Network basics

MM4220 game server programming

Kyoung-chul Kim

Jung, Nai Hoon

Outline

- Computer Network
- Packet
- Socket

What is the ‘computer network’?

- Definition
  - Two or more computers connected together using a telecommunication system.[wikipedia]

- Type of computer network
  - Ethernet
    - 10 Base 5, 10 Base 2, 1000 Base-T
  - Wi-Fi : IEEE802.11(abgn)
    - 무선랜, OlleWifi, U+Zone
  - 3G, 4G : 휴대폰
    - EV-DO, HSPA, WiMAX, Wibro, LTE
  - Other
    - Bluetooth, USB, IEEE1394, Myrinet, Infiniband

What is the ‘packet’?

- Definition
  - Formatted block of information carried by a computer network. [wikipedia]
- Why?
  - Easy Error detection
    - There can be any noise or jamming signal in physical medium.
    - 예) Check Sum, CRC
  - Host addressing
    - Need Address field (need format)
    - Modern networks usually connect three or more host computers.
  - Line sharing
    - Many computers shares same network lines
    - Holding a line without data wastes the bandwidth

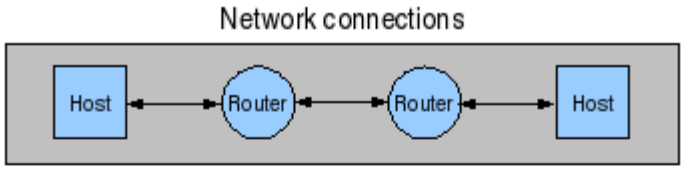
OSI reference model [생략]

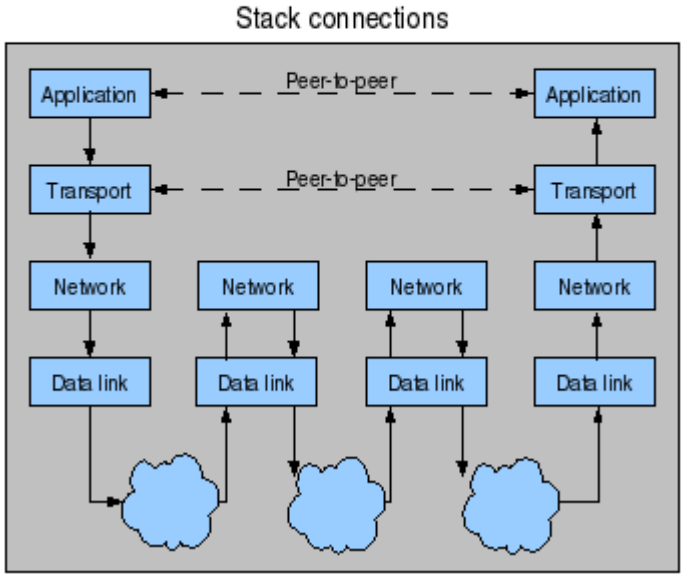
- History
  - From International Standards Organization(ISO), 1977
  - Open System Interconnection(OSI) networking suite:
    - An abstract model of networking(7-layer reference model)
    - A set of protocols
- OSI layers
  - Layer 7: Application layer
  - Layer 6: Presentation layer
  - Layer 5: Session layer
  - Layer 4: Transport layer
  - Layer 3: Network layer
  - Layer 2: Data link layer
  - Layer 1: Physical layer
- First try for layering, too old, no one use now

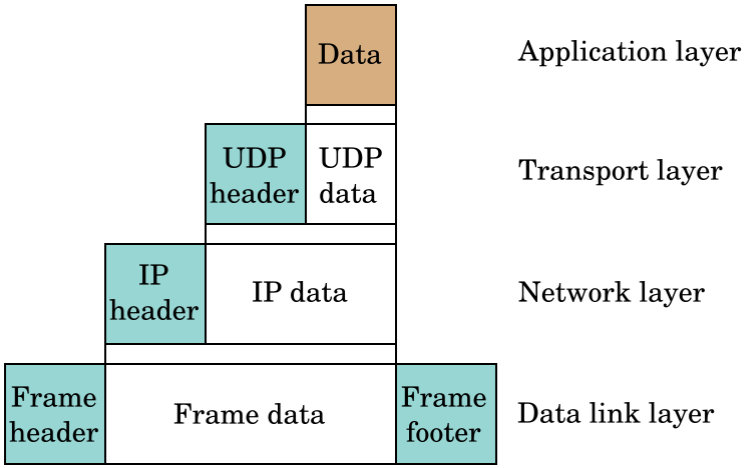
Internet protocol suite[wikipedia]

