MEASURING WIRELESS NETWORK CONNECTION QUALITY

Mike Mu

Avery Pennarun

Follow this and additional works at: http://www.tdcommons.org/dpubs_series

Recommended Citation
Mu, Mike and Pennarun, Avery, "MEASURING WIRELESS NETWORK CONNECTION QUALITY", Technical Disclosure Commons, (January 27, 2016)
http://www.tdcommons.org/dpubs_series/127

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
ABSTRACT

A system can run network tests to determine throughput for a wireless network connection between two devices. For example, a testing device can send a number of intentionally invalid test packets to a receiving device. The wireless MAC layer of the receiving device automatically sends back acknowledgment packets indicating that the test packets have been received. The timing and/or counts of the test packets and acknowledgment packets can be analyzed to determine throughput on the wireless network connection. Described features can measure network performance for any kind of wireless (e.g., Wifi™) device without installing any software on that device and without requiring any user interaction to initiate or perform tests.

BACKGROUND

In order to tune features of some types of wireless network deployments (such as coordinated channel selection, band steering, and assisted roaming), tests in the field must be performed with different algorithms and parameters, and end results compared. Lab and quality assurance tests are sufficient to find errors in the software operation, but not sufficient to determine if, in the field, the system works as well as it did in the lab. Wireless communication such as Wifi in particular tends to perform very differently for end users than it does in a test or isolation chamber.

The ability to objectively evaluate test results is desirable to determine if a particular test made network conditions better or worse. However, current data sources to obtain data for such evaluation are limited. Data sets are incomplete, e.g., device applications may only run on particular types or brands of devices, and/or tests are typically triggered only by humans, so are biased as to the conditions under which the data is collected. For example, Wifi performance is highly variable but network performance testing web sites (e.g., the speedtest.net website) only measure performance under very specific circumstances, such as when a customer actually runs
the test. This manual triggering of the web site tests can lead to sampling bias. Aggregate speedtest.net statistics are available, but browsers do not have access to useful information like client device type, Wifi connection speed, channel signal strength, etc., so cannot submit this information to the test server. Thus, measurements tend to be biased, infrequent, and hard to sort. Alternatively, test information could be obtained by using a native application running on the client device and server. However, these applications typically must be triggered by a human, must be installed by hand, and only run on certain device types (which further increases bias in results). Thus, testing of wireless network connectivity and evaluation of test results is limited using existing techniques.

DESCRIPTION

The present disclosure describes features related to measuring data throughput of a wireless network connection, such as a Wifi™ (e.g., IEEE 802.11) connection. Features described in this publication can measure performance against any kind of wireless (e.g., Wifi-compatible) device without installing any software on that device or requiring any user interaction. Described features provide wireless network tests which can be initiated entirely by a wireless access point or other device, and do not require any software to be installed on the client device, thus alleviating problems of prior wireless network tests and enabling regularly scheduled tests with unbiased results.

Maximum throughput can be determined, at any time, without requiring any special software to be installed on the client device and without any user interaction. Such features can use wireless network media access control (MAC) layer acknowledgements, such as Wifi (IEEE 802.11 standard) MAC layer acknowledgements, to determine if and/or when packets have been received. Because Wifi MAC layer acknowledgements are sent automatically by the network device hardware using the Wifi standard (e.g., network device chipset), they don't differ between device types (e.g., phone vs. tablet vs. laptop computer) or operating systems. Thus, unbiased test samples (i.e., across all devices) can be obtained at arbitrary or scheduled times that are not triggered by user interaction. For example, these tests can be triggered entirely from Wifi access point firmware, which can be controlled by a software or hardware developer.
Features described herein are enabled only with permission from the user. For example, the system can ask for user consent to gather information about the user’s devices, software, activities, network configurations, and other metadata, and to perform connectivity tests as described herein.

Operation Examples

Figure 1 shows a simplified system including a testing device 1 that is connected to a receiving device 2 via a wireless connection 3 of a wireless network. For example, testing device 1 can be any electronic device, including a network device, desktop computer, portable device (e.g., cell phone, tablet device, wearable device, etc.), game device, etc. In some examples, testing device 1 can be a wireless access point, e.g., network router, (such as a Wifi router), and/or other network device (switch, hub, etc.). Similarly, receiving device 2 can be any type of electronic device. Wireless network 3 can include any type of wireless network, e.g., a Wifi network, cell phone network, etc. Devices 1 and 2 include hardware and software capable of transmitting and receiving wireless signals enabling the transmission of information between the devices.

