AFS Troubleshooting Tools

Marybeth Schultz

Transarc Corporation

Draft of January 11 1993

Contents

Chapter 1

Troubleshooting Problems

1.1 AFS Ports

Frequently, troubleshooting AFS problems requires communication with the appropriate port. The list below describes the ports used in AFS. This list can also be retrieved from *"/usr/afsws/include/afs/cellconf*

```
#define AFSCONF_FILESERVICE       "afs"
#define AFSCONF_FILEPORT          7000
#define AFSCONF_CALLBACKSERVICE   "afscb"
#define AFSCONF_CALLBACKPORT      7001
#define AFSCONF_PROTSERVICE       "afsprot"
#define AFSCONF_PROTPORT          7002
#define AFSCONF_VLDBSERVICE       "afsvldb"
#define AFSCONF_VLDBPORT          7003
#define AFSCONF_KAUTHSERVICE      "afskauth"
#define AFSCONF_KAUTHPORT         7004
#define AFSCONF_VOLUMESERVICE     "afsvol"
#define AFSCONF_VOLUMEPORT        7005
#define AFSCONF_ERRORSERVICE      "afserror"
#define AFSCONF_ERRORPORT         7006
#define AFSCONF_NANNYSERVICE      "afsnanny"
#define AFSCONF_NANNYPORT         7007
#define AFSCONF_UPDATESERVICE     "afsupdate"
#define AFSCONF_UPDATEPORT        7008
#define AFSCONF_RMTSYSSERVICE     "afsrmtsys"
#define AFSCONF_RMTSYSPORT        7009
```

The following list describes the ports used in the AFS Backup System:

```
#FOR BACKUP SYSTEM AFS 3.1
#define BC_MESSAGEPORT          7010 /* comm. w/backup coor,3.1 */
#define BC_TAPEPORT             7011 /* comm. w/tape controller,3.1 */
#define BC_TAPEPORT             7012 /* comm. w/tape controller,3.1 */
#define BC_TAPEPORT             7013 /* comm. w/tape controller,3.1 */
#define BC_TAPEPORT             7014 /* comm. w/tape controller,3.1 */
#define BC_TAPEPORT             7015 /* comm. w/tape controller,3.1 */
#define BC_TAPEPORT             7016 /* comm. w/tape controller,3.1 */
#define BC_TAPEPORT             7017 /* comm. w/tape controller,3.1 */

#FOR BACKUP SYSTEM AFS 3.2
#define TCPORT                  7010 /* testing only,3.2 */
#define BC_MESSAGEPORT          7020 /* comm. w/backup coor,3.2 */
#define AFSCONF_BUDBPORT        7021 /* comm. w/backup database,3.2 */
#define BC_TAPEPORT             7025 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7026 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7027 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7028 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7029 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7030 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7031 /* comm. w/tape controller,3.2 */
#define BC_TAPEPORT             7032 /* comm. w/tape controller,3.2 */
```

1.2 Tools

Troubleshooting AFS problems can be very cumbersome without the right tools. The following sections discuss the tools that are currently available to assist in problem determination.

1.2.1 cmdebug

The format of the cmdebug command is:

```
% cmdebug -help
Usage: cmdebug -servers <server machine> [-port <IP port>] [-long ] [-help ]
```

Cmdebug can be used to examine the status of the cache manager and cache entries on a particular AFS client machine. This command is useful when you suspect that the cache manager is hung, or is performing poorly. As a first step in diagnosing a problem, run cmdebug without the -long switch. A machine running in a normal state (i.e. not experiencing any problems), may display some locks set, but these locks should not be continually set over multiple cmdebugs.

The -port switch is only useful with port 7001 since that is the callback port. The -long switch will display the status of all of the cache entries on the AFS client when there are no problems.

```
% cmdebug dallas
%
```