- History
  - From Defense Advanced Reseach Projects Agency(DARPA), 1970s
- Layers
  - Application
    - DHCP, DNS, TFTP, FTP, HTTP, IMAP, IRC, NNTP, POP3, SMTP, SNMP, SSH, TELNET, ECHO, BGP, RPC, PPTP, ...
  - Transport
    - TCP, UDP, DCCP, SCTP, IL, RUDP, ...
  - Network
    - IP(IPv4, IPv6), ICMP, IGMP, RSVP, Ipsec, ...
  - Data link
    - ATM, Ethernet, FDDI, Frame Relay, PPP, ARP, RARP, ...

Internet protocol suite(cont.)



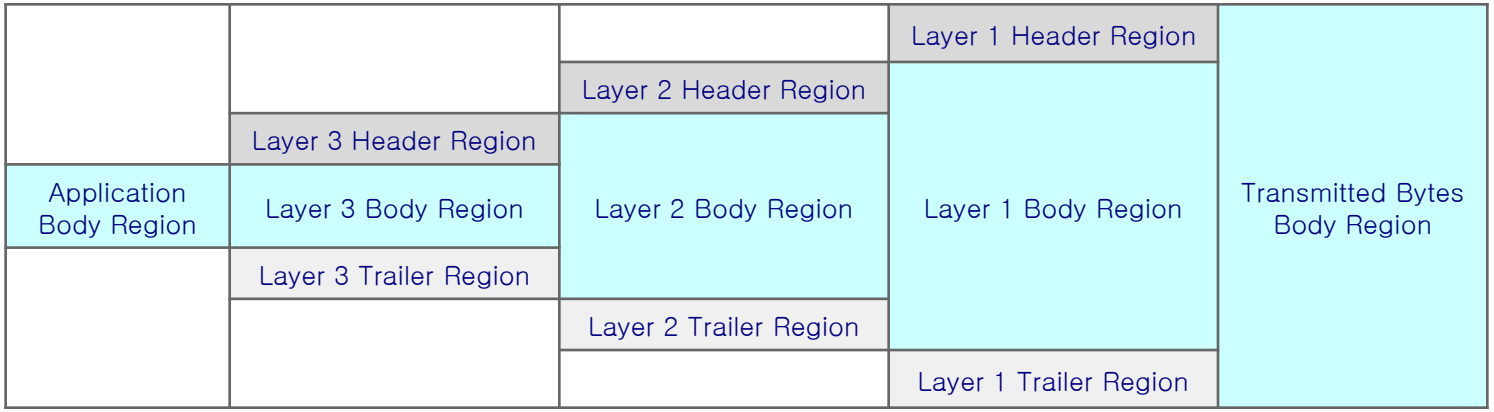




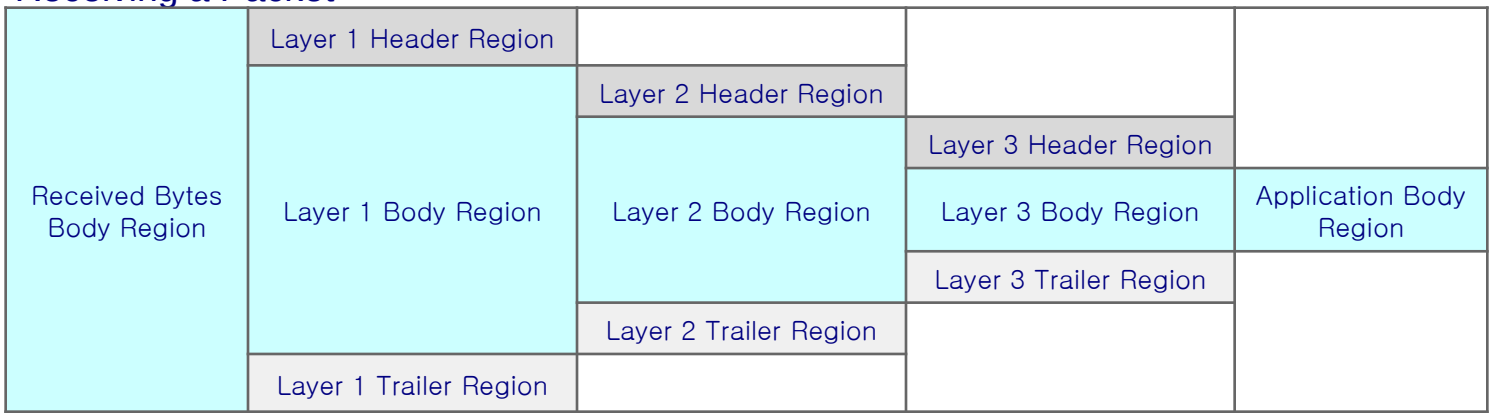
Ref: http://en.wikipedia.org/wiki/TCP/IP

