Wi-Fi Goes to Town: Rapid Picocell Switching for Wireless Transit Networks

Zhenyu Song
Princeton University
zhenyus@cs.princeton.edu

Longfei Shangguan
Princeton University
longfeis@cs.princeton.edu

Kyle Jamieson*
Princeton University
kylej@cs.princeton.edu

ABSTRACT

This paper presents the design and implementation of Wi-Fi Goes to Town, the first Wi-Fi based roadside hotspot network designed to operate at vehicular speeds with meter-sized picocells. Wi-Fi Goes to Town APs make delivery decisions to the vehicular clients they serve at millisecond-level granularities, exploiting path diversity in roadside networks. In order to accomplish this, we introduce new buffer management algorithms that allow participating APs to manage each others' queues, rapidly quenching each others' transmissions and flushing each others' queues. We furthermore integrate our fine-grained AP selection and queue management into 802.11's frame aggregation and block acknowledgement functions, making the system effective at modern 802.11 bit rates that need frame aggregation to maintain high spectral efficiency. We have implemented our system in an eight-AP network alongside a nearby road, and evaluate its performance with mobile clients moving at up to 35 mph. Depending on the clients' speed, Wi-Fi Goes to Town achieves a 2.4–4.7× TCP throughput improvement over a baseline fast handover protocol that captures the state of the art in Wi-Fi roaming, including the recent IEEE 802.11k and 802.11r standards.

CCS CONCEPTS

• **Networks → Wireless access networks**;

KEYWORDS

Wi-Fi, Handover, Transit Networks

ACM Reference format:
Zhenyu Song, Longfei Shangguan, and Kyle Jamieson. 2017. Wi-Fi Goes to Town: Rapid Picocell Switching for Wireless Transit Networks. In *Proceedings of , Los Angeles, CA, USA, August 21-25, 2017 (SIGCOMM '17),* 13 pages.
https://doi.org/10.1145/3098822.3098846

1 INTRODUCTION

Every day, billions of commuters journey in and out of the world's urban centers: many by train, light rail, or underground transport, others in vehicles that may soon become driverless in the coming decade. Today this commute is often wasted time, but we look forward to a nearby future with great demand for high-capacity wireless networks serving transportation corridors, allowing users to surf the web, complete video and audio calls or teleconferences, and stream video and music entertainment, all on the commute.

Surprisingly, compared with capacity gains from new technologies, Cooper [10] estimates—with broad consensus from cellular wireless network operators—that the overwhelming majority of capacity gains over the past 45 years can be attributed to simply decreasing the size of each *cell*, the geographic area each access point (AP) covers. However, moving to smaller cells immediately sets up a tension between the two goals of capacity and range, exacerbating the range problem mentioned above. This necessitates a *handover*, where a number of APs cooperate to serve the client as it moves from one AP to the next.

Roaming between Wi-Fi APs is standardized, with the recent 802.11r standard [22] enabling the client to make a connection with another AP before abandoning the current: this is called *make before break*. Other parts of the Wi-Fi standard, namely 802.11k [21], allow the current AP to inform the client about other nearby APs and channels, so that when the time comes for the client to abandon the current AP, it may immediately begin the association process with those APs, rather than discovering their existence from scratch. But these solutions, and others we discuss below, are too slow for the scenarios we consider here, necessitating larger cells and thus clawing back smaller cells' capacity gains.

Recently, however, two trends have arisen that may break this stalemate: the first a reflection on recent research, the second a consequence of Wi-Fi's commoditization:

(1) Recent work has demonstrated that commodity Wi-Fi APs can extract detailed channel measurements [18, 49], thus rapidly predicting the AP or APs best suited to serve a client as it moves.
(2) Today, very low cost Wi-Fi chipsets such as the ESP8266 (Figure 1) are arriving on the market for less than $5, making for the first time very high-density Wi-Fi AP installations cost-feasible.

Figure 1: The ESP8266 Wi-Fi and system-on-chip module, available *ca.* 2016 for $5.

In this work, we demonstrate a synergy between the above two trends, resulting in an opportunity to transform the design of roadside and metropolitan transit wireless networks with the widespread deployment of an array of inexpensive "Wi-Fi picocell" APs,

*K. Jamieson is also affiliated with University College London.

SIGCOMM '17, August 21-25, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-4653-5/17/08…$15.00
https://doi.org/10.1145/3098822.3098846

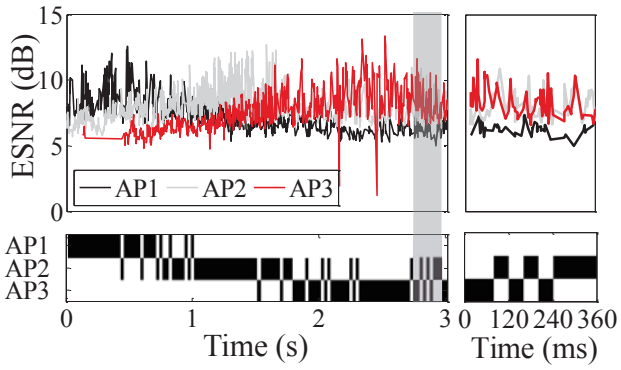


Figure 2: The vehicular picocell regime: Constructive and destructive wireless multipath fading as measured by Effective SNR (*upper plot*) conspire with vehicular-speed mobility to change the AP best able to deliver packets to a mobile client (*lower plot*) at millisecond timescales (*right detail view*). The radio coverage between APs overlaps by around 10 m.

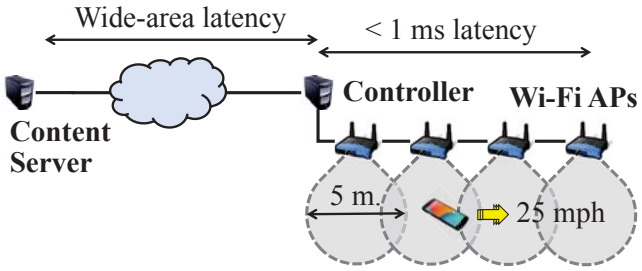


Figure 3: Wi-Fi Goes to Town architecture: cars moving at 25 mph spend about 460 milliseconds in each AP's cell.

each covering just a few meters of a busy urban transit corridor. But such a design faces the fundamental challenge shown in Figure 2: the wireless capacity (achievable throughput) to each AP exhibits two well-known wireless trends as a mobile user drives past the AP array. First, there is fading at second-level scales due to distance, but second, there is rapid, millisecond-level fast fading due to alternating constructive and destructive multipath propagation, on the spatial scale of an RF wavelength, 12 cm at 2.4 GHz. Furthermore, due to the dense deployment of APs in the array (necessary to attain good coverage), the large-scale and small-scale fades overlap in time to make the best choice of AP (shown below in the figure) change every millisecond. We term this combination of AP diversity, vehicular client mobility, and meter-level AP cell size the *vehicular picocell regime*.

This paper presents the design, implementation, and experimental evaluation of *Wi-Fi Goes to Town*, a system that operates efficiently in the vehicular picocell regime via a nearby controller exercising tight control of the AP array, as shown in Figure 3. As users travel at high speed through an underground tunnel or along a road, the controller switches downlink traffic to several APs in the array, one of which will be best able to deliver the packet to where the user's device will be tens of milliseconds later. When the packet reaches the tail of an AP's transmit queue and is ready for transmission to the client, the APs communicate directly to agree which one of them should actually attempt to deliver the packet. Since the wireless channel is so unpredictable in the vehicular picocell regime, with a coherence time (the period of time over which the wireless channel remains stable) of *ca.* two-three milliseconds at 2.4 GHz [47], Wi-Fi Goes to Town APs rely on lightweight channel state information (CSI) readings from a client's uplink transmissions, from which they compute Effective SNR [18] (ESNR), and quickly decide which AP should deliver the packet to the mobile. If the chosen AP's delivery attempt is successful, that AP cues the other APs to dequeue and discard their copies of the delivered packet.

For traffic on the uplink from the mobile to the AP array, any AP can receive each such packet, but the controller removes duplicate packets in the event that more than one AP receives the same packet from the client.

The first of two challenges involved in realizing actual wireless performance gains in this setting stems from packet queuing, which occurs in many network layers and software and hardware components in Wi-Fi networks. A modest amount of queuing (*ca.* 20 milliseconds or 100 packets) in the operating system driver and/or the Wi-Fi hardware itself allows modern 802.11 wireless link layers to combine queued packets into larger *aggregate* packets. This *aggregation* is critical to maintaining a high ratio of fixed per-frame overhead (necessary to coordinate access to the wireless medium) to useful data airtime as wireless data bit-rates increase. But frame aggregation and sequence number-based block acknowledgements lose their efficiency if the AP that receives a block acknowledgement from a client differs from the one that sends the corresponding downlink data. To address this problem, Wi-Fi Goes to Town introduces mechanisms to share block acknowledgement state between participating APs.

