Measure Wireless Network Performance Using Testing Tool Iperf

Lisa Phifer

Many companies are upgrading their wireless networks to 802.11n for better throughput, reach, and reliability, but getting a handle on your wireless LAN's (WLAN's) performance is important to ensure sufficient capacity and coverage. Here, we describe how to quantify network performance using iPerf, a simple, readily-available tool that measures TCP/UDP throughput, loss, and delay.

Getting Started

iPerf was developed to simplify TCP performance tuning by making it easy to measure maximum throughput and bandwidth. When used with UDP, iPerf can also measure datagram loss and delay (aka jitter). iPerf can be run over any kind of IP network, including local Ethernet LANs, Internet access links, and Wi-Fi networks.

To use iPerf, you must install two components: an iPerf server (which listens for incoming test requests) and an iPerf client (which launches test sessions). iPerf is available as open source or executable binaries for many operating systems, including Win32, Linux, FreeBSD, MacOS X, OpenBSD, and Solaris. A Win32 iPerf installer can be found at NLANR, while a Java GUI version (JPerf) is available from SourceForge.

To measure Wi-Fi performance, you probably want to install iPerf on an Ethernet host upstream from the access point (AP) under test -- this will be your server. Next, install iPerf on one or more Wi-Fi laptops -- these will be your clients. This is representative of a typical application flow between Wi-Fi client and wired server. If your goal is to measure AP performance, place your iPerf server on the same LAN as the AP, connected by Fast or Gigabit Ethernet. If your goal is to isolate bottlenecks, co-locate your iPerf server with real-world application servers, so that a comparable network path is traversed.

Alternatively, iPerf server and client can both be installed on Wi-Fi laptops. This can be helpful to measure client-to-client performance if you plan to support streaming video or voice calls between wireless clients. Again, make sure that iPerf traffic traverses the entire path you intend to test. For example, if you want to measure best case performance between co-located Wi-Fi clients, associate your iPerf client and server to the same AP. If you want to see how routing through upstream switches or across a WAN impacts performance, associate your iPerf server to a central AP and let your iPerf client associate with APs at various locations.

```
iPerf -c 127.0.0.1
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 127.0.0.1 port 4126 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0-10.0 sec   270 MBytes   226 Mbits/sec
```

Running iPerf

By default, iPerf clients establish a single TCP session to the iPerf server listening to Port 5001 at the

specified destination. For example, start your iPerf server by executing iperf --s at the command prompt, then open another window to start your iPerf client:

This just shows that you can measure throughput for traffic sent through the loopback address (127.0.0.1) on a single computer. If you cannot do this, you do not have iPerf installed correctly. By default, iPerf runs a 10 second test, measuring total bytes transmitted (e.g., 270 megabytes) and the resulting estimated bandwidth (e.g., 226 Mbps). Test length can be controlled by specifying time (-t seconds) or number of buffers (-n buffers). You can also view test results at regular intervals (-i seconds):

```
iPerf -c 127.0.0.1 -t 20 -i 5
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 127.0.0.1 port 4154 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0- 5.0 sec   134 MBytes   224 Mbits/sec
[1920]  5.0-10.0 sec   135 MBytes   227 Mbits/sec
[1920] 10.0-15.0 sec   133 MBytes   224 Mbits/sec
[1920] 15.0-20.0 sec   134 MBytes   225 Mbits/sec
[1920]  0.0-20.0 sec   536 MBytes   225 Mbits/sec
```

To run many tests over a long period, you may prefer to run your iPerf server as a daemon, writing server output to a log file. On Win32, this can be done by installing iPerf as a service (iPerf --s --D --o logfile.txt). When testing is finished, don't forget to remove the iPerf service (iPerf --s --R).

