WICED-SDK Iperf Application

Iperf is available in the WICED SDK as a console application. Load the application onto a WICED evaluation board, then connect with a terminal program (eg. PuTTY) with parameters 115200 8N1.

The current Iperf implementation joins to a Wi-Fi Access Point as a Wi-Fi client when the application runs. The SSID and Passphrase of the AP are specified in the <WICED-SDK>/Apps/test/iperf/wifi_config_dct.h file.

Instructions for running iPerf on WICED

Once the Iperf app boots, standard iPerf commands are available using the console. Some useful examples follow.

Start a TCP server:

```
> iperf -s
```

Start a UDP server

```
> iperf -s -u
```

Start a UDP Multicast server bound to 224.1.1.1:

```
> iperf -s -u -B 224.1.1.1
```

Start a TCP client

```
> iperf -c 192.168.1.136
```

Start a UDP client and send at voice priority (using the -S option), for 90 seconds at 10 Mbps, followed by the same command at video, best effort and background levels of priority:

```
> iperf -c 192.168.1.136 -u -S 7 -t 90 -b 10M
------------------------------------------------------------
Client connecting to 192.168.1.136, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[ ID] Interval       Transfer     Bandwidth
[  0]  0.0-90.0 sec  95.3 MBytes  8.88 Mbits/sec
[  0] Sent 67995 datagrams
[  0] Server Report:
[  0]  0.0-90.0 sec  95.2 MBytes  8.87 Mbits/sec   1.383 ms  106/67996 (0.16%)

> iperf -c 192.168.1.136 -u -S 5 -t 90 -b 10M
------------------------------------------------------------
Client connecting to 192.168.1.136, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[ ID] Interval       Transfer     Bandwidth
[  0]  0.0-90.0 sec  93.7 MBytes  8.73 Mbits/sec
[  0] Sent 66817 datagrams
[  0] Server Report:
[  0]  0.0-90.0 sec  93.7 MBytes  8.73 Mbits/sec   1.210 ms    9/66818 (0.013%)

> iperf -c 192.168.1.136 -u -S 0 -t 90 -b 10M
------------------------------------------------------------
Client connecting to 192.168.1.136, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[  0]  0.0-90.0 sec  90.6 MBytes  8.44 Mbits/sec
[  0] Sent 64607 datagrams
[  0] Server Report:
[  0]  0.0-90.0 sec  90.6 MBytes  8.44 Mbits/sec   1.449 ms    2/64608 (0.0031%)

> iperf -c 192.168.1.136 -u -S 1 -t 90 -b 10M
------------------------------------------------------------
Client connecting to 192.168.1.136, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[  0]  0.0-90.0 sec  42.1 MBytes  3.92 Mbits/sec
[  0] Sent 30037 datagrams
[  0] Server Report:
[  0]  0.0-90.0 sec  42.1 MBytes  3.92 Mbits/sec   3.077 ms   30/30038 (0.1%)
```

Start a UDP Multicast client:

```
> iperf -c 224.1.1.1 -u
------------------------------------------------------------
Client connecting to 224.1.1.1, UDP port 5001
Sending 1470 byte datagrams
Setting multicast TTL to 1
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[ ID] Interval       Transfer     Bandwidth
[  0]  0.0-10.1 sec  1002 KBytes   816 Kbits/sec
[  0] Sent 698 datagrams
```

Start iperf in a thread to enable concurrent client and server modes for bi-directional testing. The default application priority is 7 and so new threads should start at 7, as shown, however it can be set 6.

```
> thread_spawn 7 iperf -s
```

Starting a bi-directional UDP throughput test. Note that each instance of iperf must start on a separate port using "-p xxxx"

```
WICED> thread_spawn 7 iperf -s -u -i 10 -p 6000
WICED> iperf -c 192.168.1.136 -u -b 10M -t 90 -p 7000

LINUX_HOST$ iperf -s -u -i 10 -p 7000
LINUX_HOST$ iperf -c <WICED IP address> -u -b 10M -p 6000
```

Starting a bi-directional TCP throughput test. Note that each instance of iperf must start on a separate port using "-p xxxx"

```
WICED> thread_spawn 7 iperf -s -i 10 -p 6000
WICED> iperf -c 192.168.1.136 -t 90 -p 7000

LINUX_HOST$ iperf -s -i 10 -p 7000
LINUX_HOST$ iperf -c <WICED IP address> -p 6000
```

Restrictions & Known Issues

- If iperf is started in client mode and cannot connect with the server then the iperf app may reset.
- Many iperf parameters cannot be changed, for example TCP window size and max segment size.
- If the "thread_spawn" command is not used then after running an iperf server the WICED module must be reset, i.e. there is no break key to restore the console prompt.
- The "thread_kill" command is not supported.