Raw Sockets and ICMP

Srinidhi Varadarajan

Topics

- Raw sockets
- Internet Control Message Protocol (ICMP)
- Code Examples
  - Ping
  - Traceroute

Raw Sockets

- Usually, sockets are used to build applications on top of a transport protocol
  - Stream sockets (TCP)
  - Datagram sockets (UDP)
- Some applications need to access a lower layer protocol
  - Control protocols built on IP rather than UDP or TCP, such as ICMP and IGMP
  - Experimental transport protocols
- A “raw” socket allows direct access to IP
  - Used to build applications on top of the network layer

Creating a Raw Socket

- **Standard socket() call used to create a raw socket**
  - **Family is AF_INET, as for TCP or UDP**
  - **Socket type is SOCK_RAW instead of SOCK_STREAM or SOCK_DGRAM**
  - **Socket protocol needs to be specified, e.g. IPPROTO_ICMP (often left at 0 for UDP or TCP sockets)**

```
socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)
```

Socket Types

Stream socket	**SOCK_STREAM**	**1**
Datagram socket	**SOCK_DGRAM**	**2**
Raw protocol interface	**SOCK_RAW**	**3**
Reliably delivered message	**SOCK_RDM**	**4**
Sequenced packet stream	**SOCK_SEQPACKET**	**5**

Protocols

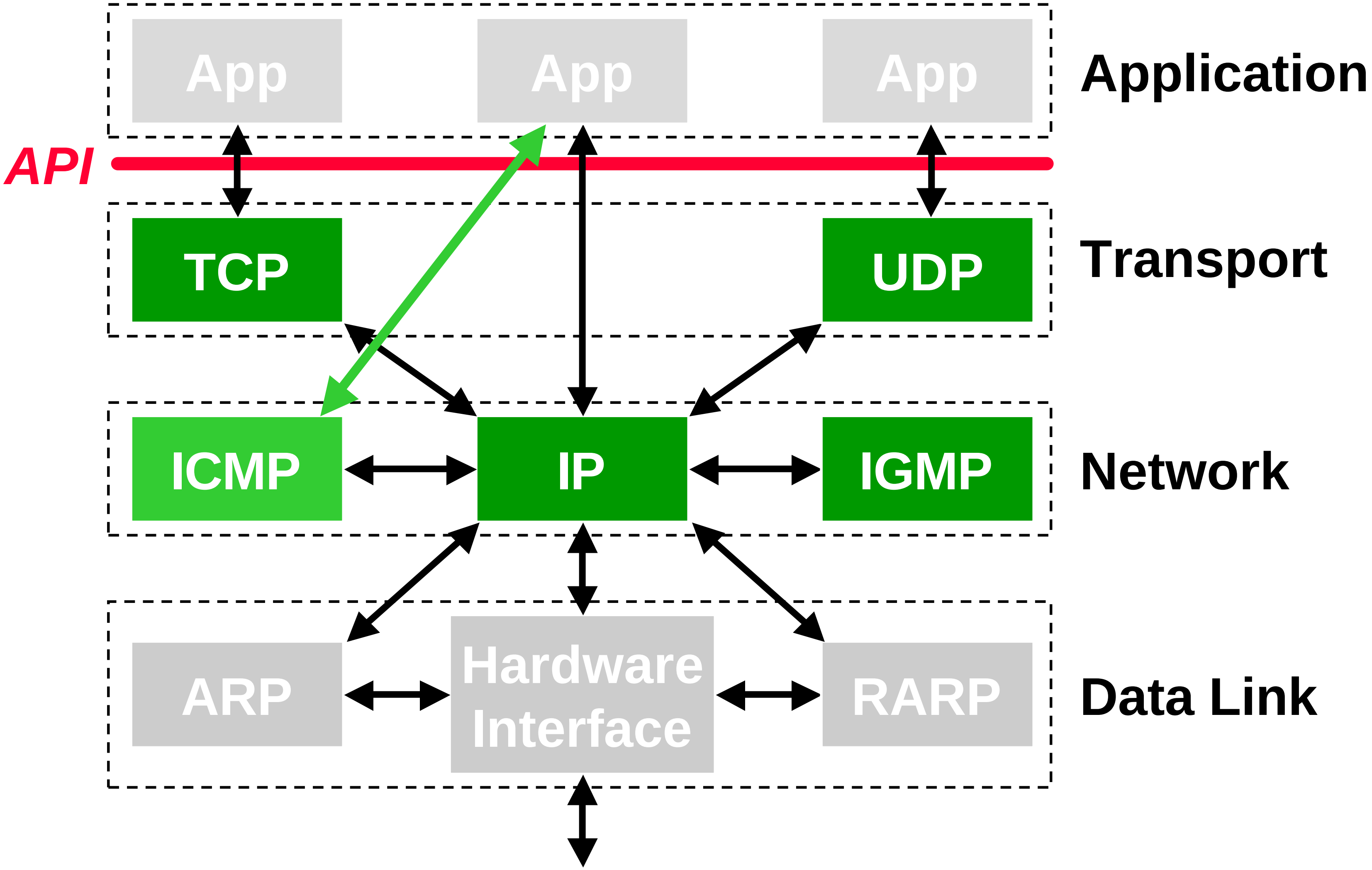
- **Protocol values**
  - **Used to define the Protocol field in the IP header**