The overall process to perform the described wireless network throughput test is as follows. The testing device 1 sends wireless test packets to the connected receiving device 2. The wireless hardware of the receiving device 2 sends back response acknowledgments that are captured by the testing device 1. The test data can then be analyzed for loss, latency, and throughput information for the wireless connection 3.

The described technique involves intentionally sending a number of intentionally invalid packets from the testing device 1 (e.g., a wireless access point) to the receiving device 2 (e.g., a connection station on the network). The testing device 1 initiates a one-way test transmission of one or more "raw" test data packets to the receiving device 2. For example, the test packets can contain useless or random data. The test packets can be of any suitable type, e.g., TCP/IP, UDP, non-IP, etc. A large number of test packets can be sent.

The packets in the test transmission are formed so that they do not elicit a standard protocol reply from a station (e.g., receiving device 2) receiving the packets. Thus, the receiving device 2 will not interpret the test packets. For example, a transmitted TCP/IP packet will
normally cause the receiving device 2 to send back a TCP acknowledgment, which is avoided in this testing technique.

To cause the receiving device 2 to skip or omit interpreting the packets and sending a standard protocol reply to the transmitting device, one of several different options can be employed. For example, a non-IP EtherType value (e.g., a subtype) can be provided in the Ethernet frame of each transmitted packet so that the packet will not be recognized as an IP packet. In other examples, an invalid destination IP address can be provided for each transmitted packet so that the packets will not be seen as valid. In another example, a random UDP port number that is not in use can be included in each transmitted packet. For example, a closed UDP port will have its Internet Control Message Protocol (ICMP) errors rate limited in short order, so the return traffic should not be significant. In another example, a non-standard IP sub protocol can be used (e.g., not UDP).

In any of these examples, the kernel on the receiving device 2 will typically drop the packet right away without attempting to send back any kind of protocol response. Thus the transmission packets are queued in the transmit queue of the testing device 1, sent through the wireless network to the receiving device (e.g., a Wifi card of the receiving device), and then promptly thrown away.

However, an acknowledgment is automatically sent from the Wifi MAC layer of the receiving device 2. Since the test transmission is sent over a wireless network such as a Wifi network, the Wifi MAC layer of the receiving device 2 will receive a packet and will reply by sending a wireless acknowledgement packet back to the testing device 1. For packets received over a wired network connection, the MAC layer of the receiving device 2 does not send a reply back to the transmitting device. Wireless networks (such as Wifi networks), however, typically use a MAC layer that includes its own acknowledge (ACK) and retransmit system, e.g., independent from the MAC layer in a protocol such as TCP. This wireless MAC layer is included in all compatible wireless network devices due to the probability of wireless packet loss typically being high (e.g., often greater than 50%) even in fairly good transmission conditions, where the reply mechanism is useful to enable retransmission of lost packets. The wireless ACKs are usually implemented in hardware of the wireless chipset providing wireless network
capability, thus enabling that the acknowledgement packets are sent in a timely fashion, and are sent regardless of the packet content or operating system in use on the receiving device 2.

Thus, a replied wireless acknowledgement is sent back to the transmitting device, and the transmitting device can transmit the next queued packet. In some implementations, the transmitting device can resend a particular transmitted packet if no wireless acknowledgement packet is received for that packet in a particular amount of time after sending it.

In some implementations, e.g., newer Wifi standards (802.11n and later), multiple wireless acknowledgments can be included in a single batch acknowledgment packet (batch ACK) sent to the transmitting device by the wireless MAC layer of the receiving device after multiple packets have been received. This allows processing of the ACK responses for a high-speed, unidirectional data stream to take a small fraction of the CPU resources needed to generate the packets in the original transmission.

Analyzing Test Data

After a number of test packets have been transmitted and wireless acknowledgments have been received at the testing device, test data can be analyzed to determine measurements of characteristics of the wireless connection from the testing device 1 to the receiving device 2. Test data can be analyzed for loss, latency, and throughput information.

