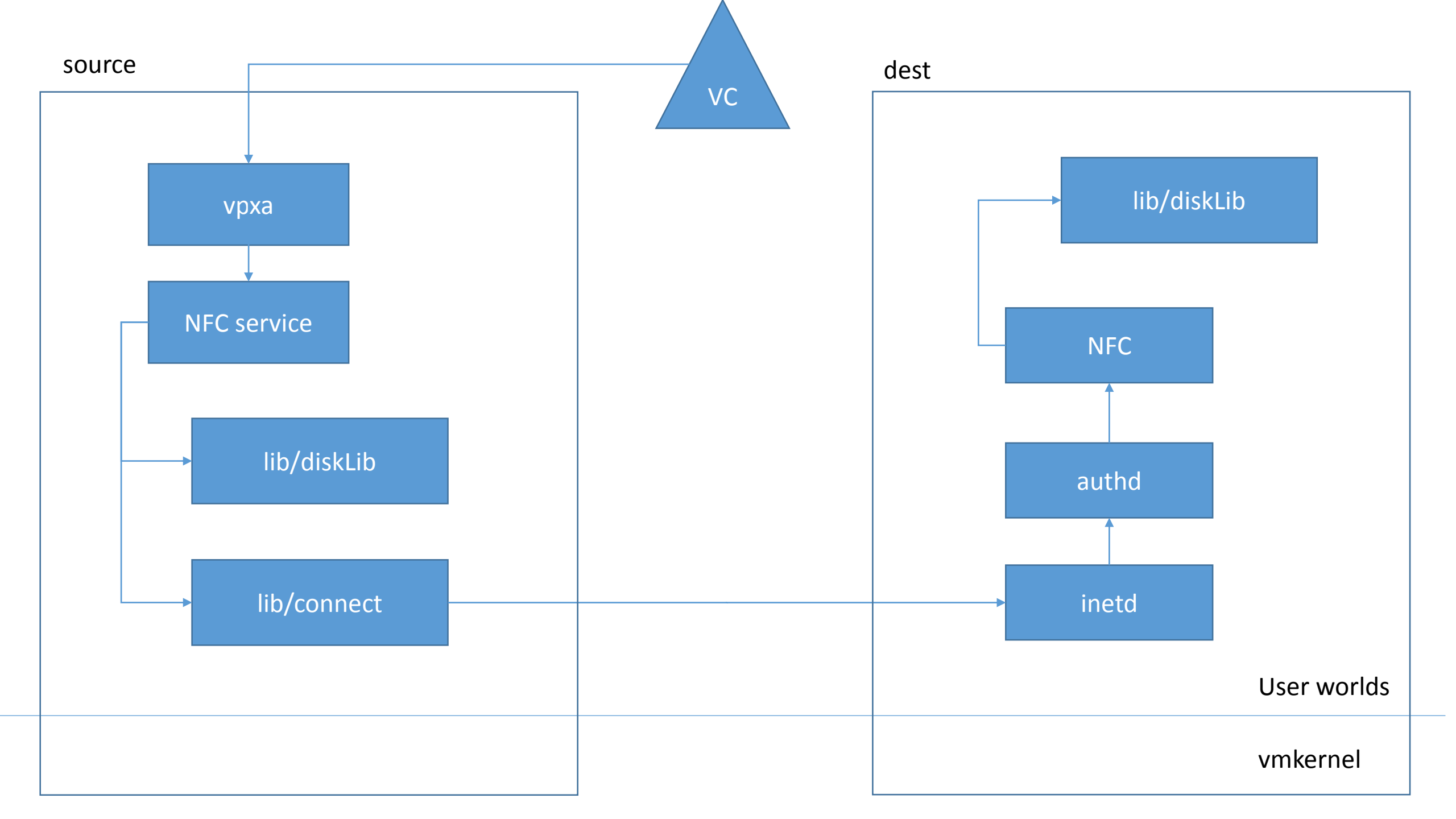
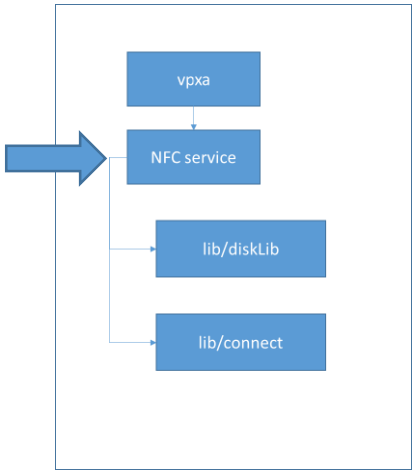
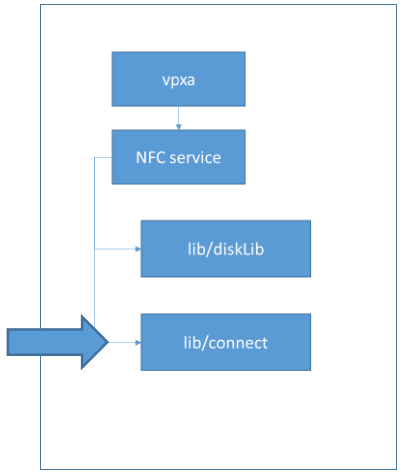
NFC