If cache entries are displayed, those entries need to be examined in further detail as described below. Typical problems that are reported are unusual lock states on either the cache manager data structures or on individual cache entries. For example,

```
% cmdebug unix1.andrew.cmu.edu
Lock afs_xserver status: (none_waiting, 3 read_locks)
Lock afs_xconn status: (writer_waitingupgrade_waiting,write_locked,6 waiters)
** Cache entry @ 0x839abc94 for 1.33818459.10158.9086
    locks: (none_waiting, write_locked)
    20 bytes    DV 1 refcnt 1
    callback 020a0280   expires 654611582
    0 opens     0 writers
    mount point, read-only, mt pt valid
** Cache entry @ 0x839b3014 for 1.201430508.1.1
    locks: (none_waiting, write_locked)
    2048 bytes  DV 1243 refcnt 1
    callback 00000000   expires 652731340
    0 opens     0 writers
    volume root, stat'd
```

```
** Cache entry @ 0x839a29cc for 1.234881617.37.40906
    locks: (none_waiting, write_locked)
    2048 bytes  DV 0 refcnt 1
    callback 00000000   expires 0
    0 opens     0 writers
    normal file
```

Using the -long switch provides more detailed information about the contents of the cache on the client machine specified, even when there are no problems to report. An example of cmdebug output, and an explanation follows (line numbers are included for this example only):

```
% cmdebug dallas.transarc.com -long
 1 Lock afs_xvcache status: (none_waiting)
 2 Lock afs_xdcache status: (none_waiting)
 3 Lock afs_xserver status: (none_waiting)
 4 Lock afs_xvcb status: (none_waiting)
 5 Lock afs_xbrs status: (none_waiting)
 6 Lock afs_xcell status: (none_waiting)
 7 Lock afs_xconn status: (none_waiting)
 8 Lock afs_xuser status: (none_waiting)
 9 Lock afs_xvolume status: (none_waiting)
10 Lock puttofile status: (none_waiting)
11 Lock afs_ftf status: (none_waiting)
12 Lock afs_xaxs status: (none_waiting)
13 ** Cache entry @ 0xff186b1c for 1.1969551160.3274.127456
14     locks: (none_waiting, write_locked)
15     2453 bytes  DV 1 refcnt 0
16     callback 9e620303   expires 718510924
17     0 opens     0 writers
18     normal file, stat'd
19 ** Cache entry @ 0xff18b16c for 1.1969551160.1236.127684
20     3015 bytes  DV 1 refcnt 0
21     callback 9e620303   expires 718510924
22     0 opens     0 writers
23     normal file, stat'd
24 ** Cache entry @ 0xff18a08c for 1.1969551160.2262.127230
25     20519 bytes DV 1 refcnt 0
26     callback 9e620303   expires 718510923
27     0 opens     0 writers
28     normal file, stat'd
...
```

Lines 1-12 show the status of the cache manager's data structures. The "none_waiting" status indicates that the structures are available for locking and/or modification (e.g. adding or deleting cache entries). If any of the entries consistently shows that processes are waiting for a lock, it may indicate a problem such as a dead-lock condition.

Lines 13-28 are typical of the output for individual cache entries. The first cache entry from the above example is explained as follows:

```
13 ** Cache entry @ 0xff186b1c for 1.1969551160.3274.127456
14     locks: (none_waiting, write_locked)
15     2453 bytes  DV 1 refcnt 0
16     callback 9e620303   expires 718510924
17     0 opens     0 writers
18     normal file, stat'd
```

Line 13 shows the internal address of the vcache entry, along with a combination of the cell identifier (1 is the local cell), the volume id, the vnode number, and a uniquifier. The combined number is used to calculate the inode number of the file represented.

```
13 ** Cache entry @ 0xff186b1c for 1.1969551160.3274.127456

    where           0xff186b1c = internal address
                    1          = cell identifier
                    1969551160 = volume id
                    3274       = vnode number
                    127456     = uniqifier
```

There is a program called *calcinode.c* that resides in */afs/transarc.com/public/afs-contrib/tools* that converts the above combined number to the inode number. For example:

```
% calcinode

Enter volume id: 1969551160
Enter vnode no.: 3274
Inode number: 2000161994
```

Or, you could use bc to calculate the inode number manually:

```
% bc       (input)
ob=16      (input)
1969551160 (input volume id)
7564F738   (output hex representation)
3274       (input vnode number)
CCA        (output hex representation)
ob=A       (input)
ib=16      (input)
77380CCA   (input last 4 hex digits volume id and last 4 hex digits
            vnode number. If first digit > or = 8, subtract 8 from it)
2000161994 (output inode number)
^D
```

Finally, determine the file name:

```
% vos exam 1969551160
user.dino                        1969551160 RW      14412 K  On-line
    bigbird.transarc.com /vicepa
    RWrite 1969551160 ROnly          0 Backup 1969551162
    MaxQuota      15000 K
    Creation    Wed Sep 25 17:19:29 1991
    Last Update Wed Oct  7 18:37:50 1992
    33856 accesses in the past day (i.e., vnode references)

    RWrite: 1969551160    Backup: 1969551162
    number of sites -> 1
       server bigbird.transarc.com partition /vicepa RW Site
% cd ~dino
% ls -lRi | grep 2000161994
2000161994 -rw-r--r--   1 dino     root   2453 Oct  1 11:52 movie.star
```

The actual inode you'll see via ls -lRi is a combo of the least significant 15 bits of the vol id, and the least significant 16 bits of the vnode.