Transmitting and receiving a packet

Transmitting a Packet

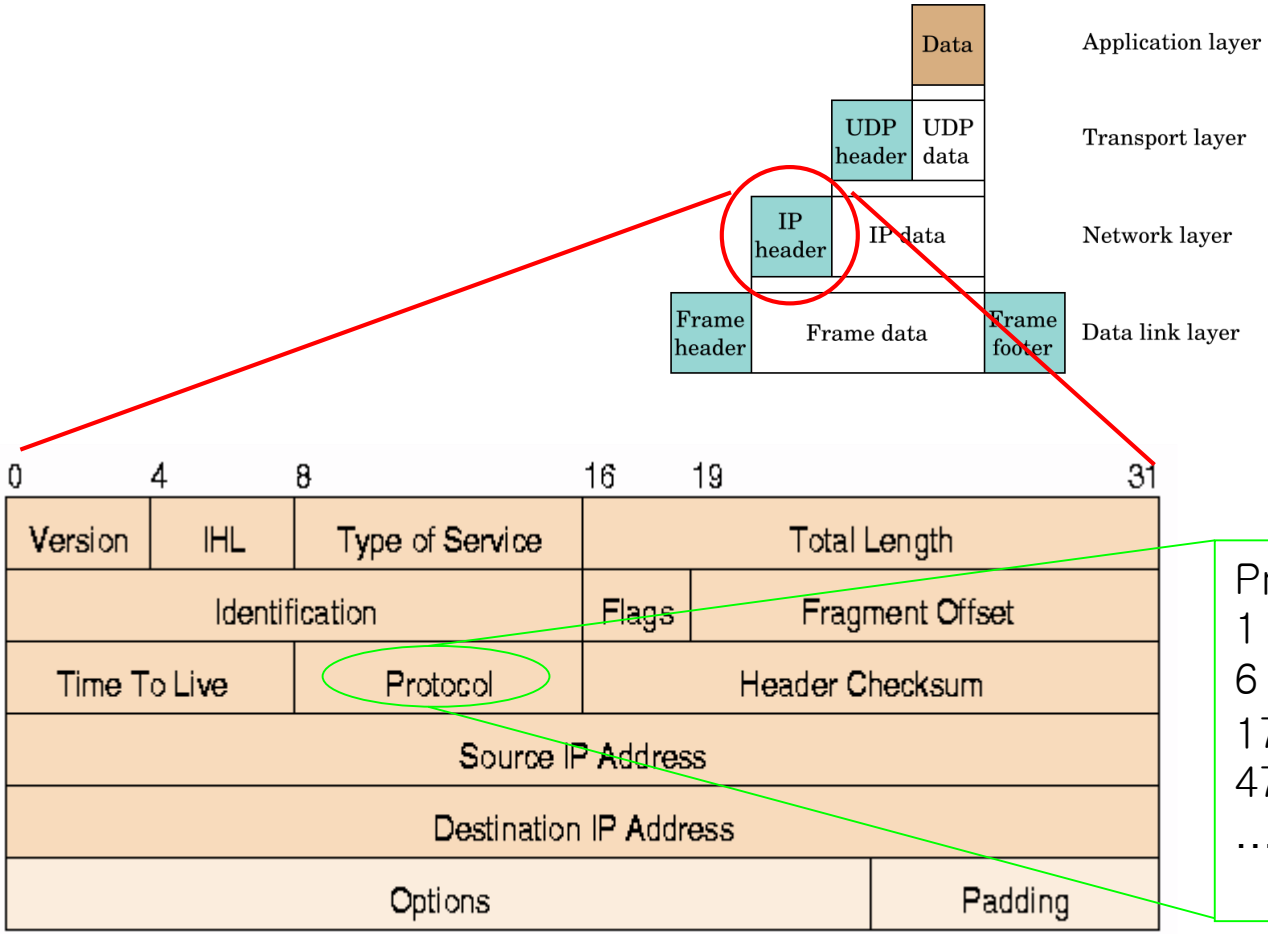


Receiving a Packet



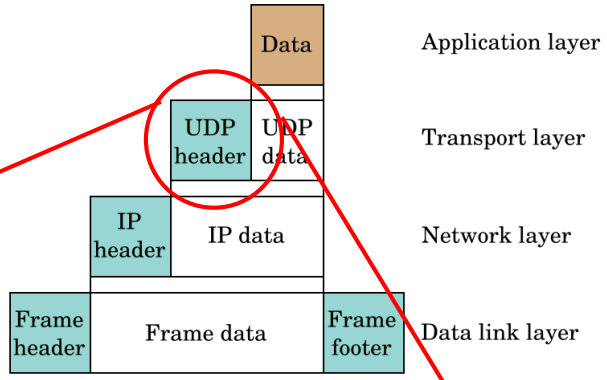
http://www.eventhelix.com/RealtimeMantra/PatternCatalog/protocol_packet_design_pattern.htm