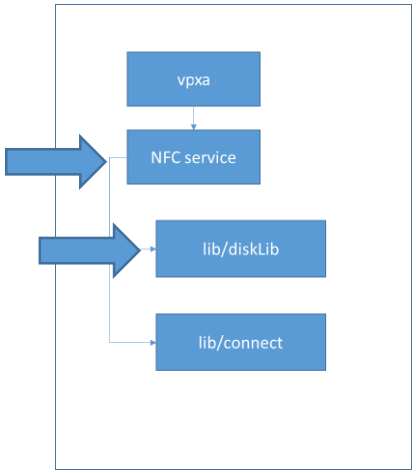

source
VC
dest
vpxa
NFC service
lib/diskLib
lib/connect
lib/diskLib
NFC
authd
inetd
User worlds
vmkernel




```
NfcManagerImpl::Copy(DataArray<CopySpec> *spec)
        Ref<NfcWorker> nfcWorker = new NfcWorker(_callbacks, GetLogger())
        NfcWorker::DoWork(DataArray<CopySpec> *copySpecs)
                NfcWorker::SetupConnections(DataArray<CopySpec> *copySpecs)
                        NfcClient::NfcClient(CopySpec)
                                CopySpec::CnxSpec *srcCnx = copySpec->GetSource()->GetCnxSpec();
                                CopySpec::CnxSpec *dstCnx = copySpec->GetDestination()->GetCnxSpec();
                                InitRemoteSession(remoteCnx->GetHost(), credentials, remoteCnx->GetPort(),
                                                  localCnx->GetHost(), remoteCnx->IsUseSSL());
                NfcWorker::TransferSpecs(copySpecs, connMap); // actual transfer
```