IP (dummy)	IPPROTO_IP	0
ICMP	IPPROTO_ICMP	1
IGMP	IPPROTO_IGMP	2
Gateway	IPPROTO_GGP	3
TCP	IPPROTO_TCP	6
PUP	IPPROTO_PUP	12
UDP	IPPROTO_UDP	17
XND IDP	IPPROTO_IDP	22
Net Disk	IPPROTO_ND	77
Raw IP	IPPROTO_RAW	255

Internet Control Message Protocol

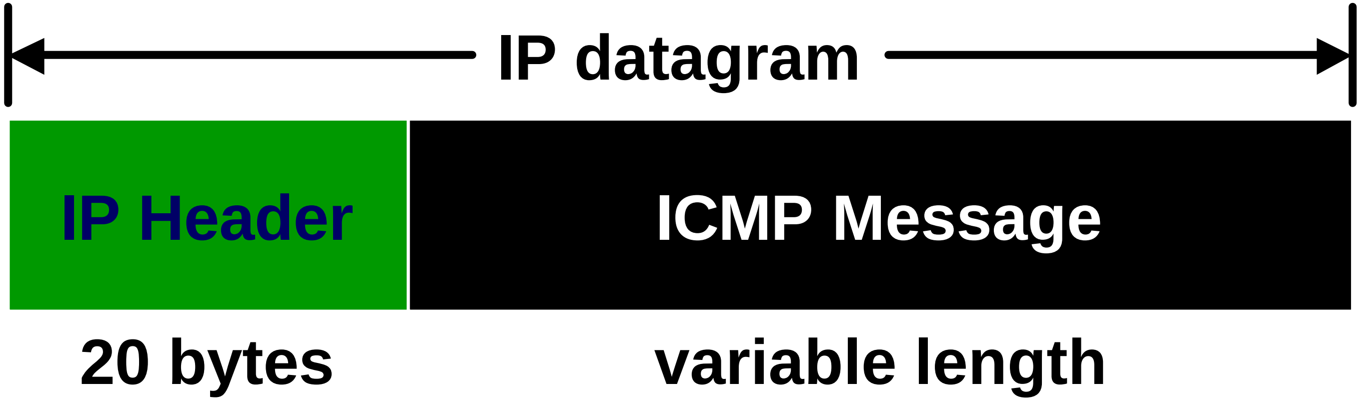
- ICMP defined in RFC 792
- ICMP messages
  - Query network node(s) for information
  - Report error conditions
- ICMP messages are carried as IP datagrams
  - ICMP “uses” or is “above” IP
- ICMP messages usually processed by IP, UDP, or TCP
  - IP, TCP, and UDP “use” or are above ICMP