For example, the measurements of when and how the packets were delivered can be obtained in various ways. One technique is to analyze the wireless MAC acknowledgement / batch acknowledgement packets (e.g., as they are received, or from storage) to determine which packets were delivered at what times. For example, the times when the acknowledgment packets are received back at the testing device can be obtained. The testing device needs to be able to measure the time difference between when the original test packet was sent and when it was acknowledged.

Examining acknowledgment packets using this technique may require more processing time than some other implementations. In some example implementations, a Linux cfg80211 monitor (e.g., Radiotap) interface can be used, e.g., filtered (in hardware) to only examine Wifi “control” frames, which can keep the processing overhead relatively low. For example, on a
very fast link, transmit aggregates may be about 64 packets per frame, and thus an acknowledgement may only need to be examined (parsed) for 1/64 of the transmitted packets.

Another technique to obtain measurements is to examine driver level packet counters that have counted various types of packets that have been received and/or sent, to obtain aggregate results. For example, count values from counters for transmit, receive, and retry operations can be retrieved. Based on the counter values, it can be determined how many transmitted packets were sent in a predetermined time period, as well as how many wireless ACKs were received in that time period, to determine throughput and error rate for the wireless connection. This technique can be faster, requiring less processing resources.

Another technique to obtain measurements is to set a status flag on transmitted packets so that a different process will determine the acknowledgment status of flagged packets and send that status to the testing process/hardware. For example, a TX_STATUS flag can be set on all transmitted packets, which is recognized by the kernel of an operating system running on the testing device. The kernel examines receives wireless acknowledgment packets and sends the testing process a notification packet for each particular transmitted packet in response to that packet being acknowledged. This allows the testing process to obtain timestamps of acknowledgments without having to parse the packets, but may take processing time to generate notification packets.

Other Implementation Details and Features

Saturating the link: The test packets are intended to be transmitted using a saturated link so that maximum throughput can be determined. In very high quality link conditions, it might be more difficult to fully saturate the transmit queue on the test device 1.

Packets forwarded via the hardware forwarding engine may generally achieve a higher throughput than packets generated on the local device, due to reduced CPU overhead. Thus, some implementations can generate packets further upstream from the test device 1 and the hardware in testing device 1 can forward those packets, thus saturating the transmit queue.

In other implementations, a low overhead packet generation mechanism can be used to generate the test packets at the desired high rate. For example, packet generation features built
into the Wifi driver can be used. In one example, the Linux kernel’s pktgen can be used. Since the transmitted packets will all be almost identical, it can save a lot of CPU processing in comparison to using the kernel’s normal networking stack. For example, the counts from packet counters (as described above) can be used to determine throughput from such generated packets.

In other implementations, a ceiling can be placed on measured bitrates for the test. For example, tests can be intended to examine particular throughput rates, e.g., less than 200 Mbps, at which it should be easy to saturate the link. Any measurement above the ceiling (e.g., 200 Mbps) can be treated as exactly the ceiling bitrate (e.g., 200 Mbps), which still allows performance differences in the range of interest to be examined.

**Retransmitting Packets:** When a packet is lost in transit, the testing device wireless chipset can do one of two things: retransmit the lost packet, or give up and process the next packet. Retransmission time should be accounted for in determining the throughput of a wireless connection. For example, the connectivity test should differentiate between low throughput and high packet loss rate.

A stream can be defined for a transmitter to provide particular retransmit characteristics. For example, the 802.11 WMM standard (mandatory in 802.11n because it is used for all aggregation) allows transmitters to define more than one stream, each called a TID, with different characteristics.

In some implementations, a stream can be defined to have retransmits enabled. To determine the throughput on a retransmit link, a series of wireless batch acknowledgment packets that refer to a given sequence number of transmitted packets can be selected. Using these acknowledgment packets as an average sample, measurements can be determined for the number of times a typical packet needs to be retransmitted, and the typical latency of a packet, to determine actual throughput.

In other implementations, a special transmit mode (e.g., TID stream) can be defined to never retransmit packets lost in transit. The raw data indicating the number of packets transmitted and indicating which (or how many) transmitted packets were never acknowledged (e.g., error rate) can be used to determine the actual throughput. Using this method might result in simpler measurements with fewer edge cases. For example, a retransmit-capable stream might
fill up a queue and stop transmitting new packets until it is sure an old packet has arrived at the
destination, thus causing artificial decrease in total throughput. A non-retransmitting stream will
always send data at the maximum possible rate.

