# A Light-Weight Wireless Handshake

Kate Ching-Ju Lin
Research Center for IT Innovation
Academia Sinica
katelin@citi.sinica.edu.tw

Yuan-Jen Chuang
Research Center for IT Innovation
Academia Sinica
johnnyc2277@citi.sinica.edu.tw

Dina Katabi
CSAIL
MIT
dk@mit.edu

## ABSTRACT

In many wireless systems, it is desirable to precede a data transmission with a handshake between the sender and the receiver. For example, RTS-CTS is a handshake that prevents collisions due to hidden terminals. Past work, however, has shown that the overhead of such handshake is too high for practical deployments. We present a new approach to wireless handshake that is almost overhead free. The key idea underlying the design is to separate a packet's PLCP header and MAC header from its body and have the sender and receiver first exchange the data and ACK headers, then exchange the bodies of the data and ACK packets without additional headers. The header exchange provides a natural handshake at almost no extra cost. We empirically evaluate the feasibility of such lightweight handshake and some of its applications. Our testbed evaluation shows that header-payload separation does not hamper packet decodabilty. It also shows that a light handshake enables hidden terminals, i.e., nodes that interfere with each other without RTS/CTS, to experience less than 4% of collisions. Furthermore, it improves the accuracy of bit rate selection in bursty and mobile environments producing a throughput gain of about 2x.

**Categories and Subject Descriptors** C.2.2 [**Computer Systems Organization**]: Computer-Communications Networks
**General Terms** Design, Experimentation
**Keywords** Handshake, Testbed

## 1. INTRODUCTION

Wireless networks would benefit from a *lightweight* handshake that enables a sender and its receiver to exchange control information before data transmission. Such a handshake can prevent packet collisions caused by hidden terminals in a manner similar to RTS-CTS, but without incurring the high overhead of today's RTS-CTS [5]. A lightweight handshake can also provide the sender with the most up-to-date receiver feedback, which is desirable in many wireless systems. For example, bit rate adaptation requires the transmitter to learn the channel state from the receiver. Existing schemes rely on past ACK packets (or their loss) to learn this information [4, 16, 18, 14, 10]. However, if the traffic has a low duty cycle or is highly bursty, the most recent ACKs may be too old to reflect the channel state at the time of transmission. The problem of stale feedback is exacerbated by mobility which causes the channel quality to vary quickly over tens of milliseconds [18]. A low-overhead handshake allows the sender to learn the channel quality on a per-packet basis; it can then pick the optimal bit rate for each packet. A similar problem appears in MIMO systems that use beamforming [20, 3] or interference alignment [8]. These systems require the receiver to inform the sender of the channel coefficients, which the sender uses for pre-coding. However, when traffic is bursty or the user's ACKs are infrequent because they are interspersed with traffic from other streams, the ACK feedback may not reflect the instantaneous channel coefficients. Stale channel state, in this case, leads to inaccurate beamforming and hence inter-stream interference [17]. A light handshake on the other hand can convey the most up-to-date channel state at the time of transmission.

Most of the overhead in a wireless handshake stems from the introduction of control frames. Even if the sender and receiver want to exchange only one bit, embedding a bit in a control frame leads to excessive overhead because the frame needs a physical layer (PLCP) header and a MAC header. Wireless headers are relatively large because they need to contain sufficient information for packet detection, channel estimation, carrier frequency estimation, sender and receiver IDs, etc. Furthermore, those control frames need to be sent at the lowest bitrate to ensure correct reception. For example, multiple prior works have shown that enabling RTS-CTS handshake in 802.11 results in a significant overhead and reduces the throughput by as much as 50% [5, 15].

This paper aims to develop a low-overhead handshake that does not require the sender or receiver to transmit additional control frames. Our approach is based on the observation that one can separate the transmission of a packet from its header without hampering packet decodability. Specifically, we propose a light-weight handshake where the sender and receiver first exchange the data packet header followed by the ACK header, and then the data packet body followed by the ACK body, without any additional headers, as shown in Fig. 1(b). The header exchange plays the role of a handshake. For example, the header exchange can play the role of RTS-CTS to prevent hidden-terminal collisions. Yet, since the nodes have to transmit the headers anyway, the overhead of such a handshake is minimal.

