August 16, 2019

Emulating WiFi networks for latency-sensitive applications

N/A

Emulating WiFi networks for latency-sensitive applications

ABSTRACT

Latency is an important performance parameter for online applications such as video conferencing, multiplayer video games, etc. The performance of such applications over WiFi networks, which sometimes gather packets in a queue until airtime becomes available, is of interest. This disclosure describes techniques to emulate WiFi transmission without actually adding WiFi networks to a hardware testbed such that the performance of latency-sensitive applications can be evaluated quickly in real time.

KEYWORDS

- Network emulator
- Latency
- WiFi
- Latency-sensitive application
- Network emulation

BACKGROUND

Latency is an important performance parameter for online applications such as video conferencing, distributed multi-player video games, etc. The performance of such applications over WiFi networks, which sometimes gather packets in a queue until airtime becomes available, is of interest to network designers, user-interface designers, and others.

Network simulators are one technique to study the performance of WiFi networks. Network simulators typically function as discrete-event simulators, which are not real-time; indeed, they can be rather time-consuming to run. Such simulators are unsuitable for testing the effects of WiFi performance on high-speed, low-latency applications such as video gaming.

Network emulators, e.g., a Linux traffic controller known as NetEm, can model a wired network performance in real time or nearly so. However, to model WiFi over a wired network emulator such as NetEm, queues at nodes have to be coordinated such that when one queue transmits, the other queues pause. To emulate the passing of control from access point (AP) to a client station (STA) and vice-versa, control has to be passed between queues within the emulator. Passing of control within NetEm is difficult because the emulator needs to be fast enough to keep up with real data transmission speeds and robust enough to forestall lock up.

DESCRIPTION

The techniques of this disclosure emulate WiFi within a wired network emulator, NetEm, such that WiFi performance can be assessed in real time, without the use of actual WiFi networks in a hardware testbed. In contrast to discrete-event network simulators that produce predicted performance results relatively slowly, the techniques deliver performance results in real time.

