The Powwow Puzzler: The Impact of Transmission Power on Network Throughput in a Congested Environment

Ian Whitfield Karl Lopker

{ianwhitfield, klopker}@cs.ucsb.edu
Department of Computer Science
University of California, Santa Barbara

Abstract

The modern conference landscape is riddled with connectivity challenges, the most notable of which is interference induced throughput degradation in 802.11 wireless networks. In this paper, we propose a solution to increase total network throughput at these events without modifying existing hardware or protocols. Our solution utilizes a dense arrangement of access points and client devices with reduced transmission power. We record metrics from a collection of consumer hardware running in our wireless testbed and observe significant improvements in per-device goodput, total network goodput, and packet loss rate as power levels are reduced.

1 Motivation

With the recent explosion in WiFi-enabled smartphone devices, a public gathering can rapidly become an extremely dense wireless network. This issue is particularly acute at technical conferences or expos, where each attendee can potentially contribute several new devices to the network. With so many devices in a small space, these networks become highly congested leading to issues such as decreased throughput, or even total connectivity loss. Solutions to this problem commonly involve a combination of careful channel assignment, adaptive power control, and AP load balancing, as described in [4].

In this paper, we propose a "quieting" of wireless interference by reducing the transmission power of the wireless clients. While a reduction in transmission power can itself lead to connectivity loss, our particular scenario of a conference environment allows for some simplifications to be made. When in a conference most users will be listening to talks or having conversations, limiting their mobility. Additionally, the conference environment allows for infrastructure including a large number of stationary access points. In our model we ignore AP load balancing and simply assign clients to the closest AP.

We intend to discover if a reduction in transmission power will, when combined with a large number of access points, lead to a number of smaller cells with less interference. In this paper, we will determine the transmission power level that results in the highest level of total network goodput in our wireless testbed.

2 Related Work

Wireless transmission power control has already been studied both as a mechanism for improving performance and increasing energy efficiency.

The authors of [1] propose a solution for mitigating interference in "chaotic" wireless

networks: networks of co-located access points managed by multiple entities and exhibiting high density and organic growth. The solution reduces the transmit power of an AP to the point where all clients are still using the highest data rate. An experiment is conducted using two pairs of nodes and the proposed power adaptation algorithms are shown to provide significant gains. This work indicates that transmission power control can be an effective tool for reducing interference. In contrast, our experiment uses more nodes (12 total, vs. 4) and wireless transmissions instead of coaxial cables. As we are attempting to optimize total network goodput, we are also willing to allow some nodes to select a lower data rate.

In [2], the authors propose an algorithm for power control where all nodes are centrally controlled by the same entity and thus, unlike [1], do not need to worry about uncooperative nearby transmitters. The algorithm jointly optimizes the transmission power of each AP and its CCA threshold (the threshold for receive power that indicates a channel is busy, i.e. carrier sense). The authors then perform a simulation study, followed by a small testbed study (3 APs, 3 clients). Once again, this transmission power-based algorithm leads to substantial performance benefits. Specialized hardware was required for the testbed implementation of this algorithm, which limited its size. We hope that our simpler approach will lead to performance gains while still using cheap commodity hardware.

In [3], the authors also study power control but, instead of attempting to find minimum values, they seek to increase transmission power to improve link quality. The authors find that for a network in which nodes do not interfere, throughput increases with transmission power. Interfering networks are more complicated. The authors find that, in their testbed, some links interfere regardless of transmission power, while some links only interfere at certain power levels. In our experiment, we are attempting to examine the latter case, but it is also possible that we will be unable to reduce interference.

3 Design and Implementation

We conducted our experiments on a small-scale wireless testbed consisting of nine smartphone devices and three wireless access points. To more accurately emulate a conference environment, the testbed uses identical access points and an assortment of common smartphones.