Supporting such a lightweight handshake requires addressing a few practical issues to ensure that the data and ACK payloads continue to be decoded correctly despite being detached from their headers. This includes frame detection, oscillator offset correction and channel estimation, which are discussed in detail in the rest of the paper.

We implement the above lightweight handshake in software radios and empirically demonstrate both the feasibility and benefit of such approach. Our evaluation focuses on three questions:

- *Can header separation harm packet decodability increasing the packet loss rate?* Our empirical results show that the impact of header separation on packet decodability is negligible. In particular, a separation as large as 500 $\mu$s causes an average increase in loss rate of 0.0005, which is insignificant for wireless channels.
- *Does the design address the hidden terminal problem?* We empirically show that a light handshake like the one in Fig. 1(b) can reduce collisions caused by hidden terminal from 60–100% to less than 4%.

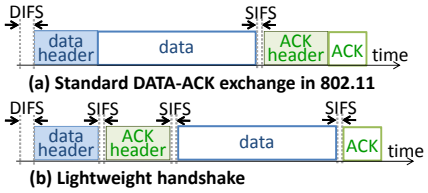


**Figure 1—Format of packet exchange:** (a) a DATA-ACK exchange in 802.11; (b) a light handshake that does not send control frames. It rather separates the headers from the packets and sends all headers early on.

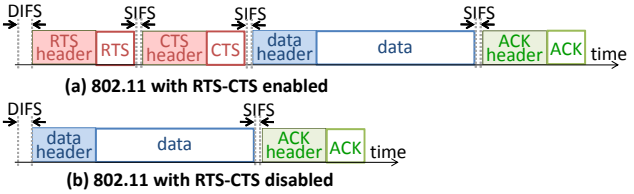


**Figure 2—Packet format of RTS-CTS exchange:** (a) a RTS-CTS exchange followed by DATA-ACK exchange in 802.11; (b) a DATA-ACK exchange in 802.11 with RTS-CTS disabled.

- *Does it improve rate adaptation in bursty and mobile environments?* We focus on a mobile transmitter that sends a stream of packets, where consecutive packets (and hence consecutive ACKs) are separated by 50 ms. This scenario arises when the transmitter has a low duty cycle or the medium is congested and hence a user's packets are interspersed with packets from other flows. Our results show that a handshake-based rate adaption protocol doubles the throughput in comparison with SampleRate, which relies on the ACK stream for its feedback [4].

We anticipate that the introduction of a low-overhead handshake can facilitate many wireless applications that benefit from fresh receiver feedback.

## 2. LIGHTWEIGHT HANDSHAKE

Handshaking in 802.11 is considered an expensive process because it requires transmitting additional control frames, like the RTS and CTS frames shown in Fig. 2(a). The overhead of such 802.11 handshake stems from the following components:

- Most of the overhead stems from the headers of the control frames. Every wireless frame is required to be transmitted with a physical-layer (PLCP) header, which includes information for packet detection, channel estimation, and carrier offset correction, and a MAC header, which includes node addresses, the used bit rate, and other control information. Furthermore, the PLCP header and these control frames are sent at the lowest bit-rate to ensure correct reception and last for hundreds of microseconds.

- There is some overhead involved in sending the feedback itself. This overheard depends on the application but in most cases is negligible in comparison with the header overhead. For example, in bitrate adaptation, the receiver needs to send the preferred bitrate. Similarly, to protect against hidden terminal collisions, the receiver can send exchange duration measured in OFDM symbols. These information requires only a few extra bits, which is less single OFDM symbol.

- Finally, switching between transmission and reception requires a SIFS interval. A SIFS in 802.11g is 10 $\mu$s, and hence is minimal in comparison with the headers.

Since most of the overhead is due to the headers, a low-overhead handshake requires amortizing the cost of the headers.