If your test traffic will pass through a network firewall, make sure that port 5001 is open or tell iPerf to use a port that is already open (e.g., iPerf --c --p 80). If your iPerf server is behind a NAT firewall, you may need to configure a port-forward rule to reach it. (This also applies to server-to-client traffic during bi-directional tests.) Finally, make sure that any personal firewalls on your iPerf client and server are disabled. Once the client can reach to the server, you can start measuring network performance.

```
iPerf -c 10.0.0.249 -F MyAppData.txt
------------------------------------------------------------
Client connecting to 10.0.0.249, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1916] local 10.0.0.170 port 4218 connected with 10.0.0.249 port 5001
[ ID] Interval       Transfer     Bandwidth
[1916]  0.0- 0.6 sec  1.39 MBytes  18.2 Mbits/sec
```

Measure Wireless Network Performance Using Testing Tool Iperf

Lisa Phifer

Measuring TCP Throughput

To determine max TCP throughput, iPerf tries to send just data as quickly as it can from client to server. Default data is sent from an 8 KB buffer, using the operating system's default TCP window size. To mimic a specific TCP application, you can tell your iPerf client to send data from a specified file (-F filename) or enter it interactively (-I). For example:

```
iPerf -c 10.0.0.249
------------------------------------------------------------
Client connecting to 10.0.0.249, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 10.0.0.170 port 4220 connected with 10.0.0.249 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0-10.0 sec  29.2 MBytes  24.5 Mbits/sec

iPerf -c 10.0.0.249 -P 2
------------------------------------------------------------
Client connecting to 10.0.0.249, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 10.0.0.170 port 4221 connected with 10.0.0.249 port 5001
[1904] local 10.0.0.170 port 4222 connected with 10.0.0.249 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0-10.0 sec  21.5 MBytes  18.0 Mbits/sec
[1904]  0.0-10.1 sec  21.7 MBytes  18.1 Mbits/sec
[SUM]   0.0-10.1 sec  43.2 MBytes  36.0 Mbits/sec

iPerf -c 10.0.0.249 -P 3
------------------------------------------------------------
Client connecting to 10.0.0.249, TCP port 5001
TCP window size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 10.0.0.170 port 4223 connected with 10.0.0.249 port 5001
[1904] local 10.0.0.170 port 4224 connected with 10.0.0.249 port 5001
[1888] local 10.0.0.170 port 4225 connected with 10.0.0.249 port 5001
[ ID] Interval       Transfer     Bandwidth
[1888]  0.0-10.0 sec  15.3 MBytes  12.8 Mbits/sec
[1904]  0.0-10.0 sec  14.9 MBytes  12.4 Mbits/sec
[1920]  0.0-10.0 sec  14.8 MBytes  12.4 Mbits/sec
[SUM]   0.0-10.0 sec  45.0 MBytes  37.6 Mbits/sec
```

Unless you specify otherwise, the iPerf client uses a single thread. You can change this by requesting multiple parallel threads (-P number). When testing Wi-Fi, multiple threads on the same laptop may slightly increase total throughput:

However, you'll need to use multiple laptops, each with its own Wi-Fi adapter, to simulate performance experienced by several discrete users. That's because multiple threads running on a single laptop are still sharing time on one Wi-Fi adapter.

On the other hand, if you have a laptop with more than one active adapter, you may wish to bind the iPerf client to one adapter using its IP address (-B IPAddress). This is particularly important when using a multi-homed laptop simultaneously connected to Ethernet and Wi-Fi (or 3G and Wi-Fi). iPerf was originally developed to assist with TCP parameter tuning, but we won't delve into TCP window size and max segment size here because that's not our goal. However, when testing high-throughput APs, you may find it necessary to tune TCP parameters to get more out of each individual iPerf client -- to learn more, see DrTCP.

Measuring UDP Loss and Delay

iPerf can also be used to measure UDP datagram throughput, loss, and delay. Unlike TCP tests, UDP tests do not send traffic as quickly as possible. Instead, iPerf tries to send 1 Mbps of traffic, packaged in 1470 byte UDP datagrams (fits into one Ethernet frame). This rate can be increased by supplying a target bandwidth parameter, specified in Kbps or Mbps (-b #K or --b #M). Here is an example:

```
iPerf -c 10.0.0.249 -u
------------------------------------------------------------
Client connecting to 10.0.0.249, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 10.0.0.170 port 4235 connected with 10.0.0.249 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec

iPerf -c 10.0.0.249 -u -b 10M
------------------------------------------------------------
Client connecting to 10.0.0.249, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1920] local 10.0.0.170 port 4236 connected with 10.0.0.249 port 5001
[ ID] Interval       Transfer     Bandwidth
[1920]  0.0-10.0 sec  11.9 MBytes  9.99 Mbits/sec
```

