

## A Network Transaction (or *How to spray*)

By Micah Altman  
Last Revision: Nov 11, 1997  
Copyright 1992-7

### Overview

This is an example of the networking operations that occur in executing the command *spray* from host *BOB*:

```
<BOB> spray target.com
```

### Types of Events

In order for the *spray* command above to be successful, three things need to happen:

1. The local must determine how to get to *target.com*
2. The local system must generate a *spray* request and send it to the next hop on its way to the destination system.
3. Intermediate hosts must forward the ip packets correctly.
4. The destination system must *receive* that request, *respond* to it, and be able to send a packet back to our local system, *BOB* .

### *Finding target.com*

- *Name to IP address resolution*: the name "target.com" has to be converted to an IP address. This might be done locally, using */etc/hosts* , or through the NIS or DNS mechanisms.

(NOTE: If you have more than one name service available, be sure to check the order in which each one is used by looking at */etc/resolv.conf* )

- *routing to first hop*: Once an IP address is found, a route must be found to that address. Your system will look in it's routing table (use *netstat -r* to see your routing tables) to determine what to do next.

( NOTE: Routing tables may have been created statically or dynamically. Static routes are often found in */etc/gateways* . Dynamic routes are usually created by the *routed* or *gated* daemons)

- *if the destination system is local* : BOB sends its message to that system

- *if the destination system is NOT local*: BOB sends its message to the next "hop" along the route to the destination. (And if all is well, the message will eventually be forwarded through all necessary hops to the target.)

- *getting the MACaddress* : If the system uses Ethernet, it will need the MAC address of the next hop. It uses *arp* to convert the IP address to the MAC address. If an entry for the destination is already in the arp table, it will use that local entry, otherwise it will send out an arp broadcast on the local lan, asking the host with the desired IP address to reply.

### *Making a Request*

- *Identifying the service:* a UDP or TCP port number for RPC is obtained from `/etc/services`, an RPC id for spray is obtained from `/etc/rpc`. (Or the appropriate NIS maps, if running NIS)
- *Requesting a port:* the spray program on BOB requests a dynamically allocated port number in order to be able to receive replies back from target.com.
- *Sending a message:* a packet of information is sent from BOB to the next hop. This packet contains both header information and application data. The header information contains, among other things, the IP address of the recipient and the sender, the port number for the recipient and sender, (in this case) and the RPC id for the service request. The application data contains information necessary for the spray daemon.

#### *Intermediate machines*

- *Getting to target.com:* If the routing information is correct, and the packet actually survives its trip through bridges, routers, gateways and backbones to target.com we move to the next section. Remember:
  - Each gateway (or other intermediate) must forward the i.p. packet to the next hop along the destination.
  - Routing tables do not specify a complete path from one system to another system, they only specify which gateway should be used *next*.

#### *Replying to a Request*

- *Listening for the request:* target.com must have a the portmapper or rpcbind daemon listening on the RPC port for requests in order to give the port # of the . (For many non-RPC services it would be inetd that would be listening). The portmapper/rpcbind will give the port # for the spray service. A daemon must also be listening on this port (in this case inetd, if inetd.conf is configured normally), to service spray requests.  
NOTE: In order for the request to be recognized correctly, the id numbers for the *spray* request must be the same on both target and destination systems. This requires that the RPC service be identically listed in the `/etc/services` file on both machine. Similarly, the RPC id for the spray service must be the same in `/etc/rpc`.
- *Authentication:* BOB must be in the access lists for RPC services.
- *Running the service:* a program is run to fulfill the client request  
NOTE: System resources such as memory, process slots, etc. must be available for this to work properly:
- *Responding with the results:*
  - *Finding a route back:* target.com must have a valid route back to BOB
  - *Identifying the client:* the client process is normally identified by its port number, which is usually dynamically allocated, and sent along with the initial request. Returning messages go to this port number.

### **Tools for Diagnosing Problems**

- Log Files/Lights:
  - Look for errors in the system log files from the network interface device driver and network daemons.
  - Some network daemons also have their own log files (e.g. named) - check

the man pages.

- Indicator lights on tranceiver can also indicate problems/congestion. Power and connection lights should be on, but collisions and erros shouldn't.

- Network Interface:

- `netstat -ia` - shows interface(s) configurations, collisions, errors in incoming/outgoing packets. You may use the interval count option to see rate of network traffic on interface.

- `ifconfig` - shows interface configuration options in addition to those shown by `netstat`

- `ping` - the simplest remote connectivity tool. Uses a very low-level ICMP echo packet. Indicates that remote interface is initialized -- but pings can be returned even when a system is hung or crashed, and pings can be blocked by routers and firewalls. Ping your own address to test basic ip configuration.

- Sockets:

- `netstat -a` - shows sockets in use, and connection states. Useful for diagnosing address in use errors and tracking opening and closing the closing of connections.

- Name Resolution:

- `yppcat [hosts.byaddr]` - shows NIS maps

- `nslookup/dig` - shows DNS maps and queries

- Routing:

- `netstat -r` - shows routes known locally

- `rtquery` - shows routes known locally, can be used to query remote routers

- `arp` - shows address resolution for IP->MAC addresses on ethernet.

- `traceroute` - attempts to show each step of route

- (note, avoid `ping -R`, which is often used for this purpose, but is unreliable)

- Testing remote side of connection:

- `ping` - tests remote network interface

- `telnet [host] [port]` - telneting to ports other than the telnet port can be used to test that the remote system is listening on that port. The echo service is a particularly simple service which can be used to test that `inetd` is running.

- Packet analysis:

- `netsnoop/snoop (IRIX)`, `tcpdump (public domain)/nfsdump (public domain)` -- shows contents of network packets

- Testing Performance:

- `netstat -i` : shows packets in/packets out, errors and collisions. Error packets can severely degrade performance. Excess collisions can indicate that the network is congested.

- `netstat -p [udf | tcp]` - shows tcp and udp statistics

- `netstat -m` - shows network memory buffers

- `nfsstat` - shows nfs stats

- `netperf` - use for benchmarking large udp/tcp transfers with various blocks sizes

- `spray` - Tests relative speed of interfaces. Floods interface with small packets - remote interface will drop packets if it can't keep up.

- `ping` - change blocks sizes, use "flood" option to get rough network turnaround times