Overview of
Dynamic Forward Routing

Edwin Brownrigg, Inventor

March 2003

Dynamic Forward Routing (DFR) is the expression of two US patents (USPTO 6,044,062, March 28, 2000; 6,249.516, June 19, 2001) for the invention of a software defined Mesh protocol. DFR, used in combination with Internet Protocol (IP), enables a metropolitan area network to configure itself dynamically. In a multi-radio wireless network, there are no pre-determined fixed paths. All routes are indeterminate. Worse, there can be many indeterminate paths, which if taken without intelligence can congest the network to the point of guaranteed failure.

Exactly how does this unique routing work? DFR enables every subscriber's radio to act not only as a source and destination of data packets, but also as a router of data packets on behalf of other users. In a network based on the client/server model, DFR overcomes the classic "hidden radio" problem. Any user of a cellular telephone who has lost a connection knows what it is to be "hidden" from the cell site. With DFR, the hidden radio asks neighboring client radios to help out and route data packets between it and the cell site. The hidden radio selects the best route through one or more neighboring radios.

The hidden radio also remembers alternate routes through other neighbors. In the event that the route of first choice becomes inoperative, the hidden radio instantly selects a cached alternate route. Even though there may be many possible multi-hop routes from a given hidden radio to a server radio, only the "best" route is selected. The radios that are in range, but not on the route, do not repeat the packet – do not interfere with those radios dynamically selected for "the" route.

When a client radio initializes, it broadcasts a "hello, server" packet. If the client receives a response from a server, they execute an authentication routine. The result is a single-hop connection. Failing the above, the client broadcasts a "hello, client" packet and accepts a

response from the first client to reply, executes the authentication routine, and immediately starts to hunt for a "better" route that the first one taken while, nonetheless, transporting payload data on behalf of the user. Here "better" is a function of quality of service measured in:

- number of hops
- end-to-end throughput
- loop-back time.

When not transporting payload data, the client is continually looking for a "better" route.

When a client broadcasts a "hello" packet, either to a server or another client, the "hello" packet is accompanied by the client's Media Access Controller (MAC) address. If the connection is one-hop between the client and the server, the server:

- adds the client's MAC address to the server's routing tree
- updates the server's Linux TCP/IP routing table
- builds a packet header with the server's MAC address as the source address and the client's MAC address as the destination address
- copies its in-memory routing tree to a string notation
- broadcasts the packet header followed by string notation of the server's in-memory routing tree to the client
- broadcasts recurring "hello" packets to test the client's presence in the network

If the client does not respond to the server within a period prescribed by the Network Operation Control (NOC), the server deletes the client from its in-memory routing tree.

Upon receiving the string notation of the server's in-memory routing tree, the client:

- copies the server's string notation of the server's in-memory routing tree, back into an in-memory routing tree
- updates the client's Linux TCP/IP routing table
- builds a return packet header with the client's MAC address as the source address and the server MAC address as the destination address.
- broadcasts recurring "hello" packets to test the server's presence in the network.

If the server does not respond to the client within a NOC-prescribed period of time, the client re-initializes.

When a new client's "hello" packet is not responded to by a server, but by another client, the responding client broadcasts its route (single- or multi-hop) to the server based on routing information provided by its adjacent client. The new client receives that routing data and builds a packet header with its MAC address as the source address, the server MAC address as the destination address, and the intervening client MAC address in sequence between the source address and the destination address. The other functions are as described above in the one-hop case.

A given client becomes aware of alternate routes by:

- examining the source address of every packet header it receives from broadcast mode
- looking up in its in-memory routing tree the acquired source address
- determining the number of hops to the server represented by that source address to the given client
- inserting that source address and its packet header into a stack of N-hop routes for future reference.

If N is less that the number of current hops between the given client and the server, the given client immediately re-routes through the newly acquired source address.

In the case when a given client looses connectivity to the server on its current route, it immediately starts to pop the stacks in ascending order of N until it successively acquires a new route. Meanwhile, should the number, N, of hops for an observed source address change, the given client updates its stacks accordingly.