



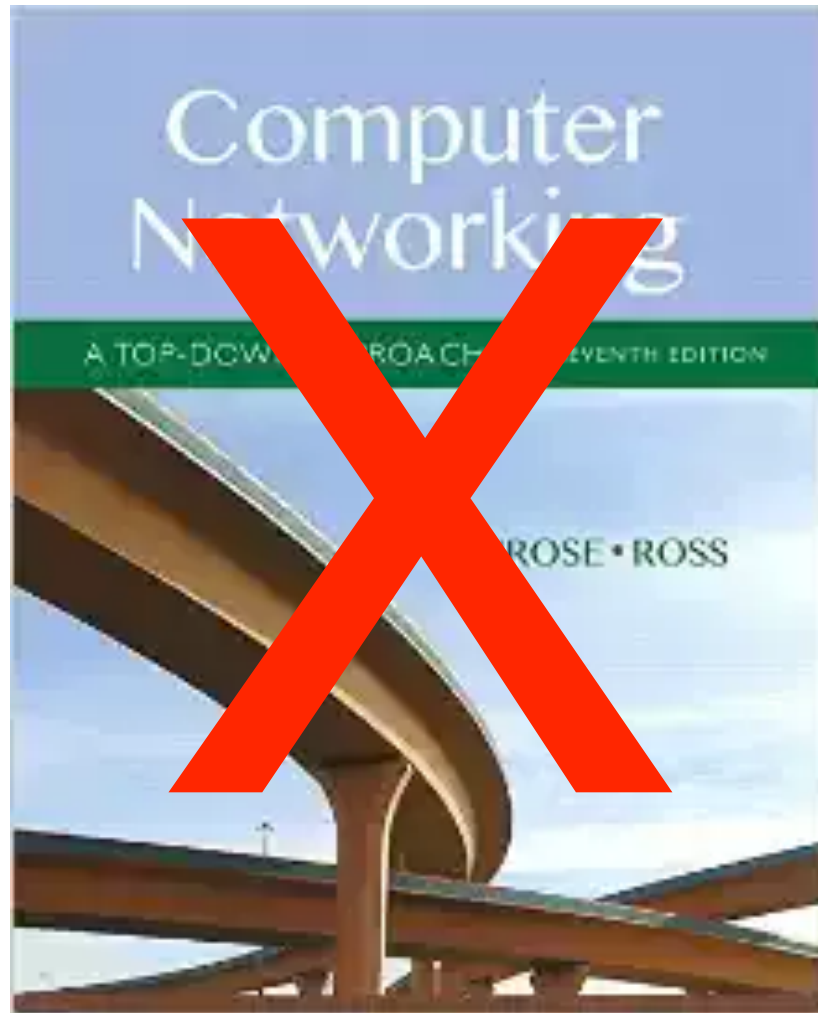
Application Layer Part 7

Mark Allman
Case / ICSI

EECS 325/425
Fall 2018

*“We’re the bad news, we’re the young guns,
We’re the ones they told you to run from”*

Reading Along ...