### 2.1 Header-Payload Separation

This paper aims to enable light handshaking before data transmission. To do so, we adopt a design that does not send control frames. Our design is based on the observation that 802.11 channel coefficients do not change for periods shorter than a few milliseconds [17]. Thus, it is not necessary to send a PLCP and MAC headers and re-estimate the channel for every control frame because the period of one packet transmission is usually shorter than the channel coherence time.

Instead, one can split a packet's headers from the packet's body, and make the sender and receiver first exchange the data and ACK headers and then exchange the data and ACK bodies without additional headers, as illustrated in Fig. 1(b). A node can then estimate the channel and the carrier frequency offset from the data or ACK header and use the estimates from the header exchange to predict the channel coefficients of the symbols in the body of the data and ACK packets. However, we note that the ACK packet actually consists of only the PLCP header and MAC header, without any other payload. Hence, we only move the PLCP header and the address field in the MAC header of an ACK packet before data exchange, and leave the rest of the MAC header acting as the real ACK and send it after the data packet. Such header-payload separation provides a natural handshake, at a minimal cost. Any receiver feedback, e.g., the optimal bit rate, can be concatenated with the ACK header.

### 2.2 Practical Issues

For the above design to work correctly, we need to ensure that the decodability of a packet's body is unharmed despite header-payload separation. This involves supporting the following functions:

**Frame Detection:** The PLCP header contains a preamble that is used for packet detection. In random access networks, packets can arrive anytime. Hence, an 802.11 receiver detects a new packet using a combination of a power detector and preamble correlation [11]. In the proposed light handshake, the header which contains the preamble is separated from the packet's body. Hence, a node needs to detect the beginning of the body frame without any known preamble. The same challenge occurs for a node that needs to detect the body of an ACK that is separated from the ACK header. The solution to this challenge is however simple. The reason standard 802.11 packets/frames need a special preamble for detection is that they can arrive at any time. In contrast, in the light handshake, a packet's body arrives at a specific time relevant to the beginning of its header. Specifically, once a node detects the header, it does not need to perform packet detection again for the body of the data or ACK packets. Instead, the node can compute the arrival time of the data packet body to be the airtime occupied by the headers and SIFS (measured in OFDM symbols). Similarly, the arrival time of the ACK can be computed given the SIFS duration and the duration of the data payload, which can be computed based on the selected bitrate and the packet length embedded in the data header. All of these estimates are accurate up to the clock resolution of the 802.11 transceiver which is significantly smaller than the margin of error in packet detection.

**Phase Tracking:** In wireless systems, it is unlikely that the oscillators on the transmitter and receiver are tuned to exactly the same frequency. The difference between their center frequency is called the carrier frequency offset (CFO), $\Delta f$. The CFO causes the signal

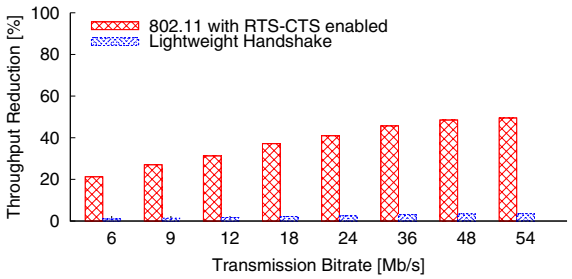


**Figure 3—In 802.11g, the throughput overhead of the light-weight handshake is less than 3.5%, whereas the overhead of RTS-CTS is 20%-50%, depending on the bitrate.**

at the receiver to keep rotating with respect to the transmitter's signal. This rotation translates to a linear increase of the signal's phase over time, which accumulates over the period between the arrival of a packet header and its body. Thus, the receiving node has to correct this phase change before it can correctly decode the body of the packet. To compensate for this rotation, the node has to estimate the CFO, $\Delta f$, from the packet header, as it does today [11]. It then compensates for the phase rotation by multiplying each signal sample, $i$, in the packet body by $e^{j2\pi\Delta f T_i}$, where $T_i$ is the interval between computing the CFO from the packet's header and the arrival of sample $i$ in the packet body. Similar to standard phase tracking in 802.11 PHY [11], one can refine the phase compensation further by leveraging the OFDM pilot bins, which are special subcarriers that carry a known data pattern and are embedded in every 802.11 OFDM symbol. Since the pilots carry known signals, they receiving known can compare the received signal after phase compensation with the expected signal to compute any residual phase error and correct for it.