The second of the two challenges comes from packet buffering. Buffering at higher layers, such as the operating system's transport-layer socket, allows applications to exploit asynchronous I/O. However, while helpful in these regards, buffering at an individual AP results in the transmission of a backlog of packets once the controller has made a decision to switch away from that AP. As we show in §3 (p. 5), buffering thus adds a significant latency to the controller's switching decision, a delay that is insignificant in current AP handover systems and systems based on Multipath TCP [15], but highly significant for the vehicular picocell regime. To address the problem of controller-AP switching latency, Wi-Fi Goes to Town maintains current amounts of buffering at APs, but introduces hooks for the controller to dequeue pending packets buffered at APs once it makes a switching decision. Microbenchmarks (§3) quantify the performance impact of buffering and the benefits of our fine-grained AP buffering control.

We have implemented Wi-Fi Goes to Town on commodity TP-Link APs, with a single Linux controller running the Click modular router [25]. Each roadside AP is equipped with a parabolic antenna of beamwidth 21 degrees. We install the Atheros CSI Tool [49] on each AP, which measures the CSI of each incoming frame and forwards it to the controller for processing. Our implementation of Wi-Fi Goes to Town is fully described in §4 below. For a comparison

point running on the same hardware, we have also implemented a performance-tuned version of the IEEE 802.11r and 802.11k fast handover protocols in Click.

We have deployed eight Wi-Fi Goes to Town APs on the third floor of an office building overlooking a nearby road. APs communicate with the controller via Ethernet backhaul. Experiments compare Wi-Fi Goes to Town with a performance-tuned version of the IEEE 802.11r and 802.11k fast handover protocol that captures the current state of the art in Wi-Fi handover techniques. Our end-to-end experiments test three different traffic workloads representative of commuting behavior: web surfing, video streaming, and bi-directional video teleconferencing. Results show a 2.4–4.7× improvement in TCP download performance and a 2.6–4.0× improvement in UDP download performance as vehicular speed ranges from five to 25 mph, and 2.4–2.6× improvement in TCP and UDP download performance in a multi-client scenario. Microbenchmarks then investigate the effect of AP density and time hysteresis for AP switching, presenting a sensitivity analysis of each of the above parameters on overall system performance.

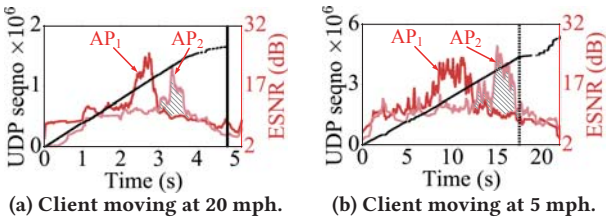
Contributions. Wi-Fi Goes to Town's contributions are two-fold. First, we design a fine-grained, highly responsive switching protocol and AP selection algorithm to send packets over the best link at millisecond-level timescales. Second, we integrate this rapid packet switching mechanism with modern Wi-Fi's frame aggregation and block acknowledgement mechanisms, critical to achieving high performance in modern 802.11 wireless networks.

Roadmap. The remainder of this paper first highlights the inability of current state-of-the-art to operate in the regime of interest (§2), then describes Wi-Fi Goes to Town's design (§3). Our system's implementation (§4) and experimental evaluation in a real roadside network testbed (§5) follow. §6 surveys related work in the area. §7 discusses future work, and §8 concludes.

2 WI-FI ROAMING AT DRIVING SPEED

Commercial Wi-Fi APs supporting the 802.11r [22] standard speed client handovers from one AP to the next by allowing the client to establish authentication with a new AP prior to de-associating with the current AP. But 802.11r is tuned for walking speed mobility with large cells. To understand its performance in the vehicular picocell regime, we evaluate Linksys 802.11r-based APs [12] in an outdoor roadside testbed (fully described in §4). The distance between two APs is 7.5 m, and the cell size of each AP is 5.2 m. We send a constant-rate stream of UDP packets using iperf3 [23] to an iPhone 6S client in a car driving by first at five mph, and then at 20 mph.

Figures 4 (a) and 4 (b) plot the client's reception of UDP packets when it drives by AP_1 and AP_2 at 20 mph and 5 mph, respectively. As a reference, we also plot the smoothed effective SNR (ESNR) of these two client-AP links. In the 20 mph test, we identify a re-association packet sent from the client to AP_1 at around 4.6 s, indicating that the client tries to switch to AP_2. However, no acks are identified at the client side even though the client retransmits the re-association packet multiple times, and so the handover fails. As the client moves further away from AP_1, link quality deteriorates significantly, with the client receiving the last packet from AP_1 at approximately 4.8 s. The handover fails because 802.11r does not



(a) Client moving at 20 mph. (b) Client moving at 5 mph.

Figure 4: Received sequence number (black curves) of a constant-rate UDP flow at a mobile client in a roadside network testbed at (a) 20 mph speed, and (b) 5 mph speed, respectively. The solid vertical line in (a) denotes the time of the last UDP packet reception, and the dashed vertical line in (b) denotes the time the client switches to AP_2. The dashed area shows the accumulated channel capacity loss. The average capacity loss is separately 20.5 Mbit/s at 20 mph and 82.2 Mbit/s at 5 mph.

make its switching decision until it collects a long 5 s history of RSSI measurements [1], longer than the client spends within hearing range of AP_1 at 20 mph. As a result, by the time the client has determined it should switch, the link condition of the current AP has already deteriorated. The client successfully switches from AP_1 to AP_2 only in the 5 mph case, as shown in Figure 4 (b), but as can be seen from the ESNR curves, the handover happens significantly later than it should, resulting in the AP needing to decrease its bitrate, sacrificing wireless capacity.

3 DESIGN

Figure 5 shows the high-level design of Wi-Fi Goes to Town (WGTT). Our AP selection and downlink queue management algorithms (§3.1) work hand-in-hand to leverage wireless path diversity at millisecond-level timescales to speed the delivery of downlink traffic as the client transitions through the grey zones of multiple APs simultaneously. Our block acknowledgement forwarding and packet de-duplication mechanisms (§3.2) work together with uplink data delivery to the same path diversity at the same fine-grained timescales to make 802.11's frame aggregation block acknowledgements more reliable, also increasing throughput.

3.1 Downlink Packet Flow

For every downlink packet to a certain client, WGTT chooses the AP best able to deliver the packet to the client (§3.1.1), and then carefully manages packet queues at all APs (§3.1.2) to ensure the selected AP delivers the selected packet to the client milliseconds later.

3.1.1 AP Selection. To operate in the vehicular picocell regime, our AP selection algorithm needs to be simultaneously accurate in its packet delivery rate predictions and agile enough to react in milliseconds. To achieve these goals, we measure link quality with the *Effective SNR* [17] (ESNR) metric, computed at each AP from channel state information (CSI) extracted from a client's uplink transmission. Each AP measures CSI on all 56 OFDM subcarriers, encapsulating these readings into a UDP packet, and delivering

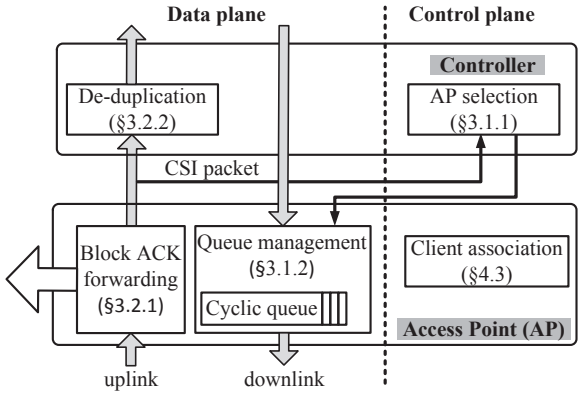


Figure 5: WGTT's high-level design, divided into control plane- and data plane-functionality.

this packet to the controller over the Ethernet backhaul. Upon receiving this packet, the controller extracts the CSI information and computes ESNR. ESNR takes SNR variations across subcarriers into account, hence more accurately predicts packet delivery probability in the presence of a highly frequency-selective channel (*i.e.* when multipath reflections are strong).

WGTT AP selection algorithm. WGTT measures the short-term history of ESNR readings from packets received over each client-AP link in a sliding window of duration W (we evaluate our choice of W in §5). For a certain client and AP a, denote the (sorted, monotonically non-decreasing) sequence of L_a ESNR readings in this window by

$$E(a) = \left[e_1(a), \ldots e_{L_a}(a)\right].$$

After sorting, we select the AP a^* with the maximal median ESNR reading in the $E(a)$ window:

$$a^* = \arg\max_a \left\{e_{\lfloor L_a/2 \rfloor}(a)\right\}.$$

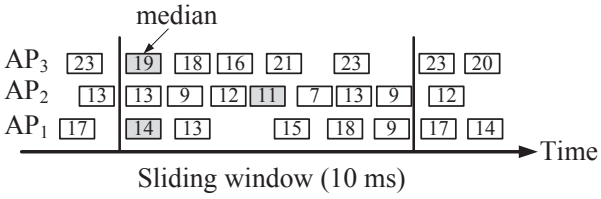


Figure 6: WGTT AP SELECTION: Choosing between three nearby APs, Wi-Fi Goes to Town examines the median ESNR reading from each, selecting in this case AP_3 with the greatest median SNR.

