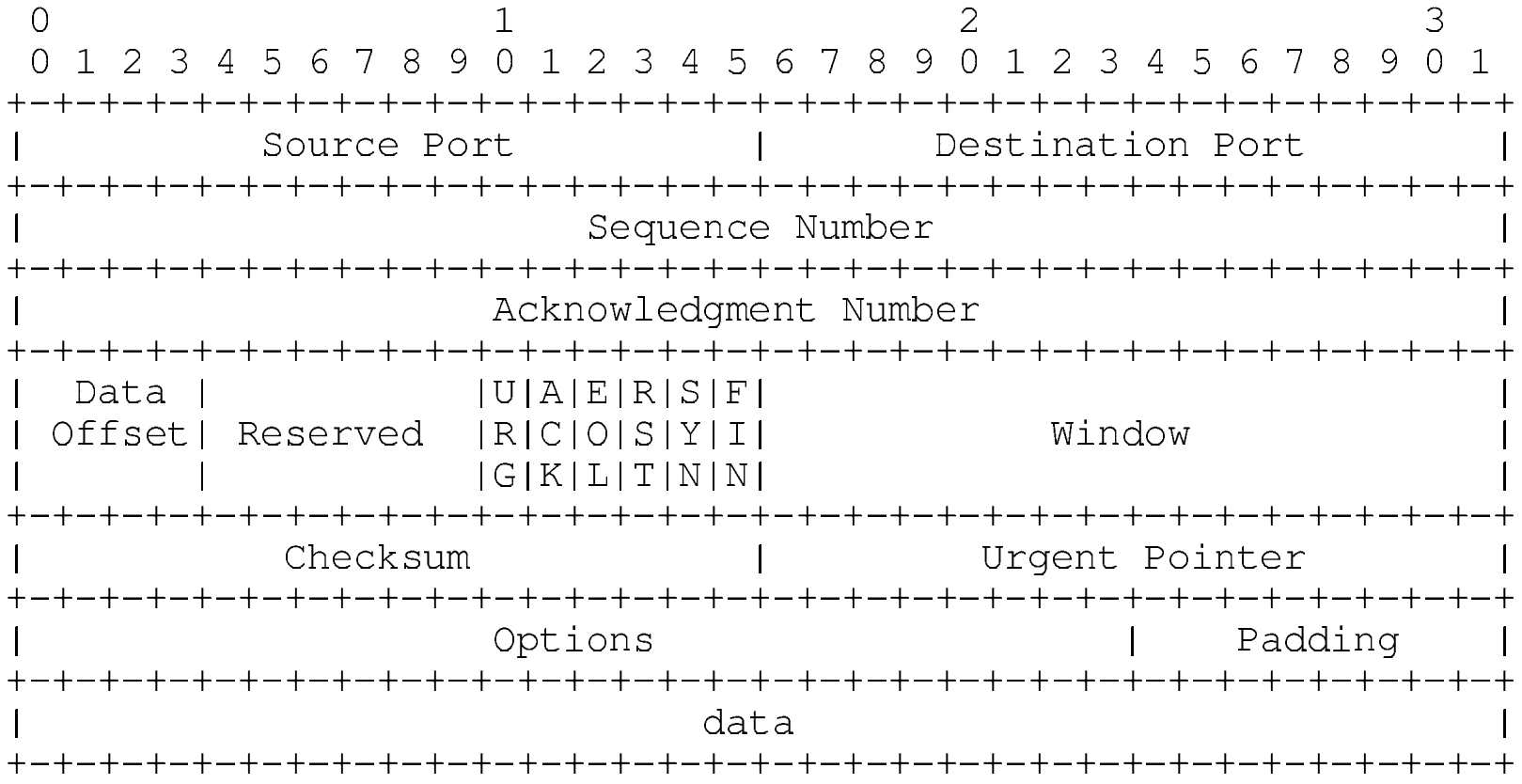
TCP

ComS 587X
Fall, 2002

TCP

- Transmission Control Protocol
- Layer 4 (Transport)
- Reliable, ordered byte stream
- Port numbers, like UDP
- Checksums payload
- Flow control
  - Sensitive to packet loss and round-trip time
- Error recovery
  - Retransmission of lost or corrupted packets
- High-throughput
  - Windowed with piggy-back acknowledgment