However, this just tells us only how quickly our iPerf client was able to transmit. To learn more about UDP delivery, we really need to look at server-side results:

```
iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.170 port 5001 connected with 10.0.0.249 port 1527
[  3]  0.0- 1.0 sec    123 KBytes  1011 Kbits/sec  1.174 ms    0/   86 (0%)
[  3]  1.0- 2.0 sec    122 KBytes  1000 Kbits/sec  1.017 ms    0/   85 (0%)
[  3]  2.0- 3.0 sec    122 KBytes  1000 Kbits/sec  0.987 ms    0/   85 (0%)
[  3]  3.0- 4.0 sec    122 KBytes  1000 Kbits/sec  0.902 ms    0/   85 (0%)
...
[  3] 29.0-30.0 sec    122 KBytes  1000 Kbits/sec  1.152 ms    0/   85 (0%)
[  3]  0.0-30.0 sec   3615 KBytes   986 Kbits/sec  1.756 ms   31/ 2518 (1.2%)
```

Here, we can see throughput (measured over one second intervals), accompanied by loss (the number lost versus the number received) and delay (i.e., jitter -- the smoothed mean of differences between consecutive transit times). The delay and loss that can be tolerated varies by application. For example, streaming video survives a good bit of delay by buffering input, while voice calls degrade quickly with delay.

UDP tests can be refined by changing datagram buffer length, specified in Kbytes or MBytes (-l #K or #M). Unlike Ethernet frames with their 1500 byte MTU (Maximum Transmission Unit), 802.11 data frames can be up to 2304 bytes (before encryption).

However, if you're testing a path that includes both Ethernet and 802.11, keep your test datagrams short enough to fit into one Ethernet frame to avoid fragmentation.

Another interesting iPerf UDP test option is Type of Service (ToS), which ranges from 0x10 (minimize delay) to 0x2 (minimize cost). In WLANs that use 802.11e to control quality of service, ToS is mapped onto Wi-Fi Multimedia (WMM) access categories. To learn more, see this tip on prioritizing Wi-Fi traffic.

Looking Both Ways

In 802.11a/b/g networks, over-the-air performance tends to be similar in both directions. For example, when distance causes data rate to drop or interference causes significant packet loss, both upstream *and* downstream application throughput are affected.

In 802.11n networks, MIMO antennas and multiple spatial streams make this less true. Data frames sent from laptop to AP may (intentionally) take a completely different air path than frames sent from AP to laptop. As a result, it is now important to measure performance in both directions. Fortunately, this capability is built into iPerf, controlled by two options:

- The --d option tells your iPerf server to immediately connect back to your iPerf client at the port given by the --L option for simultaneous tests in both directions.

- The --r option is similar, but tells your iPerf server to wait until client tests are completed, alternating to repeat each test in the reverse direction.

Finally, if you need to support multicast applications, try starting several iPerf servers with -B to specify a multicast group IP address. Then start your iPerf client, connecting to that multicast group of iPerf servers.

Graphing Results

iPerf programs can be run from the command line as shown in this tip, or launched from a Java front-end called JPerf. JPerf not only makes it easier to construct complicated command line parameters and save test results -- it also graphs those results in real-time.