Line 14 shows the lock state of the cache entry if any locks exist. While it's not unusual for entries to be locked, entries that are locked constantly, or have a number of processes waiting for locks could indicate a problem.

Line 15 shows the size of the cache entry, the data version of the file, and the reference count for this entry. The cache manager compares the data version to the version number of the file at the file server to guarantee consistency. The reference counter is used to determine which cache entries can be reused when the cache is full (A reference count of 0 means that no one is using this entry. Very high reference counts may indicate a problem).

Line 16 displays the IP address (in hex) of the server holding the callback promise for this entry, and the time at which the callback expires. The IP address can be calculated as follows:

```
16      callback 9e620303   expires 718510924

        9e620303 --> 9e.62.03.03 --> 158.98.3.3
% bc
ob=A
ib=16
9E
158
62
98
03
3
```

Use */afs/transarc.com/public/afs-contrib/tools/calctime.c* to convert the callback expiration to english:

```
% calctime 718510924
718510924: Wed Oct  7 22:22:04 1992
```

Note that the expiration date for read-only items will be very large (only in versions of AFS $<$ or $=$ to 3.2), since callbacks aren't meaningful in those cases.

Line 17 shows the number of processes currently accessing the cache entry (not the cached data).

Finally, line 18 describes both the type of item represented by the cache entry (normal file, mount point, or a volume root), and the status of the entry (most common are stat'd, read-only, mt pt valid, and mapped [Sun]).

1.2.2 fileserver debugging

In some situations, it is necessary to get more information about the activities of a particular fileserver. For example, if you believe that the server is performing poorly you might like to see exactly what requests are/are not being serviced. Fileserver debugging can be turned on in the following manner:

```
kill -TSTP <processid of fileserver>
```

If the kill command is performed once, the debugging level is set to 1. The second time that the kill command is performed, the debugging level is set to 5, and the third time, the debugging level is set to 25. Level 1 gives you a list of calls, level 5 gives you the status returned on the calls, and level 25 will tell you where the call originated. All debugging information will be written to /usr/afs/logs/FileLog.

Note that you will need to use a sleep between each command if you're using only one line of command input. A line of:

```
kill -TSTP <processid of fileserver>;
kill -TSTP <processid of fileserver>;
kill -TSTP <processid of fileserver>
```

will not necessarily set the level to 25.

To turn debugging off and set the debug level back to 0:

```
kill -HUP <processid of fileserver>
```

The output for level 1 looks like:

```
Thu Oct  8 13:01:16 1992 Set Debug On level = 1
Thu Oct  8 13:01:38 1992 SAFS_CreateFile 00000,  Did = 1969563788.1.1
Thu Oct  8 13:01:38 1992 StoreData: Fid = 1969563788.2.2
Thu Oct  8 13:01:38 1992 SAFS_CreateFile 00276,  Did = 1969563788.1.1
Thu Oct  8 13:01:38 1992 SRXAFS_FetchData, Fid = 1969563788.4.3
Thu Oct  8 13:01:39 1992 StoreData: Fid = 1969563788.4.3
Thu Oct  8 13:01:39 1992 SAFS_CreateFile 00277,  Did = 1969563788.1.1
Thu Oct  8 13:01:39 1992 SRXAFS_FetchData, Fid = 1969563788.6.4
Thu Oct  8 13:01:39 1992 StoreData: Fid = 1969563788.6.4
```

The first field in the Fid is the volume ID, the second is the vnode, and the third is the uniquifier. The inodes used for a file are made up of part of the volume id and part of the vnode. Assume the inode is an 8-digit hex number: IIIINNNN. The first part, IIII is the last four hex digits of the volume id. The last part, NNNN, is the last four hex digits of the vnode. You can use bc to calculate this: There is a program called *calcinode.c* that resides in */afs/transarc.com/public/afs-contrib/tools* that converts the above combined number to the inode number.

```
% calcinode

Enter volume id: 1969563788
Enter vnode no.: 2
Inode number: 680263682
```