The access points are ASUS RT-N10+s 802.11 b/g/n wireless routers. This model was selected for its low cost and customizable Linux-based firmware. In our testbed, the access points run the DD-WRT firmware distribution. Transmission power is set at the default level of 12.3 dBm. AP transmission power is not varied as they only send control packets during the test. The access points are configured to use 802.11g only, as most of the smartphones do not support 802.11n. All three use the same channel, 1. Each access point is surrounded by three smartphone devices, and all access points are connected (via a switch) to a central control server over wired gigabit Ethernet. The control server

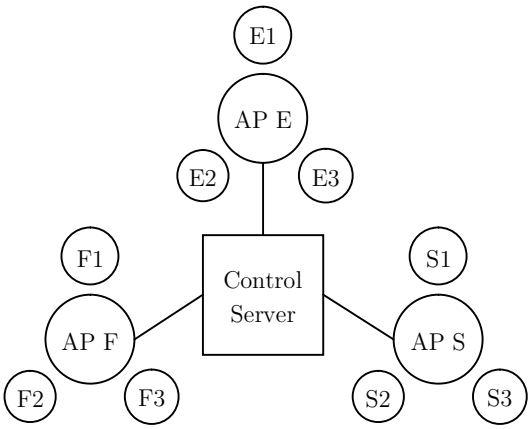


Fig. 1: Testbed layout and topology.

and smartphone devices run custom software for generating traffic and collecting metrics.

Several different types of Android devices are used to better reflect the diverse array of hardware found in a real-world wireless network. Android was selected because it allowed us to write a single client application for many different devices. The client software is written in Java with the Android SDK. When started, the client software prompts for the IP address of the control server. The client then opens a TCP connection to the server. This TCP channel is used to receive control commands from the server. TCP was selected because we do not want control packets to be lost because of network congestion. The client maintains (and displays for easy review) a log of all control commands received from the server for manual auditing. When instructed to generate traffic, the client transmits UDP packets to a predetermined port on the control server. Each packet contains a header that includes a sequence number, the link data rate at which the packet was sent, and the client's IP address. The packet is padded to a fixed size with random data.

The control server is a Dell M1330 laptop running the Xubuntu 11.10 operating system. The server runs a DHCP server for assigning IP addresses to clients, and our control server software. The server software is also written in Java for portability. The server can be used to send two different configuration commands to its clients: "set delay", and "set power". The server can also send "start" and "stop" commands for controlling the flow of generated UDP traffic from the clients. All commands are sent over the TCP connections previously established by the clients. The server is also responsible for receiving UDP test packets from the clients. The packet headers are logged, and saved to disk in CSV format for later analysis after each "stop" command.

The "set delay" command, is used to configure the delay between UDP packets sent by the clients. In early testing, we discovered that too short of a delay would overflow packet queues at various points in our testbed. This manifested as a uniform, periodic packet loss spike in our logs, even when only a single client was running. This configurable delay can be used to fine-tune the packet rate for different testbed configurations, and is used at the beginning of our evaluation procedure in the following section.

The "set power" command sets the transmission power of a client's WiFi radio in dBm. This is the primary variable changed during out test procedure as we attempt to find the transmission power required for optimal goodput.

4 Testing and Characterization

4.1 Selected Metrics

There are three metrics that we decided to collect for each device: goodput, packet loss, and data rate. We also calculate the total goodput of the entire network, as this is the value we are attempting to maximize.

We believe it is important to understand the individual contribution of each device to the total goodput of the network. It is possible that our experiment may starve some devices. Recording the goodput of each device will allow us to observe any such problems.

Packet loss is a good indicator of high congestion, insufficient transmission power, or a problem with the device. During our experiment, this metric allowed us to correct the unexpected issue of packet queue overflows in our networking hardware.

Lower signal strength may affect a device's data rate, potentially leading to less goodput. However, it is also possible that a lower data rate may still allow the maximum goodput. Monitoring the data rate will allow us to better explain observed goodput.

Before we can begin testing, the testbed must be calibrated to ensure that packet loss is being caused by congestion within the physical layer, and not by routing hardware along the path to the control server. This is done by varying the delay parameter, described in the previous section. To calibrate this parameter we used one client and started at 100ms. We then slowly decreased the delay until packet loss occurred. Then we added 1ms increments until the packet loss was gone. The additional clients were then added to verify that packet loss appeared random (as caused by congestion), and the delay was increased until any periodic losses were eliminated. For our testbed configuration, the calibrated delay value was 5 milliseconds.

