, 문자열이라 처리가 어려움
- 실제로 컴퓨터 내부적으로는 32비트 정수형으로 처리
- 예) '114.71.220.95' (문자열) ←→ 0x7247DC5F (정수)

변환 함수

- inet_aton(): 문자열 주소를 4바이트 bytes 객체로 변환
- inet_ntoa(): 4바이트 bytes 객체를 문자열 주소로 변환

```
import binascii
import socket
import sys

for string_address in ['114.71.220.95']:
    packed = socket.inet_aton(string_address)
    print ('Original:', string_address)
    print ('Packed  :', binascii.hexlify(packed))
    print ('Unpacked:', socket.inet_ntoa(packed))
```

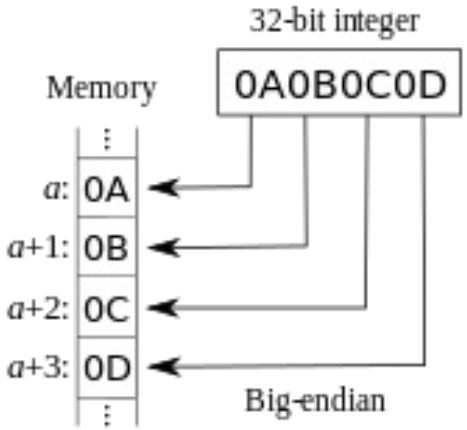
실행결과

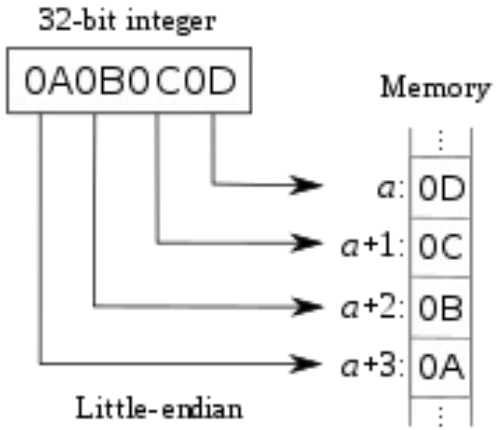
```
Original: 114.71.220.95
Packed  : b'7247dc5f'
Unpacked: 114.71.220.95
```

바이트 순서 변환

엔디언endian

- 컴퓨터 메모리에 여러 개의 연속된 대상을 배열하는 방법
- 즉, 여러 바이트로 구성된 데이터를 메모리에 저장할 때 어떤 바이트 순서로 저장할 것인지는 나타냄
- 빅 엔디언big endian
  - ✓ 사람이 숫자를 쓰는 방법과 같이 큰 단위의 바이트가 앞에 오는 방법
  - ✓ 모토로라 CPU 계열, 또는 네트워크 전송 시 사용
- 리틀 엔디언little endian
  - ✓ 작은 단위의 바이트가 앞에 오는 방법
  - ✓ 인텔 CPU





바이트 순서 변환

■ 왜 바이트 순서 변환?

- 네트워크 상에는 다양한 종류의 호스트가 존재함
- 각 호스트는 CPU에 따라 빅 엔디언 또는 리틀 엔디언을 사용함
  - ✓ 각 호스트에서 사용하는 바이스 순서를 "호스트 바이트 순서Host Byte Order"라고 함
- 빅 엔디언을 사용하는 호스트에서 리틀 엔디언을 사용하는 호스트로 데이터 전송 시 문제가 발생함
  - ✓ 예) 호스트 1(빅 엔디언) 0x1234를 호스트 2(리틀 엔디언)으로 전송
  - ✓ 호스트 2는 0x3412로 인식함

■ 해결 방법

- 네트워크로 데이터 전송 시 "정해진 바이트 순서"로 전송하도록 약속함
- 네트워크 바이트 순서Network Byte Order
  - ✓ "빅 엔디언" 사용

■ 동작

- 호스트는 데이터 전송 시 네트워크 바이트 순서(빅 엔디언)로 변환해서 전송
- 호스트는 데이터 수신 시 호스트 바이트 순서(빅 또는 리틀 엔디언)로 변환해서 데이터 사용

바이트 순서 변환

■ 바이트 순서 변환 함수

함수	내용
socket.ntohl(x)	4바이트 양의 정수를 네트워크 바이트 순서에서 호스트 바이트 순서로 변환
socket.ntohs(x)	2바이트 양의 정수를 네트워크 바이트 순서에서 호스트 바이트 순서로 변환
socket.htonl(x)	4바이트 양의 정수를 호스트 바이트 순서에서 네트워크 바이트 순서로 변환
socket.htons(x)	2바이트 양의 정수를 호스트 바이트 순서에서 네트워크 바이트 순서로 변환

```
>>> a = 1234
>>> hex(a)
'0x4d2'
>>> b = socket.htons(a)
>>> hex(b)
'0xd204'
>>> c = socket.ntohs(b)
>>> hex(c)
'0x4d2'
```

Thank you

Questions?

Contact: daeheekim@sch.ac.kr