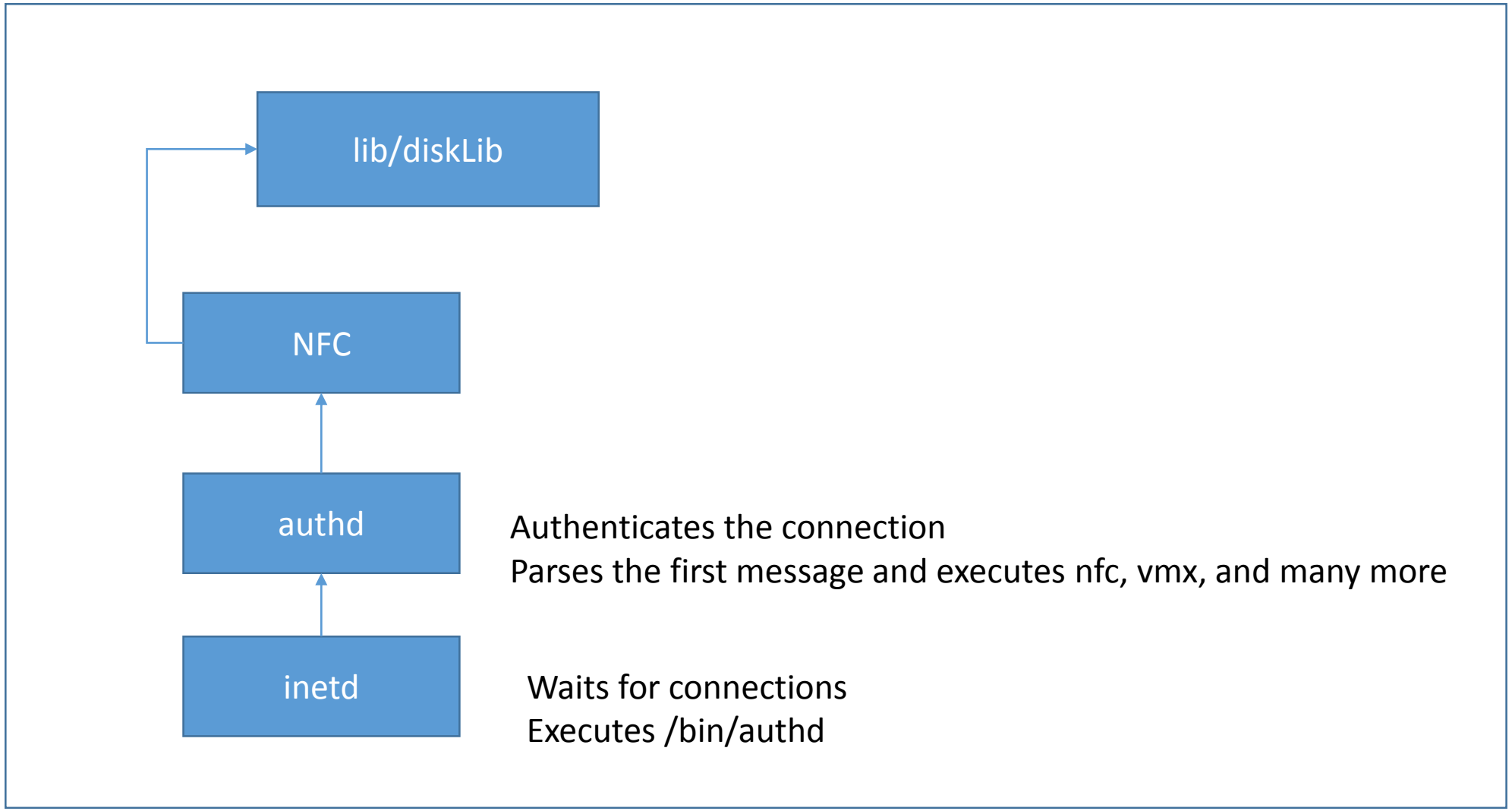


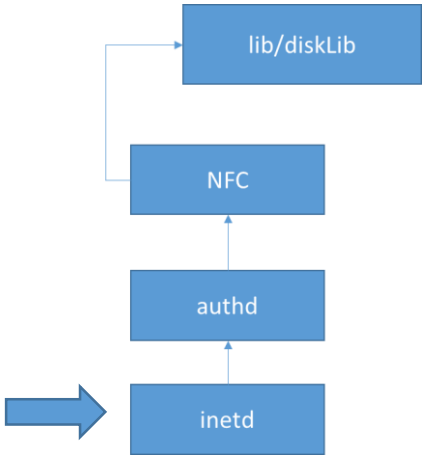
NfcClient::InitRemoteSession
Nfc_EstablishAuthdConnectionEx2
Nfc_BindAndEstablishAuthdCnx
Nfc_BindAndEstablishAuthdCnx2
Cnx_SetNetstackParams(lParams, netstack);
NfcNewAuthdConnection
CnxConnectAuthd
CnxAuthdConnect
CnxAuthdConnectTCP
CnxOpenTCPSocket




```
NfcWorker::TransferSpecs(copySpecs, connMap);
        for (unsigned int i = 0; i < _totalNumSpecs; ++i)
                CopySpec *copySpec = copySpecs->GetAt(i);
                TransferFiles(copySpec, fileList, *connMap[copySpec]);
                        for (; it != fileList.end(); ++it)
                                nfcClient.TransferFile(fInfo);
                                        switch (_connectionType) {
                                        case LOCAL:
                                                Nfc_LocalRename(_session, srcFilePath, dstFilePath)
                                                        NfcFile_Rename
                                                                Nfc_DiskLib_Rename(oldName, newName, NULL)
                                                                        _NDLCALL(DiskLibWrap_Rename)(old…)
                                        case OUTBOUND:
                                                Nfc_PutFileEx(_session, srcFilePath, dstFilePath)
                                        case INBOUND:
                                                Nfc_GetFile(_session, srcFilePath, dstFilePath)
```