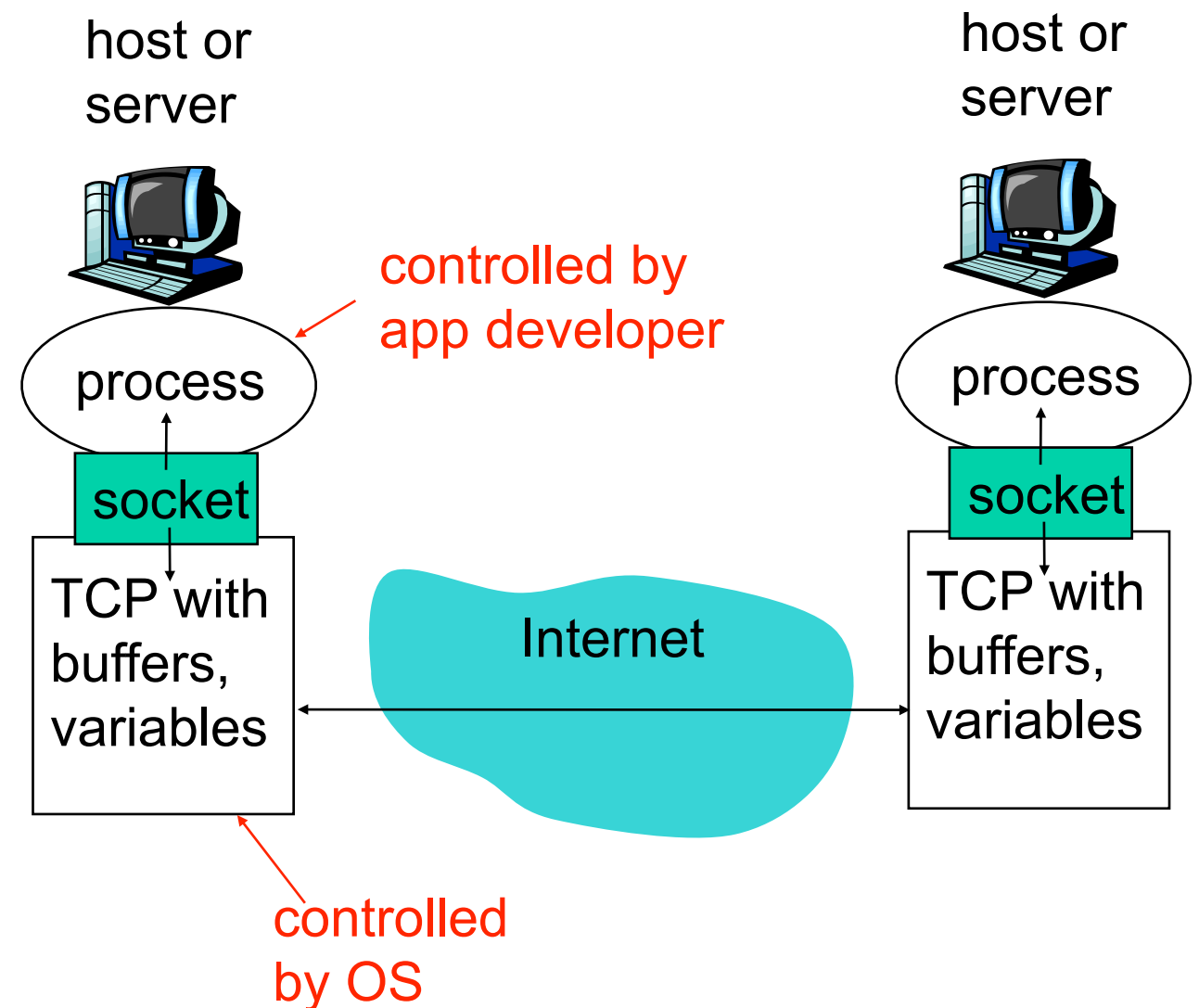


- Sockets programming (server side)

Sockets

Recall that sockets sit between the application process and the transport protocol

Sockets form the glue that allows processes to *interface* with transports (and hence all lower layers)



Example TCP Server

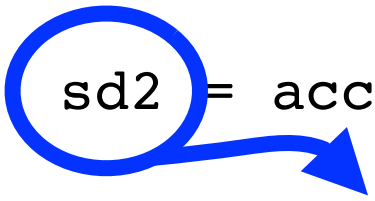
```
write (sd2,msg,strlen (msg));
```

Example TCP Server

```
write (sd2,msg,strlen (msg));
```

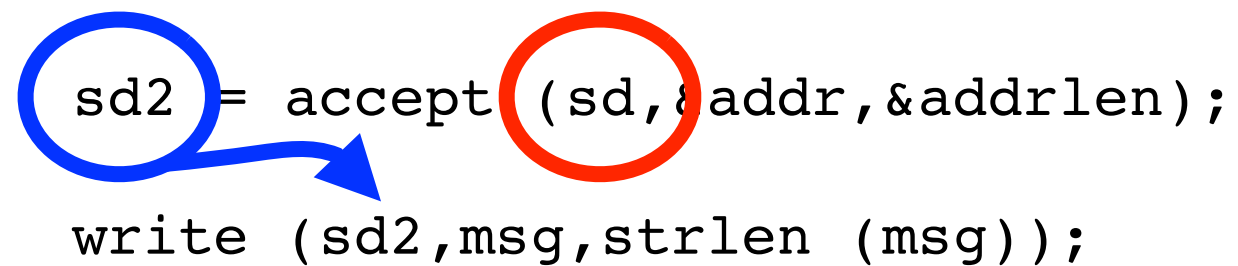
Example TCP Server

```
sd2 = accept (sd, &addr, &addrlen);  
write (sd2, msg, strlen (msg));
```



