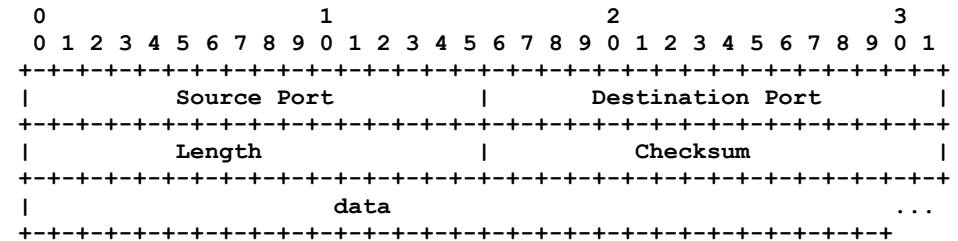


UDP: User Datagram Protocol, Other Transports, Sockets

- IP is an *unreliable datagram* protocol
 - congestion or transmission errors cause lost packets
 - multiple routes may lead to out-of-order delivery
- UDP delivers *exactly* this service to user programs
- If senders
 - send too fast, routers or receivers cannot keep up (making congestion worse);
 - compete, capacity must be fairly shared
- UDP cannot solve these problems in any way

1

UDP header



- Protocol Number or Next Header is 17 (decimal)

2

UDP header fields

- Port numbers - used to find TCBs at each end
 - Note that source port is optional. Since there is no concept of a connection in UDP, it may not be needed
- Length
 - length in bytes of UDP header plus data
 - ill-advised to exceed the available MTU
- Checksum
 - 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero bytes at the end (if necessary) to make a multiple of two bytes
 - Pseudo-header is the same as for TCP

3

A day in the life of a UDP packet

- User A: Listen (portA)
- User B: Send (AddressA, portA, DataB)
- User A: Receive (DataB)
- User A: Listen (portA)
- That's if the packet gets through the network. If it happens to be discarded due to congestion or error, we get:
 - User A: Listen (portA)
 - User B: Send (AddressA, portA, DataB)
 - ... and the rest is silence

4

Why is UDP useful?

- Because UDP offers no error recovery and no error notification, it may appear useless
- In fact, on a network less than 100% busy, UDP packets usually get delivered. But a UDP-based application *must* include its own timeouts and error recovery. Mostly, it's easier to use TCP instead
- Important UDP applications include
 - DHCP
 - DNS
 - RIP
 - SNMP (simple network management protocol)which can all survive lost packets. Each listens on a well-known port number

5

References for UDP

- A few words in Shay 11.4
- Any of the TCP/IP books listed for IPv4
- RFCs:
 - RFC 768, the original definition
 - RFC 2460 (IPv6) modifies UDP's checksum formula

6

Other transport protocols

- RTP - Real time Transmission Protocol
- SCTP - Stream Control Transmission Protocol
- DCCP - Datagram Congestion Control Protocol
- And related: ECN - Explicit Congestion Notification

7

RTP - mainly for audio/video streams

- RTP data packets run over UDP on an even-numbered port
 - normally a port number above 16384
 - RTSP (RT Streaming Protocol) is layered on top of RTP
- RTCP (RTP Control Protocol) runs over UDP on the next-higher (odd-numbered) port
- RTP provides
 - Payload-type identification
 - Sequence numbering
 - Time stamping for synchronisation and jitter management
 - Delivery monitoring
- But with the unreliability issues of UDP
 - Video or audio codecs must allow for this

8

- A reliable, congestion-friendly protocol that has learned much from TCP
- Main differences:
 - Both ends can have multiple IP addresses, and the SCTP connection can switch between addresses (for example, in case of a routing failure for one of the addresses)
 - SCTP supports multi-streaming, i.e. separate virtual connections within the main SCTP connection
- Intended use was reliable connectivity for telephony signalling over the Internet
 - SCTP is quite general in applicability
 - Quite new and not widely used yet

- DCCP behaves like a halfway house between UDP and TCP
 - TCP's reliability and in-order delivery features introduce delays that are not OK for audio/video
 - UDP's lack of congestion management causes network saturation when demand exceeds capacity
 - DCCP establishes a connection (like TCP) and reports packet delivery (unlike UDP). It does not retransmit on error or attempt in-order delivery (like UDP and unlike TCP)
 - DCCP offers two congestion management approaches
 - Also makes use of ECN (next topic)
 - Quite new and not widely used yet

