CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

UDP

UDP is bare-bones transport protocol of IP stack

Offers two services:

- (de)multiplexing
- checksum for error detection

Almost like interfacing directly with IP

Like IP, message delivery via UDP is on best-effort basis

Sent packets may or may not show up at receiver

UDP is connectionless

No handshake performed before application information sent over UDP

Simply put message in segment, send, and hope

TCP offers many features, so why would anyone use UDP?

For one, application has more control over when data is sent.

TCP provides congestion control, meaning it will throttle sending rate if necessary for health of network

Also, TCP will continue to send segment until it knows delivery was successful, even if resulting message may no longer be relevant

Another UDP advantage is that there is no overhead for connection establishment -- this speed advantage is reason DNS runs over UDP

No connections also means no need to maintain state, so server can handle communicating with more clients at once

Finally, UDP packet overhead is less than TCP's: 8 bytes vs 20 bytes

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Name translation	DNS	Typically UDP

Because of lack of congestion control, running multimedia over UDP can be harmful to network

UDP senders not "polite" -- will continue to send even when network overwhelmed with traffic

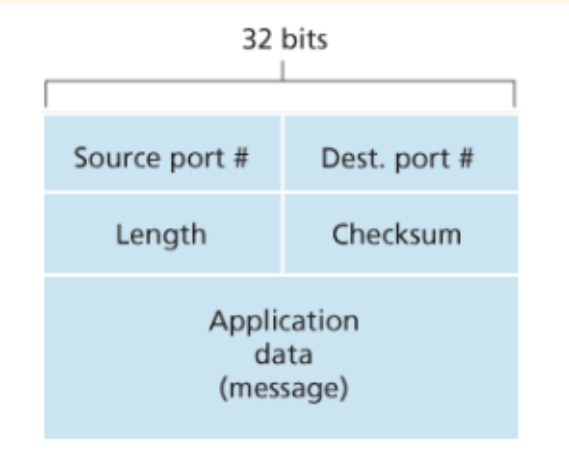
This causes TCP senders to need to back off even further

Worth noting that features of TCP could be used by applications running UDP

If application wants, for example, reliable delivery, it can implement that as part of application-layer protocol

Complicates application-layer code substantially, but allows application to pick and choose among TCP features for itself

UDP segment



Note that length field gives combined length of header and message in bytes

Mentioned previously that UDP provides error detection in form of checksum

We will study error detection and correction more when examining link layer, but briefly explain here how UDP checksum works

If error detection implemented in link layer, why bother in transport layer?

Not all link-layer protocols provide error detection, and corruption can occur in routers as well, so **end-to-end** detection must be provided by higher layer

Checksum computed by following steps:

- find sum of all 16-bit words in message
- if sum carries bit out, wrap around to right-hand side
- take 1s complement of sum (i.e., flip all bits)

Following example is taken from textbook

Assume message has three 16-bit words:

```
0110011001100000
0101010101010101
1000111100001100
```

Sum is

```
0100101011000010
```

(note that overflow wrapped around)

1s complement of

```
0100101011000010
```

is

```
1011010100111101
```

This is the checksum

How does that help?

At receiving end, all 16-bit words added up again, and result added to checksum

Because checksum was sum with all bits flipped, adding checksum to sum gives

```
1111111111111111
```

If any bits were damaged between sender and receiver, result will not be equal to 111 . . . , so we have detected error

Note that this checksum provides error detection, but not error recovery

When receiver detects error, it may drop packet or may pass to application layer with a warning

Receiver will not fix damaged segment or ask sender to resend