ICMP in the TCP/IP Suite

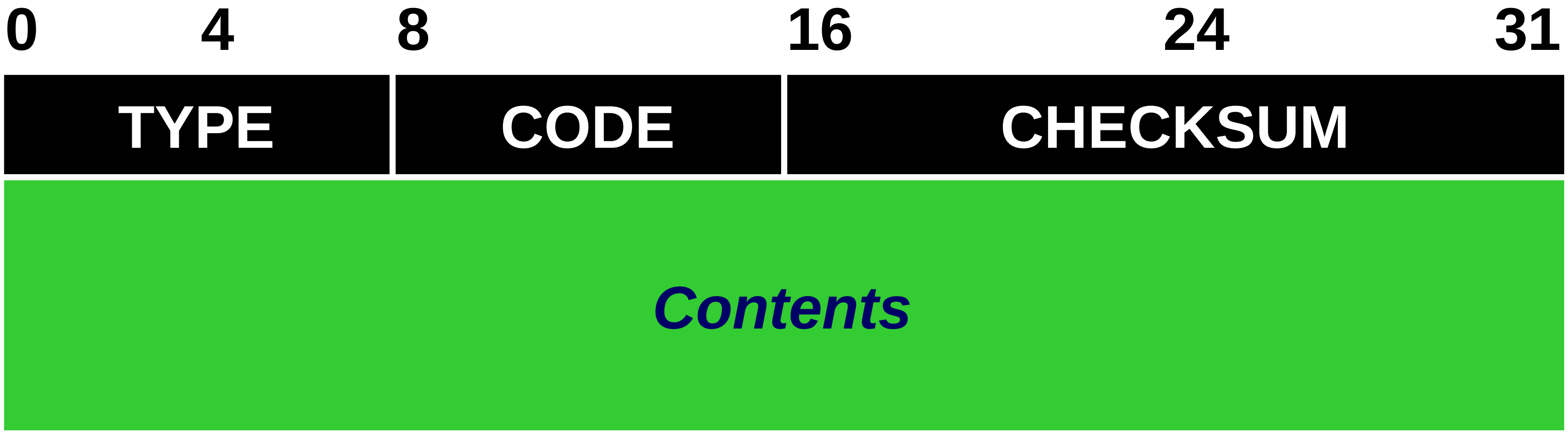


ICMP Message Format (1)

- ICMP messages are encapsulated in IP datagrams
  - IP-level routing use to move ICMP messages through a network
  - IP provides multiplexing/demultiplexing based on protocol number (IPPROTO_ICMP = 1)



ICMP Message Format (2)



0 4	8	16 24 31
TYPE	CODE	CHECKSUM
Contents		

- **TYPE: Type of ICMP message**
- **CODE: Used by some types to indicate a specific condition**
- **CHECKSUM: Checksum over full message**
- **Contents depend on TYPE and CODE**

