DOMINO: Relative Scheduling in Enterprise Wireless LANs

Wenjie Zhou†, Dong Li†, Kannan Srinivasan and Prasun Sinha
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{zhouwe, lido, kannan, prasun}@cse.ohio-state.edu
†Co-primary authors

ABSTRACT

Large-scale Enterprise WLANs are amenable to centralized control and coordination through the wired backbone for improved performance. Distributed scheduling algorithms either fail to achieve high performance in real deployments due to their myopic view of interference characteristics, or take significant time to converge to a globally optimal solution. Thus, they are not reactive to current network conditions. Centralized packet scheduling algorithms do not suffer from the performance limitations of distributed approaches, but are non-trivial to implement. Tight time synchronization requirements make proposed centralized schemes difficult to use in practice. This paper proposes *Relative Scheduling:* a technique for triggering wireless transmissions through other wireless transmissions in a domino-like fashion, thus making tight time synchronization unnecessary. Through USRP experiments and trace-driven simulations, we show that our approach can achieve up to 1.96× the throughput of Distributed Coordination Function (DCF).

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication, Network communications

General Terms

Design, Experimentation, Performance

Keywords

WLAN; Enterprise Networks; Relative Scheduling

1. INTRODUCTION

Owing to the increasing number of WiFi-capable devices, enterprise WiFi networks are becoming more prevalent in office environments, campuses, airports and malls. In addition, a number of cellular providers are deploying enterprise WiFi networks in congested areas at scale to offload the cellular traffic and thereby increase the capacity of the increasingly congested cellular networks [5]. This centralized structure has been leveraged for developing efficient solutions to various challenging problems such as channel assignment [11, 22, 26], client association [10, 11, 26], and power management [11, 24]. Another important problem in enterprise networks is channel access, which directly impacts critical parameters such as throughput and delay. Existing work in channel access can be broadly classified as follows:

- **Distributed Schemes:** Distributed Coordination Function (DCF) [1] defined in the IEEE 802.11 standard is the most widely used distributed channel access technique. Wireless devices pick a random back-off time and access the channel when the back-off timer expires. DCF has several advantages such as low implementation complexity, high scalability and robustness to failures. However, as each WiFi device makes distributed channel access decisions based on local carrier sensing, it is well known that DCF suffers from hidden and exposed terminal problems. Prior works [25, 38, 39] have shown that both of these problems are prevalent in real deployments. Some sub-optimum schemes [7, 14, 30] and throughput optimum schemes [34] have been proposed. *However, they either achieve asymptotic optimality or suffer from high-overhead and require of time synchronization.*

- **Centralized scheduling schemes (strict scheduling):** Centralized schemes are based on a central controller [9] that collects interference relationship among the nodes in the network and they make channel access decisions based on the status of all the queues in the system and the channel conditions. Although centralized schemes can achieve better performance than distributed schemes, there are two practical limitations. First, the central controller does not have information on the state of the queues at the clients. Thus, it assumes that the client always has data to transmit or it estimates the traffic load at the clients. As discussed later in Section 2, a naive solution based on piggybacking queue information, has a starvation problem. Second, the central scheduler assumes that all nodes can follow the schedules strictly. *In practice, jitter over the wired network limits the achievable time synchronization accuracy between the APs.*

- **Hybrid schemes:** Hybrid solutions like CENTAUR [38] and OmniVoice [6] schedule the downlink traffic from

CoNEXT'13, December 9-12, 2013, Santa Barbara, California, USA.

ACM 978-1-4503-2101-3/13/12 . . . $15.00.
http://dx.doi.org/10.1145/2535372.2535401.

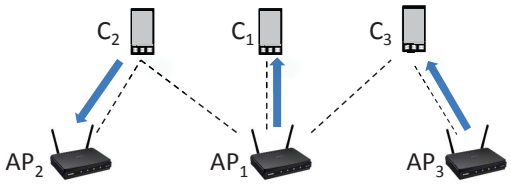


Figure 1: A network with three AP-client pairs. Dashed lines between nodes indicate that the nodes can hear each other. Solid arrows denote flow directions.

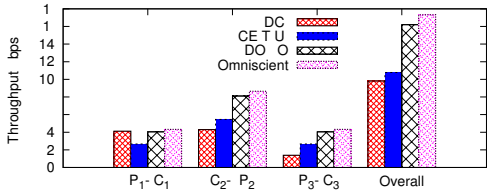


Figure 2: The throughput on different links. The overall throughput of omniscient scheme is 76% higher than DCF and 61% higher than CENTAUR. DOMINO performs close to the omniscient scheme.

the AP to the clients to avoid hidden terminals and utilize exposed terminal opportunities. *The uplink traffic still uses DCF to access the channel. The uplink traffic suffers from the same problems as discussed for the case of Distributed Schemes. Moreover, the disturbance created by uplink traffic to the downlink schedule can significantly diminish the performance gains [38].*

Although centralized schemes are difficult to implement in practice due to the reasons described above, they have the potential for very high throughput performance. We use the network shown in Figure 1 as an example. There are three AP-client pairs and three flows. AP_1 is a hidden terminal to AP_3 while C_2 and AP_1 are exposed terminals to each other. Therefore, DCF performs poorly on this network. Because of the hidden terminal problem, the link $AP_3 \rightarrow C_3$ achieves little throughput if all of the transmitters are backlogged. On the other hand, CENTAUR, a hybrid solution, schedules the $AP_1 \rightarrow C_1$ and $AP_3 \rightarrow C_3$ links to avoid the hidden terminal problem, but it is unable to schedule uplink traffic in the same slot, missing out on the opportunity for the exposed transmission $C_2 \rightarrow AP_2$. In an omniscient centralized scheduling scheme, the link $C_2 \rightarrow AP_2$ can always access the channel while the links $AP_1 \rightarrow C_1$ and $AP_3 \rightarrow C_3$ occupy the channel alternately. Figure 2 presents the throughput of different links using different scheduling algorithms.

A mechanism that can enable centralized schemes to work in practical networks can reap the benefits shown above. In this paper, we present *a framework for channel access in enterprise WLANs, called DOMINO, that can achieve the optimality of centralized schemes without depending on time-synchronization among APs.* Toward meeting this objective, we propose *Relative Scheduling*. In sharp contrast to *strict scheduling*, *Relative Scheduling* uses *wireless triggers* which are created by a set of PN-sequences transmitted by the sender and receiver at the end of a data packet transmission. The transmission events are triggered by previous events akin to a *domino* effect. In any given slot, a carefully chosen set of transmissions trigger the transmissions in the next slot. *Triggering using wireless transmissions is a new concept which is a complete paradigm shift from time synchronization-based protocol designs.* Besides, we extend the technique used in PAMAC [35] and introduce *Rapid OFDM Polling* (ROP), in which the clients use different OFDM subchannels to send their queue backlog information after receiving a polling request from the AP.

The contributions in this paper are summarized as follows:

- **Relative Scheduling:** This new paradigm of triggering wireless transmissions by other wireless transmissions has the following features: 1) it is able to work with any arbitrary centralized scheduling algorithm in a real network; 2) it does not require tight time-synchronization; 3) clients do not need to know the schedule in advance; 4) multiple triggers are used to increase robustness.
- **Experiments and Extensive Trace-driven Simulations:** Experimental results from our USRP testbed are used to derive simulation parameters. We use our USRP testbed to verify the ability of DOMINO to achieve better throughput. The simulation results show that DOMINO achieves up to 96% higher throughput compared to DCF.

The remaining part of this paper is organized as follows. Section 2 presents the motivation for relative scheduling in wireless networks. The design details of DOMINO are described in Section 3. We implement DOMINO in both a USRP testbed and ns3 and illustrate the evaluation results in Section 4. Section 5 presents the discussions followed by the related work and conclusion sections.