We also had to select an appropriate packet size. Small packets are less likely to collide and can introduce significant overhead. Large packets are more likely to collide, but extremely large packets are uncommon in real-world traffic. We selected a packet size of 1024 bytes, which is close to the average Internet packet size and simplifies data analysis. This packet size differs slightly from our original testing specification, but not enough to affect our experiment.

Finally, we had to select the range of power levels over which to perform our analysis. After examining the capabilities of each driver, it was determined that 20 dBm to 1 dBm was the largest common range.

We had originally intended to perform our tests using both UDP traffic, for its simplicity, and TCP traffic, as it is more commonly seen in the real world. However, when we were developing the client software we ran into unexpected difficulties configuring the transmission power of the devices' WiFi radios. Several devices in our initial pool were did not support power level reconfiguration, and those that did required custom solutions for interfacing with their specific drivers. This process consumed large amounts of our time, forcing us to cut the TCP tests from our analysis. While this was not ideal, TCP behavior can still be inferred from our UDP-based results.

4.2 Evaluation Procedure

1. Power off all devices for at least 1 minute.
2. Power on the server, AP(s), then client(s).
3. Associate all clients with their AP.
4. Start server code and client code. Verify all clients are connected.
5. Set client transmit power to 20 dBm via server console.
6. Set client packet delay to 5ms via server console.
7. Verify all clients received the two previous commands via client log.
8. Server sends "start" message to clients, clients begin transmitting UDP packets.
    - The packet counter in the server console should now be incrementing. This is the test for correct operation. If the counter stops, check connections and make sure all clients are still associated with their APs.
9. After 5 minutes, server sends "stop" message, clients stop sending packets and the server closes the UDP socket.
10. Server writes received packet information to disk in CSV format for offline analysis.
11. Lower power of all clients to the next power level.
12. Repeat steps 8 - 11 until lowest power level is reached, or clients start disconnecting.

4.2 Evaluation Results

We now analyze the data collected by the testing server. As discussed previously, each iteration of our testing procedure produces a log of the packets received from each client device. The log contains the packet's source (IP address), the data rate at which it was transmitted, and its sequence number.

As each packet has been padded to a fixed size, the number of packets received can easily be translated into a total amount of data. Since each test runs for a fixed time duration, the following simple formula was used to calculate the average goodput from each device.

$$\text{goodput}_{\text{avg}}(i) = \frac{\text{packets}(i) * \text{SIZE}}{\text{TIME}}$$