Example ICMP Message Types

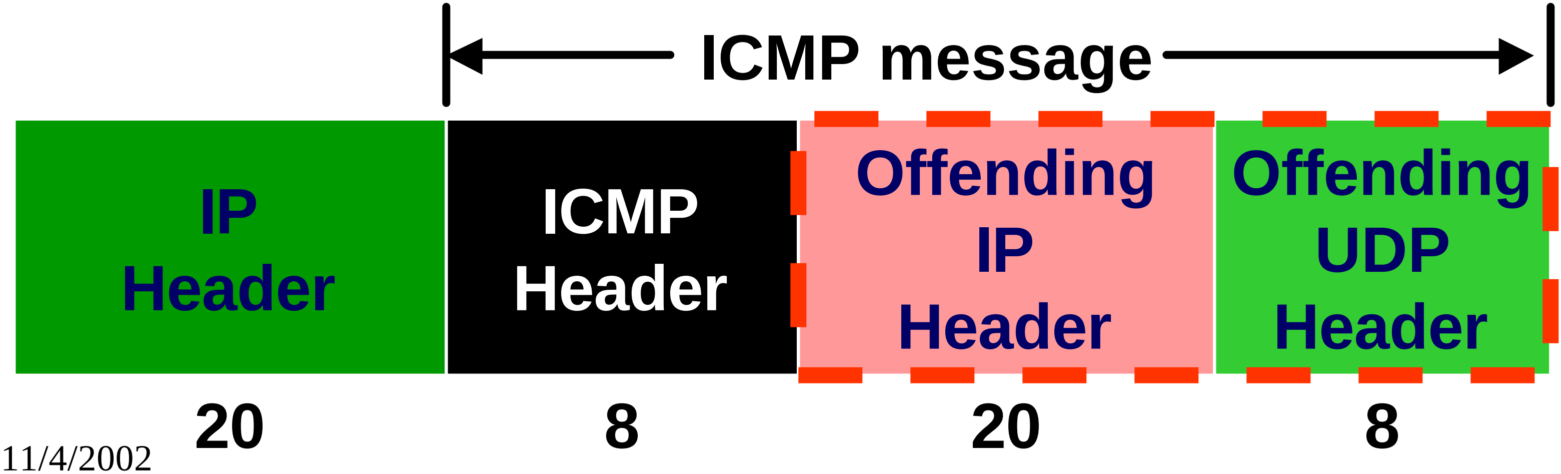
- Queries
  - TYPE = 8: Echo request
  - TYPE = 0: Echo reply
  - TYPE = 13: Time stamp request
  - TYPE = 14: Time stamp reply
- Errors
  - TYPE = 3: Destination unreachable
    - CODE = 0: Network unreachable
    - CODE = 1: Host unreachable
    - CODE = 2: Protocol unreachable
    - CODE = 3: Port unreachable
  - TYPE = 11: Time exceeded
    - CODE = 0: Time-to-live equals 0 in transit

Error Example: Port Unreachable

- Port unreachable error occurs when a receiving host receives a packet with an unknown (inactive) port number
- IP datagram is valid -- reaches addressed host
- UDP datagram contains a port that is not in use (e.g. 8000 and no application has a socket bound to an address with that port)
- UDP replies with an ICMP “Destination Unreachable/Port Unreachable” message
  - TYPE = 3, CODE =3

ICMP Error Messages

- ICMP error messages include header and first 8 bytes of offending IP datagram
  - All of IP header
    - Destination address, protocol number, etc.
  - For UDP, all of UDP header including source and destination port numbers
- ICMP message for port unreachable