Example: IP packet header



Ref: http://www.freesoft.org/CIE/Course/Section3/7.htm

Example: UDP packet header

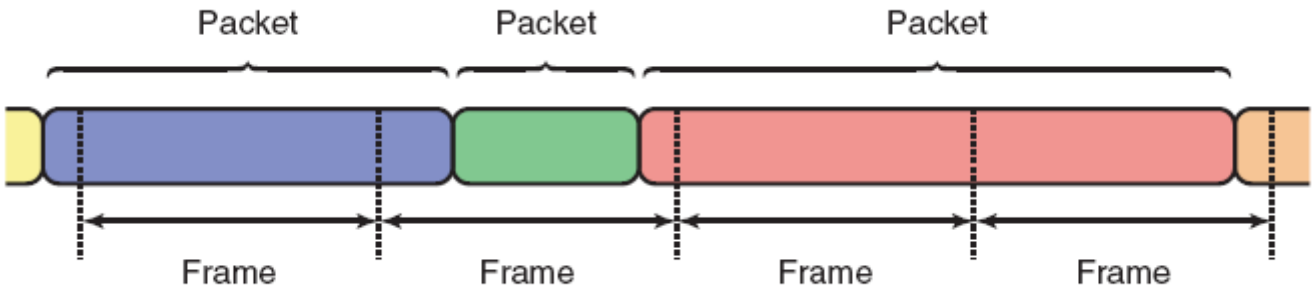


+	Bits 0 – 15	16 – 31
0	Source Port	Destination Port
32	Length	Checksum
64	Data	

Port:
11 – SYSTAT
13 – DAYTIME
53 – DNS
67 – DHCP
123–NTP
…

Ref: http://en.wikipedia.org/wiki/User_Datagram_Protocol

Packets and frames

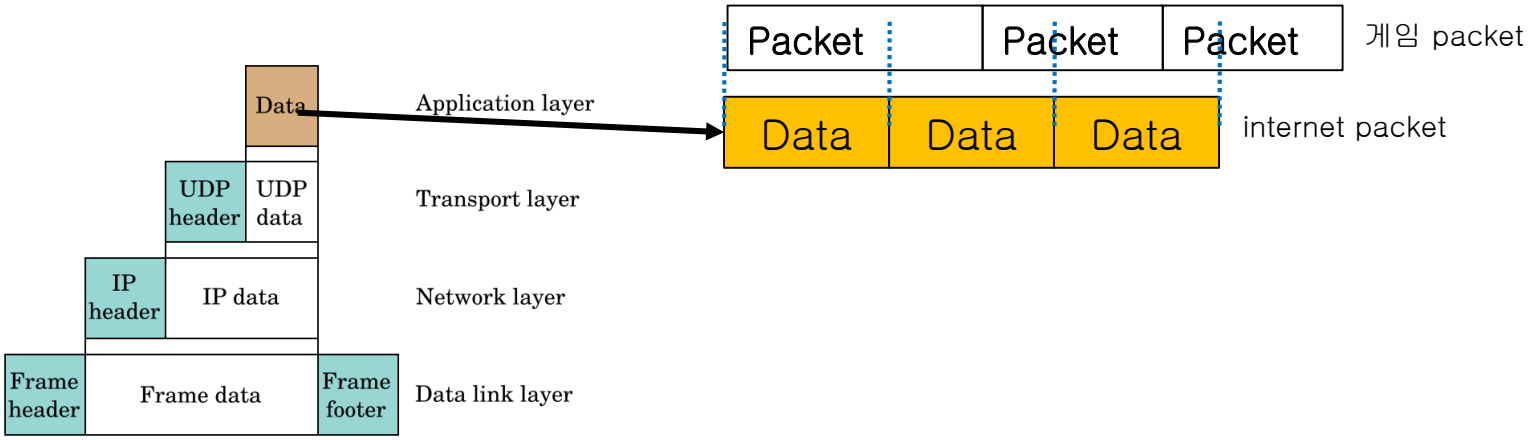


Ref: http://en.wikipedia.org/wiki/Image:Packets-and-Frames_illustration.png

Packet: logical unit of tx/rx
Frame: physical unit of tx/rx

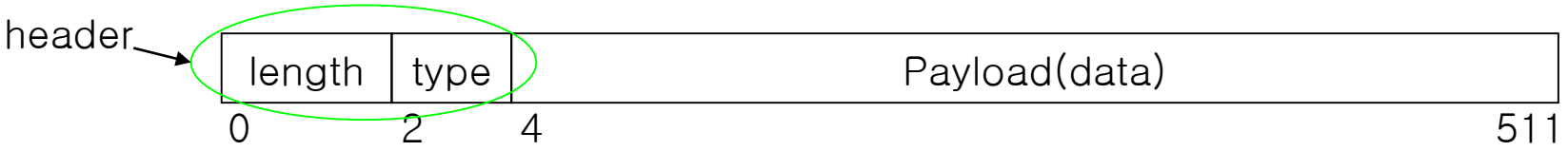
Packet in Game

- 필요성
  - 오고 가는 정보를 그룹 지어서 관리한다.
    - 연관된 정보를 하나로 묶어서 해당 처리 모듈로 전송할 수 있게 한다.
- 성격
  - 네트워크의 packet과는 다른 이야기



Packet in Game

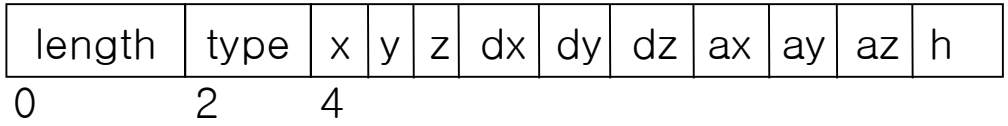