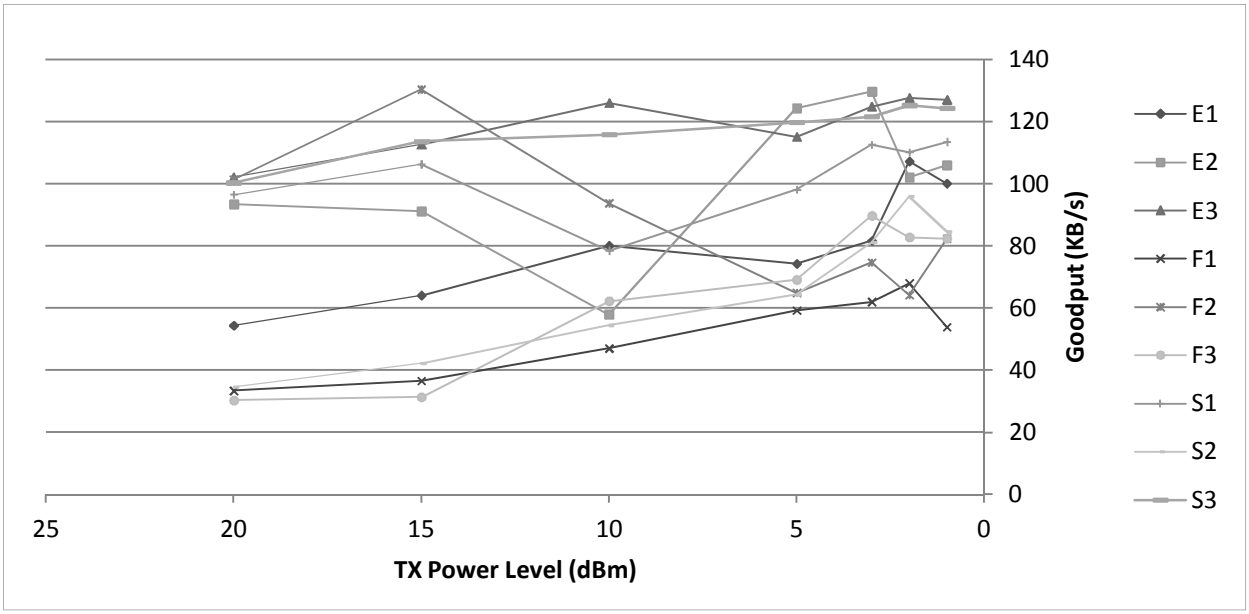
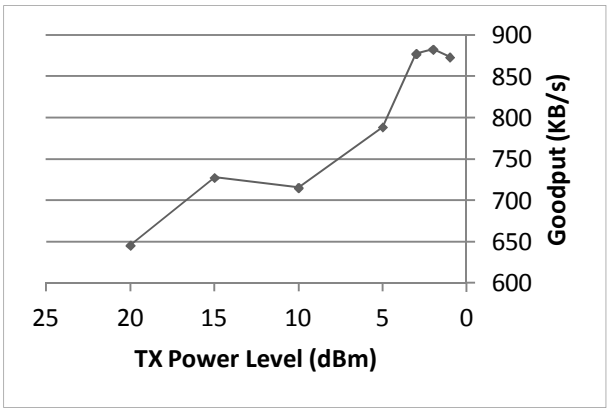


Fig. 2: Per-client goodput at each power level.



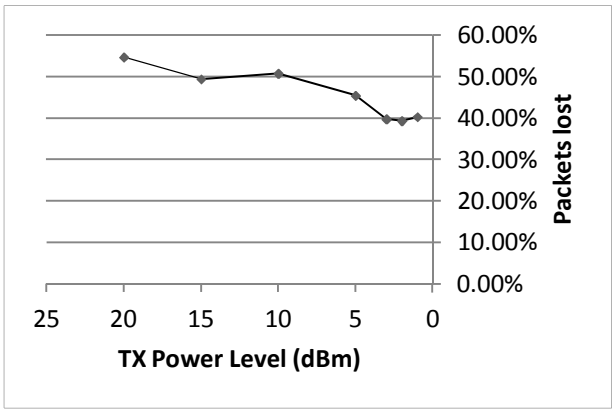


Fig. 3, 4: Total network goodput and packet loss at each power level.

Figure 2 shows the goodput for each device at each network power level.

- All devices performed best below the maximum power level.
- Six of the nine devices showed a positive trend in throughput until 2 dBm.
- Most devices stopped gaining throughput around 2 dBm.

Device F2 (a Droid X) experienced its maximum throughput at 15 dBm, significantly higher than the other devices. We believe this is an artifact of the test bed configuration, as the other Droid X device (E2) reached its maximum throughput at 3 dBm.

The total network throughput can be calculated by summing the throughputs of the individual devices.

$$\text{goodput}_{\text{total}} = \sum_{i} \text{goodput}_{\text{avg}}(i)$$

Figure 3 shows the total throughput of the entire network at each power level.

- Total network throughput increased as power level decreased, with a maximum at 2 dBm.

Since each packet contains a sequence number, it is possible to measure packet loss by counting the number of skipped sequence numbers (e.g. if packets 1,2,5,6 are logged, packets 3 and 4 must have been lost). This observation was used to create figure 4, which shows the number of packets lost at each power level, as a fraction of the total number of packets sent.

Each packet also contains the link data rate at which it was transmitted. There was some initial concern that lower transmission power might negatively impact data rate, thus decreasing total network goodput. However, as seen in Figure 5, the majority of devices selected 54 Mb/s for each test. The highest data rates occurred at 50% of the maximum transmission power, indicating that packet loss has a more significant impact on our clients' rate selection algorithms than SNR. Additionally, no device changed its data rate after the beginning of a test, allowing .