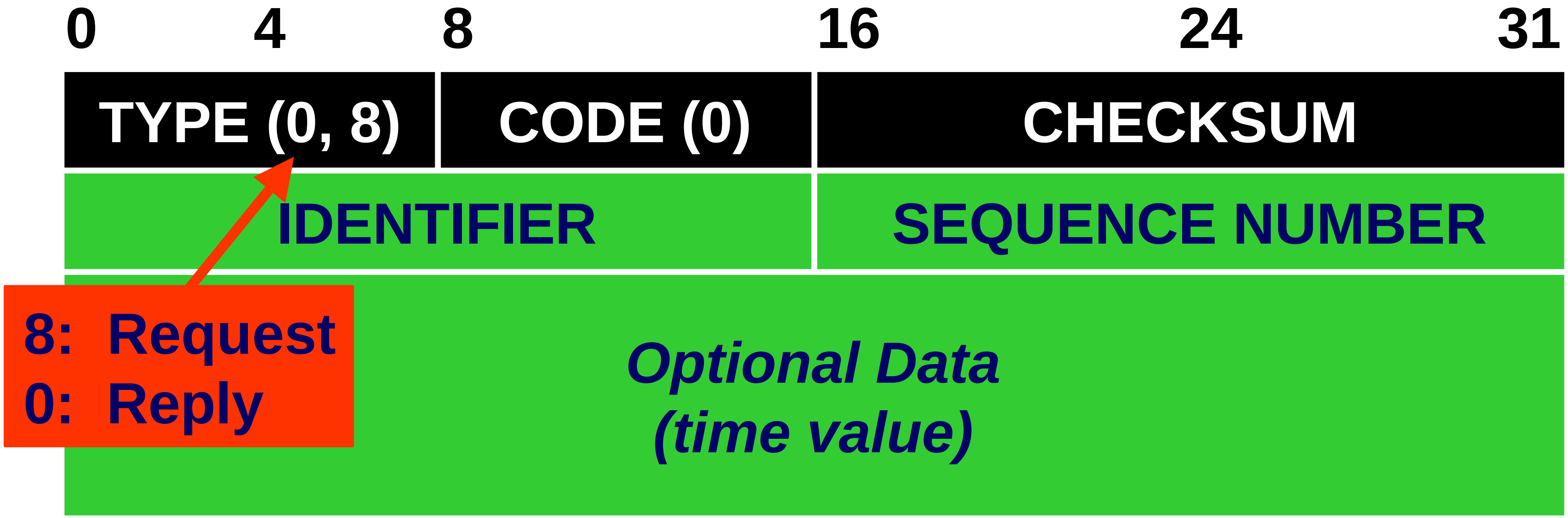
Ping Example

- “Ping” utility
  - Tests whether or not a host is reachable
  - Provides a round-trip time
  - Written by Mike Muuss in 1983 to diagnose network problems
- Operation
  - ICMP echo request (TYPE = 8) sent to host
  - Host replies with ICMP echo reply (TYPE = 0)
- Client-server roles
  - Host sending echo request is the *client*
  - Host sending echo reply is the *server*
  - Server usually implemented in TCP/IP code

Ping Algorithm

1) Initialize echo request
2) Send echo request
3) Wait for echo reply (or time out)
4) Receive reply
5) Report results
6) Go back to 1 until complete

Echo Request/Reply Format (1)



0 4	8	16 24 31
TYPE (0, 8)	CODE (0)	CHECKSUM
IDENTIFIER		SEQUENCE NUMBER
Optional Data (time value)		

8: Request
0: Reply

- **IDENTIFIER: Means to identify sending instance of "ping"**
  - Process id in UNIX
- **SEQUENCE NUMBER: Means to identify lost or misordered replies**

Echo Request/Reply Format (2)

- **Common ICMP echo reply/request header definition from icmp.h code example**

```
typedef struct tagICMPHDR
{
    u_char  Type;         // Type
    u_char  Code;         // Code
    u_short Checksum;     // Checksum
    u_short ID;           // Identification
    u_short Seq;          // Sequence
} ICMPHDR, *PICMPHDR;
```

Echo Request

- **Echo request will include**
  - **Common request/reply header**
  - **Time stamp (32 bits)**
  - **Filler data (REQ_DATASIZE bytes)**

```
typedef struct tagECHOREQUEST
{
  ICMPHDR icmpHdr;               // Header
  int     dwTime;                // Time
  char    cData[REQ_DATASIZE];// Fill data
} ECHOREQUEST, *PECHOREQUEST;
```

```
      static ECHOREQUEST  echo_req;
```

Initializing the Echo Request

```
echo_req.icmpHdr.Type          = ICMP_ECHOREQ;
echo_req.icmpHdr.Code          = 0;
echo_req.icmpHdr.Checksum      = 0;
echo_req.icmpHdr.ID            = id++;
echo_req.icmpHdr.Seq           = seq++;

// Fill in some data to send
memset(echo_req.cData, ' ', REQ_DATASIZE);

// Save tick count when sent (milliseconds)
echo_req.dwTime = gettime …;

// Put data in packet and compute checksum
echo_req.icmpHdr.Checksum = in_cksum(…);
```