2. MOTIVATION

The basic assumption of strict scheduling scheme is that nodes in the network behave exactly according to the time schedule. However, this assumption does not hold without microsecond-level time synchronization. Several time synchronization schemes have been proposed for wired and wireless networks. In wired networks, existing work shows that the Network Time Protocol (NTP) [4] can achieve a time accuracy of about 1000 μs in a quiet Ethernet network. Given that a WiFi slot time is only 9 μs, this coarse synchronization is intolerable. The Precision Time Protocol [2] is designed to achieve microsecond level accuracy. However, it requires specialized and expensive hardware. In wireless domain, a recent work [31] achieves nanosecond level synchronization, which is enough for strict scheduling scheme. However, the synchronization has to be done for every transmission and is limited to a single collision domain. To the best of our knowledge, the most accurate time synchronization scheme over multiple collision domains is proposed only in RBS [15]. It uses reference broadcast on the wireless channel to achieve microsecond level synchronization. Although it achieves around 10μs accuracy in a 4-hop network, the performance decreases as the number of hops and the number of nodes in one hop increases, making it unsuitable for large and dense networks. In addition, since clock skew is influenced by the environment (e.g., temperature and supply voltage), this synchronization scheme has to be frequently executed to update the time. Extra hardware such as cellular networks [8] or GPS [28] could also be used to realize synchronization, which however increases the system cost and

Parameter	WiFi	ROP
number of subcarries	64	256
subcarriers per subchannel	–	6
guard subcarriers	–	3
number of subchannels	–	24
CP duration	0.8 μs	3.2 μs
symbol duration	4 μs	16 μs

Table 1: Parameters used for the OFDM symbol to convey the queue length of clients

complexity. In addition, GPS is not accurate in indoor environments. Besides precise time synchronization, the strict scheduling scheme requires collecting the queue status of all the nodes, and distributing the schedule to the clients.

Instead of using an absolute time tag with the scheduling decision, DOMINO uses a *relative* transmission tag. It notifies the next transmission to start at the end of the current transmission. Decoding the ongoing packet and estimating the stop point provides a naive solution for the relative tag. However, the relative tag fails when the decoding fails. This happens quite often in case of hidden transmission and multiple exposed transmissions. To address this issue, we use node signatures to enable our relative triggering scheme.

3. DOMINO DESIGN

This section presents the key components of DOMINO: Rapid OFDM Polling, Relative Scheduling and the mechanism for converting any schedule produced by an arbitrary scheduler to a schedule suitable for DOMINO. First, we discuss the method to identify hidden and exposed links.

Identifying hidden and exposed links: The central server requires the interference information between different links to calculate the schedule. Exposed links could transmit simultaneously while hidden links should be scheduled in different slots. In DOMINO, a central interference map consisting of the received signal strength between all node pairs, is maintained at the server. This map could be utilized to calculate the interference between different links and create a conflict graph $G(V, E)$ [19, 33]. The method used in [19] requires $O(N)$ steps to build the map, where N is the number of nodes in the network. Each node in V stands for a link (AP-client or client-AP pair). An edge in E indicates that those two links interfere with each other according to the interference map and should not transmit together. Thus, all of links that form an independent set in $G(V, E)$ could transmit simultaneously.

3.1 ROP: Rapid OFDM Polling

As there is no direct connection, it is non-trivial for the central server to collect queue status of the clients efficiently. One indirect solution is to use the AP to which it is associated with as a relay node and piggyback the queue status in packets sent to AP [42]. The disadvantage is that if the client stays silent for a period and then enqueues a new packet, the AP does not get to know about this new packet and the client may get starved. By taking advantage of Orthogonal frequency-division multiplexing (OFDM), PAMAC [35] obtains the queue status at all the clients through one polling action. However, PAMAC only obtains a coarse status of the queue, which is not sufficient for the central controller to compute the schedule. In addition, it does not consider the difference in the received signal strength from different clients. We introduce ROP in DOMINO to address these issues. Back2F [37] also has a similar scheme in using different OFDM subcarriers to convey channel contention information from different devices. However, Back2F studied the affect of self-interference (one subcarrier) on subcarrier detection, while we study the interference between subchannels (multiple subcarriers) from different clients. Besides, we design a complete AP-client polling system in ROP while Back2F focused on channel contention.

OFDM is a modulation scheme widely used in wireless communication. Instead of transmitting over a wideband, the system could be viewed as transmitting slowly on multiple independent narrow-band channels, called subcarriers. To confront multipath fading, a cyclic prefix (CP) is attached to each symbol. In 802.11 a/g, the 20MHz bandwidth is divided into 64 subcarriers, of which 48 are used to transmit useful data. One OFDM symbol takes 4 μs in 802.11 while the CP duration is 0.8 μs.

To obtain the queue status in DOMINO, the channel is separated into several subchannels, each consisting of several subcarriers. When a node is associated with an AP, a unique subchannel is assigned to it. In practice, there are several problems that affect this OFDM system. **First**, the frequency offset between clients and AP breaks the orthogonality between different subcarriers and causes inter-subcarrier interference. **Second**, the clients have to be synchronized and send at the same time. The return OFDM symbol from different clients should overlap at the AP for at least a duration of one FFT window. **Third**, since the analog-to-digital converter (ADC) at the AP has limited resolution, it could get saturated by the stronger signal. So the difference between the receive signal strength from different clients also affects the final decoding result.

Figure 4 shows the process of how DOMINO obtains the queue status from the clients. First, the AP broadcasts a polling packet. This packet contains a preamble that each client can use to tune the frequency offset. It also behaves as a reference broadcast to synchronize the clients. Then, after receiving this packet, each client waits for one standard slot time in WiFi (9 μs) and then transmits its queue size using the assigned subchannel. Because the distance between an AP and its clients varies, the signals from different clients reach the AP at slightly different times. However, the AP is still able to find a suitable FFT window using a large enough CP duration.

Instead of using the default values in WiFi, we use a different set of values for this special control OFDM symbol as shown in Table 1. Considering the maximum WiFi communication range to be 300 meters, the longest turnaround propagation delay is 2 μs. To account for this delay, the CP duration is chosen to be 3.2 μs. Each subchannel contains 6 subcarriers, to encode a maximum queue size of $63(= 2^6 - 1)$ if binary phase-shift keying (BPSK) is used as the modulation. When there are more than 64 packets in the queue, we can report 63 first packets and keep track of the number of unreported packets. Moreover, to prevent interference between different subchannels, several subcarriers are used to create a guard interval. Our experiments at the end of this section show that 3 subcarriers are enough to tolerate a mismatch of up to 38 dB in the received signal strength from different clients. A total of 24 subchannels are available for the AP to assign. Because of DC offset, the center

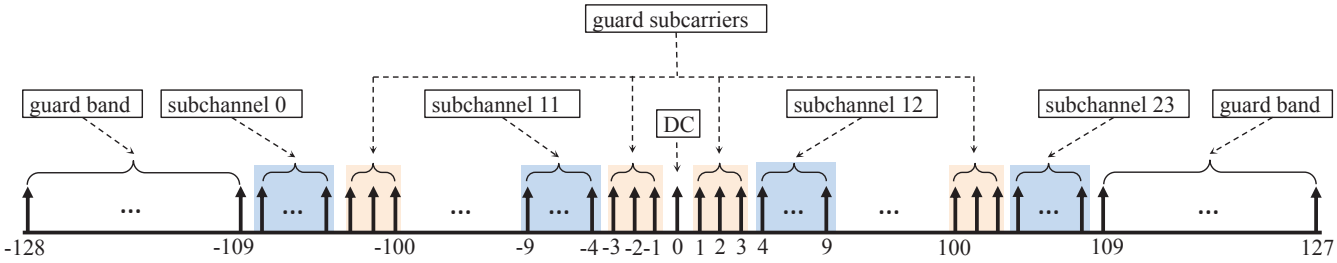


Figure 3: The construction of one OFDM symbol

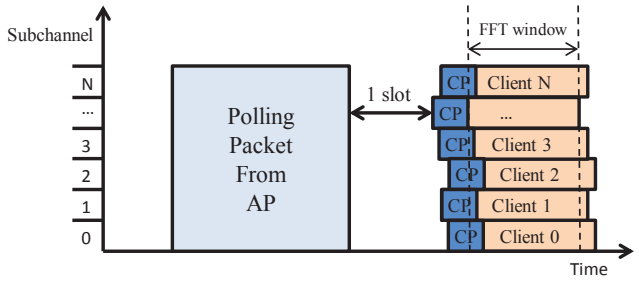


Figure 4: The process of obtaining queue status from clients.

subcarrier is not used. The remaining 39 subcarriers are used as guard band between different wireless channels as in 802.11 (11 out of 64 subcarriers are guard band). Figure 3 shows the details of how the control OFDM subcarriers are assigned.

