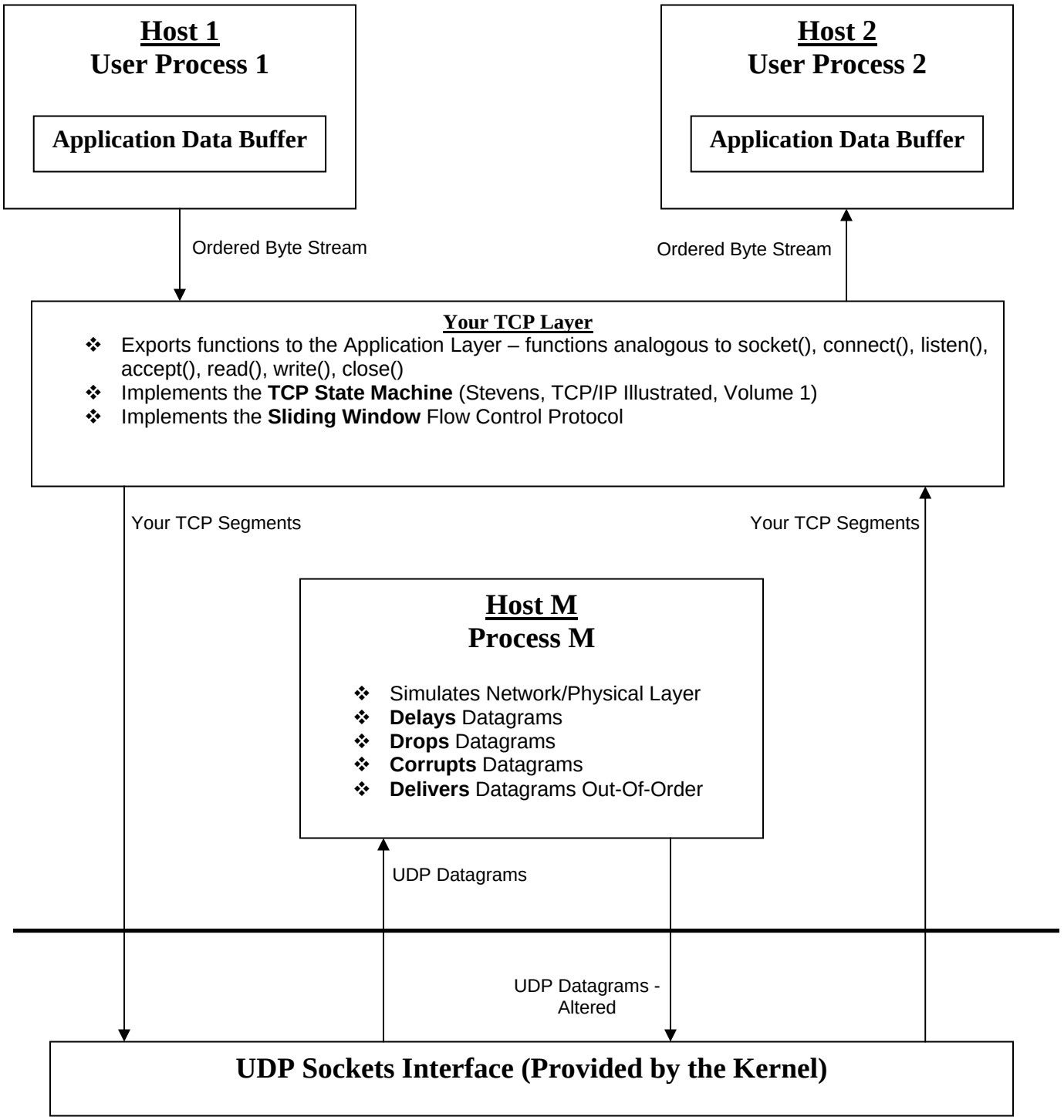
1. **TCP Simulation over UDP**

Implement the TCP protocol on top of the UDP Datagram Delivery Service. Refer to the Diagram Below.

Processes 1 and 2 are two user-level processes exchanging data between themselves using your TCP services.



Since on a LAN, UDP is practically as reliable as TCP, we need another process M to simulate the Network Layer and Physical Layer. It introduces various anomalies in the UDP Datagrams. This process should be invisible to the actual communicating processes.