**The Impact of Tx/Rx Switching on the CFO:** In some hardware, switching between transmission and reception may disturb the carrier frequency and change the CFO. This will introduce errors in phase tracking, described above. We counter this effect as follows. Off-the-shelf radios such as the USRP [1] have both a receive and a transmit chain connected to the same antenna; Typically, there is a switch that turns the transmit chain off while receiving. However, to prevent oscillator disturbance, we use the technique in [7]. Specifically, we keep both chains always on and set the power of the transmitted signal to zero whenever the radio is in reception mode, so that no switching is needed. This can be done even with USRP radios.

**Sender and Receiver Identification:** In the light handshake, the transmissions of data header, ACK header, data body and ACK body are separated by SIFS. This means that the channel is never left idle for a DIFS period and hence no other nodes would interrupt the data-ACK exchange. Consequently, a node pair can get the identification of each other from the header exchange without additional identification exchange before the data and ACK bodies.

**CRC Check:** To ensure that the information in the headers can be decoded correctly, we add an extra CRC check in the tail of each header. If a node pair misses or incorrectly decodes one of the data or ACK headers, it needs to contend for the medium again.

## 2.3 Overhead of Light Handshake

The total overhead from the light handshake is two SIFS's, two CRC checks and the receiver feedback in the ACK header. For rate adaptation, the feedback is 4 bits, which is sufficient to identify one of the available 802.11g bit rates. Hidden terminal detection requires sending the duration of the transmission measured in OFDM symbols, which can be expressed using 12 bits. Fig. 3 shows a comparison between throughput reduction caused by RTS-CTS and the light handshake in 802.11g. All the airtime that is not occupied by the frames and IFS, as shown in the top graph of Fig. 1, is considered as the overhead. We compute the throughput of 1500-byte data packets with respect to different bit rates according to the 802.11 specifications, in which the PLCP header with a long preamble occupies 192 $\mu$s, the ACK is sent at the lowest rate 6 Mb/s and the MAC header is sent at the same rate with the data packet. We exclude the airtime occupied for random backoff.[1] The figure shows that our light-weight handshake reduces the throughput overhead from 20%-50% to 1%-3.5%, which can be compensated by the benefit of enabling handshaking.

# 3. EXPERIMENTAL ENVIRONMENT

**(a) Testbed:** We evaluate the light handshake in the testbed shown in Fig. 4. For each experiment, we place two nodes randomly in the marked locations in Fig. 4. These locations include both line-of-sight (LOS) and non-line-of-sight (NLOS) scenarios and span an SNR range of 5 to 25 dB, which is the SNR range spanned by 802.11 [16].

**(b) Hardware:** Each node in the testbed is a 3.07GHz Intel i7 machine equipped with a USRP2 board [1] and a RFX2400 daughterboard, which communicates on a 10 MHz channel and has a central frequency of 2.43GHz.

**(c) Software:** We implement the light handshake using a software radio. Since software radios process the signal in the user mode on the PC, they cannot support realtime events, i.e., they cannot support the timing of the 802.11 MAC, nor can they detect a header and turn around to transmit within a SIFS interval. To work around the lack of realtime support in software radios, we implement header-body separation for the data packet and the ACK packet separately.

**(d) Physical Layer:** We implement the main functions of the 802.11 PHY layer, including OFDM, QAM modulation, convolutional codes, and the Viterbi decoder. In particular, we leverage the GNURadio OFDM code base, which provides different 802.11 modulations (BPSK, 4QAM, 16QAM, and 64QAM) and the corresponding convolutional codes [2]. However, unlike 802.11, which operates over a 20MHz channel, USRP2 operates over a 10MHz chanel. Hence, USRP2 supports data rates between 3MB/s and 27 Mb/s. The optimal bit rate in this range depends on the instantaneous channel quality. Our implementation uses the same number of OFDM data and pilot subcarriers as 802.11 (a total of 64 subcarriers). The implementation also supports packet detection, carrier frequency offset estimation, channel estimation, and phase tracking [11].