- Makes use of bits 6 and 7 in the IPv6 Traffic Class field or the IPv4 Differentiated Services field
 - 00 - ECN not in use
 - 01 - unused
 - 10 - ECT flag
 - 11 - CE flag
- ECT means “sender is ECN-capable”
- CE means “router is congested” and is interpreted by the receiving transport protocol
- A transport protocol that supports ECN will invoke a “slow down” mechanism when it receives a CE flag
 - Quite new and not widely used yet

Other Transport References

- RTP - Shay 11.4, RFC 3550
- RTSP - RFC 2326
- SCTP - RFC 4960
- DCCP - RFC 4340, 4341, 4342
- ECN - RFC 3168

Sockets

- All transport protocols need a mechanism for upper layer software to access the transport
- The general concept is a notional “socket” that the application plugs into, embodied as a Socket API
 - Originated as “Berkeley sockets” on 4.2BSD Unix
 - Standard API is defined as part of Posix standard
- API includes calls to resolve DNS names into IP addresses, open and close sockets, and send and receive data
 - Plus many socket options for various purposes

13

Socket API overview (not a programmer's guide)

- **socket** is a function that creates a new socket data structure and returns a handle for it.

mySocket = socket(domain, type, proto)

domain is **AF_INET** for IPv4, **AF_INET6** for IPv6

type is **SOCK_STREAM** for TCP,
SOCK_DGRAM for UDP

proto is **IPPROTO_IP** as a default

- **close(mySocket)** gets rid of a socket

14

Finding addresses (1)

- **gethostbyname()** is a function that takes a DNS name as a string and returns a structure containing the corresponding IP address
- In other words, it invokes the DNS resolver and the whole DNS lookup process
- Works well in an IPv4 network with one IP address per DNS name
- Inadequate for an IPv4/IPv6 network with multiple addresses per name

15

Finding addresses (2)

- **getaddrinfo()** overcomes the shortcomings of **gethostbyname()** for IPv4/IPv6 coexistence
- Allows user to express IPv6 preference
- Note that it doesn't return “the” address. The code that calls **getaddrinfo** is supposed to choose from a set of addresses
 - by default, assume addresses are ordered by system preference
 - if the first address doesn't answer, try the next one...
- The user code is more complex than with **gethostbyname**, but results in more robust application behaviour when there is any kind of network problem

16

Finding addresses (3)

- An **AF_INET6** socket can be used for IPv6 or IPv4
 - if a “mapped” address like `::FFFF:10.1.2.3` is used, the socket will automatically use IPv4
 - Shay 11.3 is wrong to suggest that *routers* recognise these addresses - it is the sending host that decides to use IPv4 when mapped addresses are used with a socket
- The socket option **IPV6_V6ONLY** will force a socket to work only in IPv6 mode

17

Getting ready to talk

- **bind()** assigns an address to a socket
- **listen()** asks a socket to listen for incoming connections (TCP) or datagrams (UDP)
- **connect()** launches a connection (TCP SYN/ACK), or connects a program to a local socket (UDP)
 - **shutdown()** disconnects (TCP FIN/ACK)
- **accept()** accepts an incoming connection request (TCP)

18

Talking

- **send()** and **recv()** calls
 - and variants, or use **write()** and **read()**
- Streaming mode for TCP - no relationship between individual **send/recv** calls and individual TCP segments
 - Hence the PUSH option in TCP, to force data into the receiver
- Datagram mode for UDP - one **send/recv** for each datagram, and lost datagrams are truly lost
 - checksum errors give error returns from **recv()**

19

Securing the socket layer: Transport Layer Security (TLS)

- Protects TCP sessions
 - Earlier versions known as SSL
- Uses a handshake procedure to negotiate crypto algorithm
- Uses server's public key to securely negotiate keys for the session
 - server presents a certificate to the client, including its public key. The certificate is cryptographically signed by a trusted *certificate authority* using *its* public key
- Following these negotiations, no 3rd party can intercept or inject traffic

20

Socket References

- Shay 12.2 (TLS is in 7.5)
- See your favourite Unix book or <http://www.rt.com/man/>
- With IPv6: RFC 3493 (and 3542)
- The Wikipedia article on Berkeley sockets is pretty good (but doesn't use **getaddrinfo**)
- POSIX standard: IEEE Std. 1003.1-2004 Standard for Information Technology -- Portable Operating System Interface (POSIX)
See http://www.unix.org/version3/ieee_std.html