

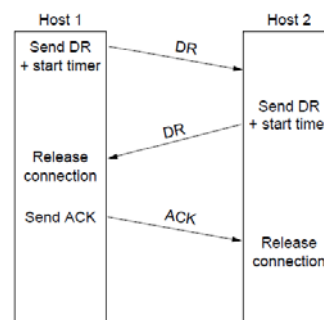
# Transport Layer Contd

COMP90007  
Internet Technologies

1

## Strategies for Connection Release

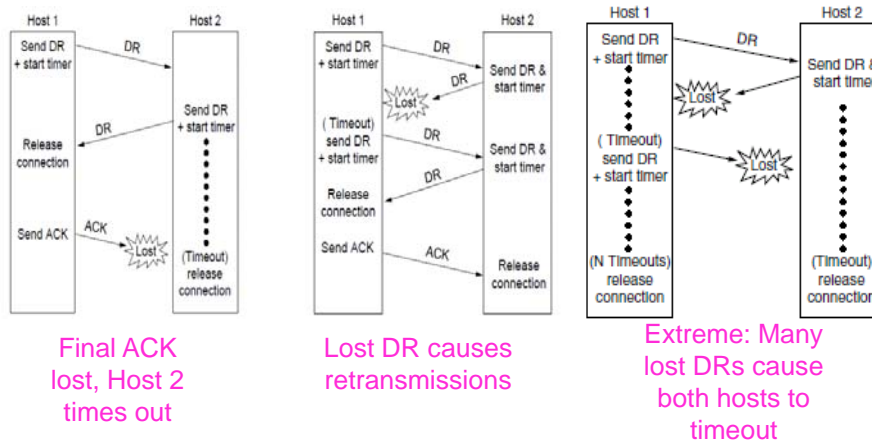
- 3 way handshake
- Finite retry
- **Timeouts**
- Normal release sequence, initiated by transport user on Host 1
  - DR=Disconnect Request
  - Both DRs are ACKed by the other side



2

## Connection Release (Error Cases)

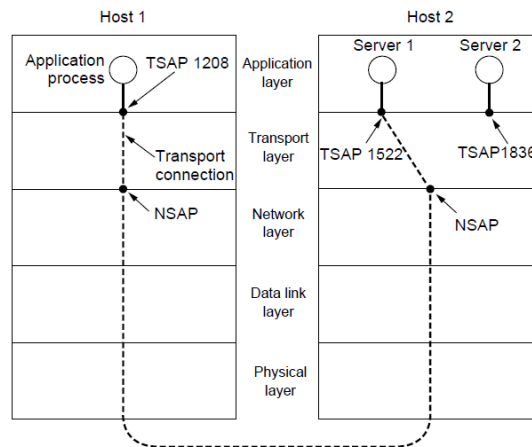
- Error cases are handled with timers and retransmission



## Addressing

- Specification of **remote process to connect to** is required at application and transport layers
- Addressing in transport layer is typically done using **Transport Service Access Points (TSAPs)**
  - on the Internet, a TSAP is commonly referred to as a port (e.g. **port 80**)
- Addressing in the network layer is typically done using **Network Service Access Points (NSAPs)**
  - on the Internet, the concept of an NSAP is commonly interpreted as simply an **IP address**

## TSAPs, NSAPs and Transport Layer Connections Illustrated



5

## Types of TSAP Allocation

### 1. Static

- Well known services have standard allocated TSAPs/ports, which are embedded in OS

### 2. Directory Assistance – Port-mapper

- A new service must register itself with the portmapper, giving both its service name and TSAP

### 3. Mediated

- A process server intercepts inbound connections and spawns requested server and attaches inbound connection
- cf. Unix /etc/(x)inetd