Or, you could use bc to calculate the inode number manually:

```
% bc       (input)
ob=16      (input)
1969563788 (input volume id)
7565288C   (output hex representation)
2          (input vnode number)
2          (output hex representation)
ob=A       (input)
ib=16      (input)
288C0002   (input last 4 hex digits vol id and last 4 hex digits vnode number)
680263682  (output inode number)
^D
```

Finally, determine the file name:

```
% vos exam 1969563788 -cell afs32b
vsu_ClientInit: Could not get afs tokens, running unauthenticated.
test.mb                          1969563788 RW        270 K  On-line
...
% cd ~mb/test.mb
% ls -lRi | grep 680263682
680263682 -rw-r--r--  1 mb             0 Oct  8 13:05 00000
```

The output for level 5 is very similar to the output for level one except that the return status is displayed. A return status of 0 is a success.

```
Thu Oct  8 13:04:35 1992 Set Debug On level = 5
Thu Oct  8 13:04:47 1992 SRXAFS_FetchData, Fid = 1969563788.1.1
Thu Oct  8 13:04:47 1992 SRXAFS_FetchData returns 0
Thu Oct  8 13:04:48 1992 SAFS_StoreStatus,  Fid = 1969563788.2.2
Thu Oct  8 13:04:48 1992 SAFS_StoreStatus returns 0
Thu Oct  8 13:04:48 1992 SAFS_StoreStatus,  Fid = 1969563788.4.3
Thu Oct  8 13:04:48 1992 SAFS_StoreStatus returns 0
Thu Oct  8 13:04:48 1992 StoreData: Fid = 1969563788.4.3
Thu Oct  8 13:04:48 1992 SAFS_StoreData returns 0
Thu Oct  8 13:04:48 1992 SAFS_StoreStatus,  Fid = 1969563788.6.4
Thu Oct  8 13:04:48 1992 SAFS_StoreStatus returns 0
Thu Oct  8 13:04:49 1992 StoreData: Fid = 1969563788.6.4
Thu Oct  8 13:04:49 1992 SAFS_StoreData returns 0
```

The output for level 25 looks like:

```
Thu Oct  8 13:05:11 1992 Set Debug On level = 25
Thu Oct  8 13:05:23 1992 SRXAFS_FetchData, Fid = 1969563788.1.1
Thu Oct  8 13:05:23 1992 SRXAFS_FetchData returns 0
Thu Oct  8 13:05:23 1992 SAFS_StoreStatus,  Fid = 1969563788.2.2
Thu Oct  8 13:05:23 1992 BCB: BreakCallBack(all but c037cf11.7001,
                         (1969563788,2,2))
Thu Oct  8 13:05:23 1992 BCB done
Thu Oct  8 13:05:23 1992 SAFS_StoreStatus returns 0
Thu Oct  8 13:05:23 1992 SAFS_StoreStatus,  Fid = 1969563788.4.3
Thu Oct  8 13:05:23 1992 BCB: BreakCallBack(all but c037cf11.7001,
                         (1969563788,4,3))
Thu Oct  8 13:05:23 1992 BCB done
Thu Oct  8 13:05:23 1992 SAFS_StoreStatus returns 0
Thu Oct  8 13:05:23 1992 StoreData: Fid = 1969563788.4.3
Thu Oct  8 13:05:24 1992 BCB: BreakCallBack(all but c037cf11.7001,
                         (1969563788,4,3))
```

The numbers of the form c037cf11.7001 represent the ethernet address of the machine making the call and the port number from which it is being made. You can use bc to translate the ethernet address.

```
c037cf11 --> c0.37.cf.11 --> 192.55.207.17

% bc
ob=A  (input)
ib=16 (input)
C0    (input)
192   (ouput)
37    (input)
55    (output)
CF    (input)
207   (output)
11    (input)
17    (output)
^D
```

To kill a fileserver safely (assuming no bosserver running), use

```
kill -QUIT <processid of fileserver>
```

To make sure the fileserver doesn't run the salvager when you restart it, delete /usr/afs/local/SALVAGE.fs.

