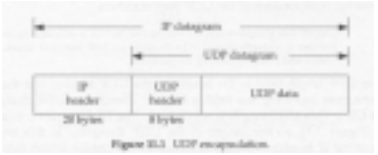
Chpater 11:
UDP: User Datagram Protocol

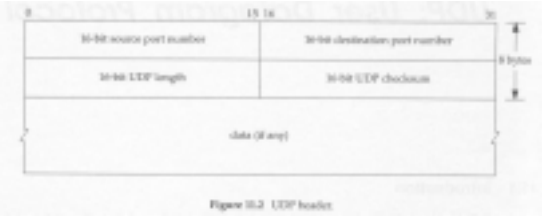
Introduction

- **UDP encapsulation:**



  - UDP is a simple, *datagram-oriented* (different from *stream-oriented*), transport layer protocol
  - UDP provides no reliability
- **UDP Header**
  - The port numbers identify the sending process and the receiving process
  - UDP length field is the length of the UDP header and the UDP data in bytes.The minimum value is 8 bytes

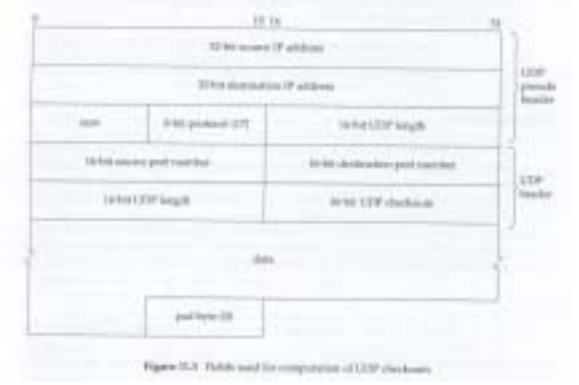
UDP Header



- **Compare TCP Checksum and UDP Checksum**
  - TCP checksum is mandatory
  - UDP checksum is optional
- **Differences between UDP checksum and IP checksum**
  - UDP datagram can be an odd number of bytes
  - Both UDP and TCP include a 12-byte pseudo-header purpose to let UDP double-check the data has arrived at the correct destination

UDP Checksum

- UDP checksum is an end-to-end checksum.If the receiver detects a checksum error, UDP datagram is simply discarded.



UDP Checksum (Cont.)

- Value 0 means the sending host did not calculate the checksum
- The UDP checksums cannot detect an error that swaps to of the 16-bit values

- Some Statistics

Layer	Number of checksum errors	Approximate total number of packets
Ethernet	446	170,000,000
IP	34	170,000,000
UDP	5	160,000,000
TCP	350	30,000,000

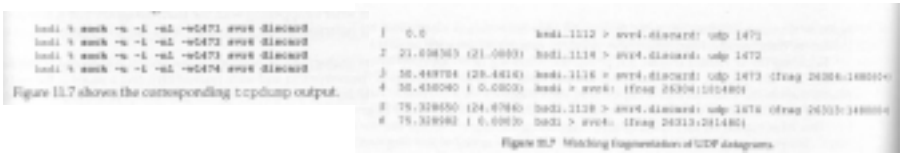
Figure 11.5 Counts of corrupted packets detected by various checksums.

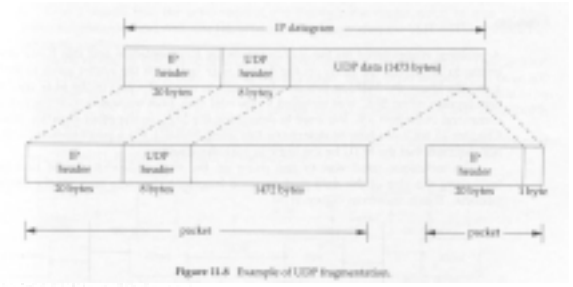
A Simple Example

- There is no communication between the sender and receiver before the first datagram is sent
- There are no acknowledgments by the receiver when the data is received

IP Fragmentation

- **Example:**





IP Fragmentation (Cont.)

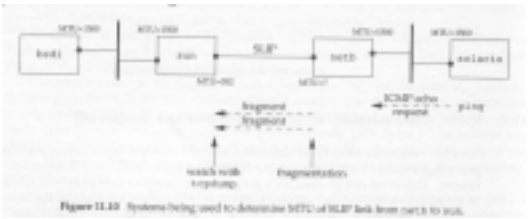
- **The physical network layer limits the size of the frame.**
- **Fragmentation can take place either at the original *sending host* or at an *intermediate router*.**
- **The IP layer at the *destination* performs the reassembly.**
- **All IP fragments have the same ID number.**
- ***Fragment offset*: the offset from the beginning of the original datagram**
- **One fragment lost: entire datagram must be retransmitted.**
- **Data portion of a fragment: multiple of 8 bytes (other than the final one)**
- **The port numbers only occurs in the first fragment.**
- **Any transport layer header appears only in the first fragment.**

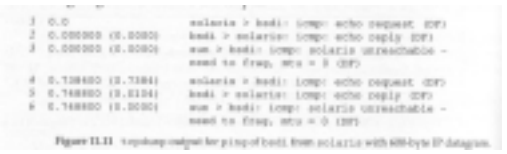
ICMP Unreachable Error (Fragmentation Required)

- **When occurs the ICMP unreachable error**
  - a datagram that requires fragmentation, but the DF (don't fragment) flag is turned on

ICMP Unreachable Error (Cont.)