5 Discussion

In this paper, we set out to solve the throughput problems caused by highly congested networks such as those at technical conferences and expositions. We designed a small-scale test bed using a number of WiFi-

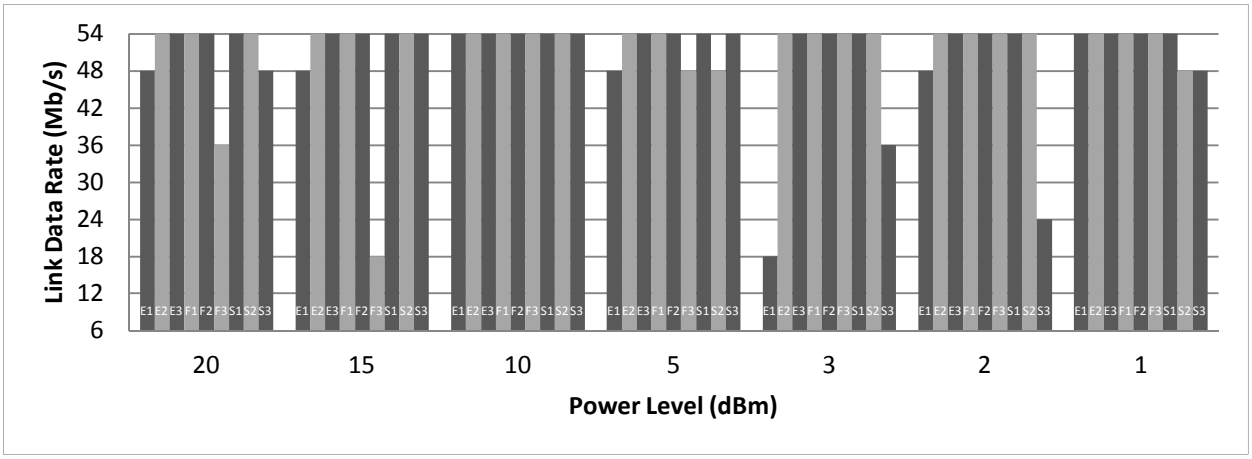


Fig. 5: Client data rates at each power level.

enabled smartphone devices arranged around several access points. Each device ran custom client software that configured the WiFi chipset's driver to transmit packets at a fixed, specified power level. The client software transmitted packets to a central data collection server that logged the data for later analysis. Upon analyzing the data, we observed a clear correlation between decreased transmission power and increased goodput at both the individual device and aggregate network level.

Given these results, it appears that our proposed low-power, many-AP topology can be a viable solution to the challenge of congested wireless in conference environments.

6 References

[1] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. "Self-management in chaotic wireless deployments." *Proceedings of the 11th annual international conference on Mobile computing and networking (MobiCom), Cologne, Sept. 2005.*

[2] V. P. Mhatre , K. Papagiannaki , and F. Baccelli. "Interference Mitigation through Power Control in High Density 802.11 WLANs." *Proceedings of IEEE Infocom, 2007.*

[3] I. Broustis, J. Eriksson, S. V. Krishnamurthy and M. Faloutsos. "Implications of Power Control in Wireless Networks: A Quantitative Study." *Proceedings of the Passive and Active Measurement Conference (PAM), 2007.*

[4] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. P. Mhartre. "MDG: Management-Driven Guidelines for 802.11 WLAN Design." *Proceedings of the international conference on Mobile computing and networking (MobiCom), Montréal, Sept. 2007.*

Device ID	Device Model	Wireless Chipset
E1	Samsung Galaxy Tab 10.1	Broadcom BCM4330
E2	Motorola Droid X	Texas Instruments WL1251
E3	Motorola Droid	Texas Instruments WL1251
F1	Motorola Droid	Texas Instruments WL1251
F2	Motorola Droid X	Texas Instruments WL1251
F3	Samsung Galaxy Tab 10.1	Broadcom BCM4330
S1	Motorola Droid	Texas Instruments WL1251
S2	Samsung Galaxy Tab 10.1	Broadcom BCM4330
S3	LG Revolution	Broadcom BCM4329

Table 1: Table of client devices.