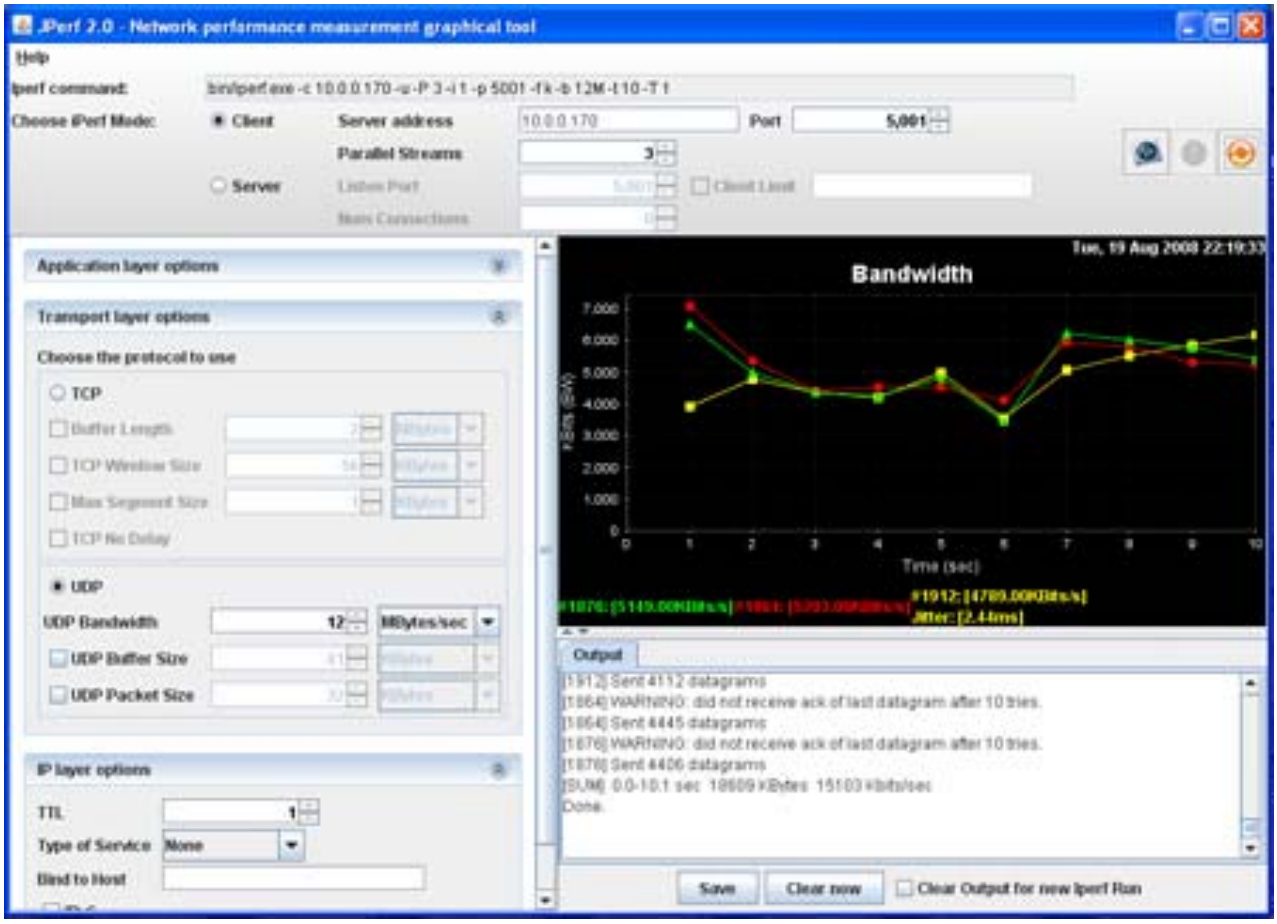

Figure 1. Using JPerf to launch iPerf

.

In fact, iPerf testers have been embedded in other network traffic analysis tools -- including lower-layer LAN analysis tools like AirMagnet. For example, the following screen snapshot illustrates a Wi-Fi laptop running AirMagnet being used as an iPerf client, interacting with an ordinary iPerf server installed on the wired network.