dest


lib/diskLib
NFC
authd
inetd
Authenticates the connection
Parses the first message and executes nfc, vmx, and many more
Waits for connections
Executes /bin/authd




```
main()
{
        parse_config_file()
                // for each service
                        sep->se_fd = socket();
                        listen(sep->se_fd);
        for (;;) {
                ctrl = accept(sep->se_fd, (struct sockaddr *)0, (socklen_t *)0);

                pid = fork();

                if (pid) { // parent
                  continue;
                }

                dup2(ctrl, 0);
                close(ctrl);
                dup2(0, 1);
                dup2(0, 2);

                execv(sep->se_server, sep->se_argv);
        }
}
```

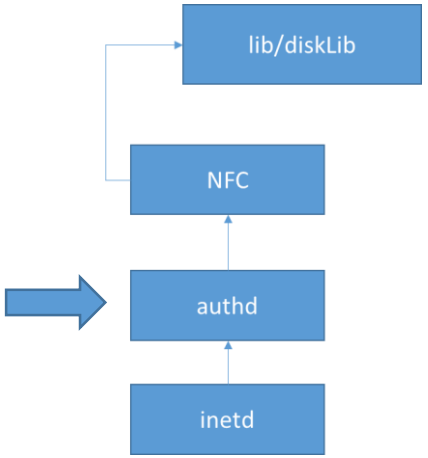
```
[root@waubesa:~] cat /var/run/inetd.conf
# Internet server configuration database

# Remote shell access

ssh      stream   tcp   nowait   root   /usr/lib/vmware/openssh/bin/sshd
sshd ++group=host/vim/vimuser/terminal/ssh -i
ssh      stream   tcp6  nowait   root   /usr/lib/vmware/openssh/bin/sshd
sshd ++group=host/vim/vimuser/terminal/ssh -i

# VMware authentication daemon
authd   stream    tcp   nowait   root   /sbin/authd           authd
authd   stream    tcp6  nowait   root   /sbin/authd           authd
```

Integer value	Name	<unistd.h> symbolic constant[1]	<stdio.h> file stream[2]
0	Standard input	STDIN_FILENO	stdin
1	Standard output	STDOUT_FILENO	stdout
2	Standard error	STDERR_FILENO	stderr