3.1.2 AP Queue Management. To rapidly switch between APs, the WGTT controller forwards each downlink packet to all APs within communication range of the client,[1] while allowing just one at a time (as determined by the AP selection algorithm) to transmit packets to the client. Each other AP buffers downlink packets in

[1]Those APs that have received a packet from the client within the AP selection window W.

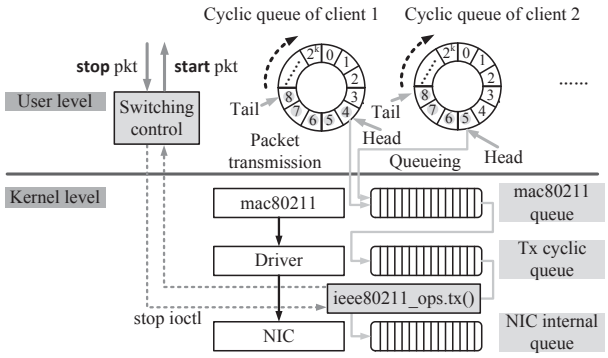


Figure 7: Packet queueing in the WGTT AP.

the *cyclic queue* shown in Figure 7, which also summarizes all other locations in the WGTT AP where packets are buffered. Both packet switching and queue management require an index number to identify each data packet. In WGTT, we define an m-bit *index number* for each data packet, which increments by one for each packet destined to a certain client. We set $m = 12$ to guarantee the uniqueness of each index number in each client's cyclic buffer.

When the controller switches from one AP to the next (*e.g.*, AP_1 to AP_2), there are roughly 1,600 (at 50 Mbit/s UDP offered load) to 2,000 packets (at 90 Mbit/s UDP offered load) backlogged in AP_1's queues, at various layers of the networking stack as shown in the figure 7. Unless dequeued, AP_1 will attempt to deliver these backlogged packets to the client, likely failing, thus sacrificing channel capacity and disrupting any ongoing TCP flows to that client.

WGTT's switching protocol. When the controller determines that the client should be switched from AP_1 to AP_2, it instructs AP_1 to tell AP_2 which packets are backlogged in its queues. Since these backlogged packets are already buffered in AP_2's cyclic queues even before the switch, AP_2 can then deliver them to the client almost immediately. This switching protocol consists of the following three steps:

(1) The controller sends a stop(c) control packet to AP_1, instructing it to cease sending to the client c. The stop packet contains the layer-2 addresses of c and AP_2.
(2) After receiving stop(c), AP_1 ceases sending to c, and sends to AP_2 a start control packet containing c and the index k of the first unsent packet destined to c: start(c, k).
(3) After receiving start(c, k), AP_2 sends an ack control packet back to the controller, and begins transmitting packets from its cyclic queue at index k to the client.[2]

After the switch, AP_2 continues delivering new downlink packets received from the controller to c. In the absence of control packet loss, the switch will be accomplished after these three steps. However, both the control packet and the ack packet may be lost, and so we set a timeout for the control packet retransmission. If the controller does not receive the ack within 30 ms, it retransmits the

[2]The controller will not issue another switch until the current issued switch is acknowledged.

stop packet. On the other hand, since the control and ack packets manage the downlink packet switching, these packets should be processed promptly for switching delay minimization. In the WGTT AP, incoming control packets are prioritized, bypassing the cyclic queue. Control packets are thus given higher priority, so that they are always been processed ahead of data packets and thereby minimizing switching delay.

Table 1: Measuring the running time of the switching protocol in different data rate settings.

Data rate (Mb/s)	50	60	70	80	90
Mean execution time (ms)	17	19	21	19	17
Standard deviation (ms)	3	5	4	5	3

Running the switching protocol takes over 17 ms on average (as shown in Table 1), During the time that the switch is happening, we allow AP_1 to send the backlogged packets buffered in its hardware NIC queue. These packets take 6 ms to deliver, thus although they are sent over AP_1's inferior link, the capacity loss is minimal.[3]

Implementing the switch. We modify the ieee80211_ops_tx() function in the Linux kernel to keep track of the index of the last packet destined to each client just before it enters the NIC's hardware queue. When AP_1 receives a stop(c) packet from the controller, it queries the index number for c in the kernel through an ioctl system call: this is the first unsent packet destined to c (index number k). The ieee80211_ops_tx() function replies and then monitors the backlogged packets that flow out of the driver's transmit cyclic queue and filters out packets destined to c. Upon receiving the packet index from the kernel, AP_1 sends start(c, k) to AP_2.

3.1.3 Packet addressing and tunneling. In the controller, both the layer-2 and layer-3 headers of the downlink packet have the destination set to the addresses of the clients. We cannot change them to the AP's addresses, otherwise the AP cannot decide which client the packet should be delivered to, so we tunnel downlink packets in an IP packet with the AP's IP address in the destination field.

3.2 Uplink packet flow

On the uplink, WGTT introduces a new technique, *block acknowledgement forwarding*, that integrates with frame aggregation to make block acknowledgements more reliable, reducing retransmissions on the downlink.

3.2.1 Block acknowledgement forwarding. The Wi-Fi block acknowledgement mechanism (first introduced in the 802.11e standard) improves channel efficiency by aggregating multiple packet acknowledgements into one frame. The block acknowledgement (Block ACK) contains a bitmap to selectively acknowledge individual frames in a window of packets. When the client moves at vehicular speed, its Block ACK is prone to loss due to the constructive and destructive wireless multipath fading, especially near the edges of an individual AP's coverage. In this case, the AP retransmits all packets that should be acknowledged in the lost Block ACK, hurting throughput and channel utilization. In WGTT, we exploit path diversity, designing a link-layer protocol to allow APs not currently talking to the client to forward an overheard Block ACK to the client's current AP over the Ethernet backhaul.

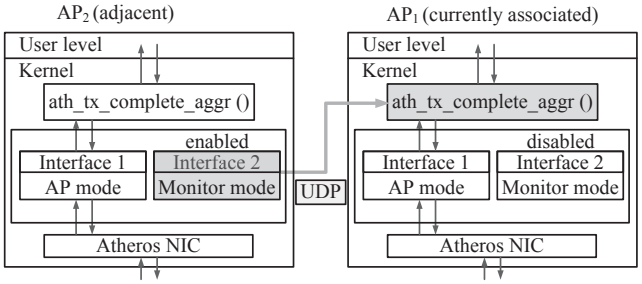


Figure 8: WGTT's Block ACK forwarding design.

Specifically, we create two virtual NIC interfaces for each AP, with one working in *AP mode* to handle normal uplink/downlink packet flows, and another working in *monitor mode* to overhear packets and captures block ACKs. The monitor mode interface is disabled in the AP that the client currently associated with. As shown in Figure 8, upon receiving a block ACK, AP_2 extracts the layer-2 source address (client's address), the sequence number of the first packet that should be acknowledged in this Block ACK, and the Block ACK bitmap, encapsulating them into a UDP packet, and forwarding this UDP packet to AP_1. Upon receiving the information, AP_1 first checks whether this Block ACK has been received before (from its own NIC or from other APs). If so, AP_1 drops the forwarded block ACK. Otherwise, it updates the ath_tx_status data structure using the received information, and inputs this data structure to the function ath_tx_complete_aggr()[4], where the newly updated block ACK bitmap is examined. The result is that the effective block ACK loss rate will decrease.

3.2.2 Packet addressing and tunneling. Uplink packets sent from a client are received by one or more APs, which encapsulate the packet in an UDP/IP and 802.3 header, putting the source layer-2 and layer-3 address as the received AP, and the destination host as the controller. Consequently, the controller can record from which AP the received packet is sent. The controller then strips the tunneling header of the packet and de-duplicate packets by checking the source IP address and the IP sequence number of incoming packets. To speed up the de-duplication process, we use a hashset and compose a 48-bit key unique to a specific packet using the source IP address and the IP identification field of this packet.[5]

3.2.3 Packet de-duplication. As all APs in the network are successfully associated with the client, they all forward uplink packets heard from the client to the controller, resulting in packet duplication, which can lead to spurious TCP retransmissions, harming throughput. Hence the controller needs to de-duplicate uplink packets before forwarding them to the Internet.

[3]We intend to further optimize switching time with kernel-level Click cyclic queue implementations in future work.

[4]While all ath functions are specific to the Atheros driver, similar functions can be found in other drivers due to the generic interface of Linux OS.

[5]Not all packets have IP header: for those without an IP header, we only consider ARP packets, where we don't need de-duplication.