To study the performance of ROP, we implement the mechanism shown in Figure 4 on GNURadio [3] and test it using a USRP testbed. Because only one OFDM symbol is sent back to the AP, it is difficult to estimate the phase offset. Thus, 2-amplitude-shift keying (2ASK) is used for modulation instead of BPSK since phase offset does not affect the amplitude of the samples.

Figure 5(a) plots the result of two clients sending on adjacent subchannels with similar received signal strength (RSS). The bits sent on subchannel 1 are 111111, while the bits on subchannel 2 is 011111 where the first bit is set to 0 to show the interference between different subchannels. This figure shows that all the bits on both the subchannels are received correctly. When there is 30 dB difference in the RSS from the two subchannels, the decoded samples are plotted in Figure 5(b). The sequence 111111 is sent on both the subchannels. However, the first three subcarriers of subchannel 2 are affected by the strong signal on subchannel 1. Figure 5(c) shows the result after separating them by three subcarriers. It reveals that that separation helps in reducing the interference between different subchannels. Figure 6 shows the relationship between difference in RSS and the number of guard subcarriers. A separation of three subcarriers is shown to be sufficient as long as the RSS difference is no more than 38 dB. We measured the RSS between different nodes in a testbed with 40 wireless nodes. Only 0.54% of all link pairs have an RSS difference greater than 38 dB, which indicates that separation of 3 subcarriers is enough for most cases. In the extreme case where the RSS difference from two clients is indeed higher than 38 dB, the AP should assign them non-adjacent subchannels to relieve the interference. Another interesting question is – *how low the signal-to-noise ratio (SNR) could be for reliable detection?*. Experiment results reveal that as long as the SNR is higher than 4 dB, an OFDM symbol can be decoded correctly. Studies show that the SNR should be at least 4 dB to allow reliable WiFi transmission with the lowest data rate of 6 Mbps [29]. This set of experiments proves that the design of using one OFDM symbol to extract the queue information from the clients is effective.

3.2 Relative Scheduling

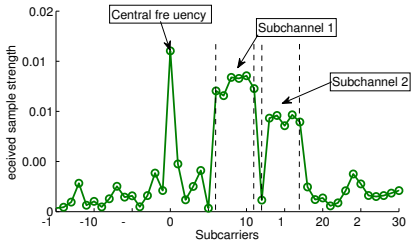
The schedule calculated by the central server requires microsecond level synchronization between the APs. Otherwise, the slots will overlap and degrade throughput. DOMINO takes advantage of the broadcast nature of wireless communication and introduces a novel concept called *relative scheduling*, which removes the need for tight synchronization.

To show the basic idea of relative scheduling, we use a network with 4 AP-client pairs shown in Figure 7 as an example. Let's consider downlink traffic only. Figure 7(c) presents one possible schedule according to the conflict graph in Figure 7(b). In relative scheduling, this scheduling decision is turned into two chains:

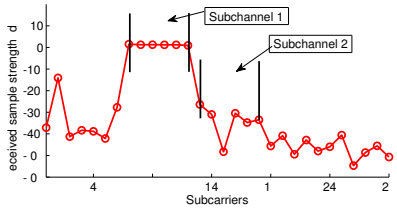
$$\text{Chain1}: AP_1 \rightarrow C_1, AP_2 \rightarrow C_2, AP_1 \rightarrow C_1, AP_2 \rightarrow C_2$$
$$\text{Chain2}: AP_4 \rightarrow C_4, AP_3 \rightarrow C_3, AP_4 \rightarrow C_4, AP_3 \rightarrow C_3$$

The transmitters in the same chain transmit in sequence. The central controller does not need to give the exact time when a link should be active. Instead, it informs that a link should be *relatively* activated after another one.

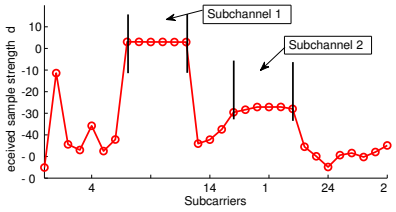
In Figure 7, the signals from AP_2 and AP_3 collide at AP_1, making it impossible for AP_1 to correctly decode the packet. Thus, AP_1 is unable to detect which link is currently active in the relative chain. To trigger the transmission of the next packet, DOMINO utilizes node signatures, a concept which has been used in several recent works [23, 36, 41] for various types of control signaling. At the end of each transmission, the signature of the next transmitter is explicitly appended. Instead of trying to decode the ongoing packet and listening for the end of this transmission, each node keeps on running a correlator for its own signature and starts transmitting once the signature is detected. Signatures could be decoded under high interference, which makes it possible for AP_1 to receive the notification from AP_2, even when the packet cannot be decoded correctly. Here, we assume that only the transmitter that has the maximum RSS at the next transmitter is in charge of sending the signature. If the network topology is changed with AP_4 and C_4 removed from Figure 7(a), AP_1 has to inform both AP_2 and AP_3 to start sending



(a) The decoded OFDM samples at the AP with 2 clients using adjacent subchannels without any guard interval. The received signal strength from both clients are similar.



(b) The decoded OFDM samples at the AP with 2 clients using adjacent subchannels without any guard interval. There is a 30 dB difference in the received signal strength from the 2 clients.



(c) The decoded OFDM samples at the AP with 2 clients using adjacent subchannels with guard interval. There is a 30 dB difference in the received signal strength from the 2 clients.

Figure 5: Received OFDM samples from two clients. The system can tolerate 30 dB signal strength difference when using three subcarriers as guard interval.

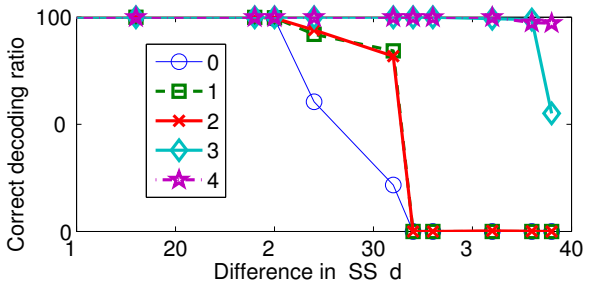
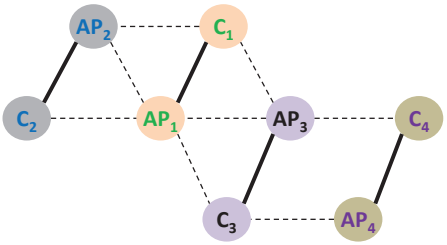
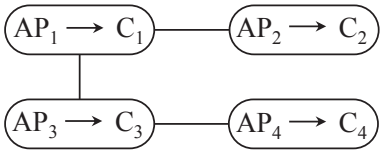


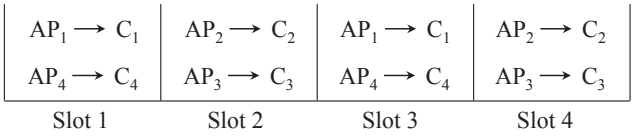
Figure 6: The relationship between number of guard subcarriers and the difference in RSS



(a) Network topology (Dashed line indicates that the two nodes interfere with each other, while solid line denotes AP-client association)



(b) Conflict graph of the downlink traffic



(c) One possible schedule of the downlink traffic

Figure 7: A network with four AP-client pairs.

at the end of its transmission. In this case, AP_1 sends the sum of AP_2 and AP_3's signatures. Because different signatures are orthogonal to each other, they are still detectable.