1.2.3 fs debug

The format of the fsdebug command is:

```
% fs debug -help
Usage: fs debug -debug <'on' or 'off'> [-dafs <afs debug level>]
[-dnet <network debug level>] [-help ]
```

Fsdebug allows a user to turn cache manager debugging on/off. When on, the cache manager will record information about its activities in /usr/vice/etc/AFSLog (or other location if specified by the -logfile which on afsd startup). Note that this command does require root privileges, both for initialization and for reading /usr/vice/etc/AFSLog (unless another location is specified).

An example of an fs debug session and its analysis is included below:

```
% su rootl
Password:
# ls -l /usr/vice/etc/A*
-rw-------  1 root           0 Feb 12  1992 /usr/vice/etc/AFSLog
# fs debug on -dafs 1
# ls -l /usr/vice/etc/A*
-rw-------  1 root       44841 Oct  8 09:06 /usr/vice/etc/AFSLog
# fs debug off
# more /usr/vice/etc/AFSLog
lookup dir ff17f718, name transarc.com
lookup fid is 1.75647bbe.72.64d0
crossing mt point ff18a4c4
Volume's root.cell (ff1602e0) state=0
fid 1.75647bc6.0.ff17f718 in volume ff160350(state=1)
mount point is to 1.75647bbe.72.64d0
lookup returning ce ff1858d4
lookup dir ff1858d4, name @sys
lookup fid is 1.75647bc6.f0.421d
crossing mt point ff189984
Volume's sun3x_411 (ff1606d0) state=0
mount point is to 1.75647bc6.f0.421d
lookup returning ce ff1824cc
lookup dir ff1824cc, name usr
lookup fid is 1.7564e57d.2.2
crossing mt point ff187b48
Volume's sun3x_411.usr (ff160740) state=0
mount point is to 1.7564e57d.2.2
```

```
lookup returning ce ff1822b0
lookup dir ff1822b0, name afsws
lookup fid is 1.7564e580.10.27e5
crossing mt point ff1853e8
```

To determine what volume ids are being accessed, you can run */afs/transarc.com/public/afs-contrib/tools/analyze-logs/AFSLog.getvol*. Note that you you need to change the cellname in the script first.

```
% AFSLog.getvol < /usr/vice/etc/AFSLog|sort|uniq > ~/AFSLog.getvol
% more ~/AFSLog.getvol
75647bbe
75647bc6
75647bc6
7564e57d
7564e580
```

To translate the volume ids from hex:

```
% bc
ob=A        (input)
ib=16       (input)
75647BBE    (input)
1969519550  (output)
75647BC6    (input)
1969519558  (output)
7564E57D    (input)
1969546621  (output)
7564E580    (input)
1969546624  (output)
```

These volume ids can then quered to determine that actual volume name using "vos examine".

```
% vos examine 1969519550
root.afs.readonly                1969519550 RO        101 K  On-line
...
% vos examine 1969519558
root.cell.readonly               1969519558 RO        282 K  On-line
...
% vos examine 1969546621
sun3x_411                        1969546621 RW          3 K  On-line
...
```

```
% vos examine 1969546624
sun3x_411.usr                    1969546624 RW         10 K  On-line
...
```

To determine what files/directories are being accessed by the cache manager, you can run */afs/transarc.com/public/afs-contrib/tools/analyze-logs/AFSLog.getfiles*.

```
% AFSLog.getfiles < /usr/vice/etc/AFSLog > ~/AFSLog.files
% more ~/AFSLog.files

transarc.com/@sys/usr/afsws/
```

Note that you might get garbled pathnames if the machine is being used by many different users.

1.2.4 rxdebug

The format of the rxdebug command is:

```
rxdebug -servers <server machine> [-port <IP port>] [-nodally ]
[-allconnections ] [-rxstats ] [-onlyserver ] [-onlyclient ]
[-onlyport <show only <port>] [-onlyhost <show only <host>]
[-onlyauth <show only <auth level>>] [-help ]
```

Rxdebug is used to look at the RPC status of the servers by communicating with the appropriate port. Please note that the AFS version of rxdebug that you are running must match the AFS version of client and fileserver binaries that you are running. If there is a mismatch, you may get unreliable results.

"-allconnections" displays both server and client connections.

"-nodally" is used to look at the active connections that are not in an idle state. Dally mode is a state in which the connection is idle/waiting.

"-onlyauth" displays only those connections with the specified authentication level.