Waiting for Echo Reply

- **Time-out is important since ping will often be used when a host is unreachable**
- **select() used with a time-out value to wait for echo reply**

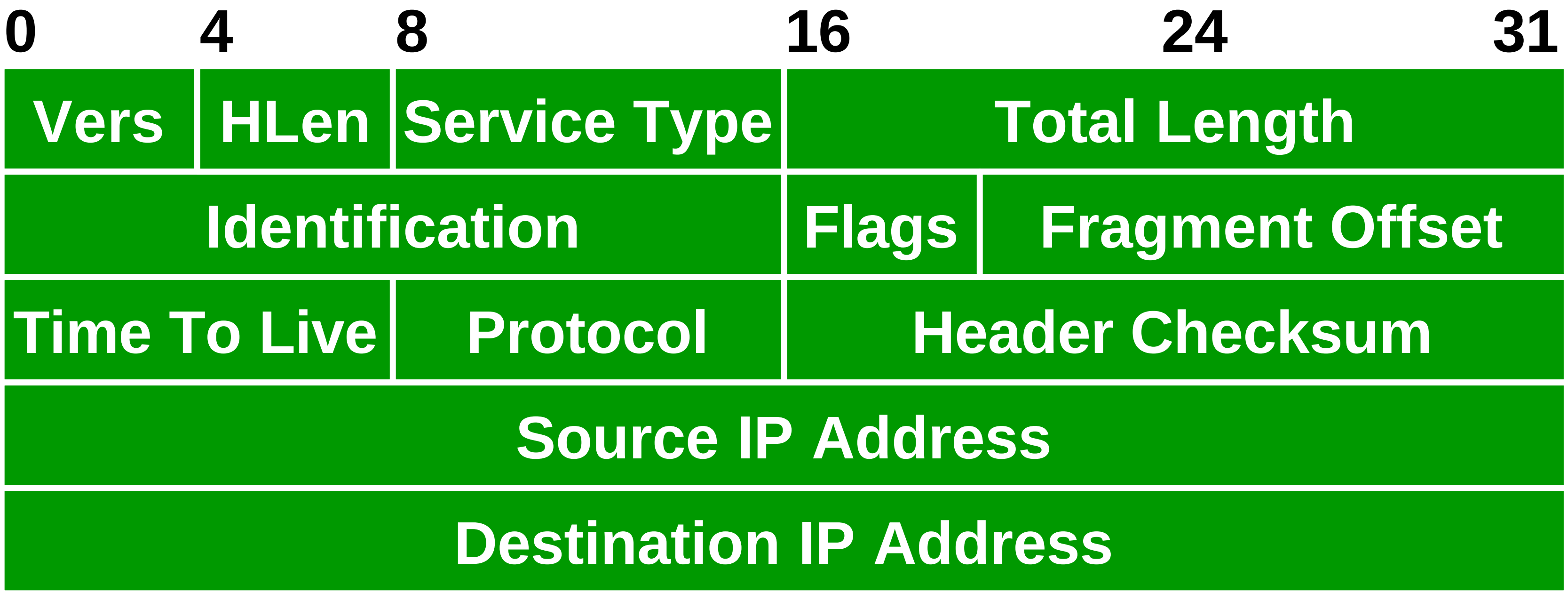
```
readfds.fd_count = 1;       // set size
readfds.fd_array[0] = raw; // socket set
timeout.tv_sec = 10;       // timeout (s)
timeout.tv_usec = 0;       // timeout (us)

if((rc = select(1, &readfds, NULL, NULL,
&timeout)) == SOCKET_ERROR)
  errexit("select() failed %d\n", perror());
```

Echo Reply

- Raw socket returns IP header
- Received datagram contains
  - IP header
  - ICMP echo request/reply header
  - Echo request message
  - Potentially, additional fill data

```
typedef struct tagECHOREPLY
{
   IPHDR        ipHdr;
   ECHOREQUEST  echoRequest;
   char         cFiller[256];
} ECHOREPLY, *PECHOREPLY;
```