Gold codes [17], because of their outstanding cross correlation property, are used as the node signatures. Longer Gold codes have higher difference between self-correlation and cross-correlation values, thus increasing the robustness and providing more signatures, albeit with higher overhead. In DOMINO, we use a set of 129 Gold codes with length 127. With 20 MHz bandwidth and BPSK as the modulation scheme, it takes 6.35 μs to transmit one signature. As will be mentioned later, two codes are reserved for special use. Thus, the system can support up to 127 nodes in one collision domain. Note that the signatures could be reused across different collision domains. Since there is a central controller in the system, we assume that every wireless node will be assigned a unique signature when it joins the network.

In the above discussions, it is assumed that the next transmitter receives the signature from the previous transmitter. However, only the AP obtains the schedule from the central controller. So if a client is the current transmitter, it will not know the next transmitter. Another issue to note is that notifying a hidden terminal (AP_3 and AP_4 in Figure 7) to start transmission is not easy. To solve these problems, both the transmitter and receiver send out the signatures of its surrounding nodes at the same time. The mechanism used by the AP to inform the client the signatures to send is divided into two cases as shown in Figure 8. When the AP is the transmitter (Figure 8(a)), it transmits the samples of the signature that the client should send (S_1) at the end of the packet. When the AP is the receiver (Figure 8(b)), it transmits S_1 at the end of the ACK. In either case, the client stores those samples. One slot after the ACK, both the AP and clients transmit the signatures together. The signatures sent by the AP and client should be different because they only need to inform their own surrounding next transmitters. The one with stronger RSS is responsible for sending the signatures of the nodes in the common area. To distinguish the signatures sent from the AP to the client and the final signatures broadcast to the next transmitter, a special signature, the START signature (S^prime) is sent after the latter one. So, the next transmitters could start transmitting only if both its own signature and the START signature are detected in sequence.

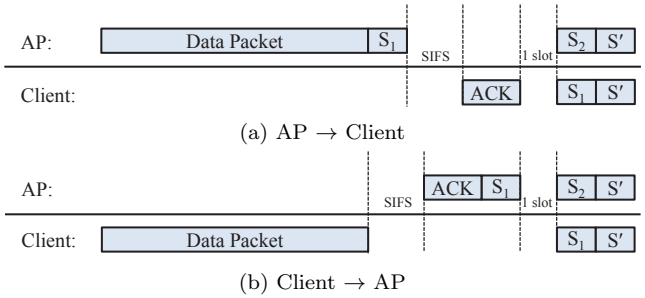


Figure 8: The timeline of the transmission between AP and client. S_1 is the signature that should be sent by the client, S_2 is the signature that should be sent by the AP, and S' is the START signature.

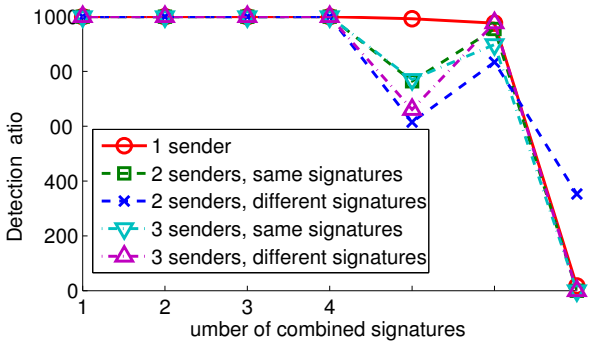


Figure 9: Detection ratio of multiple signatures

In practice, a packet received incorrectly prevents the sending of signatures. Also sometimes signature detection fails. If any of these happens, the chain is broken and all of the following transmitters would not be able to transmit. To make the system robust, we add cross-links between different chains so that one transmitter could get the notification from multiple nodes. Thus, backup notifications are created and could be used if the normal notification is ineffective. However, it is possible that the signatures from different triggering transmitters collide at the next transmitter.

Using USRPs, we studied how many signatures can be added together and yet received correctly even in presence of interference using USRP. Five different experiment setups are evaluated. In the first setup, there is only one transmitter and one receiver. In the second and third setups, there are two transmitters with similar RSS at the receiver. Note that DOMINO chooses the top two nodes with the highest RSS as the triggering nodes. So, our choice of picking two transmitters with similar RSS is the worst case as they result in the highest interference to each other. Both of the transmitters send the same signatures in the second setup. We used different signatures in the third setup to study the interference of non-correlated signals. The fourth and fifth setups are similar to the second and third one except that there are three senders. Figure 9 plots the result of signature detection ratio from 1000 runs. The signature detection ratio is nearly 100% in all experiments when the total number of combined signatures is no more than 4 and the false positive ratio is below 1% all the time (not shown in the figure). DOMINO uses 4 as the maximum number of signatures to combine.

3.3 Schedule Converter

The third component of the central server is the converter, which is a series of procedures that convert a strict schedule made by an arbitrary scheduler to a relative schedule. Before introducing the converter, we define a few terms. The sender of a link l is denoted by $l.sender$ and the receiver by $l.receiver$. Link l is either an uplink or a downlink. Thus, either $l.sender$ or $l.receiver$ must be an AP, which is denoted by $l.ap$. For a node n, link l could *trigger* n if and only if the signature sent by $l.sender$ or $l.receiver$ can be received by node n. Link l_1 could trigger link l_2 if and only if l_1 can trigger $l_2.sender$. The $l.inbound$ is the number of signatures that $l.sender$ receives. Larger $l.inbound$ is more robust to transmission failures because it indicates that more links could trigger l. However, larger inbound also reduces the signature detection reliability. Therefore, the maximum inbound is set to 2. The *outbound* is the number of signatures that a node could broadcast, which indicates the number of links the node could trigger. The experiment result in Section 3.2 suggests that the maximum outbound for each node should be 4.

A strict schedule with k time slots can be denoted by $S = [s_1, s_2, \cdots, s_k]$, where s_i is a set of links that can transmit concurrently in slot i. In a strict schedule produced by an arbitrary scheduler, there is no guarantee that links in $s_i, i \in [1, \cdots, k-1]$ can trigger all the links in s_{i+1}. On the other hand, a relative schedule requires that the links in slot s_{i+1} can be triggered by at least one link in s_i. To satisfy this requirement, the converter uses the following two techniques:

Fake link insertion: For each slot $s_i \in S$, we insert all the links that are not conflicting with the links in s_i to create a maximal cover in the link conflict graph. The inserted links are marked as fake links. The purpose of adding the maximal number of fake links is to keep all links in the network being triggered frequently to synchronize with the rest of the network. When a node is indicated to send a packet to destination d by the schedule or the received signature, but the node has no packet for d in its queue, the node will send a fake packet to d. Note that a node only need to send the header of the fake packet, instead of sending the entire fake packet. Thus the interference introduced by these fake packets will be mitigated. Also note that the WiFi chip will not consume more energy when sending fake packets than idle listening [41].