2. **FTP Client/Server**

FTP Client/Server should provide the following facilities:

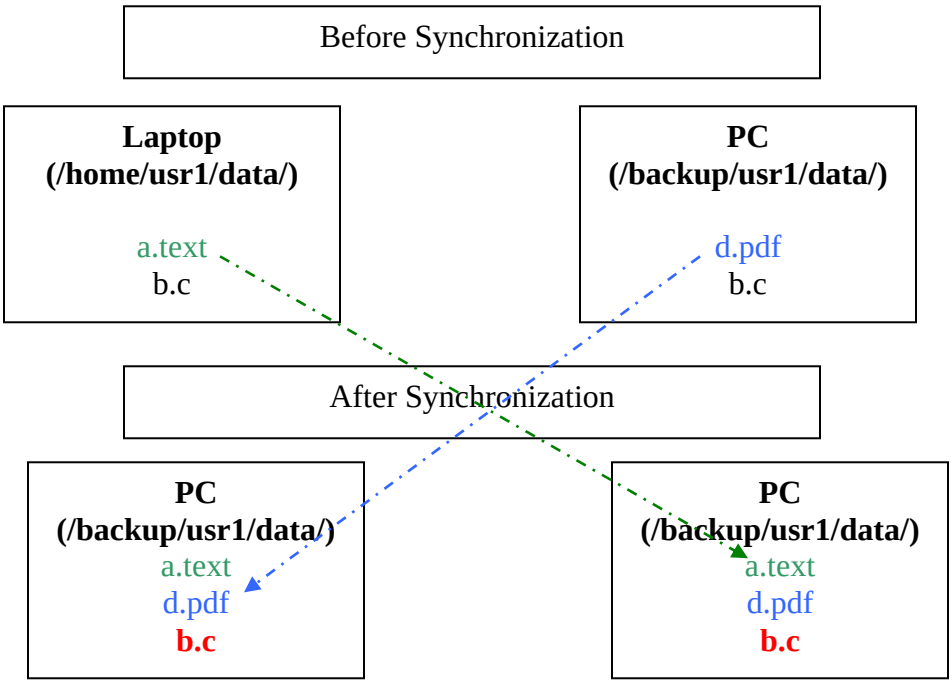
a. Separate connections for data and control.
b. Local Commands (on the ***client*** machine):
    i. lls (the initial *l* stands for *local*)
    ii. lpwd
    iii. lcd
c. Remote commands (on the ***server*** machine):
    i. ls -l
    ii. pwd
    iii. cd
d. Data transfer commands
    i. Get
    ii. Put
    iii. Mget
    iv. Mput
e. Authentication
    i. Connect to remote server and logon with username/password.
f. Resume Facility
    i. If a file upload/download gets interrupted, next time around the upload/download of this file should resume from the point at which it was left off.
g. Never use character by character transfer – use block transfers.
h. Ensure high throughput of data through socket – investigate.

Refer to RFC 959, http://www.faqs.org/rfcs/rfc959.html. You need not implement all of the protocol as specified in this document. You can simplify the protocol.

3. **File Synchronization**

Consider the following scenario:

1. You work on a laptop at home and on a PC in the lab. You frequently need to manually copy files between the laptop and PC to ensure that both the machines have all their files synchronized (i.e., same files, same versions, etc, diagram below).



In the diagram above, both the machines have the latest copies of each file, except for **b.c**, which was updated independently on both the machines – this is a conflict (if b.c was modified *only* on the PC, however, it would have been copied to the laptop).

2. You are a lab administrator and need to synchronize files among all the machines under your control.

Write a file synchronization application.

a. The architecture can be peer to peer or centralized – whichever seems suitable to you.
b. Given a list of IP's and a top directory (e.g. /home/usr1/data above), the application must recursively synchronize all the files starting from the top directory.
c. At the end of a run, it should generate a report – list the conflicts and the files copied (Source? Destination?).
d. In case of conflicting files the report should show the *diff*.
e. If there are two entirely *different* files with the same name, the application should be able to point this fact out to the user.
f. It should handle both scenarios 1 (two machines) and 2 (multiple machines) efficiently).

4. **A Network File Sharing Application**

    a. A number of workstations with files/directories that are shared.
    b. A number of client processes running on each of the workstations.
    c. Facilities on the client side:
        i. Add shares.
        ii. Remove shares.
        iii. Connect to a central server.
        iv. Maintain a local index of the files that are shared.
    d. There will be a central server facilities:
        i. Database of users.
        ii. List of connected users.
        iii. Browse files belonging to individual users.
        iv. General file search.
        v. Search for alternatives – i.e. same file with different names (think of some digest, e.g. CRC).
    e. File download from one user to another MUST be peer to peer.
    f. Points about the architecture of the application.
        i. The central server should not maintain the entire database of the files shared on the network.
        ii. When an individual connects to the central server, the latter retrieves an index of files shared from that client.
        iii. Suppose a download is in progress. If the connection to the central server is lost, the download should still continue (benefits of peer to peer download).