Measure Wireless Network Performance Using Testing Tool Iperf

Lisa Phifer

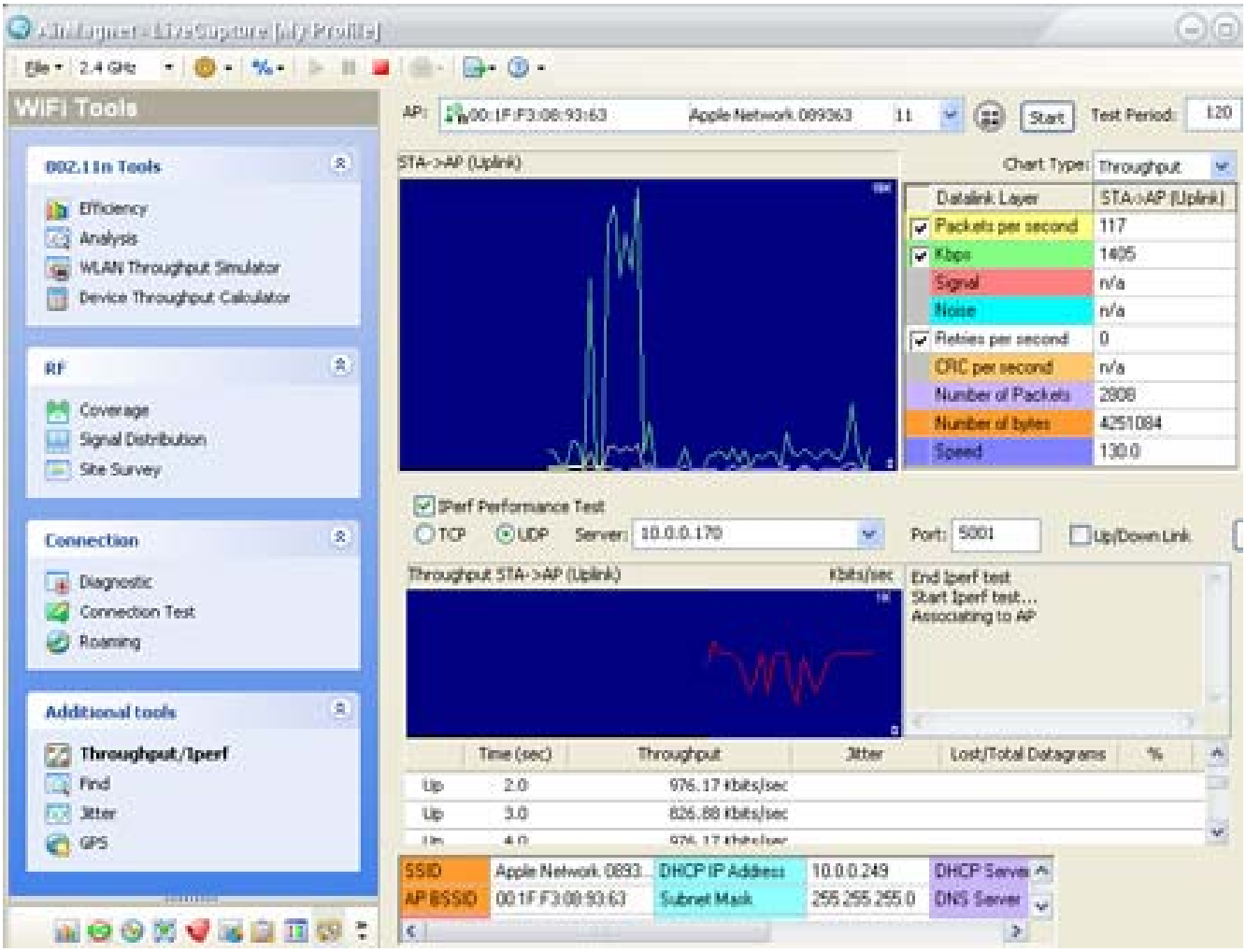

Figure 2. Using AirMagnet to launch iPerf

Conclusion

As we have seen, iPerf makes it easy to quantify end-to-end performance for TCP-based stream applications and UDP datagram applications. However, iPerf cannot simulate every kind of application -- for example, it isn't particularly good for simulating interactive Web surfing. Also, the WLAN adapters used for iPerf Wi-Fi testing can impact your measurements -- for best results, use a representative adapter in a configuration similar to your "real world" users.

Nonetheless, iPerf can be an extremely handy tool to help you generate and eyeball WLAN application traffic. Also, because iPerf is readily available as open source, it's a good way to create test scenarios that must be replicated elsewhere -- at branch offices, by vendor tech support, etc. To learn more about iPerf, see older documentation hosted at NLANR or visit the new project page at SourceForge.