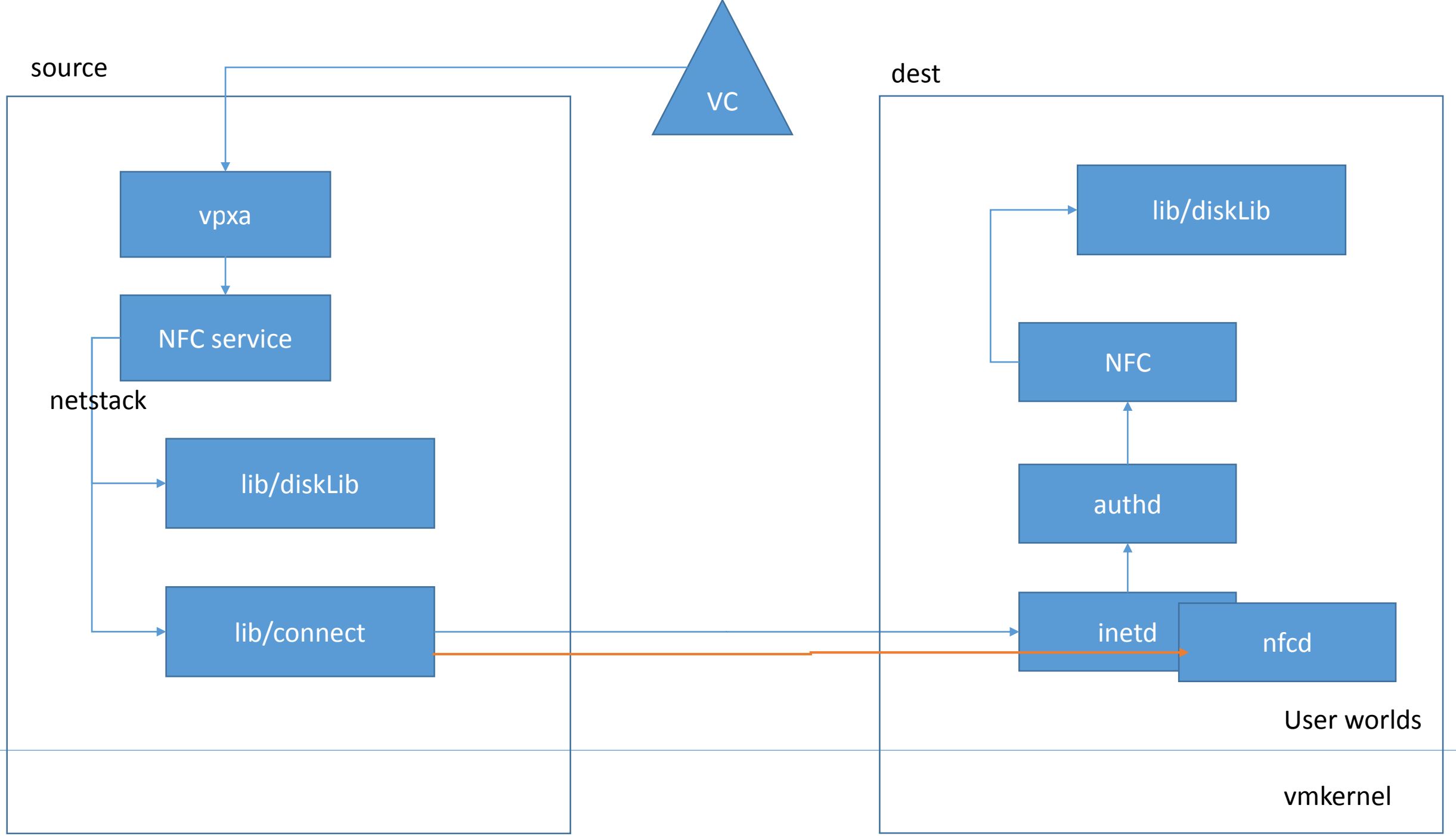


apps/vmauthd/vmauthd.c

```
main()
        VMAuthdCommandLoop(&specific);
            info = VMAuthdGetCommand(&arg);
                for (info = vmauthdCommands; info->command; info++) {
                    if (strncasecmp(cmd, info->command)
                        return info;
            info->callback(info->command, info->flags & CMD_USERFLAG)
```

```
static VMAuthdCommand VMAuthdBANNERCommand;
static VMAuthdCommand VMAuthdTHUMBPRINTCommand;
…
static VMAuthdCommand VMAuthdGLOBALCommand;
[illegible] VMAuthdCONNECTVPXACommand;
static VMAuthdCommand VMAuthdCONNECTCommand;
static VMAuthdCommand VMAuthdCONNECTARGVCommand;
```

```
VMAuthdCONNECTVPXACommand(socket_name)
    VMAuthdVpxaConnect(char *socket_name)
        VMAuthdDoConnect
            VMAuthdStartProcess(exec_name,...)
                fork();
                Posix_Execv(argv[0], argv);
```


source
VC
dest
vpxa
NFC service
netstack
lib/diskLib
lib/connect
lib/diskLib
NFC
authd
inetd
nfcd
User worlds
vmkernel

The nfc daemon (nfcd)

- Watchdog
- Nfcd.c

Nfcd.c

VmkCtl_Exec("/etc/init.d/nfcd restart", false, true)

Nfcd restart

- **apps/esxcli/plugins/network/IpInterfaceAdd.cpp**
- **vim/hostd/hostsvc/provider/vmkernel/networkSystemVmkImpl.cpp**