"-onlyclient" displays only client connections.

"-onlyhost" displays information specific to the named host.

"-onlyport" displays all connections to the named port.

"-onlyserver" displays only server connections.

"-rxstats" displays rx statistics. (Each statistics is not explained in detail here, but important ones are noted when appropriate).

In each of the examples that follow, the following apply:

"Free packets" displays the number of free packets available for use for RPC communications. This number is relative to the number initially allocated in rx and can vary from server to server based upon the switches used when the fileserver is started. Attention is necessary when the number of free packets is low because rx will discard received packets when there is not enough buffers. This causes the sending rx to retransmit upon timeout.

"packet reclaims" is not implemented yet.

"calls" displays the number of RPC calls successfully made to the specified server.

"used FDs" displays the number of user level file descriptor processes used, i.e. the number of files and sockets that are open. For client connections, this value should always be 64. For server connections, this value should always be less than 15.

"not waiting for packets" or "waiting for packets" displays the availability or inavailabilty of packet buffers available for RPC communications. "not waiting for packets" is normal. "waiting for packets" occurs when you run out of packets. In this case, processes are waiting for packet buffers so that they can send more data. Under normal circumstances, you should never see this. If you do, the number of packet buffers can be increased by using a switch on the server (report as a defect).

If there are any RPC requests being serviced, the next several lines display information about the current connection to the specified server.

"Connection from host ###.###.###.###" is the internet address of the remote host.

"port" displays the port number that the RPC is running on.

"Cuid" is a connection id number for the RPC request.

"serial" is the serial number of the last packet sent.

"security index" displays the type of authentication used between RPC nodes.

```
(0) no authentication
(1) oldstyle auth2 authentication ( < AFS 3.1b )
(2) kerberos style authentication
```

You will usually see (2) unless the connection is between servers, in which case (0) is used.

Each connection is broken into multiple calls. Typically, only call 0 is used. However, if the system is under heavy load, you may see others used.

"#" displays the call id number.

"state" describes the state of the connection.

```
active  - the call is currently active
dally   - the connection is idle/waiting
precall - initialized but before call
```

"mode" describes the ... of the connection.

```
eof       - completed
error     - an error has been received
receiving - actively receiving data
sending   - actively sending data
unknown   - none of the above
```

"flags" describe ...

```
call_cleared        - obselete
reader_wait         - the reader is waiting for the next packet from
                      the sender
receive_done        - the process has finished receiving packets
wait_packets        - corresponds to ``waiting_for_packets''
                      described above
waiting_for_process - the RPC is waiting to be serviced
window_alloc        - the RPC is waiting for the first window on a
                      connection
window_send         - an entire window full of packets has been
                      sent, but none have been acknowledged.  The
                      sender stops and waits for an acknowledgment.
                      This is rudimentary flow control.
```

For further detail, please refer to the AFS Programmer's Reference manual.

Example of the fileserver port:

```
% rxdebug ernie 7000
Trying 158.98.3.2 (port 7000):
Free packets: 319, packet reclaims: 0, calls: 2975439, used FDs: 6
not waiting for packets.
Connection from host 192.55.207.120, port 7001, Cuid ad7d0af5/ac564c14
  serial 1207, security index 0, client conn
    call 0: # 593, state dally, mode: receiving, flags: receive_done
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Connection from host 192.55.207.24, port 7001, Cuid ad7d0af5/ac565460
  serial 112, security index 0, client conn
    call 0: # 47, state dally, mode: receiving, flags: receive_done
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Connection from host 192.54.226.1, port 7001, Cuid ad7d0af5/ac564b7c
  serial 9932, security index 0, client conn
    call 0: # 5061, state dally, mode: receiving, flags: receive_done
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Done.
```

Example of the callback port:

```
% rxdebug ernie 7001
Trying 158.98.3.2 (port 7001):
Free packets: 65, packet reclaims: 0, calls: 4653, used FDs: 64
not waiting for packets.
Done.
```

To see what vlserver a particular client is talking to:

```
rxdebug client 7001 -allcon -onlyport 7003
Trying 192.55.207.40 (port 7001):
Free packets: 65, packet reclaims: 0, calls: 956, used FDs: 64
not waiting for packets.
Showing only connections on port 7003
Connection from host 192.55.207.52, port 7003, Cuid a59f9490/a39dec60
  serial 156, security index 0, client conn
    call 0: # 105, state not initialized
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Connection from host 128.2.10.11, port 7003, Cuid a59f9490/a39decc0
  serial 86, security index 0, client conn
    call 0: # 60, state not initialized
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Done.
```