Our packet structure:



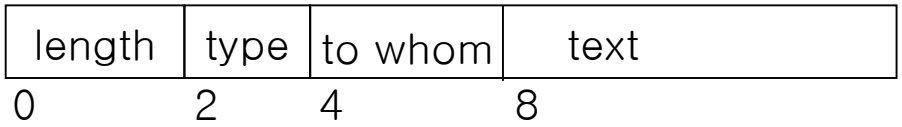

```
short buf[256]

buf[0] = length;
buf[1] = type;
buf[2] = any_other_info;
...
send( fd, buf, (size_t)buf[0], 0 );
```

We need many type of packet

For move



For chat




```
short buf[256]

buf[0] = length;
buf[1] = OP_CHAT;
*((long *)(&buf[2])) = to_whom;
strncpy( &buf[4], text, length-8);

...

send( fd, buf, (size_t)buf[0], 0 );
```

```
short buf[256]

buf[0] = length;
buf[1] = OP_MOVE;
*((float *)(&buf[2])) = x;
*((float *)(&buf[4])) = y;
*((float *)(&buf[6])) = z;
*((float *)(&buf[8])) = dx;
*((float *)(&buf[10])) = dy;
*((float *)(&buf[12])) = dz;
*((float *)(&buf[14])) = ax;
*((float *)(&buf[16])) = ay;
*((float *)(&buf[18])) = az;
*((int *)(&buf[20])) = h;
...
send( fd, buf, (size_t)buf[0], 0 );
```