IP Header (1)



0 4 8 16 24 31

Vers | HLen | Service Type | Total Length

Identification | Flags | Fragment Offset

Time To Live | Protocol | Header Checksum

Source IP Address

Destination IP Address

IP Header (2)

```
typedef struct tagIPHDR
{
   u_char  VIHL;           // Ver, Hdr length
   u_char  TOS;            // Type of service
   short   TotLen;         // Total length
   short   ID;             // Identification
   short   FlagOff;        // Flags, Frag off
   u_char  TTL;            // Time-to-live
   u_char  Protocol;       // Protocol
   u_short Checksum;       // Checksum
   struct  in_addr iaSrc; // Source IP addr
   struct  in_addr iaDst; // Dest IP addr
} IPHDR, *PIPHDR;
```

Extracting Results from Reply

- **Ping client can extract IP, ICMP, and echo information from the received datagram**

```
…
ECHOREPLY echo_reply;
…
type = echo_reply.echoRequest.icmpHdr.Type;
ttl  = echo_reply.ipHdr.TTL;
…
```

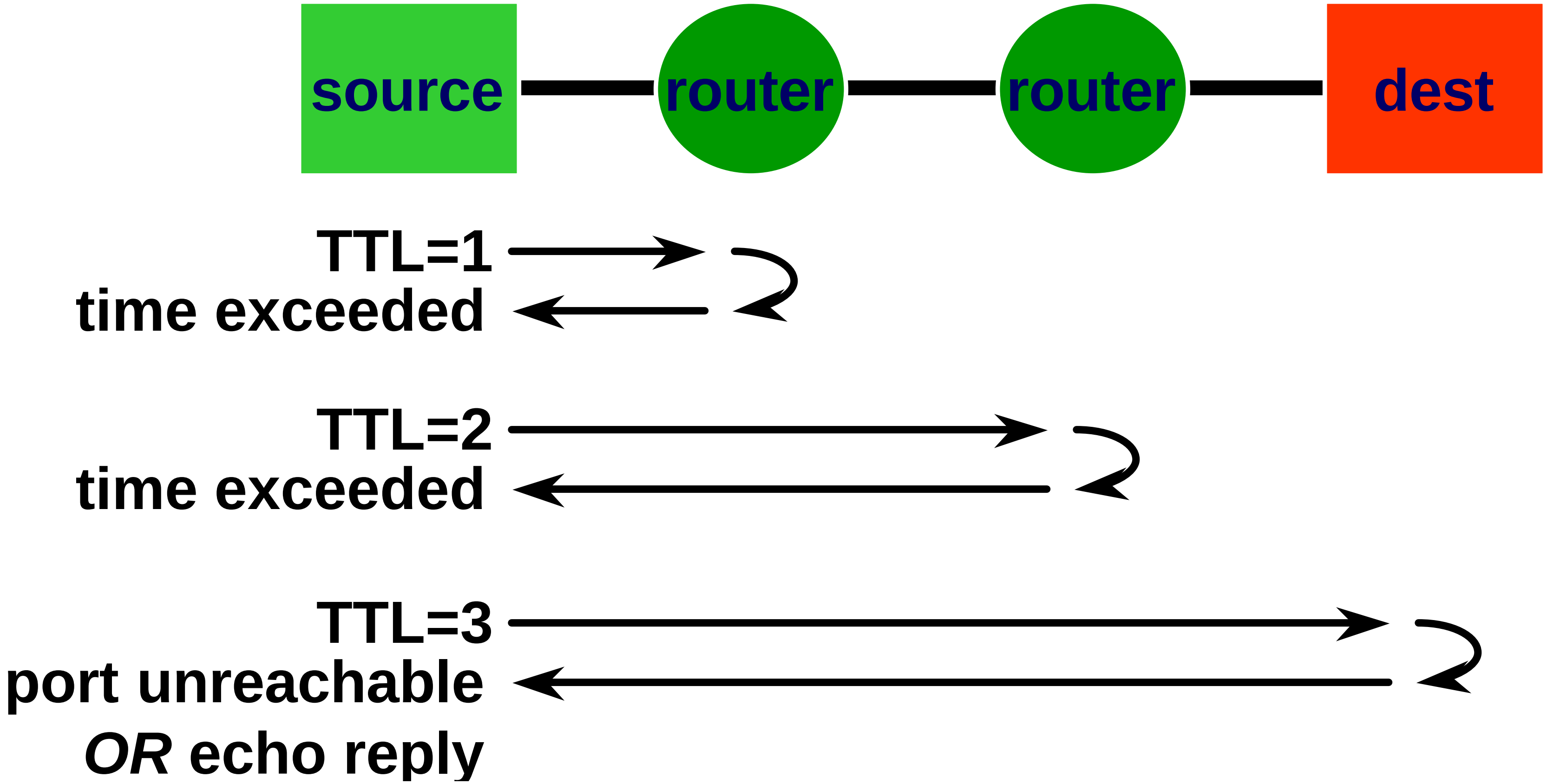
Traceroute Example

- Traceroute
  - Reports the route used by an IP datagram from source to destination
  - Provides a round-trip time
  - Written by Van Jacobson as a network diagnostic and debugging tool
- Operation
  - Sends ICMP or other datagram toward destination
  - IP time-to-live (TTL) value is controlled to limit extent
  - Intermediate nodes return ICMP time exceeded error -- includes router address

IP TTL Value

- **IP *Time-To-Live* Value: Maximum number of routers through which the datagram may pass**
  - **Decremented at each router**
    - **May be decremented once per second**
    - **Decremented at least once per router**
  - **Used to prevent looping in the network**
- **Basis for Traceroute**

Traceroute Operation



source

router

router

dest

TTL=1