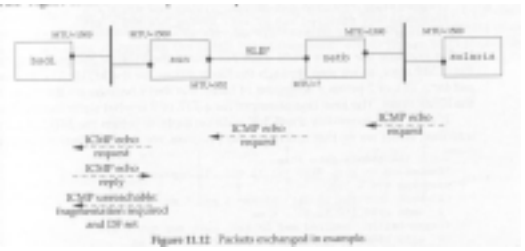
- **Example:**





ICMP Unreachable Error (Cont.)

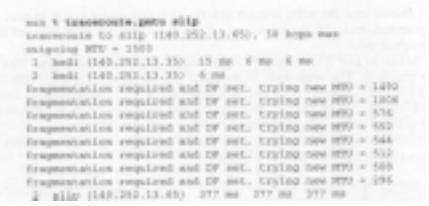
- **DF flag is set and causes *sun* to generate the ICMP unreachable error back to *bsdi* (where it's discarded)**



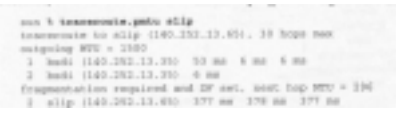
- **Determining the path MTU using traceroute**
  - Whenever we receive an ICMP "can't fragment" error, we'll reduce the size of the packet.

Determining the Path MTU Using Traceroute

- **The router *bsdi* does not return the MTU**

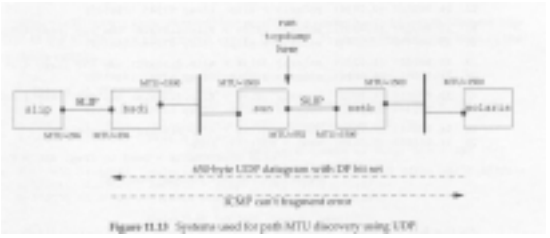


- **The ICMP code on *bsdi* to return the MTU**



Path MTU Discovery with UDP

- **Example:**



- The following command generates ten 650-byte UDP datagrams, with a 5-second pause between each datagram:

```
solaris % sock -u -i -n10 -w650 -p5 slip discard
```


Path MTU Discovery with UDP

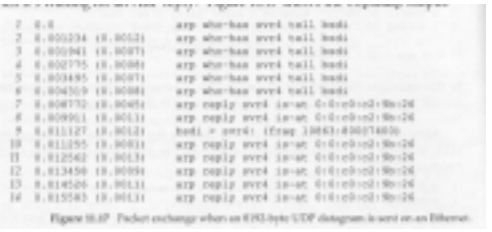
Path MTU Discovery with UDP

- **Four fragments generated by the router bsdi**

- **Three fragments generated by the router bsdi return the next-hop MTU in the ICMP "can't fragment" error**

Interaction Between UDP and ARP

- **Example:**



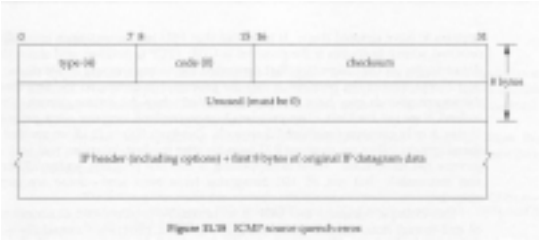
- Six ARP requests are generated before the first ARP reply is returned
- Only the last fragment is sent, first five fragments have been discarded
- Unexplained anomaly in output seven ARP replies, not six

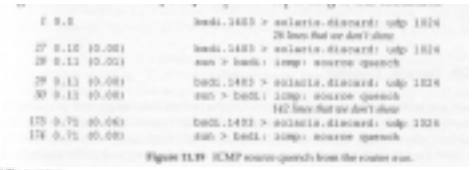
Interaction Between UDP and ARP (Cont.)

- **Why we don't see the ICMP message**
  - Most Berkeley derived implementations never generate this error
  - The first fragment which containing the UDP header was never received
- **Maximum UDP Datagram Size**
  - Just over 8192 bytes for the maximum size of a UDP datagram that can be read or written
  - Limit the size of an IP datagram to less than 65535 bytes
- **How to deal with received datagram exceeds the size**
  - The traditional Berkeley => discarding any excess data
  - The sockets API under SVR4 => does not truncate the datagram
  - The TLI API => Instead a flag is returned

ICMP Source Quench Error

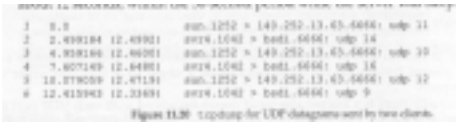
- This is an error that may be generated by a system (router or host) when it receives datagrams at a rate is too fast to be processed





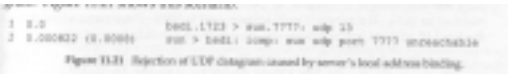
UDP Server Design

- **Client IP Address and Port Number**
  - When an application receives a UDP datagram, it must be told by the operating system who sent the message--the source IP address and port number
- **Destination IP Address**
  - Who the datagram was sent to, that is, the destination IP address
- **UDP Input Queue**
  - A single server process handles all the client requests on a single UDP port

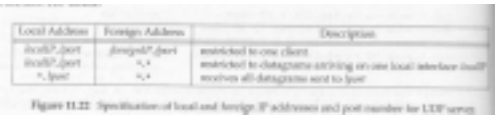


UDP Server Design (Cont.)

  - The application is not told when its input queue overflows
  - Nothing is sent back to the client to tell it that its datagram was discarded
  - UDP input queue is FIFO, ARP input queue was LIFO
- **Restricting Local IP Address**



- **Restricting Foreign IP Address**
- **Multiple Recipients per Port**



Summary

- UDP is a simple protocol
- The services it provides to a user process are port numbers and an optional checksum
- Path MTU discovery using Traceroute and UDP
- The ICMP source quench error can be sent by a system that is receiving IP datagrams faster than they can processed