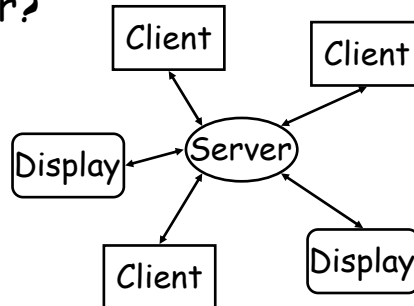



## *Network Programming*

- How do Server, Client, and Display applications communicate with each other?



## *Network Programming in RESOLVE/C++*

- Unidirectional channels: one application sends, the other receives
- Two channels are needed for bi-directional communication (underlying TCP communication links are bi-directional, but RCPP API makes them unidirectional)



## *Identifying the Receiver*

- IP address and port number
  - IP address uniquely determines the computer (e.g., 164.107.112.14). We use the host name (e.g., beta.cis.ohio-state.edu)
  - Port number is an integer to distinguish connections and services (e.g., ftp uses 21, BugsWorld Server uses 7277)



## *Connection Protocol*

- Each receiver application advertises its host name and port number, and "listens" for other applications that want to "connect"
- "Sender" application tries to open a connection to "Receiver" application using known host name and port number
- When receiver detects sender's attempt, the connection is established

## *RESOLVE/C++ API*

- Sender sends information through Character\_OStream
- Receiver receives information through Character\_IStream
- Seamless integration of network programming in the existing RCPP I/O framework!

## *Opening a Connection: Sender*

```
object Character_OStream ochannel;  
object Text host_name;  
object Integer port_number;  
  
...  
  
while (not ochannel.Is_Open ())  
{  
    // trying to connect to host_name, port_number  
    ochannel.Open_External (host_name, port_number);  
}  
  
...
```

## *Opening a Connection: Receiver*

```
object Character_IStream ichannel;
```

```
...
```

```
while (not ichannel.Is_Open ())  
{  
    // waiting for reverse connection  
    ichannel.Open_External ();  
}
```

```
...
```

## *Receiver: Finding Host Name and Port Number*

```
object Character_IStream ichannel;
```

```
object Text host_name;
```

```
object Integer port_number;
```

```
...
```

```
ichannel.Get_Host_And_Port (host_name, port_number);
```

```
...
```



## *Communication*

---

- Once the connection is established, the sender writes to the output stream, and the receiver reads from the input stream exactly as they would from any input/output stream



## *Client-Server Protocol*

---

1. Server advertises host name and port number, then waits for connection from any client.
2. Client attempts to connect with server using server's host name and port number. Once connection is established, client can send information to server. So client sends its own host name and port number to server. Then it waits for reverse connection.
3. Server, having established the connection, reads the client's host name and port number, and initiates its own attempt at connecting with the client. Since the client is waiting for the server, the reverse connection is established. Now the server can send information to the client.