time exceeded

TTL=2

time exceeded

TTL=3

port unreachable

OR echo reply

- IP packets sent by source (*traceroute*)
- ICMP packets returned by routers and host

Traceroute Algorithm

1) Set TTL value to 1
2) Initialize echo request
3) Send echo request
4) Wait for echo reply or time exceeded error (or time out)
5) Receive reply
6) Report results
7) If echo reply, then done; else increment TTL and return to 2

May want to do echo multiple times per TTL

Setting the TTL Value

- **Need to control the IP TTL value**
- **Raw socket with ICMP does not let us write IP header values**
- **Use setsockopt() to set TTL value**

```
setsockopt(raw, IPPROTO_IP, IP_TTL,
     (char *) &ttl, sizeof(ttl))

                or

int on = 1;
setsockopt(raw, IPPROTO_IP, IP_HDRINCL,
     &on, sizeof(on))
```

Basic Traceroute Loop

```
ttl = 0;
do {
   ++ttl;

   if(setsockopt(raw, IPPROTO_IP, IP_TTL,
   (char *) &ttl, sizeof(ttl)))
     errexit("setsockopt() failed: %d\n",
     perror());

   done = PingTarget(raw, target_addr);

} while (!done && ttl < MAX_TTL);
```

Potential “Bells and Whistles”

- Multiple pings for each TTL value to better assess round-trip time
- Modify amount of data sent in echo request
- Calculate link delay and other statistics
  - Delay[i] = RTT[i] - RTT[i-1]
- Look up intermediate host names using gethostbyaddr()
- Graphical features

ICMP, Ping, Traceroute Reference

W. Richard Stevens, *TCP/IP Illustrated, Volume 1, The Protocols*, Addison-Wesley Publishing Co., Reading, MA, 1994 (Chapters 6-8).

You should now be able to …

- Describe the use of ICMP for queries and replies
- Analyze ICMP message format
- Analyze the operation of Ping and Traceroute applications
- Analyze, design, and implement network applications using raw sockets