**(e) TX/RX Switching:** To make header-body separation work properly, we need to ensure that a node keeps synchronized between the arrival of a header and its body. However, as explained in §2 switching the transmit and receive chains on and off can perturb the CFO. To avoid such perturbations, we turn off the TX/RX switch in USRP, which leaves both the transmit and receive chains connected to the antenna all the time. During reception, we simply set the transmitted signal samples to zero.

[1]Due to the space limit, we only plot the overhead of the default 802.11g with the long preamble enabled. However, even when the optional short preamble is used, our scheme can decrease the overhead from 16.69%-44.74% to 1.05%-4.84%. In 802.11a, which has a much shorter PLCP header (20 $\mu$s), our scheme can still reduce the overhead from 5.17%-25.89% to 1.7%-10%. For a 2x2 mimo system in 802.11n, the overhead is reduced from 14.3%-39.1% to 5.8%-19.2%.

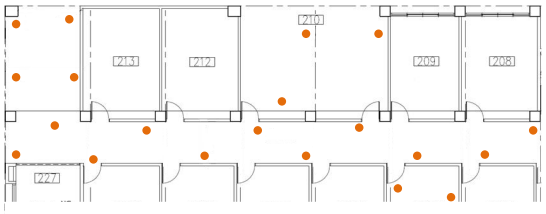


**Figure 4—The testbed. Dots refer to node locations.**

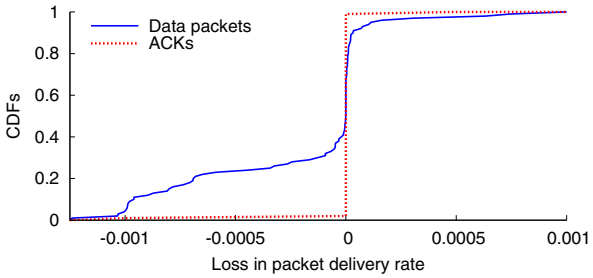


**Figure 5—Reduction in packet delivery rate due to header-body separation.** The figure shows that the impact of separating a packet header from its body as required by our light-weight handshake is negligible.

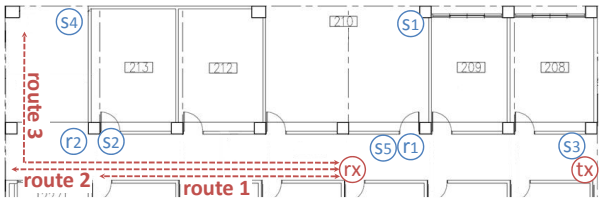


**Figure 6—The testbed for hidden terminal and rate adaptation experiments.** Blue nodes refer to hidden terminals. Red lines illustrate three movement paths.

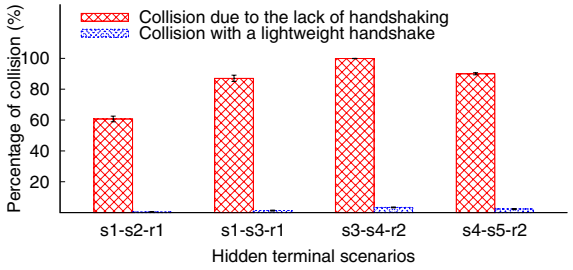


**Figure 7—Hidden terminals:** The figure shows the percentage of collisions with and without the light handshake.

# 4. EMPIRICAL STUDY

We use testbed evaluation to study the feasibility of the light handshake and its potential benefits.

## 4.1 Feasibility of Header-Payload Separation

We start by examining the impact of header separation on packet decodability, and hence the feasibility of such design.

