CTCP: Coded TCP "How To" for Debian Squeeze

1. INSTALLATION
Install CTCP using:

```
sudo dpkg -i ctcp_1.0_i386.deb
```

2. STARTUP
Start CTCP proxies using:

```
proxy_remote start
proxy_local start
```

Check that listening on ports using:

```
netstat -a
```

The output should include two lines similar to the following:

```
tcp        0      0 *:socks      *:*      LISTEN
tcp        0      0 *:1081       *:*      LISTEN
```

The commands `man ctcp`, `man proxy_local`, `man_proxy_remote` provide further information.

3. INITIAL TESTING
Install proxychains (http://proxychains.sourceforge.net/) using:

```
sudo apt-get install proxychains
```

Edit file `/etc/proxychains.conf` as follows:

a. Uncomment the line containing `dynamic_chain`
b. Comment out `strict_chain`
c. Uncomment `quiet_mode`
d. Comment out line `proxy_dns`
e. Comment out last line `socks4 127.0.0.1 9050`
f. Add line: `socks5 127.0.0.1 1080`

Proxychains will now redirect connections via the CTCP proxy. Now check that CTCP proxy is functioning by typing:

```
proxychains telnet 127.0.0.1 22
```

This will connect to the local SSH port via the CTCP proxy. The output should look similar to:

```
ProxyChains-3.1 (http://proxychains.sf.net)
Trying 127.0.0.1...
Warning while making the logs directory: No such file or directory
Request for a new session: Client address 127.0.0.1 Client port 53628
Connected to 127.0.0.1.
Escape character is '^]'.
SSH-2.0-OpenSSH_5.5p1 Debian-6+squeeze3
```

To exit the connection just press the `<enter>` key. Now check external connection via the proxy by typing:

```
proxychains wget -4 www.google.com
```

(if need be, install `wget` first using `apt-get install wget`; note that the "-4" option ensures use of IPv4 since the CTCP proxy currently is untested with IPv6). The output should look similar to:

```
ProxyChains-3.1 (http://proxychains.sf.net)
--2013-02-24 19:25:20--  http://www.google.com/
Resolving www.google.com... Request for a new session: Client address
127.0.0.1 Client port 45634
173.194.66.104
Connecting to www.google.com|173.194.66.104|:80... Request for a new
session: Client address 127.0.0.1 Client port 53983
connected.
HTTP request sent, awaiting response... 302 Found
Location: http://www.google.ie/ [following]
--2013-02-24 19:25:20--  http://www.google.ie/
Resolving www.google.ie... Request for a new session: Client address
127.0.0.1 Client port 55852
173.194.67.94
Connecting to www.google.ie|173.194.67.94|:80... Request for a new session:
Client address 127.0.0.1 Client port 41543
connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: `index.html'

    [ <=>] 10,811      --.-K/s   in 0.02s

2013-02-24 19:25:20 (528 KB/s) - `index.html' saved [10811]
```

and if successful there will be a file `index.html` saved to the current directory.

4. TESTING ACROSS A NETWORK PATH

The setup we consider here is with two CTCP proxy servers, a local proxy running on the client machine and a remote proxy running on the server machine.

Client APP <-----> proxy_local <----------------------> proxy_remote <-----> Server

The local proxy listens on a local port and forwards connections from the client app to the remote proxy, which in turn passes the connection on to the content server e.g. a web server. The downlink connection from the remote proxy to the local proxy uses the CTCP transport, and so is protected against packet loss. Other connections (between client application and local proxy, between remote proxy and content server) use standard TCP and are not protected against packet loss.

By default (i.e. after initial installation as above), the local proxy binds to port 1080 on the local machine and the remote proxy binds to port 1081 on the local machine. That is, the CTCP transport is being used across an internal network path, which is high bandwidth and zero loss.

To test across an external path, two machines are needed a client machine and a server machine. Firstly, install CTCP on both machines as described above. On the client machine, type:

```
proxy_remote shutdown
```

to stop the remote proxy running on the client machine. On the server machine, type:

```
proxy_local shutdown
```

to stop the local proxy running on the server. We should now have only the `proxy_local` daemon running on the client machine, and only the `proxy_remote` daemon running on the server machine.

We need to tell the `proxy_local` daemon where the `proxy_remote` daemon is running. To do this, edit file `/etc/ctcp/proxy_local.conf` on the client machine and change the `UP_PROXY_ADDR` entry to equal to IP address of the server machine. Restart `proxy_local` by typing

```
proxy_local shutdown; proxy_local start
```

on the client machine. We can now test that a connection has been successfully established between the client proxy_local daemon and the server proxy_remote daemon by typing:

```
proxychains telnet 127.0.0.1 22
```

on the *client* machine. Since traffic is sent via the proxies, this should establish a connection between the client machine and the SSH port 22 on the server machine (note that the 127.0.0.1 is interpreted by proxy_remote to be its local machine i.e. the server machine). On the client machine, type:

```
netstat -a
```

and the output should be similar to the following:

```
tcp      0   0 localhost:socks         localhost:51691 ESTABLISHED
tcp      0   0 localhost:51691         localhost:socks ESTABLISHED
tc       0   0 149.157.192.240:59093   149.157.192.3:1081 ESTABLISHED
```

The first two lines are the connection to the local client proxy proxy_local. The third line is the connection between proxy_local and proxy_remote (in this example the server machine running proxy_remote has IP address 149.157.192.3 and proxy_remote is listening on port 1081, the client machine running proxy_local has IP address 149.157.192.240). On the server type:

```
netstat -a
```

and the output should be similar to the following:

```
tcp      0    0 149.157.192.3:1081    149.157.192.240:59093 ESTABLISHED
tcp      0    0 localhost:ssh         localhost:55100     ESTABLISHED
tcp      0    0 localhost:55100       localhost:ssh       ESTABLISHED
```

The first line is the connection between proxy_local and proxy_remote and mirrors the third line above from the client machine. The second and third lines are the connection from proxy_remote to the content server, which in this case is SSH listening on port 22 of the server machine.

5. EMULATING A LOSSY PATH

For testing, loss can be artificially introduced into a path using either Linux command tc or iptables. Using the iptables command:

```
iptables -A INPUT -m statistic --mode random --probability 0.05 -j DROP
```

introduces 5% packet loss on incoming traffic. To undo this command type:

```
iptables -D INPUT -m statistic --mode random --probability 0.05 -j DROP
```

and use iptables –L to confirm. For example, adding using iptables to add 5% loss on a link we can compare the download throughput achieved with and without CTCP by first running:

```
wget -4 http://www.hamilton.ie/seminars/videos/53-b_radunovic_hi.mp4
```

to obtain the throughput using standard TCP. This command downloads a large from from the Debian mirror. With 5% loss on a link, the above command achieves a download rate of about 900Kbps. Now use:

```
proxychains wget -4 http://www.hamilton.ie/seminars/videos/53-
b_radunovic_hi.mp4
```

to obtain the throughput using CTCP. On the same 5% lossy link the download rate increases to about 15Mbps.