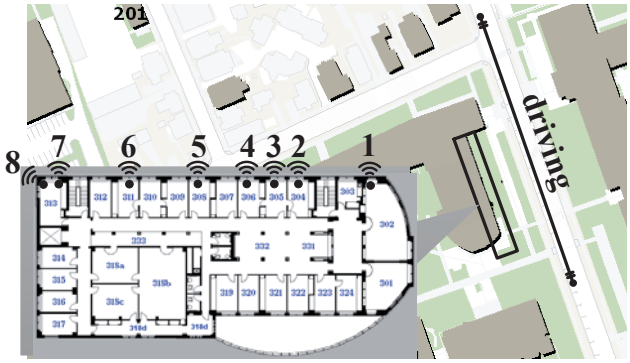


Figure 9: Experiment setup: we deploy eight WGTT APs on the third floor of an office building overlooking a side road with speed limit 25 mph. The radio coverage overlaps between adjacent APs.

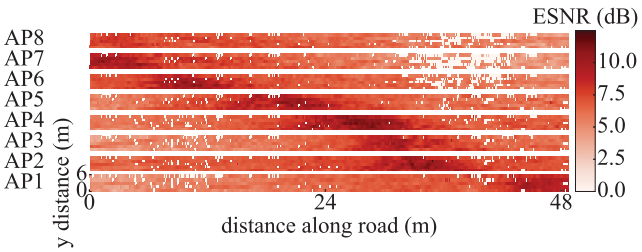


Figure 10: Effective SNR heatmap measured at each AP. In each heatmap the x-axis refers to distance along the road, y-axis refers to distance across the road. The AP radio coverage overlaps by between 6 m and 10 m.

4 IMPLEMENTATION

We implement the WGTT AP and controller logic on commodity routers and laptops using the Click modular router [9], and deploy a testbed on the third floor of an office building overlooking a side road with speed limit 25 mph (as shown in Figure 9). The testbed is composed of eight WGTT APs interconnected with each other through the Ethernet backhaul. The controller connects to the routers through Ethernet backhaul as well. The router works on the channel 11 of the 2.4 GHz frequency band, without modification of the default rate control algorithm. WGTT's small cell limits the delay spread to a value similar to an indoor environment. So the length of the standard Wi-Fi cyclic prefix is sufficient. Figure 10 shows the ESNR heatmap of the road we measured at each AP. We can see the ESNR distribution is coherent with the location distribution of eight APs deployed along the roadside.

4.1 Controller

Hardware. The WGTT controller is a Lenovo Thinkpad T430 laptop[45], equipped with a Intel Core i5-3320M CPU, 8 GB DDR3 RAM, and a 160 GB Solid State Drive (SSD). We install two USB ethernet adaptors on it, one for LAN packet processing and another for the WAN.

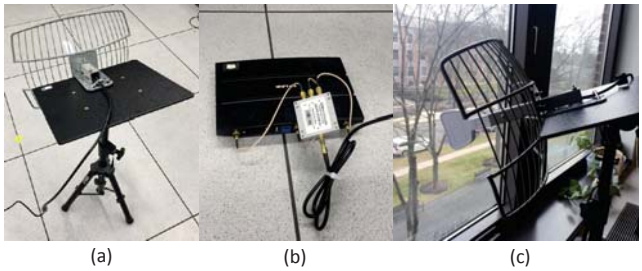


Figure 11: The WGTT AP is composed of (a) a directional antenna that connected to three ports of TP-Link AP via a splitter (b). (c): the AP is deployed in front of a window.

Implementation. The controller runs Ubuntu Linux v14.04 LTS. We write click elements for our control logic and install rules blocking the Linux kernel from receiving any packets received from the NIC, so Click is the only application with access to the NIC.

4.2 Access Point

Hardware. We build the WGTT AP using a TP-Link N750 AP [46] equipped with an Atheros AR9344 NIC, which measures the CSI of each incoming frame and forwards it to the controller for processing. We detach the default omnidirectional antennas of this router and connect it to a 14 dBi, 21-degree beamwidth *Laird* directional antenna [26] using a *Mini-Circuits* ZN3PD-622W-S+ RF splitter-combiner (as shown in Figure 11).[6] Notice that since the WGTT software design is hardware-agnostic, it is possible to replace the directional antenna with small-cell omni-directional antenna.

Implementation. The TP-Link router runs openwrt Chaos Calmer v15.05.1 [35]. We write click elements for AP control logic and a click configuration ap.click on it to (i) manage the packet queue and (ii) encapsulate uplink packets and forward them to the controller. The Atheros NIC on the TP-Link router computes the CSI of each uplink packet (using the CSI tool [18]), encapsulating the CSI and client information into a UDP packet, and delivering this packet to the controller through the Ethernet.

4.3 Client Association

Like other wireless local area network designs that utilize "thin APs" coupled with a centralized controller, WGTT APs all share the same 802.11 *basic service set identifier* (BSSID), and so appear as one AP to the client. When a client associates with the first AP (*e.g.* AP_1), WGTT synchronizes the association with all APs in the network. To achieve this goal, we modify hostapd in the user space of the Linux wireless system, letting AP_1 send the client information (layer-2 address, authorization state *etc.*) to other APs through the Ethernet backhaul. Specifically, at the end of the client association with AP_1, the hostapd of this AP will receive an *association callback*, signaling that hostapd's association confirmation to the client has been received. AP_1 then moves the client information sta_info struct to a

[6] As all cables to the splitter-combiner are short and of equal length, this results in one spatial stream to the client. We leave the design and experimentation of a multiple spatial stream roadside AP for future work.

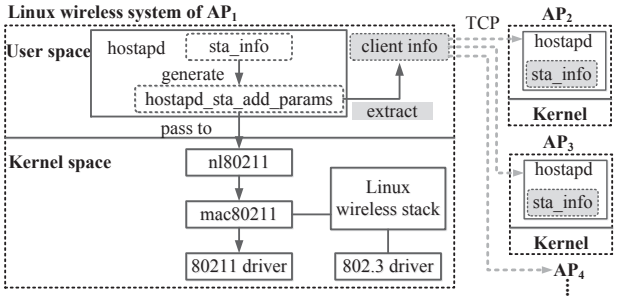


Figure 12: WGTT client association. Each AP shares client association state with the others over the Ethernet backhaul.

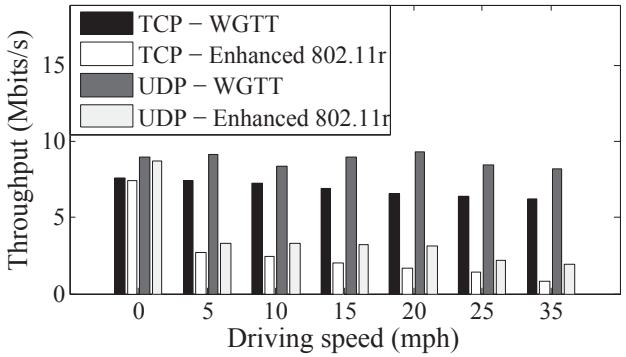


Figure 13: TCP and UDP throughput when the client moves at different speeds.

new hostapd_sta_add_params struct, passing it to the kernel level mac80211 and the driver. We add code to extract the client information within hostapd_sta_add_params struct of AP_1, open a TCP connection to all other APs in the network, and transmit the client information sta_info to those APs. On the other end, the receiving AP is listening for this TCP connection. When the TCP connection is set up, the information in the received packet is transferred back into a hostapd_sta_add_params struct and passed into mac80211 and the driver on the received AP. Figure 12 illustrates this process.

5 EVALUATION

In this section, we first conduct field studies to evaluate the end-to-end performance of WGTT and compare it with a performance-tuned version of the 802.11r fast roaming protocol [22]. We then present micro-benchmark experiments to provide further insight into which factors impact WGTT's performance. After that, we conduct three real-world case studies to show WGTT's capability of handling online video streaming, remote video conferencing, and web browsing at driving speed.

5.1 Methodology

Three Lenovo L512 laptops with Atheros AR9590 wireless card serve as clients in our experiments. The client transits through eight deployed APs at different driving speed, ranging from 5 mph to 35 mph. For each experiment, we log packet flows sent to and from both the controller and the client using tcpdump for data analysis.

Comparison scheme. We implement a performance-tuned version of the IEEE 802.11r fast roaming protocol and deploy it on our testbed for comparison. In most 802.11r implementations, the client does not switch to another AP until it collects a number of RSSI readings from the AP it currently associated with, but as we showed above (§2) this fails in the vehicular picocell regime. We therefore enhance a combination of the standard 802.11r and 802.11k [21] protocols and our best understanding of centralized controller WLAN products in the straightforward way we expect the industry to proceed:

(1) Each AP beacons every 100 ms, from which the client discovers their presence and estimates RSSI.
(2) We set an RSSI threshold below which a client switches to another AP with the highest RSSI value once the RSSI of the current AP is lower than this threshold, with a time hysteresis of one second.
(3) After the first client association, other APs learn the authentication and association information of that client (as current products based on a centralized controller implement, to the best of our knowledge) thus can forward to other APs any authentication or association frame from the client in the uplink direction.

We term this scheme *Enhanced 802.11r*, using it as a performance benchmark in the remainder of our evaluation.

5.2 End-to-end Performance

We first evaluate the end-to-end performance of WGTT delivering bulk TCP and UDP data flows.