Batch connection: Since the strict scheduler creates schedules in batches, we also need to create the triggering connection between two neighboring batches. After the current relative schedule is created, the last slot of the relative schedule is retained in the converter, and will be used as the first slot of the next batch. The exception is that in the very first batch, the first slot has no preceding slots and therefore cannot be triggered by any links. In this case, the APs will individually start executing the schedule. If the link in the front of the schedule is a downlink, then the AP will send a packet according to the schedule. Otherwise, the AP will send a signature to the sender of that link. Because the APs are not synchronized, collisions could happen. However, we will show that relative scheduling heals itself and

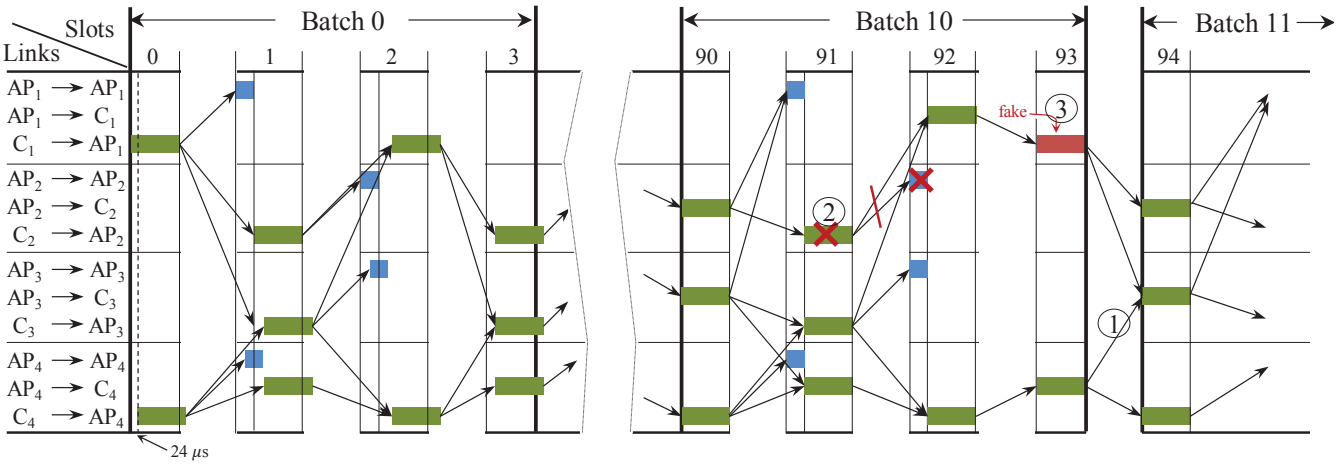


Figure 10: The timeline of the network in Figure 7. The link $AP_i \rightarrow AP_i$ stands for polling operation. The arrow between different links indicates the triggers between different slots.

synchronizes the schedules between different APs within a few slots in Section 4.2.2.

After inserting fake packets into the strict schedule, we are ready to create the triggering relation between the links in the neighboring slots. Given two neighboring slots s_i and s_{i+1}, the objective is that for each l in s_{i+1}, find at least one link in s_i that can reliably trigger l, while the inbound and outbound constraints of each link is satisfied. For each link l in s_{i+1}, we first select one node n in s_i, such that n has the highest SNR at $l.sender$. After assigning one trigger to each link in s_{i+1}, links with saturated inbound or outbound links are excluded in the following steps. Then, we repeat the previous step on the remaining nodes to find the secondary possible triggering node for each link in s_{i+1}. This process stops when no more trigger can be added. Note that even after inserting fake links, it is still not guaranteed that each link in s_{i+1} has a triggering link in s_i. Such a scenario happens rarely in our experiments. The scheduler will reschedule such links.

The last step is to insert ROP slots into the relative schedule. The duration of the ROP slot is the time needed for an AP to execute the ROP protocol. During an ROP slot, all links that interfere with the links associated with the polling AP must be silent. So, two APs can share the ROP slot if and only if none of their links are conflicting. We insert at most one ROP slot between two neighboring slots. If an ROP slot is inserted between slots s_i and s_{i+1}, the links in s_i will trigger links in s_{i+1} with a special signature, which is called an ROP signature, instead of the signature S' as shown in Figure 8. Once links in s_{i+1} receives the ROP signature, they will wait for one ROP slot before starting transmission. The ROP slots are inserted into the relative schedule in a greedy fashion. To insert ROP schedule for AP A into relative schedule $S = [s_1, s_2, \cdots, s_k]$, we first check if s_i has nodes that can trigger A. If s_i can trigger A, and there is no ROP slot that has been inserted between s_i and s_{i+1}, then we insert an ROP slot between s_i and s_{i+1}. Otherwise, we check if A can poll together with the APs in the existing ROP slot. If they can execute ROP together, assign the ROP schedule for A in the existing ROP slot. Otherwise, increase i and continue to check the next slot in S.

At the end, each link l in the created relative schedule will be distributed to AP $l.ap$. During executing the relative schedule, the AP sends data packet or executes ROP according to the schedule upon reception of its own signature, and sends appropriate signatures to its clients to trigger the senders in the next slot. Each client sends data packets when receiving its own signature; triggers other links based on the signatures received from its AP; and, returns queue size to its AP when a polling packet is received.

3.4 DOMINO Under Microscope

To take a closer look at the overall system, we again use the example shown in Figure 7. However, all the uplink and downlink flows are saturated with payloads. Figure 10 presents the transmission timeline from our trace-driven simulation. Because of the jitter in the wired backbone, the packets in slot 0 are actually sent with a 24 μs time difference. This delay passes to slot 1 and slot 2. However, in slot 2, link $C_1 \rightarrow AP_1$ receives two triggers from link $C_2 \rightarrow AP_2$ and $C_3 \rightarrow AP_3$ (this can be decoded as C_1 is waiting for a polling slot). Since the transmitter uses the last correctly received trigger as time reference, it gets synchronized in transmission with link $C_4 \rightarrow AP_4$. The transmission in the following slots are then all synchronized. This result reveals the robustness of DOMINO to time jitter.

We marked some interesting points in the figure to emphasize the design of DOMINO. ❶ indicates the trigger from link $AP_4 \rightarrow C_4$ to $AP_3 \rightarrow C_3$. As shown in Figure 7, AP_3 and AP_4 are hidden to each other. ❶ presents an instance where the receiver of the former transmission triggers the next transmitter. At ❷, we assume this transmission fails. Thus, the following two triggers are missing. However, as a result, only one polling transmission is missed, which indicates that the effect of transmission failure in DOMINO is limited. ❸ presents the introduction of fake packets to increase the network coverage of triggers[1]. Link $AP_2 \rightarrow C_2$ in slot 94 could not be triggered without this fake packet.

3.5 Practical Issues

[1]Note that fake packets are also scheduled by the central server.

Scheme	SC	HT	ET
DOMINO (Kbps)	4.25	5.42	9.18
DCF (Kbps)	2.76	1.62	2.72

Table 2: Aggregate throughput in 3 different scenarios with USRP prototype

Although we take care of many practical issues in the design of DOMINO, there are still some that require further discussions:

- **Different packet sizes and data rates:** In our design, we use a fixed slot time and assume that all the data packets consume the same amount of time, which will likely not hold at all times in practice. However, techniques, such as packet splitting and aggregation, could help to produce virtual packets that take the same amount of time. A simple calculation with the fixed packet duration, packet size and data rate will suffice. Then, instead of reporting how many packets are queued to the central sever, wireless nodes calculate and forward the total number of virtual packets.
- **Number of clients per AP could support:** The wireless channel is divided to 24 subchannels, which limits the total number of clients per AP. In case the number of clients is more than 24, we could divide the clients into different sets, with each set with no more than 24 nodes. And then the AP could poll once for each set.
- **Missed ACKs:** When the ACK for packet p is missing, the sender adds a new transmission request. Instead of waiting for the request to be scheduled again by the server, the sender will retransmit as long as the following two conditions are met: 1) the sender is a client and it receives its trigger; 2) the sender is an AP and the schedule at the top of the schedule list has the same destination as $p.receiver$.