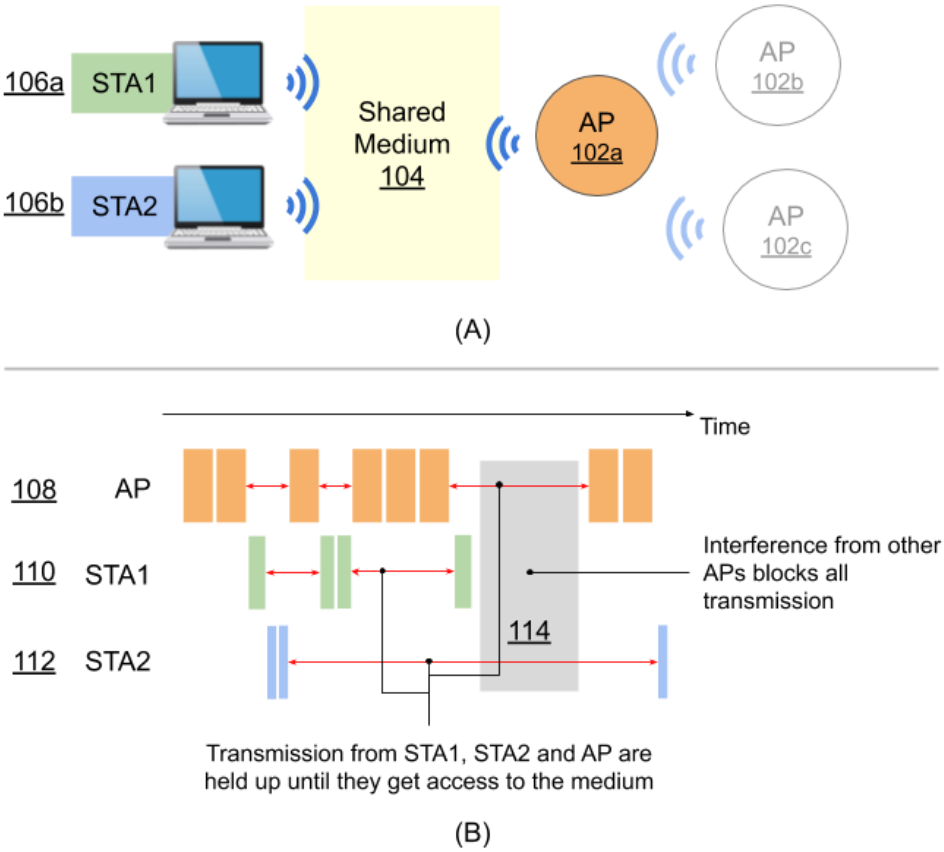

Fig. 1: (A) A typical WiFi environment; (B) Congestion in WiFi networks

Fig. 1A illustrates a typical WiFi environment, which comprises several access points (102a-c) and WiFi client stations (106a-b) communicating over a shared medium (104). The shared medium is radio frequency spectrum used by access points (AP) and stations (STA) for download as well as upload. Packets can be sent using time-division duplex, e.g., in only one direction at a time, and when the airwaves are free. This causes traffic to be sent in bursts or slots within which only one AP-STA pair has control of the medium at a time. This can cause congestion, as illustrated in Fig. 1B, which illustrates a timeline of packets originating from the AP (108), STA1 (110), and STA2 (112). For certain time intervals, other AP-STA pairs occupy the shared medium (114), thereby causing AP, STA1, and STA2 to pause transmission, in turn causing a queue build-up at these network nodes, leading to higher latency.

The techniques of this disclosure leverage NetEm [1], a Linux traffic control facility that enables an experimenter to add delay, packet loss, duplication and other characteristics to packets outgoing from a selected network interface.

Extracting WiFi Models

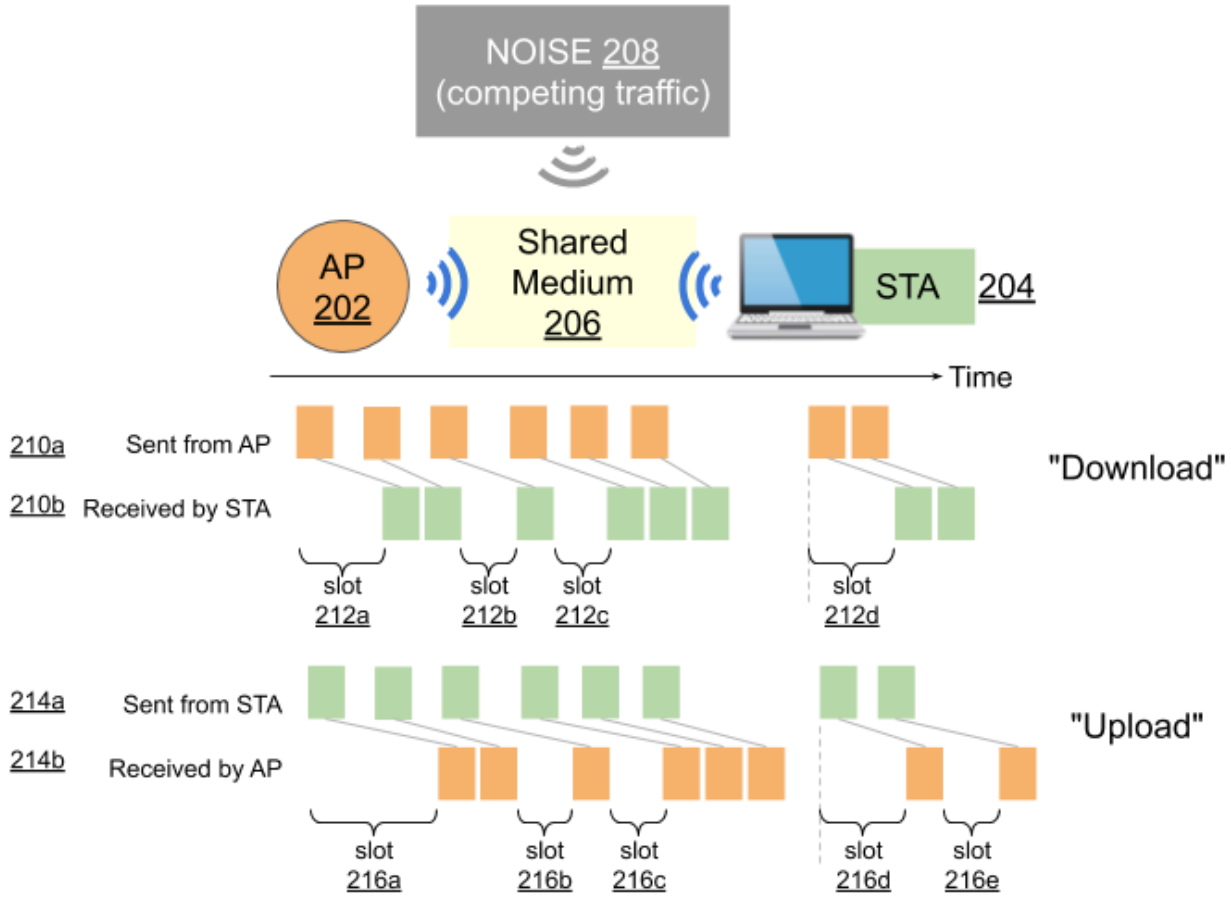

Fig. 2: Obtaining slot statistics using a test bed

To emulate the congestion and latency of WiFi over NetEm, statistics of slot intervals are obtained using a hardware test bed. Fig. 2 illustrates obtaining slot statistics using a test bed, per techniques of this disclosure. The test bed comprises an AP (202) in WiFi communication with a client station STA (204) over a shared radio medium (206) to which noise (208) of varying levels can be added. The noise represents competing traffic, and can have various models based on the

number of competing the APs/STAs, type of noise traffic (e.g., TCP/UDP, video, audio, etc.), the attenuation levels, etc.