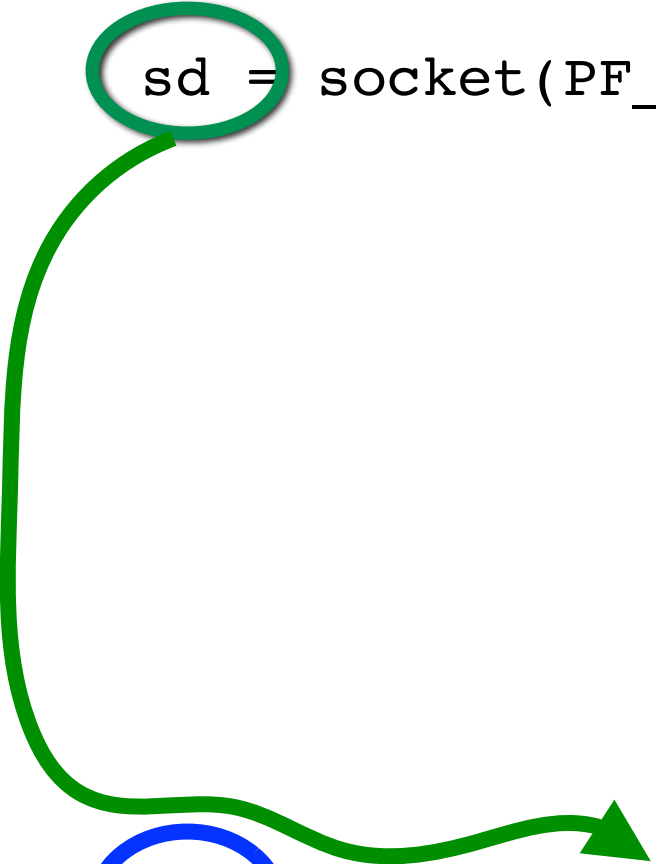
Example TCP Server

```
sd2 = accept (sd, &addr, &addrlen);  
write (sd2, msg, strlen (msg));
```

A diagram illustrating variable usage in a code snippet. The variable 'sd2' in the first line is circled in blue. The variable 'sd' in the same line is circled in red. A blue arrow points from the blue circle around 'sd2' to the red circle around 'sd', indicating that 'sd2' is the variable being assigned the value of 'sd'.

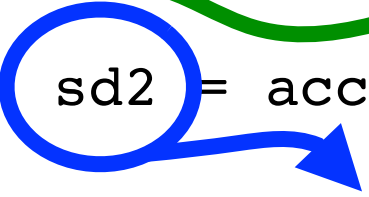
Example TCP Server

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```



A green curved arrow originates from the variable 'sd' in the first line of code and points to the 'sd' argument in the 'accept' function call of the second line of code.

```
sd2 = accept (sd, &addr, &addrlen);
```