Example of -rxstats switch:

```
% rxdebug ernie -rxstats
Trying 158.98.3.2 (port 7000):
Free packets: 311, packet reclaims: 0, calls: 6314559, used FDs: 6
not waiting for packets.
rx stats: free packets 0, allocs 17854337, alloc-failures(rcv 0,send 0,ack 0)
   greedy 1, bogusReads 0 (last from host 0), noPackets 0, noBuffers 0,
   selects 12835770, sendSelects 0
   packets read: data 7455052 ack 1415546 busy 0 abort 33922 ackall 8251
   challenge 4 response 38040 debug 9 params 0 (null) 0
   other read counters: data 7455019, ack 1415464,dup 18440 spurious 116
   packets sent: data 8853134 ack 373507 busy 0 abort 639446 ackall 1839
   challenge 38851 response 4 debug 0 params 0 (null) 0
   other send counters: ack 373507, data 8751666 (not resends),
   resends 101468,pushed 39579, acked&ignored 7554024
   Average rtt is 0.320, with 1356406 samples
   Minimum rtt is 0.002, maximum is 21.398
   229 server connections, 134 client connections, 134 peer structs,
   154 call structs, 111 free call structs
   0 clock updates
Connection from host 192.55.207.36, port 7001, Cuid ac833739/accf143c
  serial 397, security index 0, client conn
    call 0: # 200, state dally, mode: receiving, flags: receive_done
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Connection from host 192.55.207.74, port 7001, Cuid ac833739/accf0c48
  serial 1600, security index 0, client conn
    call 0: # 801, state dally, mode: receiving, flags: receive_done
    call 1: # 0, state not initialized
    call 2: # 0, state not initialized
    call 3: # 0, state not initialized
Done.
```

"noBuffers" is very important. This is the number of times that rx has discarded a received packet because it had insufficient memory allocated to copy it into. This forces the sender to retransmit under a timeout. If this field is large, it could indicate that the server resources have been saturated.

"pushed" are the packets which were explicitly NAKed by the receiver. For example, packet number n+6 was received before packet n+5 was received.

"resends" are the packets resent due to a timeout.

1.2.5 udebug

Udebug allows the System Administrator to look at the ubik status of a server by specifying the port of the database server process they wish to check. The format of the udebug command follows. As with other AFS commands, the "-long" switch is used to get more detailed output.

```
% udebug -help
Usage: udebug -servers <server machine> [-port <IP port>] [-long ] [-help ]
```

In each of the examples that follow, the following apply:

"Host ###.###.###.###" displays the IP address of the host that you are talking with.

"his time is 0" indicates that clocks are synchronized relative to the client's time.

"his time is n" indicates the number of seconds of skew between the local workstation clock and the server's clock. This number should be two or less. A number greater than 5 or 10 can be the source of synchronization problems. Note, it may be the workstation's clock that is off - compare servers' times.

"Vote: Last yes vote for ###.###.###.### at -xx" displays the server site that it last voted for and the number of seconds that have passed since the vote was performed. A vote is taken every 15 seconds to determine a sync site. Usually, the only server conducting a vote will be the sync site. However, if any server goes for 75 seconds without hearing from the sync site, then that server will initiate a vote. When a server is voting for an "unknown" file server you will see its vote for address 255.255.255.255. Note, the "-xx" should be less than 90.

"Last yes vote started at -#########" is the number of seconds that have passed since a yes vote was received. This in uninitialized or less than 90.

"Local db version is #########.#" is the version of the database at the local site. This should match the version listed with "Sync site's db version" below. The first part of the version number is a time stamp indicating when the current sync site was elected. The second part (after the "."), starts at one and is incremented on every update made to the database.

"I am (not) sync site" will display status of server that you are talking with.

"I am sync site until xx (x servers)" displays only if the server that you are talking with is the sync site.