**Method:** We make the transmitter send a stream of normal packets, followed by a stream of packets where the header is separated from the data by 500 $\mu$s, which is strictly larger than the separation required in our scheme (2 SIFS's + an ACK header). We repeat the same experiment for ACK packets. ACK headers however are separated from ACKs' payload by 12.5 ms which is strictly larger than the separation in our design (2 SIFS's + 1500B data at the lowest rate.) ACKs are also sent at the lowest bit rate as in 802.11. The receiver decodes these packets offline using a standard OFDM and Viterbi decoder. If the header is separated from the packet body, the receiver decodes the header, jumps by the inserted separation period, compensates for the channel rotation as described in §2 and continues decoding the packet body. We randomly assign two nodes to the marked locations in Fig. 4. We repeat the experiment with 150 different random assignments of node locations, and ensure that the results span the entire 802.11 operational SNR range.

**Result:** Fig. 5 shows the CDFs of reduction in packet delivery rate due to header-payload separation. The figure shows that for data packets the reduction in delivery rate is on average 0.0005 packet, which is negligible. The reduction is even lower for ACK packets despite that they suffer a longer separation. This is because ACKs are sent at the lowest bit rate. They are also much shorter than data packets and hence have a lower packet loss rate for the same bit error rate.

## 4.2 Application to the Detection of Hidden Terminals

Next, we investigate whether light handshakes can prevent collisions caused by hidden terminal scenarios. In this case, the data header plays the role of RTS. The receiver uses the packet size embedded in the data header and its selected bit rate to compute the duration of the data transmission, and includes this information in its ACK header. The ACK header plays the role of CTS; senders that hear the ACK header abstain from transmitting for the duration of the data-ACK exchange, specified in the ACK header. We would like to check whether this application of the light handshake is effective at preventing collisions caused by hidden terminals.

**Method:** We consider four different hidden terminal scenarios illustrated in Fig. 6, where transmitters s1 and s2 transmit to receiver r1 in the first scenario, s1 and s3 transmit to r1 in the second scenario, s3 and s4 transmit to r2 in the third scenario, and s4 and s5 transmit to r2 in the forth scenario. In each scenario, the two transmitters are located such that they cannot sense each other's signal and therefore might transmit simultaneously, resulting in a collision. To capture the actual number of collisions in these scenarios, we put 802.11 nodes in those hidden terminal positions, measure how often they have collisions, and then force the USRPs to have exactly as many collisions as the 802.11 nodes in their locations.

To work around the lack of realtime support in software radios, we run each experiment in three phases: First, one sender, say s1, transmits 10,000 packets, while the other sender, say s2, computes the percentage of packets (preambles) it cannot detect and as a result can cause collisions at the receiver. Second, reversely, s2 transmits, while s1 computes how many packets cannot be detected. Last, one sender transmits a data header, the receiver transmits an ACK header, and the second sender tries to detect the ACK header if it fails detecting the preamble from the first sender. We compute the percentage of collisions as the percentage of ACK headers that the second sender failed to detect.

**Results:** We show the percentage of packet collision in Fig. 7. The figure shows that, without handshaking, the two senders in the four hidden terminal scenarios in our testbed are very likely to transmit simultaneously resulting in collisions because most of packets sent by one sender cannot be detected by the other sender. Instead, our light-weight handshake, which operates similarly to RTS-CTS, can prevent two senders from transmitting their packets simultaneously,

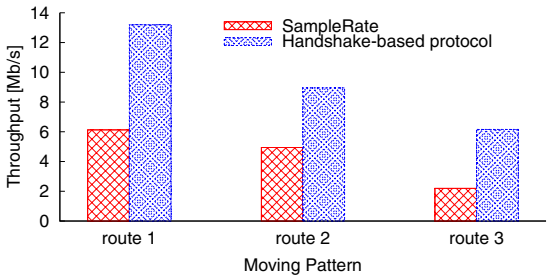


**Figure 8—Rate adaptation.** The figure shows that light-weight handshaking can pick the best rate for each packet and as a result can adapt to the high variability of mobile channels. In contrast, traditional rate selection algorithms, like SampleRate, which use past receiver feedback cannot track the channel state if the flow transmits at a low packet rate and hence has a low ACK rate.

reducing the percentage of packet collisions from 60-100% to less than 4%.