4. IMPLEMENTATION AND EVALUATION

4.1 Implementation and Experimentation

We implemented a simple version of DOMINO on USRP and compared its performance with DCF. We assume that the queue in the clients are saturated and the transmission schedules are already loaded in each AP. Four USRPs are used to simulate two AP-client pairs and the flows on two links are created. Then, we studied the aggregate throughput in three different scenarios: (i) those two links are exposed to each other (ET); (ii) they are hidden links to each other (HT); (iii) they are in the same contention domain and are neither exposed or hidden to each other (SC). Table 2 shows the aggregate throughput. Because DOMINO does not incur the overhead of backoffs, it achieves 54% higher throughput even in the setting SC. In case of hidden and exposed terminals, DOMINO obtains more than 3× the throughput of DCF. As there exists a significant latency variance between the USRP and the host, implementing CENTAUR's carrier sensing mechanism to align exposed transmissions using USRP devices is difficult. So, we leave the comparison of DOMINO and CENTAUR to trace-driven evaluations.

4.2 Large Scale Evaluation

The above prototype with USRP shows the potential advantages of DOMINO. However, the limited number of USRP platforms and the latency with USRP prohibits a large scale implementation. Thus, we perform measurements in a network of 40 WiFi nodes spread across 2 buildings and use the RSS trace between different nodes to conduct a large scale evaluation in ns3.

4.2.1 Evaluation Setup

Let's denote $T(m, n)$ as a network topology with m APs and n clients per AP. To create $T(m, n)$, we first sort the nodes from our trace by the number of nodes in their communication range in a decreasing order. Then, we select the first node as one AP, and randomly pick n nodes that could communicate with the AP as clients. We repeat this process to select the remaining $(m-1)$ APs and their clients.

Unless specified, the default traffic (UDP or TCP) rate for uplink and downlink is 10 Mbps. The evaluation results are based on a run of 50 seconds. The physical layer data rate is set to 12 Mbps and the data packet size is 512 Bytes. Wired connections are created between the APs and the central server. The latency on the wired connection is set following a normal distribution with mean 285 μs and variance 22 μs according to [38].

In DOMINO, we use the scheduler modified based on RAND [32], a greedy algorithm. To calculate the schedule for each slot, the first link l from the queue of links Q that has data to send is added to a set $C(l)$. Then we add another link l' from $Q - C(l)$ to $C(l)$ if l' is not conflicting with any link in $C(l)$. This adding process is repeated until no more links could be added. All the links in $C(l)$ are then scheduled in this slot. To improve the fairness, we move the links in $C(l)$ to the end of Q. The following slots are scheduled in the same way.

4.2.2 Time for transmissions to reach synchronization

Because of jitter over the wired network, the transmissions for slot 0 may not be well aligned. We use the network $T(10, 2)$ to study how long it takes for the misalignment to converge. The result is presented in Figure 11. The wired latency variance is changed from 20 μs to 80 μs. The figure shows that although the maximum misalignment varies from 10 μs to 20 μs, it is reduced to 1 or 2 μs within 4 slots. This result indicates that the our scheme does not need a long warm-up time and also it is robust to initial misalignment.

4.2.3 Throughput and fairness

To evaluate and compare the throughput and delay between DOMINO, CENTAUR and DCF, we still use $T(10, 2)$ with the downlink data rate fixed to 10 Mbps. The uplink data rate varies from 0 to 10 Mbps. There are 10 hidden link pairs and 62 exposed link pairs out of 720 possible link pairs. For both CENTAUR and DCF, the MAC parameters are set according to 802.11g standard. The throughput fairness among all links is calculated using the Jain's fairness index [18]. As shown in Figure 12(a), DOMINO outperforms DCF by 74% when there is only downlink UDP traffic. Although the throughput gain decreases to 24% as the uplink UDP traffic data rate increases, DOMINO has high fairness around 0.78, compared with 0.47 for DCF (Figure 12(c)). For TCP traffic, the throughput gain of DOMINO varies

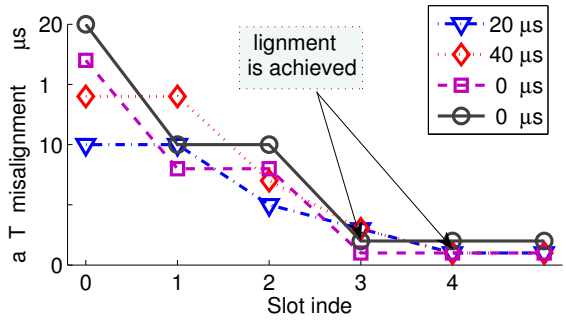


Figure 11: Maximum transmission misalignment at the start of contention free period

Topology	DOMINO	CENTAUR	DCF
Figure 13(a) (Mbps)	32.72	28.60	9.97
Figure 13(b) (Mbps)	33.85	18.35	22.13

Table 3: Aggregate throughput with 4 pairs of exposed links

within 10% to 15% (Figure 12(d)), while the fairness gain is between 17% and 39% (Figure 12(f)). The reason why TCP traffic does not produce as high throughput gain as UDP is that we treat the TCP ACK packet as a regular data packet and it takes one whole slot to transmit the TCP ACK, which wastes the channel resource. We believe that aggregating TCP ACKs will help improve the gain, but leave it as a future work.

Figure 12 surprisingly shows that the performance of CENTAUR is worse than DCF when the uplink data rate is low. We looked deep into the simulation results. Although CENTAUR has 0 ACK timeout compared with 57386 times for DCF when the uplink data rate is 0 Mbps, CENTAUR is sensitive to network topology and behaves worse than DCF when the scheduled downlink traffic is not transmitted in the way it is supposed to be. The assumption that exposed links transmit concurrently just does not hold in some network topologies. To illustrate this argument better, we use the topologies shown in Figure 13 as examples. Assume that only downlink traffic exists. In both examples, all of the downlinks are not conflicting with each other and are scheduled in the same batch as in CENTAUR. However, AP_1, AP_2 and AP_3 are not in the communication range of each other in Figure 13(b). Since CENTAUR uses carrier sensing and fixed back-off intervals to synchronize the transmissions, the transmissions could not be synchronized. Thus, in each batch, AP_1, AP_2 and AP_3 have higher chance than AP_4 to access the channel and finish transmitting earlier. But the scheduled packets for the next batch would not arrive until all of the packets at AP_4 are sent. In DCF, however, AP_1, AP_2 and AP_3 always have packets in their queues and keep contending for the channel. Table 3 presents the aggregate throughput for both topologies. For Figure 13(a), the throughput of both DOMINO and CENTAUR are around 3× the throughput of DCF. However, the throughput of CENTAUR is lower than DCF for Figure 13(b) while DOMINO provides the same throughput in both scenarios.

4.2.4 *Delay*

Figures 12(b) and 12(e) plot the average delay of different schemes. The delay is defined as the duration from the time a packet is queued to the time it is successfully delivered. The delay of DCF is 2× higher than DOMINO. Because the UDP traffic data rate is high, the MAC layer queue gets saturated quickly. So queuing delay significantly contributes to the packet delay. DOMINO promises higher throughput, which means that packets get delivered faster. The packet delay for TCP traffic is shown in Figure 12(e). Because TCP congestion control, the MAC queue increases slower than UDP traffic. On the other hand, the congestion control window size grows faster for DOMINO than for DCF. So higher throughput indicates faster packet delivery as well as more queued packets. These two factors have opposite effects on the packet delay, resulting in comparable packet delay for DOMINO and DCF.