Make packet struct/union

For chat

```
typedef struct {
        short length;
        short type;
        short to_whom;
        char text[PKTLEN-6];
} t_chat;
```

For move

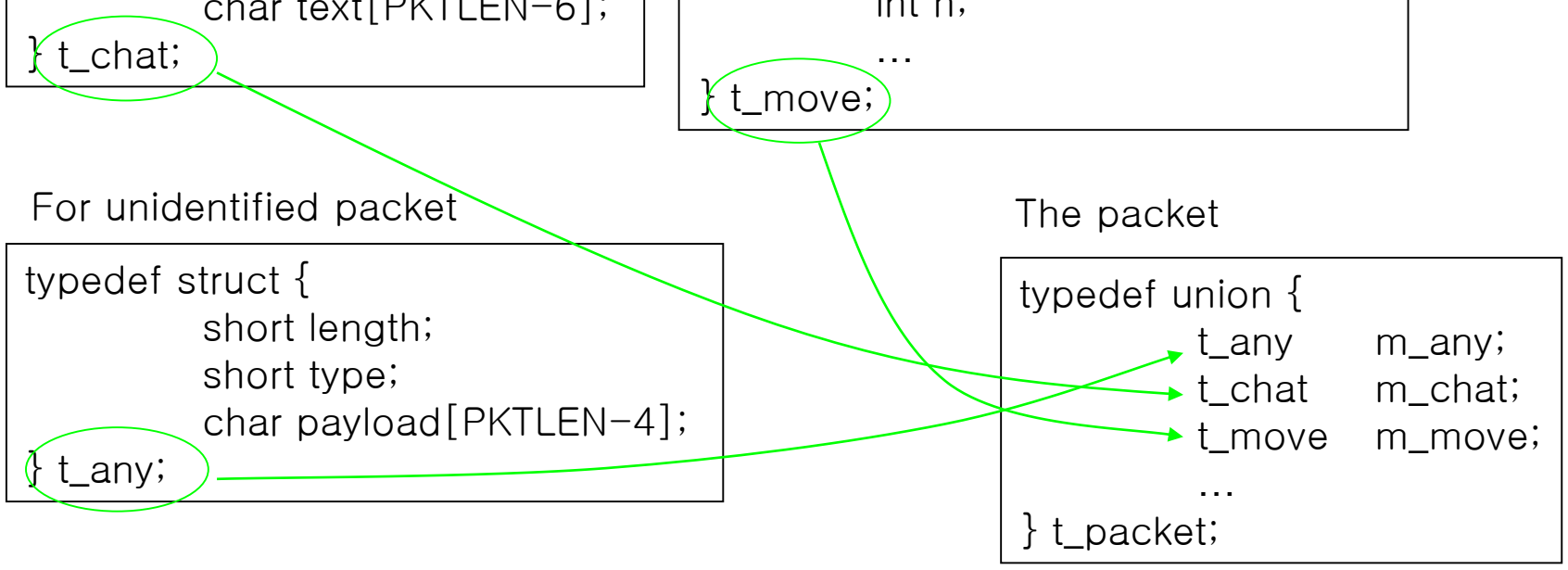
```
typedef struct {
        short length;
        short type;
        float x,y,z,dx,dy,xz,ax,ay,az
        int h;
        ...
} t_move;
```

For unidentified packet

```
typedef struct {
        short length;
        short type;
        char payload[PKTLEN-4];
} t_any;
```

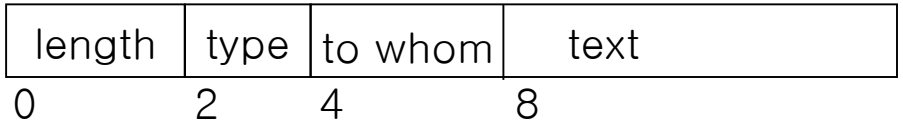
The packet

```
typedef union {
        t_any     m_any;
        t_chat    m_chat;
        t_move    m_move;
        ...
} t_packet;
```