5.2.1 Single-client experiments. We examine WGTT's throughput at different client moving speeds, and test TCP and UDP throughput of WGTT and the Enhanced 802.11r fast roaming protocol. As shown in Figure 13, WGTT achieves a slightly higher throughput than its counterpart in the static case. As the client moves, WGTT achieves a constantly high throughput at both low (5 mph) and high (35 mph) moving speeds, with an average throughput of 6.6 Mbits/s for TCP and 8.7 Mbits/s for UDP. In contrast, Enhanced 802.11r achieves only 2.7 Mbits/s and 3.3 Mbits/s throughput for TCP and UDP at 5 mph driving speed. When the client at 35 mph, the TCP and UDP throughput of Enhanced 802.11r further drops to 0.8 Mbits/s, and 1.9 Mbits/s, respectively.

To better understand the sources of WGTT's throughput gain over Enhanced 802.11r, we plot the TCP throughput against time and a timeseries showing which AP the client is associated with during its movement in Figure 14. As shown, WGTT keeps switching from one AP to another at high frequency (around five times per second), providing the client with the best link at each period of time. Benefiting from fast link switching, WGTT's throughput maintains at a relatively stable level (around 5 Mbits/s) throughout the client's transition over eight APs. In contrast, the TCP throughput of Enhanced 802.11r increases as the client moves to the associated AP, and then drops to zero at about 2.5 s in the experiment as the client moves out of the AP's radio range. This is because Enhanced 802.11r fails to switch promptly as the client moves near the edge of the current AP's coverage, where the current link deteriorates

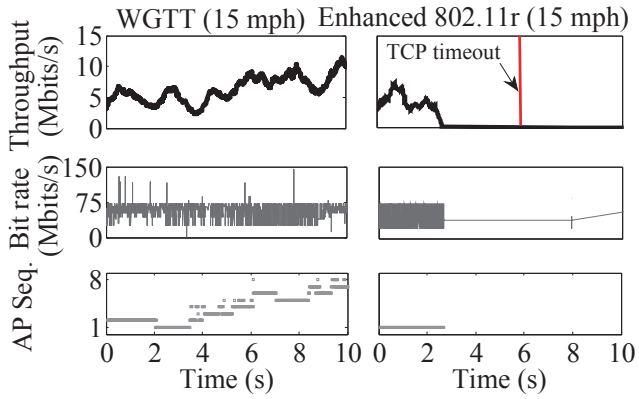


Figure 14: TCP throughput of WGTT and Enhanced 802.11r during a single client's 15 mph drive.

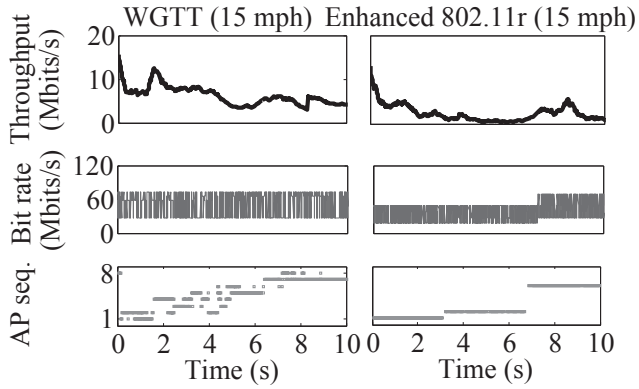


Figure 15: UDP throughput of WGTT and Enhanced 802.11r during a single client's 15 mph drive.

significantly. TCP timeout occurs at around 5.86 s (the vertical line in the figure), causing the TCP connection to break thereafter. We observe a result with UDP transmission (Figure 15), where WGTT switches between multiple client-AP links frequently to enjoy the better link quality, thereby keeping a relatively stable transmission rate throughout the client's movement. Enhanced 802.11r switches only three times during the entire transition period (10 s), achieving a low and unstable throughput.

WGTT's link bit rate. We next examine the link bit rate of WGTT during the client's movement. In this experiment, the client transits through eight APs at a constant speed (15 mph) and sends TCP and UDP packets to the AP during its movement. Figure 16 shows the CDF of the link bit rate measurements. We find that WGTT achieves a 90% quantile of around 70 Mbits/s, which is 30 Mbits/s higher than Enhanced 802.11r.

Accuracy of the AP switching algorithm. We define *switching accuracy* of a handover algorithm as the fraction of the time that the algorithm chooses the *optimal* AP to deliver the packet, where the optimal AP is the AP with maximum ESNR to the client at any instant in time. In this experiment, we send both TCP and

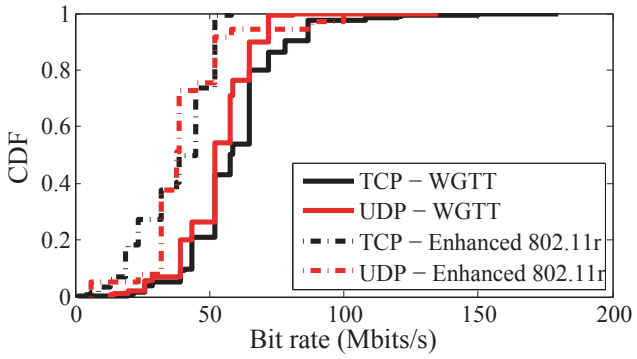


Figure 16: CDF of the link bit rate of TCP and UDP transmission. The client moves at 15 mph.

UDP packets at the maximum rate to a vehicle moving at 15 mph that transits across eight APs, and test the switching accuracy of WGTT and Enhanced 802.11r. Table 2 shows the result. As shown, WGTT achieves over 90% switching accuracy for both TCP and UDP transmissions. In contrast, Enhanced 802.11r's switching accuracy is only 20.24% for TCP transmission, it then drops further to 18.72% for UDP transmission. The reason behind this result is that the optimal link changes from one AP to another rapidly in the vehicular-picocell regime due to fast fading wireless channels, while Enhanced 802.11r chooses to switch only when the current link deteriorates significantly.

Table 2: Switching accuracy of WGTT and Enhanced 802.11r for TCP and UDP flows from a single client moving at 15 mph.

	WGTT (%)	Enhanced 802.11r (%)
TCP	90.12	20.24
UDP	91.38	18.72

Combined with the previous result, this result demonstrates that contrary to many other wireless networking designs, better packet switching decisions, instead of physical-layer bit rate adaptation, are responsible for most of WGTT's gain.

5.2.2 Multiple-client experiments. Here we test the ability of WGTT to improve the performance of multiple clients moving on the road simultaneously. We vary the number of clients from one to three, measuring per-client TCP and UDP throughput of WGTT and the Enhanced 802.11r fast roaming protocol. As shown in Figure 17, WGTT achieves an average per-client 5.3 Mbits/s TCP throughput and 8.2 Mbits/s UDP throughput in the single client case, which is 2.5× and 2.1× of the TCP and UDP throughput achieved by Enhanced 802.11r. As we gradually increase the number of clients on road, the throughput gap between WGTT and Enhanced 802.11r increases to 2.6× and 2.4 × for TCP and UDP transmissions, respectively. The reason behind is that multiple vehicles (clients) moving around will introduce dynamic multi-path, and so a higher packet loss rate. Accordingly, the throughput of Enhanced 802.11r drops significantly. In contrast, WGTT benefits from the uplink diversity:

each AP overheard the uplink packet will forward it to the server, resulting in a reduced amount of retransmissions and so a higher throughput.

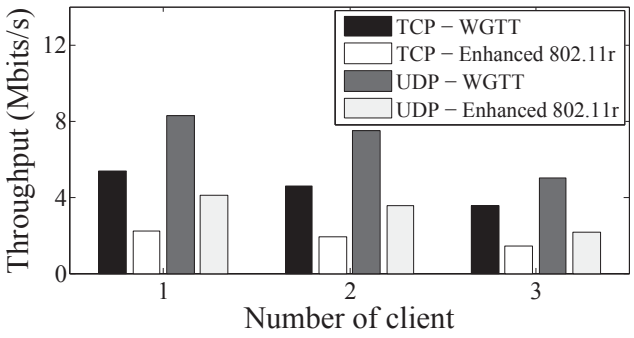


Figure 17: Average per-client downlink throughput with different numbers of clients. All clients move at 15 mph.

To demonstrate this, we draw the packet loss rate of three clients' uplink UDP flow during the client's transition among eight APs. We also plot the packet loss rate when the client transmits uplink packets through only one AP (Enhanced 802.11r) during its transition. As shown in Figure 18, with a single up link, the packet loss rate changes abruptly for all these three clients. In contrast, with multiple up links, the packet loss rate maintains in a very low level (below 0.02) for these three clients.

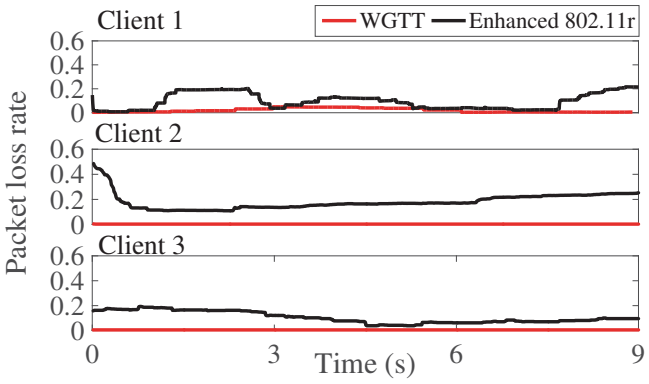


Figure 18: Packet loss rate of uplink UDP packets sent by three mobile clients.