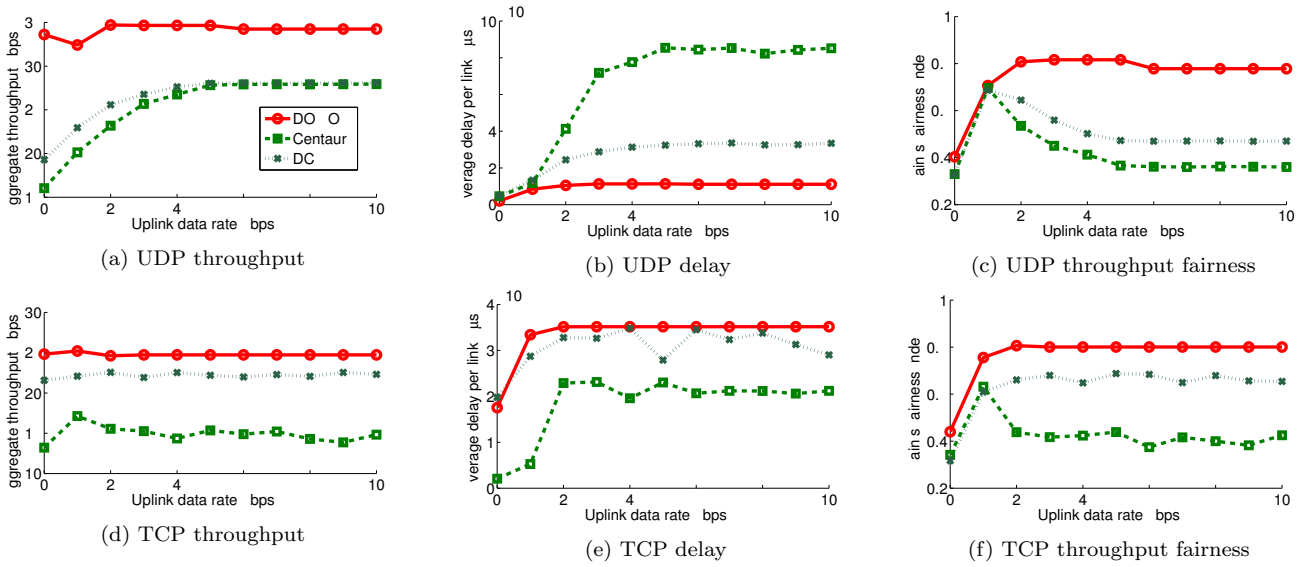
4.2.5 *Simulation with a random network*

The above trace only consists of 40 nodes, which limits the scale of the network that we can simulate. In this section, we uses the default path loss model in ns3 to compute the RSS between different nodes instead of manually setting the RSS from the trace. We randomly placed nodes in an 800×800 m^2 area and create a topology $T(20, 3)$, which consists of 80 nodes. We repeat the simulation for 50 times with UDP traffic and plot the CDF of the throughput gain of DOMINO over DCF in Figure 14. The throughput gain varies from 22% to 96% with a median of 58%.

5. DISCUSSION

Building conflict graph dynamically: In our current design, we assume that the conflict graph does not change over time, which does not hold in mobile scenarios. Updating the conflict graph with low overhead remains a challenge. Kashyap et al. [19] have provided a scheme that updates the conflict graph of a network with time Nt, where N is the number of nodes in the network, and t is the time of sending one beacon packet. Since non-interfering nodes could send the beacons concurrently, the time complexity could be reduced to $t(\Delta + 1)$, where Δ is the maximum degree of the two-hop connected graph. The two-hop connected graph is created by connecting any two vertices that are within two hops in the interference graph. The interference graph can be estimated based on previous known channel status. Fu et al. [16] have shown that in the 2.4 GHz spectrum, the channel coherence time of walking is 125.1 ms, which is the maximum conflict graph updating period. Therefore the overhead of periodically generating the conflict graph is $t(\Delta + 1)/(125.1\ ms)$. When $\Delta = 40$ and each beacon takes 40 μs, the overhead is only 1.3%, which is negligible compared with the throughput gain.

Co-existence with current networks: Enterprise networks may be subject to external interference such as from external WiFi networks. To co-exist with existing networks, DOMINO divides time into two parts: a centralized contention free period (CFP) and a carrier sensing contention period (CoP) as shown in Figure 15. The contention free period contains several slots and supports concurrent transmissions in each slot. To reserve the channel during this duration, we set the Network Allocation Vector (NAV) duration to the end of the CFP period in the MAC header of each transmitted packets. External nodes have to defer their transmission upon receiving the NAV. In the contention pe-



(a) UDP throughput

(b) UDP delay

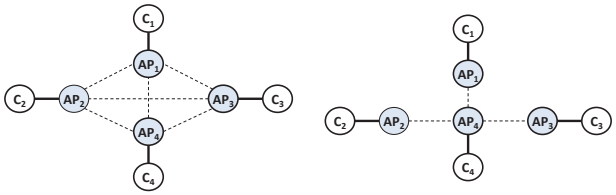
(c) UDP throughput fairness

(d) TCP throughput

(e) TCP delay

(f) TCP throughput fairness

Figure 12: TCP and UDP throughput, delay and fairness of $T(10, 2)$. The downlink data rate is fixed to 10 Mbps and the uplink data rate varies from 0 to 10 Mbps.



(a) 4 links that are exposed to each other

(b) 3 links that share a common exposed link

Figure 13: Exposed links example. Dashed links indicate nodes are interfering with each other and solid links denote AP-client pair.

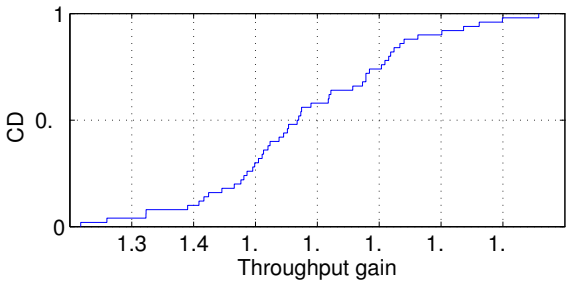


Figure 14: The CDF of throughput gain of DOMINO over DCF with 50 runs

riod, all nodes use carrier sensing to access the channel. The server estimates the amount of external traffic and internal traffic during the contention period, and adjusts the durations of the following CFP and CoP to provide fair access to all traffic.

Light traffic load: DOMINO improves the throughput of the network under heavy traffic. However, with light data arrival rate, the throughput gain will not be high and the control overhead increases the packet delay. In network topology $T(6, 5)$ with traffic rate 6 KBps (this is lower than typical web browsing, considering that the home page of Yahoo! is around 1.9 MB), the delay of DOMINO is only 1.14× higher than the delay of DCF, which is not extremely high. In addition, we can utilize the CFP and CoP duration as discussed above, to solve this problem. Under light traffic, we set CFP duration to 0 to turn off scheduling.

Energy saving: It is straightforward to implement energy saving mechanism in DOMINO. For example, the server can schedule an energy constraint device to sleep for a duration within which it does not need to send or receive packets.

Number of signatures: DOMINO uses signatures with 127 bits and supports 127 nodes in one collision domain. To support mores nodes in one collision domain, there are several choices. First, instead of using 127 bits as the signature length, we can use 255 and 511, supporting 255 and 511 nodes in one collision domain respectively. Second, the combination of those 127 signatures could be used to identify one node. Both of the choices results in larger signature duration, which increases the overhead. So an algorithm to estimate the node density is required to choose the best signature length.

Polling frequency: Currently, DOMINO polls the queue information of the clients in every batch, which may result in a waste of channel resource. Intuitively, the APs should not send polling packets when the scheduler has enough packets to send. However, this could cause starvation at some clients. We run a set of simulations to evaluate the delay and throughput of UDP traffic in $T(10, 2)$ when varying the batch size (the reciprocal of polling frequency). The simulation results show that when the network traffic is heavy (5Mbps per link), as the batch size increases, the delay slightly decreases and the throughput slightly increases. However, when the traffic is light (500Kbps per link), the delay increases when the batch size increses. We leave the design of a better polling scheme as future work.

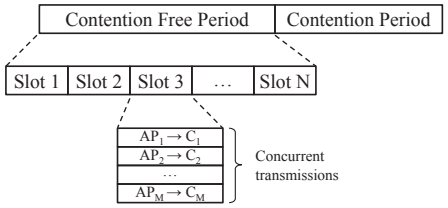


Figure 15: DOMINO consists of contention free period (CFP) and contention period (CoP). The CFP is divided into different slots and each slot supports multiple concurrent transmissions.