## 4.3 Application to Rate Adaptation

We examine whether the proposed light handshake can improve the performance of rate adaptation. We study its potential use by focusing on the scenarios where such a handshake is beneficial. That is, we let a mobile transmitter sends a stream of packets, where consecutive packets (and hence consecutive ACKs) are separated by 50 ms. This scenario arises when the transmitter has a low duty cycle or the medium is congested and hence a user's packets are interspersed with packets from other flows. This scenario is challenging for existing rate adaptation algorithms because they all rely on ACK packets for receiver feedback [4, 16, 18, 14, 10] and become ineffective if the channel changes faster than the ACKs arrive.

We compare two rate selection protocols: The first protocol is SampleRate, the default autorate algorithm implemented in Atheros Madwifi driver (version v0.9.4) [4]. SampleRate periodically samples the throughput of different bitrates every $10^{th}$ packet, and picks the bitrate with the highest average throughput [4]. The second protocol leverages the light handshake and is based on the effective SNR rate selection protocol introduced in [10]. The effective SNR (ESNR) is similar to the SNR but it takes into account the impact of frequency selectivity. In this protocol, the receiver uses the data header to compute the ESNR and selects the optimal bitrate by looking up the ESNR-to-rate mapping table as in [10]. [2] The receiver sends the optimal bitrate to the sender in the ACK header, hence allowing the sender to transmit the data packet at the optimal bitrate selected by the receiver. The rate adaptation protocol can thus be performed on a per-packet basis.

**Method:** We set up the receiver in one corner, and move the transmitter along one of the three paths indicated by the red lines in Fig. 6. During the movement, the transmitter sends a stream of packets for 2 minute. The packet interarrival time is set to 50 ms. The packet stream cycles between transmitting at each of the 802.11 bitrates, and for each rate it sends an 1000-byte packet followed a packet where the header and payload are separated.[3] A whole cycle is completed every 50 ms. We repeat each experiment multiple times. We collect the received packets and process them offline with both SampleRate and the proposed handshake-based protocol. We make SampleRate pick the best rate based on whether the packet was decoded correctly. For the handshake-based protocol, we make it pick the rate based on the ESNR of the data header of each packet.

**Results:** Fig. 8 shows a comparison between the throughput obtained with SampleRate and the handshake-based protocol. The figure shows that SampleRate produces a low throughput because it might select an inappropriate rate based on the out-of-date ACK feedback. The handshake-based protocol however can select the best bitrate according to the up-to-date feedback in the handshake ACK header, and as a result almost doubles the throughput of SampleRate. Note that the throughput difference between the three paths is due to that SNR decreases with distance. It is also interesting to note that the gains of the handshake protocol are highest for the third path. This is because the third path spans non-line-of-sight locations and experiences more SNR variations, and hence benefits more from up-to-date feedback.

# 5. RELATED WORK

Past work has empirically or analytically estimated the overhead of enabling RTS-CTS, and reported a throughput reduction of up to 50% for 802.11g [5] and MIMO systems [15]. Several systems [13, 19] try to leverage such handshake, while optimize its overhead by grouping multiple RTS frames as a GRTS (group RTS) [13] or by dynamically disabling RTS-CTS when a node detects no collision [19].

Some prior work proposes alternative solutions to the hidden terminal problem, such as ZigZag [6] or interference cancellation [9]. In comparison to these schemes, the lightweight handshake is significantly simpler and does not require changing the basic packet decoding algorithms in the 802.11 PHY.