Comparison

For chat

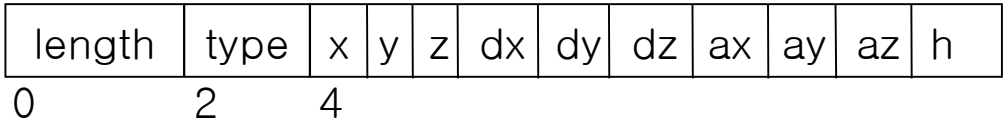



```
short buf[256]

buf[0] = length;
buf[1] = OP_CHAT;
*((long *)(&buf[2])) = to_whom;
strncpy( &buf[4], text, length-8);

...

send( fd, buf, (size_t)buf[0], 0 );
```

```
t_packet pkt;

pkt.m_chat.length = length;
pkt.m_chat.type = OP_CHAT;
pkt.m_chat.to_whom = to_whom;
strncpy( pkt.m_chat.text, text, length-8);

...

send( fd, &pkt, (size_t)pkt.m_chat.length, 0 );
```

Comparison

For move




```
short buf[256]

buf[0] = length;
buf[1] = OP_MOVE;
*((float *)(&buf[2])) = x;
*((float *)(&buf[4])) = y;
*((float *)(&buf[6])) = z;
*((float *)(&buf[8])) = dx;
*((float *)(&buf[10])) = dy;
*((float *)(&buf[12])) = dz;
*((float *)(&buf[14])) = ax;
*((float *)(&buf[16])) = ay;
*((float *)(&buf[18])) = az;
*((int *)(&buf[20])) = h;
...
send( fd, buf, (size_t)buf[0], 0 );
```

```
t_packet pkt;

pkt.m_move.length = length;
pkt.m_move.type = OP_MOVE;
pkt.m_move.x = x;
pkt.m_move.y = y;
pkt.m_move.z = z;
pkt.m_move.dx = dx;
pkt.m_move.dy = dy;
pkt.m_move.dz = dz;
pkt.m_move.ax = ax;
pkt.m_move.ay = ay;
pkt.m_move.az = az;
pkt.m_move.h = h;
...
send( fd, &pkt, (size_t)pkt.m_move.length, 0 );
```

Another Method

- OpenSource Projects
  - Example : Protocol Buffer, Flat Buffers
  - Various Language : C, C# , Java, JSON
  - Various Platform : Windows, Linux, Iphone, Unity3d

Socket

- Definition
  - Communication end-point
  - From 4.2BSD UNIX, 1983
  - *De facto* standard API for internet application
- Everything in UNIX is a file
  - File descriptor for network communication
  - socket() returns socket descriptor
  - read()/write() vs. send()/recv()
  - Internet socket, Unix socket, X.25 socket...
- Two type of Internet socket
  - SOCK_STREAM - TCP
  - SOCK_DGRAM - UDP

Socket

- Definition
  - Communication end-point
  - From 4.2BSD UNIX, 1983
  - *De facto* standard API
- Everything in UNIX is a file
  - File descriptor for network communication
  - socket() returns socket descriptor
  - read()/write() vs. send()/recv()
  - Internet socket, Unix socket, X.25 socket...
- Two type of Internet socket
  - SOCK_STREAM - TCP
  - SOCK_DGRAM - UDP

System calls 1/2

- WSAStartup(WORD wVersionRequested, LPWSADATA lpWSAData) : 소켓 사용 시작
- WSACleanup() : 소켓 사용 끝
- SOCKET socket(int af, int type, int protocol) : 소켓 생성
- Int closesocket(SOCKET s) : 소켓 반환
- Int connect(SOCKET s, const struct sockaddr FAR* name, int namelen) : 소켓 끼리 연결
- Int bind(SOCKET s, const struct sockaddr FAR* name, int namelen) : 소켓에 주소지정
- Int listen(SOCKET s, int backlog) : 소켓 연결을 기다림

System calls 2/2

- SOCKET accept(SOCKET s, struct sockaddr FAR* addr, int FAR* addrlen) : 소켓 접속을 받는다.
- Int send (SOCKET s, const char FAR *msg, int len, int flags) : 소켓으로 데이터를 보낸다.
- Int recv(SOCKET s, char FAR *buf, int len, int flags) : 소켓으로 온 데이터를 읽는다.