We further test the TCP throughput in three multiple-client scenarios: following driving, parallel driving, and driving in opposing direction, as shown in Figure 19. In each testing, the server sends UDP packets at a constant rate (15 Mb/s) to the clients moving at 15 mph.

Figure 20 shows the TCP and UDP throughput in these three testing cases. As shown, we achieve the highest TCP and UDP throughput in case (c): two cars driving in opposite direction. This is due to the fact that the two clients stay far way from each other

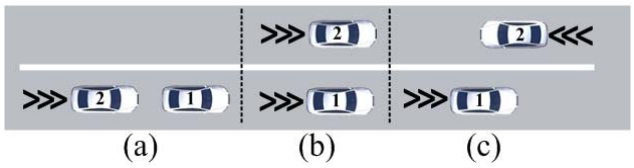


Figure 19: TCP/UDP throughput measurement testing in multiple client scenario, including (a) following driving with a spacing of 3 m; (b) parallel driving, and (c) driving in opposing direction. The clients move at 15 mph in each experiment.

during their transition period in this case. Hence each client experiences minimum link contention. We observe that both the lowest TCP (4.8 Mbits/s) throughput and UDP throughput (5.0 Mbits/s) appear in parallel driving case (case (b)). This is because the two clients could carrier sense each other, thus experiencing a higher link contention. Nevertheless, WGTT achieves consistently better performance than Enhanced 802.11r fast roaming protocol in all three multi-client testing cases. This is because WGTT leverages the link diversity to let all APs forward their overheard packet to the remote sever, thereby reducing the packet retransmissions.

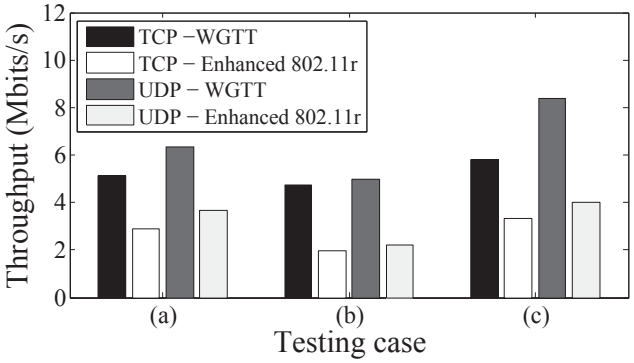


Figure 20: TCP/UDP throughput in different multi-client testing case. The client moves at 15 mph.

5.3 Microbenchmarks

We now present microbenchmarks, aimed at understanding the impact of certain parameters on our system's performance.

5.3.1 Choosing a proper window size. WGTT uses a time window w to compare ESNR readings from different APs (§3.1.1), so w is critical to the accurateness and agility of our AP selection algorithm. We do an emulation-based experiment to determine w. We drive at a constant speed (15 mph) and collect 10 runs of ESNR data. Based on that, we vary the window size and compute the average channel capacity loss of 10 test cases. The result is shown in Figure 21. As shown, the capacity loss decreases as we enlarge the window size to 10 ms, and then increases as we expand the window size further. Suggested by the experiment result, we set w to 10 ms, which achieves the minimal channel capacity loss. We

further perform a sensitivity analysis of w at various vehicle speeds and find that it remains unchanged.

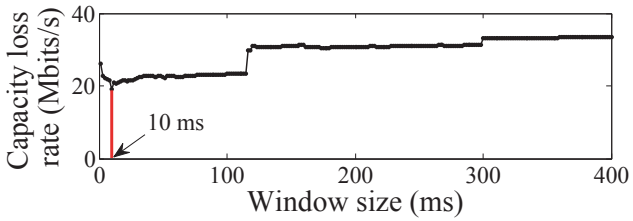


Figure 21: Capacity loss rate for different window sizes.

5.3.2 Link layer ACK collision rate. As all APs in the network are simultaneously associated with the client, they all reply with link layer (block) acknowledgements after hearing an uplink packet, resulting in potential collisions at the client, and so uplink packet retransmissions. To understand how frequently link layer acknowledgement collisions happen in our system, we first turn RTS/CTS off and send a constant-rate stream of UDP packets using iperf3 and measure the number of uplink retransmissions to find the upper-bound of the frequency of link layer acknowledgment collisions. The result is shown in Table 3. Link layer acknowledgement collisions rarely happen in our system, with only 0.001% for 70 Mbits/s UDP sending rate, and 0.004% for 90 Mbits/s UDP sending rate. We speculate the reason is that TP-Link AP issues an HT-immediate block ACK with some backoff. We find that the interval between the last MPDU and block ACK varies in the range of *us* instead of being a fixed time. If backoff is performed, side lobes of our parabolic antenna prevent link layer acknowledgement collisions from happening. Given our experiment results, we conclude that such a small fraction of collision have minimal impact on WGTT's throughput.

Table 3: Link layer ACK collision rate at the client side.

Data rate (Mbits/s)	**70**	**80**	**90**
Ack collision rate (%)	0.001	0.003	0.004

5.3.3 Impact of time hysteresis for AP switching. We next examine the impact of time hysteresis for AP switching on the throughput. In this experiment, we vary the time hysteresis from 120 ms down to 40 ms, and let a client transit across eight APs to receive TCP packets sent from the server. Figure 22 shows the throughput against time and a timeseries showing which AP the client is associated with during its movement. As shown, the throughput changes in a similar trend for the three different time hysteresis settings, fluctuating due to channel variations but never dropping to zero due to prompt AP switches, as shown below in the figure. As we decrease the time hysteresis, we can see the throughput grows gradually from 1.3 Mbits/s to around 6.4 Mbits/s at 2 s. The throughput gain is due to the fact that the channel condition changes frequently at driving speed. A smaller time hysteresis renders our switching algorithm more able to adapt to the fast channel changes, and so achieving a higher throughput.

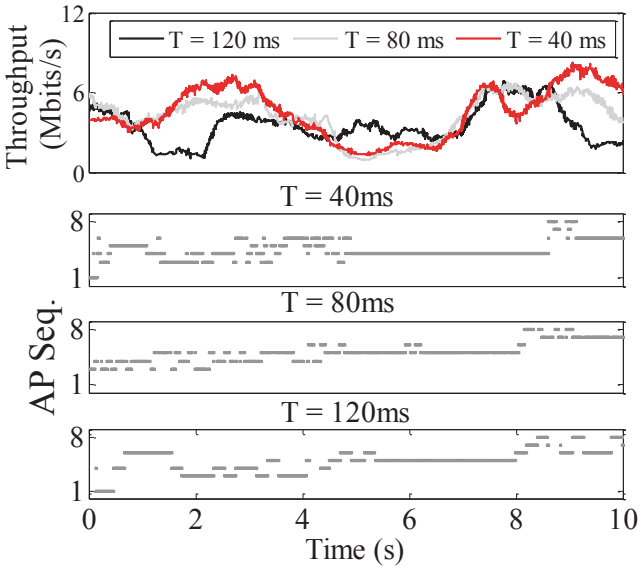


Figure 22: The TCP throughput against time in different and a timeseries showing which AP the client is associated with during its movement. The client moves at 15 mph.

5.3.4 Impact of AP density. We now examine the impact of AP density on the throughput. In this experiment, we vary the driving speed of the client and measure the UDP throughput in both a dense and sparse AP deployment area of our testbed. Figure 23 shows the result. As expected, WGTT achieves a consistently high UDP throughput in both sparse and dense AP deployment under different driving speeds. The UDP throughput increases from 6.7 Mbits/s on average to around 9.3 Mbits/s as the client moves from the area of low AP density to the area of high AP density. This is due to the fact WGTT benefits from uplink diversity of nearby APs and receives packets through multiple paths.

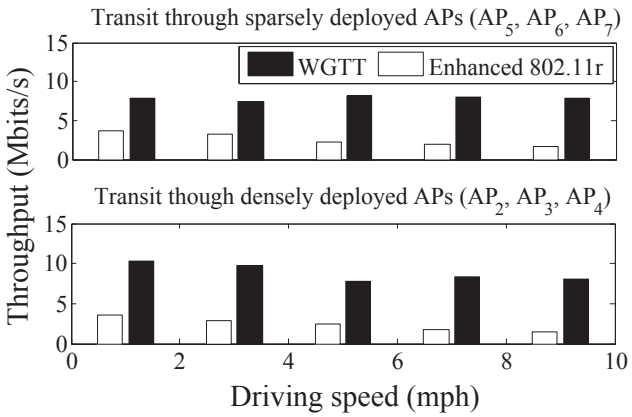


Figure 23: UDP throughput in areas of different density in our testbed deployment.