Testing Length and Initiation: A test cycle can have a length determined by any of various
factors. For example, each test cycle can have a length that is balanced between unnecessarily
loading the network (causing performance glitches for both the user of the testing device and any
other devices on the communication channel) and getting statistically significant data (for
example, some Wifi drivers might take time before they achieve their maximum transmit rate).

A test can be triggered to initiate in a variety of ways. For example a command line tool
can be used. The tool can run a test against a list of receiving devices (e.g., communication
stations identified by MAC address) or all attached receiving devices, either sequentially or in
parallel, and print the results in a machine-parseable format.

In some implementations, a connected server can trigger a test with a request sent to the
testing device. Options can be provided to run tests periodically and log the results.

In some cases, periodic tests can try to avoid times when the network is being heavily
used. In other cases, periodic tests can spread the test sampling to different times to more easily
measure performance under heavy load conditions of the network.

Output of Results: Test results can be sent to and stored in a system log containing device
information and metadata (as described below). The logs can be processed to generate
aggregated reports that can be output in any of various ways. For example, a table of overall
system performance can be displayed in a configuration user interface (UI). Such a table can be
used to help users view connectivity status and debug connectivity problems. Network
connection information based on the described measurements can be displayed in a customer UI
(e.g., a wireless router UI) and/or a customer service UI.

Figures 2 and 3 show example output that can be displayed, e.g., in a user interface to a
user, network administrator, etc. on a display of an electronic device. Fig. 2 shows one example
of a graphical display presenting network connectivity data in the form of a graph. The graph
shows a signal strength (RSSI) in the in the horizontal direction and a throughput value in
megabits per second in the vertical direction. The data obtained from the network tests described above can be used to form the displayed plots on the graph. Different plots can be shown for different frequency bands used by the wireless network, such as 2.4 GHz and 5 GHz in this example. Fig. 3 shows another example of a displayed graph presenting network connectivity data. In this example, signal strength (RSSI) vs. throughput value in megabits per second is shown for various percentiles, including maximum, 90th percentile, 10th percentile, average, and predicted results. A variety of other forms of graphs, statistics, tables, and/or data formats can be displayed to present network test data and results.

Metadata Collection and Privacy

Some implementations can collect metadata describing the characteristics of the testing device 1, the network connection 3, and/or other related system components, if such information is available. For example, such information can assist in analyzing network connectivity characteristics and problems. Examples of such metadata can include user account ID, MAC address of the testing (e.g., client) device, MAC Organizationally Unique Identifier (OUI) of client device, DHCP Hostname of client device, DHCP Fingerprint of client device, operating system and web browser application reported by device, serial number / model number / software version of client device (e.g., access point), Wifi signal strength (RSSI), Wifi average PHY rate (if available), Wifi number of antennas available, Wifi channel width in use (e.g., 20/40/80), Wifi band (e.g., 2.4 vs. 5 GHz), Wifi standard supported, Wifi channel number, number of Wifi stations on same access point, total number of Wifi access points on a LAN or other network, total number of stations on a LAN or other network, ping time, download speed, and upload speed.

In various examples, a metadata server or other storage can be used to store metadata, which can be accessed when providing wireless connection analysis, or providing results to a user interface.

Privacy of the user can be maintained by requesting user consent for measuring network connection status and/or using data describing the user's account, testing devices, and/or network connections. In some cases, user metadata entries can be expired after a predetermined time limit elapses after the information is stored.
In some examples, privacy features can be used for collected information about testing devices. For example, instead of the actual MAC address of the device, a pre-anonymized variant of the MAC address, with a random salt, can be used to allow differentiating test results between clients without allowing identification and tracking of clients.

Metadata can be used in the analysis of network test data to assist determining the cause of network errors or reduction in performance. For example, the test data and metadata may show that one type of device, at a given wireless network configuration, performs with throughput of less than the expected amount for 25% of time, but a different type of device performs as expected for 100% of time. This may indicate that particular types of devices are suitable for particular configurations. Statistical modeling can allow compatibility problems to be detected and their causes found.

The described techniques may be implemented in a software program, by computer hardware, or by a combination of software and hardware.
FIG. 1

Avg Mbps at each RSSI, by band

FIG. 2

Mbps breakdown at each RSSI (signal strength)

FIG. 3