Additionally, there is a large literature on rate adaptation including proposals for mobile scenarios. Some protocols use an optimized version of RTS-CTS [12, 13]. Others including SampleRate [4], SoftPHY [18], SNR-based [14, 10] and BER-based protocols [16, 18] do not use RTS-CTS, but rely on receiver feedback from the ACK stream. The approach in this paper is significantly lower overhead than schemes that use RTS-CTS. In comparison, with schemes that rely on ACK feedback, the approach in this paper can track channel variations even when ACKs are infrequent either because the flow sends a low packet rate or it shares the medium with many other flows and hence it is forced to a low rate by the MAC and the congestion control protocol. In contrast, protocols that rely on ACK feedback may be unable to track channel variations in such scenarios.

# 6. CONCLUSION

This paper introduces a light-weight per-packet handshake for 802.11 networks. Our approach is based on separating the data and ACK headers from their payloads and having the sender and receiver first exchange the header then the payload without headers. We show that such header separation does not hamper the packets' decodability. We empirically demonstrate two potential applications of such light handshake: hidden terminal detection and rate adaptation in for low rate mobile flows. We believe that such a light handshake is beneficial in a variety of additional scenarios where the receiver needs to send frequent feedback to the sender. This includes beamforming, multi-user MIMO systems, and interference alignment.

[2] It is typical to the receiver chain to make computation and act immediately. For example, the receiver computes CFO and channel coefficients from the first few symbols and applies their value to correct all subsequent data symbols. Hence, it can similarly real-time compute the SNR of the first few symbols and select the best rate by looking up the mapping table.

[3] Cycling all transmission bit-rates (from 3 Mb/s to 27 Mb/s) requires $2 * 1000 * 8 * (1/3 + 1/4.5 + 1/6 + 1/9 + 1/12 + 1/18 + 1/24 + 1/27)\mu s = 16.8ms$.

## 7. REFERENCES


[1] Ettus Inc., Universal Software Radio Peripheral. http://ettus.com.

[2] *IEEE Std 802.11-1997*, pages i –445, 1997.

[3] E. Aryafar, N. Anand, T. Salonidis, and E. W. Knightly. Design and experimental evaluation of multi-user beamforming in wireless lans. In *ACM MobiCom*, 2010.

[4] J. Bicket. Bit-rate selection in wireless networks. 2005.

[5] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *ACM SIGCOMM*, 2006.

[6] S. Gollakota and D. Katabi. Zigzag decoding: Combating hidden terminals in wireless networks. In *ACM SIGCOMM*, 2008.

[7] S. Gollakota and D. Katabi. Physical layer wireless security made fast and channel-independent. In *IEEE INFOCOM*, 2011.

[8] S. Gollakota, S. D. Perli, and D. Katabi. Interference Alignment and Cancellation. In *ACM SIGCOMM*, 2009.

[9] D. Halperin, T. Anderson, and D. Wetherall. Taking the sting out of carrier sense: interference cancellation for wireless lans. In *ACM MobiCom*, 2008.

[10] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *ACM SIGCOMM*, 2010.

[11] J. Heiskala and J. Terry. *OFDM Wireless LANs: A Theoretical and Practical Guide*. Sams Publishing, 2001.

[12] G. Holland, N. Vaidya, and P. Bahl. A rate-adaptive mac protocol for multi-hop wireless networks. In *ACM MobiCom*, 2001.

[13] Z. Ji, Y. Yang, J. Zhou, M. Takai, and R. Bagrodia. Exploiting medium access diversity in rate adaptive wireless lans. In *ACM MobiCom*, 2004.

[14] G. Judd, X. Wang, and P. Steenkiste. Efficient channel-aware rate adaptation in dynamic environments. In *ACM MobiSys*, 2008.

[15] T. H. Kim, J. Robert W. Heath, and S. Choi. The hidden cost of overhead in MIMO wireless LANs. Technical report, Feb. 2007.

[16] K. C.-J. Lin, N. Kushman, and D. Katabi. ZipTx: Harnessing partial packets in 802.11 networks. In *ACM MobiCom*, 2008.

[17] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[18] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *ACM SIGCOMM*, 2009.

[19] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *ACM MobiCom*, 2006.

[20] T. Yoo, S. Member, and A. Goldsmith. On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming. *IEEE J. Select. Areas Commun*, 24:528–541, 2006.