5.4 Case studies

We next conduct three real-world case studies to examine WGTT's performance in streaming applications: online video streaming, remote video conferencing, and web browsing.

Online video. In this case study, we test WGTT's ability to reliably stream video to mobile clients. We employ the video rebuffer ratio as the metric to measure the video's quality of experience (QoE). Video rebuffer measures number of times and the duration of interruptions due to re-buffering. The video rebuffer ratio is defined as the fraction of rebuffers over the time duration that the client transits through eight APs. To minimize the impact of Internet latency on video rebuffers, we cache the video in the local server. In this experiment, a volunteer is asked to watch a HD video (1280 × 720 resolution) through our testbed. We then vary the driving speed of the client and test the rebuffer ratio of this video using the VLC media player [48] and play online streamming via FTP. We set the pre-buffer length to be 1,500 ms. The result is shown in Table 4. As shown, WGTT achieves a smoothed video playback with zero rebuffer ratio in both low (5 mph) and high (20 mph) driving speed. In contrast, the video rebuffer ratio reaches to 0.69 at 5 mph driving speed and drops gradually to 0.54 at 20 mph driving speed when the client transits through APs through Enhanced 802.11r fast roaming protocol. The decreasing trend of the rebuffer ratio here is due to the significant decreasing of the transition time at high moving speed.

Table 4: Video rebuffering ratio at different moving speed.

Client speed (mph)	5	10	15	20
WGTT	0	0	0	0
Enhanced 802.11r	0.69	0.64	0.61	0.54

Remote video conferencing. In this case study, we test WGTT's ability to provide reliable video streaming to mobile clients. Unlike the previous case study, remote video conferencing requires the mobile client to simultaneously upload and download real-time video streams, hence have an even higher link quality requirement. In this experiment, we run a two-user video conference, with one user on the moving vehicle and the other in a conference room. Both applications periodically present their fps on the application's user interface. We use scrot, a screenshots software to record the fps every 1 s. We then measure the frames per second (fps) of the downlink video on each client side and show the CDF of the averaged value in Figure 24. As shown, we achieve an 85% percentile of 20 fps using Skype [42] on both 5 mph and 15 mph driving speeds. The fps increases to 56 when we turn to Google Hangouts [19]. This is because Google Hangouts automatically reduces image resolution of each frame.

Web browsing. In this case study, we test WGTT's ability to load web pages quickly for a mobile client. We invite one volunteer to browse the eBay homepage (2.1 MB) during its fast transition among eight APs. We then vary the client moving speed, and measure the duration of time that the system launches the web browser until the webpage is fully loaded in the web browser. To minimize the impact of Internet latency on our measurement result, we store the webpage on the local server, allowing the client to load the webpage directly. At each driving speed, we repeat the experiment 10 times and average the result (Table 5). As shown, we achieve a constantly stable webpage loading time at different driving speeds when employing WGTT as the default roaming protocol. In contrast, the webpage loading time increases to around 18 s at 10 mph as when we change the roaming protocol to Enhanced 802.11r. As we increase the driving speed further, the client never successfully loads the whole page during its short driving time across eight APs.

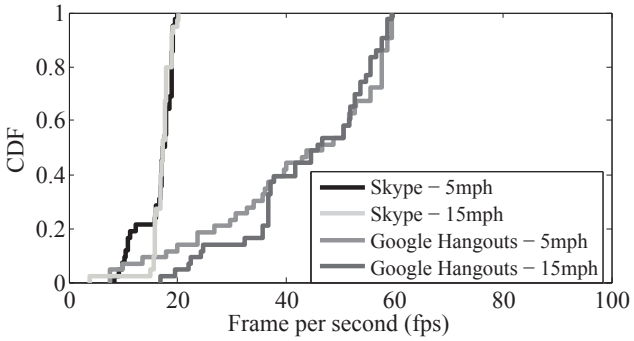


Figure 24: CDF of the frame rate of video conferencing in different moving speed settings.

Table 5: Web page loading time at different driving speeds.

Client speed (mph)	5	10	15	20
WGTT	4.44	4.64	4.34	4.47
Enhanced 802.11r	15.49	18.21	∞	∞

6 RELATED WORK

Pack *et al.* [37] survey much of the older related work in the area; in this section we focus on newer work. While Wi-Fi handover is a well-trodden area *in name*, prior work largely focuses on walking and stationary scenarios, or sporadic, delay-tolerant patterns of opportunistic network connectivity while driving. We thus argue that while still nominally "handover," the objective of Wi-Fi Goes to Town qualitatively differs from the objectives of the following prior work.

6.1 Wi-Fi-based systems

Streamlining AP discovery. Taking handoff itself as a given, these techniques reduce its latency. Multiscan [3] leverages multiple radios to make a connection to a new AP before breaking the current one. Syncscan [39] and DeuceScan [7] time-synchronize APs to accelerate the standard 802.11 handoff process, and Neighbor Graphs [41] allow APs to provide channel information about their neighbors, as was later standardized in 802.11k.

Multiple AP association at the link layer. This group of approaches connects clients to multiple APs simultaneously, switching packets at the link and/or network layers. FatVAP [24] balances traffic between multiple APs at time granularities of two seconds,

and so is designed for clients with good continuous connectivity to multiple APs at a time. DenseAP [30] targets the same scenario, updating load metrics every 20 seconds and client locations every 30 seconds. Multiscan [3] equips each client with two radios, allowing it to "pre-associate" with one AP while actively communicating with another, thus eliminating handoff overhead. The later 802.11k and 802.11r standards reduce handoff latency, but without the need for multiple radios. VirtualWiFi (also known as MultiNet) [6] and Juggler [34] allow clients to associate with more than one AP at a time, but do not balance transport-layer flows across APs.

SWIMMING [27] is a simulator-based proposal that uses energy detection to classify any burst of energy of similar length to an acknowledgement as such. Thus while it may correctly classify multiple simultaneous ack packets from more than one AP as an ack, as noted by the authors, energy detection is prone to false positives resulting in data loss and degrading end-to-end performance (based on Matlab simulations). In comparison, Wi-Fi Goes to Town contributes novel fine-grained AP switching algorithms that SWIMMING does not anticipate, specifically to handle the vehicular picocell regime. We experimentally evaluate the prevalence of downlink ack collisions (§5), showing that the issue has minimum effect on end-to-end throughput performance. We also claim the first practical implementation and evaluation in a testbed of an entire seamless roadside hotspot system.

ViFi [2] is a system targeted at cars, allowing nearby APs to "salvage" uplink packets from vehicular clients APs, and develops a probabilistic algorithm for de-duplicating packets on the uplink. However, ViFi APs have large cell sizes resulting from use of low-rate 1 Mbit/s bit-rate transmissions from rooftop-mounted omnidirectional antennas, maximizing range, but sacrificing spatial reuse. MRD [28] leverages diversity reception and the combination of different parts of the same frame received by multiple APs in the uplink in order to recover packets. Divert [29] finds that frame loss history, combined with a hysteresis mechanism to limit the speed of AP switching yields performance gains for walking-speed clients. While these techniques share some of our proposed mechanism, they don't push the envelope in terms of cell size, nor do they leverage fine-grained location at fine timescales to preemptively place packets at the right access point before the client arrives at that location, the keys to our proposed method of making the network more reliable for users on the go.

Approaches at the transport layer. Classic approaches to mobility at the transport layer include rapidly rebinding TCP endpoints in response to connectivity changes at the network layer or below, as TCP Migrate [43] proposes, or rerouting packets at the network layer, as Mobile IP [38] proposes. But these methods are intended for mobility on the temporal scale of minutes to hours and longer, in contrast to the much finer timescales of our work. Recent Multipath extensions to TCP enable its operation over multiple Internet paths (MP-TCP [15]). Croitoru *et al.* [11] exploit MP-TCP, lettering the client associate to multiple nearby APs and send one subflow over each connection. But again, their approach is tuned for walking-speed mobility and hence slow to react to link quality changes: indeed, the authors observe in their experiments a latency of 1–2 seconds for traffic to shift to a better AP once links change.

Industrial efforts. In addition to, and/or sometimes extending the 802.11 standards for fast handover 802.11r and 802.11k (described in detail above, §2), multiple Wi-Fi vendors have developed their own solutions for fast handover of Wi-Fi clients, all of which target walking speed-mobility, and many of which are not described in full technical detail in the associated white papers we reference here.

Like Wi-Fi Goes to Town, commercial offerings from Extricom, Meru, and Ubiquiti also present multiple APs as one AP to the client, with Extricom and Meru using a controller to make handoff decisions, and Ubiquiti lets APs talk to each other to determine the best one for packet delivering. However, these systems are either at a high cost (*e.g.*, the controller of Extricom costs over 6,800 usd [14]) or performing worse in fast moving scenario (as we experimentally demonstrated in §2). Linksys, Meraki and Cisco also offer fast handover APs, but addressing on the fast client authentication.