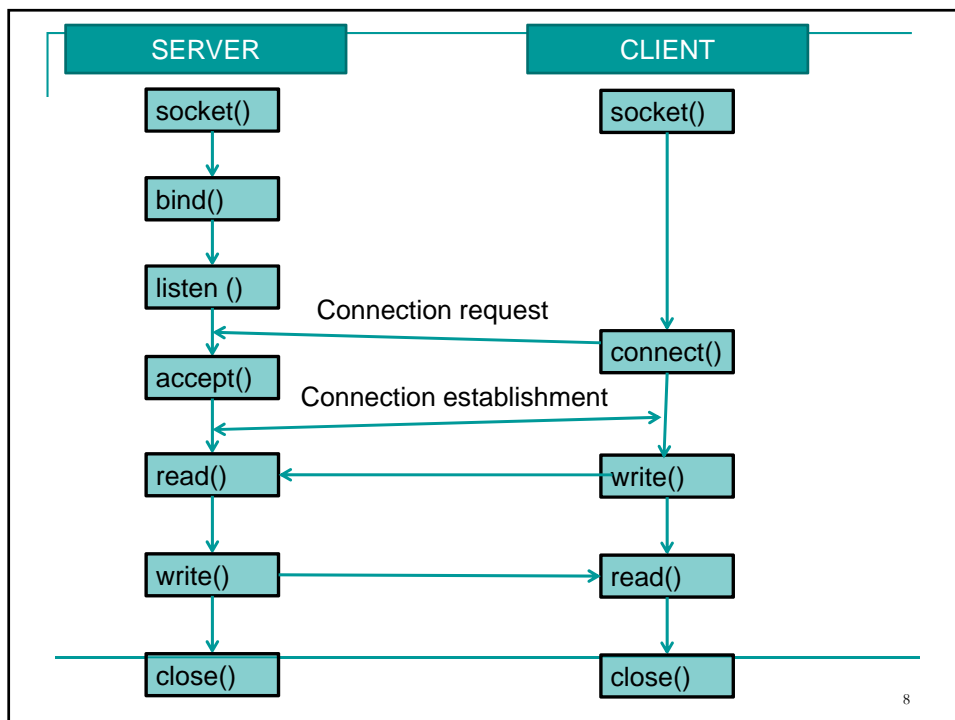
6

## Sockets

- Sockets widely used for interconnections
  - “Berkeley” sockets are predominant in internet applications
  - Notion of “sockets” as transport endpoints
  - Like simple set plus SOCKET, BIND, and ACCEPT

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

7



8

## Another Socket Example – Internet File Server

Remember our first simple network program. . .

Here is another example: from the book, with more details but the essence is the same...

```
s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);  
if (s < 0) fatal("socket");  
memset(&channel, 0, sizeof(channel));  
channel.sin_family = AF_INET;  
memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);  
channel.sin_port = htons(SERVER_PORT);
```

```
c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
```

9

## Socket Example – Internet File Server Contd.

Server code. . .

```
memset(&channel, 0, sizeof(channel));  
channel.sin_family = AF_INET;  
channel.sin_addr.s_addr = htonl(INADDR_ANY);  
channel.sin_port = htons(SERVER_PORT);
```

```
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);  
if (s < 0) fatal("socket failed");  
setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));
```

```
b = bind(s, (struct sockaddr *) &channel, sizeof(channel));  
if (b < 0) fatal("bind failed");
```

**Assign  
address**

```
l = listen(s, QUEUE_SIZE);  
if (l < 0) fatal("listen failed");
```

**Prepare for  
incoming  
connections**

. . .

10

## Socket Example – Internet File Server

Server code contd..

```
while (1) {  
    sa = accept(s, 0, 0);  
    if (sa < 0) fatal("accept failed");  
    read(sa, buf, BUF_SIZE);  
    /* Get and return the file. */  
    fd = open(buf, O_RDONLY);  
    if (fd < 0) fatal("open failed");  
    ....  
}
```

**Block waiting for the next connection**

**Read (receive) request**

The server can also create a new thread to handle the connection on the new socket and go back to waiting for the next connection on the original socket...

11

## Simple Example with Multi-Threading

```
ServerSocket serverSocket = new ServerSocket([parameters]);  
  
While (true) {  
    Socket socket = serverSocket.accept();  
    MultiThreadMyServer server = new MultiThreadMyServer();  
    server.setMyService([some parameters]);  
    server.setSocket(socket);  
    new Thread(server).start();  
    ....  
}
```

- (Code from OO Programming with Java; Chp. 14)

12