TCP Header


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
U R G
A C K
E O L
R S T
S Y N
F I N
Window
Checksum
Urgent Pointer
Options
Padding
data

Misc

- Advantages over UDP
  - Automatic error control
  - Reliability
  - Ease of use
- Some well-known services and port numbers
  - SSH: 22
  - Telnet: 23
  - FTP: 20/21
  - SMTP (Mail delivery): 25
  - HTTP: 80
  - POP (Mail retrieval): 110

Client/Server Paradigm

- Servers provide some service
  - E.g., mail delivery
  - Accept requests
  - Return results
- Clients use the service
  - Make requests
  - Accept results
- Switching roles midstream difficult

Clients

- Clients typically
  - Connect to a server
  - Make requests
  - Process responses
- The **protocol** defines the way in which the client and server communicate
  - Protocol often defined by an RFC (Request for Comments)
  - Well-known ports often imply the protocol used
    - E.g. port 80 == HTTP
- Protocol mismatches don't work
  - E.g. An HTTP client can't talk to an SMTP server

Servers

- Server provides services, typically at a predefined port
- Server typically runs for a long time (i.e., a daemon)
  - Constantly accepting new connections and processing client requests
  - Usually started by the operating system at boot time

TCP Sockets in Java

- Separate classes:
  - java.net.ServerSocket
    - For accepting connections
    - Typically used by servers
  - java.net.Socket
    - For making connections
    - Typically used by clients
- Different than UDP
  - Single socket, DatagramSocket, used for both servers and clients

Socket

- java.net.Socket provides bi-directional communication between TCP ports
- Constructors include:
  - Socket(String hostname, int port)
  - Socket(InetAddress addr, int port)
  - Socket(InetAddress addr, int port, InetAddress localAddr, int localPort)
  - Socket(String hostname, int port, InetAddress localAddr, int localPort)

Creating a Socket

- Typically-used constructors define the remote host and port to which to connect
- No separate connect() method to set or change the remote host and port
- Socket constructors may block while waiting for the remote host to respond
  - Depends on operating systems and timeout settings
  - When performance is critical, programmer may want to create a separate thread to create a Socket

Socket Methods

- void close()
- InetAddress getInetAddress()
- InputStream getInputStream()
- OutputStream getOutputStream()
- boolean getKeepAlive()
- InetAddress getLocalAddress()
- int getLocalPort()
- int getPort()
- int getReceiveBufferSize()
- int getSendBufferSize()
- int getSoLinger()
- int getSoTimeout()
- boolean getTcpNoDelay()
- void setKeepAlive()
- void setReceiveBufferSize()
- void setSendBufferSize()
- static void setSocketImplFactory(SocketImplFactory factory)
- void setSoLinger(boolean onFlag, int duration)
- void setSoTimeout(int duration)
- void setTcpNoDelay(boolean onFlag)
- void shutdownInput()
- void shutdownOutput()

Socket Input & Output

- After construction, a socket is ready for I/O
- Obtain the socket InputStream and/or OutputStream

```
try {
  String s;
  Socket socket = new Socket("www.cs.iastate.edu", 80);
  BufferedReader reader = new BufferedReader( new
   InputStreamReader( socket.getInputStream() ) );
  PrintStream pstream = new PrintStream(
   socket.getOutputStream() );
  pstream.println("GET /");
  while ((s = reader.readLine()) != null) {
    System.out.println(s);
  }
} catch (Exception e) {
  System.err.println("Error: " + e);
}
```

Socket Options

- SO_KEEPALIVE
  - Periodically transmit empty messages to keep the connection open
- SO_RCVBUF
- SO_SNDBUF
- SO_LINGER
- TCP_NODELAY
- SO_TIMEOUT

Summary

- TCP
- Sockets & ServerSockets
- Socket Options
- Simple client