6.2 Opportunistic, disconnected systems

A large body of work has focused on predicting rough mobility and connectivity patterns, but operates at much more coarse temporal (*e.g.* 10 seconds) and spatial (100 m × 100 m) scales [8, 32, 33] than Wi-Fi Goes to Town. A 2006 study of vehicular Wi-Fi hotspot access [4] shows that Wi-Fi hotspots are often usable from the road at speeds of up to 60 kph, but performance significantly degrades at speeds above 30 kph. Gass *et al.* [16] measure the antenna coverage and packet loss rates at various vehicle speeds without considering AP switching. Ott *et al.* [36] measure transmission characteristics of TCP/UDP packets in a moving vehicle that passes by one or more APs. However, they do not consider the packet buffering problem during AP handover. The Cartel system [20] and the QuickWiFi protocol [13] develop an optimal scanning strategy based on an *a priori* distribution of AP channel assignments, and streamline the 802.11 association process, but do not coordinate traffic flows between and among roadside APs. Spider [44] uses models of a vehicular client's join probability, simultaneously associating with multiple APs on different channels and exposing a virtual interface for each AP, like VirtualWiFi. Mobisteer [31] uses large transmit beam-forming antennas on clients to steer transmissions towards roadside APs.

7 DISCUSSION

System cost. The WGTT prototype costs around $120 USD per AP. This cost can be further cut down to $15 USD by building a dedicated WGTT AP. *e.g.*, using a $5 USD ESP8266 system-on-chip Wi-Fi chipset, a $5 USD Raspberry Pi Zero [40], and a less than $5 USD "Pringle Can" antenna [5].

Picocells vs. macrocells. Picocells increase network capacity by re-using the same frequencies between adjacent APs. In contrast, due to inter-cell interference, macrocells achieve lower throughput and spectral efficiency. This in turn reduces the data rate of a user which is detrimental to the network as a whole.

Multi-channel settings. The current WGTT prototype works on a single channel. Letting adjacent APs work on different wireless channels would avoid serious wireless interference and improve the absolute throughput of the system. However, spectrum efficiency would then drop significantly. Moreover, the nearby APs working

on different channels would be unable to forward overheard packets, resulting in a higher uplink packet loss rate and thus hurt the throughput.

Large area deployment. The current WGTT prototype consists of eight APs. In the future, we plan a large deployment and a large-scale measurement study, *e.g.*, measuring the achievable network capacity. Network providers are also well-motivated to deploy WGTT because they would gain advertising opportunities, and a transit system would gain revenue, either directly through customer billing or indirectly through deals.

8 CONCLUSION

We have presented Wi-Fi Goes to Town, the first Wi-Fi based roadside hotspot network designed to operate in a new performance regime for wireless networks, the vehicular picocell regime. Our design uses a nearby controller commanding tight control of the AP array. Our controller and APs cooperate to implement an AP selection, downlink queue management algorithm, and uplink acknowledgement sharing protocol that work hand-in-hand to leverage wireless path diversity to precisely switch downlink traffic, and make uplink acknowledgements more reliable. Wi-Fi Goes to Town is the first step in a line of work that will scale out the wireless capacity of roadside hotspot networks using small cells. Many interesting problems in this performance regime remain open: the choice of antenna and Wi-Fi chipset technology; the application of MIMO techniques ranging all the way from basic link-level MIMO to Distributed multi-user MIMO beamforming; and further mitigation of inter-AP interference.

ACKNOWLEDGEMENTS

We thank our shepherd Prof. Bhaskaran Raman and the anonymous reviewers for their constructive feedback. This material is based upon work supported by the National Science Foundation under Grant No. 1617161 and by a Google Research Award.

REFERENCES

[1] 802.11r, 802.11k, and 802.11w Deployment Guide, Cisco IOS-XE Release 3.3. Website.
[2] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, J. Zahorjan. Interactive Wi-Fi connectivity for moving vehicles. *SIGCOMM*, 2008.
[3] V. Brik, A. Mishra, S. Banerjee. Eliminating handoff latencies in 802.11 WLANs using multiple radios: Applications, experience, and evaluation. *IMC*, 2005.
[4] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, S. Madden. A measurement study of vehicular internet access using *in situ* wireless networks. *MobiCom*, 2006.
[5] Building a Can antenna. Website.
[6] R. Chandra, P. Bahl, P. Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. *Infocom*, 2004.
[7] Y.-S. Chen, M.-C. Chuang, C.-K. Chen. DeuceScan: Deuce-based fast handoff scheme in IEEE 802.11 wireless networks. *IEEE Transactions on Vehicular Technology*, 2008.
[8] Y.-C. Cheng, Y. Chawathe, A. LaMarca, J. Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. *MobiSys*, 2005.
[9] The Click modular router project. Website.
[10] Cooper's Law. Website.
[11] A. Croitoru, D. Niculescu, C. Raiciu. Towards Wi-Fi mobility without fast handover. *NSDI*, 2015.
[12] Linksys EA7500 MAX-STREAM AC1900 Wi-Fi router. Website.
[13] J. Eriksson, H. Balakrishnan, S. Madden. Cabernet: Vehicular content delivery using WiFi. *MobiCom*, 2008.
[14] Extrecom multiSeries 1000 wireless LAN switch. Website.
[15] A. Ford, C. Raiciu, M. Handley, O. Bonaventure. TCP extensions for multipath operation with multiple addresses. Tech. rep., RFC-6824, 2013.
[16] R. Gass, J. Scott, C. Diot. Measurements of in-motion 802.11 networking. *WMCSA*, 2005.
[17] D. Halperin, W. Hu, A. Sheth, D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. *SIGCOMM*, 2010.
[18] ——. Tool release: Gathering 802.11n traces with channel state information. *ACM SIGCOMM CCR*, 2011.
[19] Google Hangouts. Website.
[20] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, S. Madden. CarTel: A distributed mobile sensor computing system. *SenSys*, 2006.
[21] IEEE 802.11k-2008—Amendment 1: Radio Resource Measurement of Wireless LANs, 2008.
[22] IEEE 802.11r-2008—Amendment 2: Fast Basic Service Set (BSS) Transition, 2008.
[23] Iperf3. Website.
[24] S. Kandula, K. Lin, T. Badirkhanli, D. Katabi. FatVAP: Aggregating AP backhaul capacity to maximize throughput. *NSDI*, 2008.
[25] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 2000.
[26] Laird GD24BP 2.4G directional antenna. Website.
[27] P. Lv, X. Wang, X. Xue, M. Xu. SWIMMING: Seamless and efficient WiFi-based internet access from moving vehicles. *IEEE Transactions on Mobile Computing*, 2015.
[28] A. Miu, H. Balakrishnan, C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. *MobiCom*, 2005.
[29] A. Miu, G. Tan, H. Balakrishnan, J. Apostolopoulos. Divert: Fine-grained path selection for wireless LANs. *MobiSys*, 2004.
[30] R. Murty, J. Padhye, R. Chandra, A. Wolman, B. Zill. Designing high performance enterprise Wi-Fi networks. *NSDI*, 2008.
[31] V. Navda, A. Subramanian, K. Dhanasekaran, A. Timm-Giel, S. Das. MobiSteer: Using steerable beam directional antenna for vehicular network access. *MobiSys*, 2007.
[32] A. Nicholson, Y. Chawathe, M. Chen, B. Noble, D. Wetherall. Improved access point selection. *MobiSys*, 2006.
[33] A. Nicholson, B. Noble. BreadCrumbs: Forecasting mobile connectivity. *MobiCom*, 2008.
[34] A. Nicholson, S. Wolchok, B. Noble. Juggler: Virtual networks for fun and profit. *IEEE Transactions on Mobile Computing*, 2010.
[35] Openwrt Chaos Calmer v15.05.1. Website.
[36] J. Ott, D. Kutscher. Drive-thru internet: Ieee 802.11 b for" automobile" users. *INFOCOM*, 2004.
[37] S. Pack, J. Choi, T. Kwon, Y. Choi. Fast handoff support in IEEE 802.11 wireless networks. *IEEE Communication Surveys and Tutorials*, 2007.
[38] C. Perkins. IP Mobility Support. RFC 2002, IETF, 1996.
[39] I. Ramani, S. Savage. SyncScan: Practical fast handoff for 802.11 infrastructure networks. *Infocom*, 2005.
[40] Raspberry Pi . Website.
[41] M. Shin, A. Mishra, W. Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. *MobiSys*, 2004.
[42] Skype. Website.
[43] A. Snoeren, H. Balakrishnan. An end-to-end approach to host mobility. *MobiCom*, 2000.
[44] H. Soroush, P. Gilbert, N. Banerjee, B. Levine, M. Corner, L. Cox. Concurrent Wi-Fi for mobile users: Analysis and measurements. *CoNEXT*, 2011.
[45] Lenovo Thinkpad T430 laptop. Website.
[46] Tp-link N750 gigabit router. Website.
[47] D. Tse, P. Viswanath. *Fundamentals of Wireless Communication.* Cambridge University Press, 2005.
[48] VideoLAN. Website.
[49] Y. Xie, Z. Li, M. Li. Precise power delay profiling with commodity Wi-Fi. *ACM MobiCom*, 2015.