"Recovery state xx" will appear only on the sync site. A hex value is displayed that shows its status. The normal status is "1f" which is what is printed if all of the bits are turned on. Anything other than "1f" could indicate problems, and the status of each bit should be investigated.

```
-The rightmost, or 1 bit (1), shows that it is the sync site.
-The second bit (2) shows that all of the database versions it sees
on the other database servers are ok
-The third bit (4) shows that the sync site has fetched
the best database that it sees
-The fourth bit (8) shows that it has changed the version number
of its database
-The fifth bit (16) shows that ever since it became the sync site,
it has sent a copy of its database to the other database servers.
```

"Lowest host ###.###.###.### at -x" displays the IP address of the server with the lowest IP address and the number of seconds that have passed since it voted for the lowest IP address host.

"Sync host ###.###.###.### at -x" displays the IP address of the file server machine that is the sync site, and the number of seconds that have passed since it voted for the sync site.

"Sync site's db version is #########.#" which is the version of the database that is housed at the sync site. This should match the version listed with "local db version" above.

"n locked pages, n of them for write" indicates the status of currently active updates, if any. This almost always displays 0. If it appears to be stuck at a value other than 0, contact AFS Product Support.

"This server started at -#########", displays number of seconds since the last restart of the file server. This field is not always reinitialized.

EXAMPLES

When a server has just started, and before it votes for another server, the output will look like this:

```
% udebug server3 7002
Host 192.55.207.13, his time is 1
Vote: Last yes vote for 255.255.255.255 at -76 (not sync site);
Last vote started at -637421183
Local db version is 637420836.1
I am not sync site
Lowest host 192.54.226.1 at -5
Sync host 192.54.226.1 at -5
Sync site's db version is 0.0
0 locked pages, 0 of them for write
This server started at -637421183
```

When the file server that is not a sync site votes for a sync site and finds one, the output looks like this:

```
% udebug server2 7002
Host 192.55.207.13, his time is 0
Vote: Last yes vote for 192.54.226.1 at -9 (sync site);
Last vote started at -9
Local db version is 637420836.1
I am not sync site
Lowest host 192.54.226.1 at -9
Sync host 192.54.226.1 at -9
Sync site's db version is 637420836.1
0 locked pages, 0 of them for write
This server started at -637421591
```

The output from a sync site is displayed below. Note that the status of the other database servers are also displayed with the following information:

"Server ###.###.###.###: (db #########.#)" shows the server IP address and the version of the database that server has.

"last vote rcvd at -#" indicates the number of seconds that have passed since that server voted for the sync site (if vote was yes). This value is normally < 16, and is always < 90 unless one or more servers are down or unreachable.

"last beacon sent at -#" indicates the number of seconds that have passed since the sync site sent beacons to the other database servers to verify that they are up. This value is normally < 16, and is always < 90 unless one or more servers are down or unreachable.

"last vote was ..." displays the status of the last vote taken.

"dbcurrent=n" shows that the database on that server is up to date with the database on the sync site. (n = 0 (not current), n = 1 (current))

"up=n" displays if the server is responding to beacons. (n = 0 (not responding), n = 1 (responding))

"beaconSince=n" displays that number of seconds that have passed since the last response to a beacon was received. This number should be small if the server is up.

```
% udebug bigbird 7002
Host 158.98.3.3, his time is 1
Vote: Last yes vote for 158.98.3.3 at -1 (sync site); Last vote started at -1
Local db version is 719493856.463
I am sync site until 59 (3 servers)
Recovery state 1f
Sync site's db version is 719493856.463
0 locked pages, 0 of them for write
This server started at -5048149

Server 192.55.207.52: (db 719493856.463)
    last vote rcvd at -2, last beacon sent at -2, last vote was yes
    dbcurrent=1, up=1 beaconSince=1

Server 158.98.3.2: (db 719493856.463)
    last vote rcvd at -2, last beacon sent at -2, last vote was yes
    dbcurrent=1, up=1 beaconSince=1
```

When running a single database server, a vote is never performed. Therefore, it will display "Vote: Last yes vote for 255.255.255.255", and the times displayed will get very large. In the first few lines of the output, the server believes that it is not the sync site. However, it indicates that it is the sync site later on in the output.

Here is what the output looks like for a single database server system:

```
% udebug server3 7003
Host 192.55.207.5, his time is 0
Vote: Last yes vote for 255.255.255.255 at -60244 (not sync site);
Last vote started at -637421924
Local db version is 637361685.1
I am sync site until 1510061723 (1 servers)
Recovery state 1f
Sync site's db version is 648534534.1
0 locked pages, 0 of them for write
This server started at -60239
```