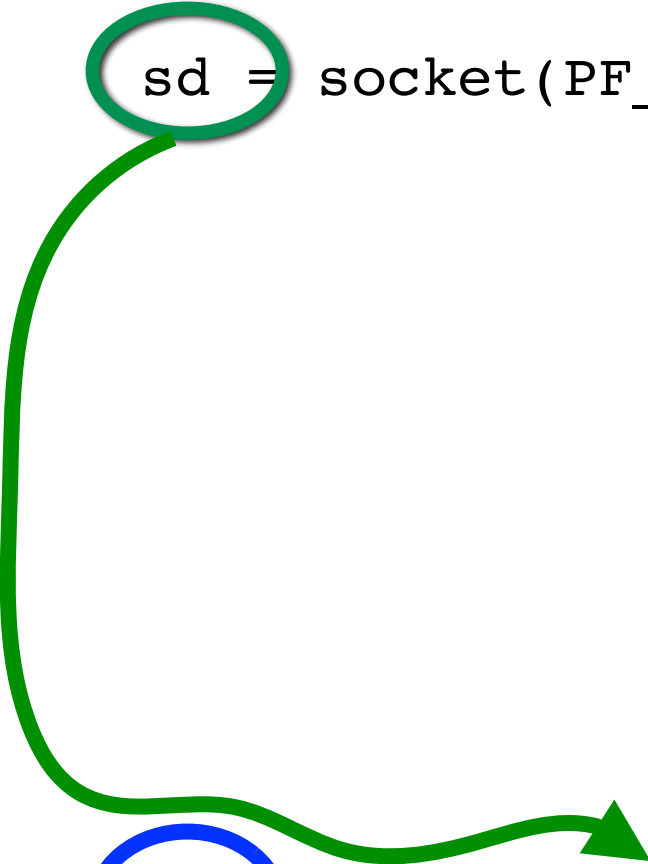


A blue curved arrow originates from the variable 'sd2' in the second line of code and points to the 'sd2' argument in the 'write' function call of the third line of code.

```
write (sd2, msg, strlen (msg));
```

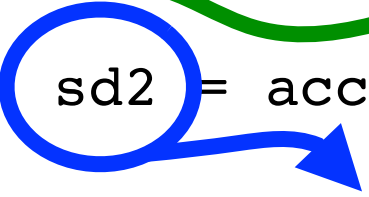

Example TCP Server

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```



A green curved arrow originates from the variable 'sd' in the first line of code and points to the 'sd' parameter in the 'accept' function of the second line of code.

```
sd2 = accept (sd, &addr, &addrlen);
```



A blue curved arrow originates from the variable 'sd2' in the second line of code and points to the 'sd2' parameter in the 'write' function of the third line of code.

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
listen (sd, QLEN);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
listen (sd, QLEN);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
bind (sd, (struct sockaddr *)&sin, sizeof(sin));
```

```
listen (sd, QLEN);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
bind (sd, (struct sockaddr *)&sin, sizeof(sin));
```

```
listen (sd, QLEN);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
bind (sd, (struct sockaddr *)&sin, sizeof(sin));
```

```
listen (sd, QLEN);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```

Example TCP Server

```
protoinfo = getprotobyname ("tcp");
```

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);
```

```
/* setup endpoint info */  
memset ((char *)&sin, 0x0, sizeof (sin));  
sin.sin_family = AF_INET;  
sin.sin_addr.s_addr = INADDR_ANY;  
sin.sin_port = htons ((u_short) atoi (argv [PORT_POS]));
```

```
bind (sd, (struct sockaddr *)&sin, sizeof(sin));
```

```
listen (sd, QLEN);
```

```
sd2 = accept (sd, &addr, &addrlen);
```

```
write (sd2, msg, strlen (msg));
```


Example TCP Server

```
protoinfo = getprotobyname ("tcp");
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);

/* setup endpoint info */
memset ((char *)&sin, 0, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = INADDR_ANY;
sin.sin_port = htons (atoi (argv [PORT_POS]));

bind (sd, (struct sockaddr *)&sin, sizeof(sin));

listen (sd, 5);

sd2 = accept (sd, NULL, &addrlen);
write (sd2, msg, strlen (msg));
```

Error Checking

```
sd = socket(PF_INET, SOCK_STREAM, protoinfo->p_proto);  
if (sd < 0)  
    errexit ("cannot create socket");  
  
if (bind (sd, (struct sockaddr *)&sin, sizeof(sin)) < 0)  
    errexit ("cannot bind to port");
```

Socket Example

Socket Example

Server:

```
% hostname  
eecslab-5  
% ./socketsd 1947 "hello world"
```

Socket Example

Server:

```
% hostname  
eecslab-5  
% ./socketsd 1947 "hello world"
```

Client:

```
% hostname  
eecslab-6  
% ./sockets eecslab-5.case.edu 1947  
hello world
```