During a download phase, packets sent by the AP (210a) are received by the STA (210b). Packets can be, for example, randomly generated UDP traffic in one or both directions, represent specific types of traffic, e.g., audio, video, etc. For the first slot, the difference in time between the transmission of a packet by the AP and its reception by the STA is the slot interval (212a). When packets pile up, the slot interval is the time between bursts (212b-c). When there is a break in traffic, the next transmitted packet is treated as if it were first in the series (212d). A histogram of slot intervals is formed for the download phase.

During an upload phase, packets sent by the STA (214a) are received by the STA (214b). Slot intervals (216a-e) are measured in a manner similar to the download phase. A histogram of slot intervals is formed for the upload phase.

While making slot interval measurements, care is taken to keep the traffic uniformly paced and to use sufficient data (e.g., more than 200,000 packets) so as to capture rare events.

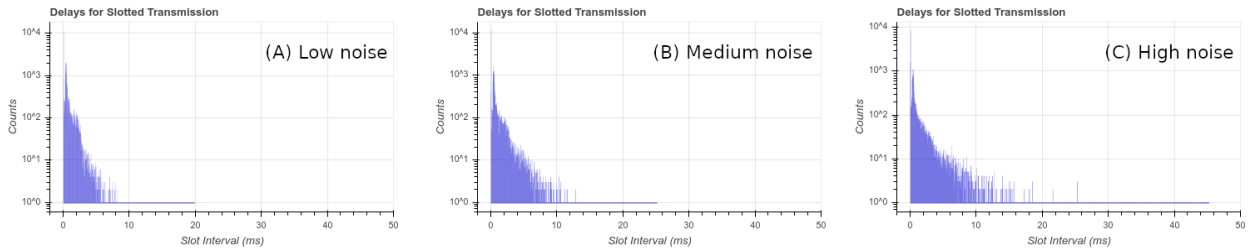

Fig. 3: Histograms of slot intervals under various noise conditions (A) Low noise (noise attenuation = 60 dB); (B) Medium noise (noise attenuation = 35 dB); (C) High noise (noise attenuation = 25 dB)

Fig. 3 illustrates example histograms of slot intervals obtained under different noise conditions. The measured histograms are transformed into a cumulative distribution function that is used to generate random slot intervals with a distribution that precisely matches the empirical

data. To improve the fit with the overall one-way delay, bandwidth limitations and other transmission delays are introduced as follows. Transmission delays are added either as a small uniform or Pareto distribution. Bandwidth impairments are introduced by limiting the number of packets that can be transmitted in a single burst. In this manner, by using an empirical slot distribution and by introducing extra delays and bandwidth restrictions, the NetEm model closely produces the same overall one-way delay as measured in the lab.

Emulating a WiFi Network over NetEm or other wired-network emulator

With the statistics of slot intervals obtained as described above, the techniques of this disclosure augment NetEm with a packet transmission control model called slot, which enables aggregation of packets until ready for transmission. By controlling the amount of data transmitted in each slot and the time between slots in accordance with slot interval statistics obtained as described above, packet aggregation that occurs on typical WiFi networks can be emulated, e.g., in cloud-based machines or other simulation environments. The random behavior of a real-time WiFi network is effectively shaped by burst patterns, e.g., by the probability distribution of slot intervals under various noise conditions.

In this manner, WiFi models can be studied for their effect on latency in live application sessions, e.g., gameplay sessions, conferencing sessions, etc. Such studies can also include components that request participants to rate their perception of performance. The results of such a study can be used towards better network design, improving user experience, etc.

CONCLUSION

This disclosure describes techniques to emulate WiFi transmission without actually adding WiFi networks to a hardware testbed such that the performance of latency-sensitive applications can be evaluated quickly in real time.

REFERENCES

1. Hemminger, Stephen. “Network emulation with NetEm.” 2005. https://pdfs.semanticscholar.org/9180/aa7b7978c62363e4af3a9053371775fbcbdc.pdf accessed on Jun. 25, 2019
2. Ns-3 Network Simulator. https://www.nsnam.org/