6. RELATED WORK

WiFi centralized control plane schemes: Existing work, such as [10, 22, 24], focused on the channel assignment, client association and power control. MDG [11] studied the relationship between three different functions with centralized solution: channel allocation, load balancing and power control and proposed a joint framework of different schemes. DenseAP [26] introduces the idea of dense AP deployment. A central controller determines the AP for each client to be associated with dynamically and the channel to assign to each AP. A network architecture and a set of APIs are defined in Dyson [27] to provide an easy way for network observation and implementing control polices. These papers focused on the control plane of centralized network, while our proposed scheme targets channel access.

Centralized and hybrid data plane schemes: PCF, defined in the 802.11 standard, is proved to be promising in supporting real-time traffic [13] with a single AP. MIFI [9] extends the use of PCF to multiple AP scenario. In the contention-free period (CFP), non-interfering APs polls the traffic simultaneously. This work, however, creates more exposed links originating from their definition of non-interfering APs. CENTAUR [38] proposed a hybrid centralized scheme with the assumption that the user devices will remain unmodified. The downlink traffic are scheduled according to the exposed and hidden relationships between different links. However, the unscheduled uplink traffic disturbs the performance in an unpredictable way. DOMINO does not suffer from a similar problem because it controls all the traffic, including both uplink and downlink. OmniVoice [6] only schedules downlink traffic. It divides time into uplink and downlink slots so that the uplink traffic does not disturb the downlink schedule. It is designed for VoIP applications in a small network. However, the synchronization accuracy degrades with increase in the size of the backbone network. XPRESS [21] uses backpressure algorithms to obtain optimal throughput in multi-hop wireless networks. It could be extended to enterprise networks. However, it also only schedules downlink traffic.

Clients queue status: To obtain queue information of the clients, some recent works have leveraged the fact that there is room between the channel capacity and real data rate so that the ongoing transmission could tolerate some interference. Side Channel [40] focused on ZigBee networks, and used interfering signals on different chip positions or different chip intervals to convey the request for uplink transmission. Flashback [12], on the other hand, worked on the OFDM system. Interfering signals, called flashes, are sent on a given frequency and the frequency interval between the adjacent flashes encodes the clients queue information. These two techniques promise high performance in a network with a single AP. However, with multiple APs and exposed transmissions, the benefits of side channels or flashes decreases. In [20], each client is assigned a unique bit sequence, named transmission request. The sequences assigned to different clients are orthogonal to each other so that these requests can be detected by the AP. This method suffers from different received power from the clients and it only transmits 1 bit information back to the AP.

7. CONCLUSION

In this paper, we proposed *Relative Scheduling*, which allows wireless nodes to transmit relatively one after the other. The use of node signature as transmission trigger in relative scheduling is also verified using the USRP platform. Then, we developed DOMINO, a centralized scheduling framework for enterprise WLANs based on the concept of Relative Scheduling. Our evaluation results show that DOMINO significantly outperforms DCF by up to 96% higher throughput.

Acknowledgements

We thank the anonymous reviewers and our shepherd, Bozidar Radunovic, for their valuable feedback that helped us improve the paper.

References

[1] IEEE Std. 802.11-2007, part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications, June 2007.

[2] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages c1–269, 2008.

[3] GNU Radio, accessed Jan 2013. `http://gnuradio.org`.

[4] Network Time Protocol, accessed Jan 2013. `http://www.ntp.org/`.

[5] Facts tagged with WiFi, accessed Mar 2013. `http://www.factbrowser.com/tags/wifi`.

[6] Nabeel Ahmed, Srinivasan Keshav, and Konstantina Papagiannaki. OmniVoice: a mobile voice solution for small-scale enterprises. In *Proc. of ACM MobiHoc*, pages 5:1–5:11, 2011.

[7] Eyjolfur Ingi Asgeirsson and Pradipta Mitra. On a game theoretic approach to capacity maximization in wireless networks. *CoRR*, 2010.

[8] Telecommunications Industry Association and Electronic Industries Association. *CDMA2000 High Rate Packet Data Air Interface Specification*. TIA document. Telecommunications Industry Association, 2003.

[9] Y. Bejerano and R.S. Bhatia. MiFi: a framework for fairness and QoS assurance for current IEEE 802.11 networks with multiple access points. *IEEE/ACM Transactions on Networking*, 14:849 –862, aug. 2006.

[10] Yigal Bejerano, Seung-Jae Han, and Li (Erran) Li. Fairness and load balancing in wireless LANs using association control. In *Proc. of ACM MOBICOM*, pages 315–329, 2004.

[11] Ioannis Broustis, Konstantina Papagiannaki, Srikanth V. Krishnamurthy, Michalis Faloutsos,

and Vivek Mhatre. MDG: measurement-driven guidelines for 802.11 WLAN design. In *Proc. of ACM MOBICOM*, pages 254–265, 2007.

[12] Asaf Cidon, Kanthi Nagaraj, Sachin Katti, and Pramod Viswanath. Flashback: decoupled lightweight wireless control. In *Proc. of ACM SIGCOMM*, pages 223–234. ACM, 2012.

[13] Constantine Coutras, Sanjay Gupta, and Ness B. Shroff. Scheduling of real-time traffic in IEEE 802.11 wireless LANs. *Wirel. Netw.*, 6:457–466, December 2000.

[14] Michael Dinitz. Distributed algorithms for approximating wireless network capacity. In *Proc. of INFOCOM*, pages 1397–1405, 2010.

[15] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *SIGOPS Oper. Syst. Rev.*, 36:147–163, December 2002.

[16] Ruijun Fu, Yunxing Ye, Ning Yang, and Kaveh Pahlavan. Doppler spread analysis of human motions for body area network applications. In *IEEE PIMRC*, pages 2209–2213, 2011.

[17] R. Gold. Optimal binary sequences for spread spectrum multiplexing. *IEEE Transactions on Information Theory*, 13:619 –621, october 1967.

[18] Raj Jain, Dah-Ming Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Technical Report, Digital Equipment Corporation, DEC-TR-301*, 1984.

[19] Anand Kashyap, Samrat Ganguly, and Samir R. Das. A measurement-based approach to modeling link capacity in 802.11-based wireless networks. In *Proc. of ACM Mobicom*, pages 242–253, 2007.

[20] Jae-Hoon Ko, Soonmok Kwon, and Cheeha Kim. Orthogonal signaling-based queue status investigation method in IEEE 802.11. *Comput. Commun.*, 34:1033–1041, June 2011.

[21] Rafael Laufer, Theodoros Salonidis, Henrik Lundgren, and Pascal Le Guyadec. XPRESS: a cross-layer backpressure architecture for wireless multi-hop networks. In *Proc. of ACM MOBICOM*, pages 49–60, 2011.

[22] K.K. Leung and B.J. Kim. Frequency assignment for IEEE 802.11 wireless networks. In *Proc. of IEEE VTC*, volume 3, pages 1422 –1426, oct. 2003.

[23] Eugenio Magistretti, Omer Gurewitz, and Edward W. Knightly. 802.11ec: Collision avoidance without control messages. In *Proc. of ACM MOBICOM*, pages 65–76, 2012.

[24] V.P. Mhatre, K. Papagiannaki, and F. Baccelli. Interference mitigation through power control in high density 802.11 WLANs. In *Proc. of IEEE INFOCOM*, pages 535 –543, may 2007.

[25] Kimaya Mittal and Elizabeth M. Belding. RTSS/CTSS: Mitigation of exposed terminals in static 802.11-based mesh networks. In *Proc. of IEEE WiMesh Workshop*, Sept. 2006.

[26] Rohan Murty, Jitendra Padhye, Ranveer Chandra, Alec Wolman, and Brian Zill. Designing high performance enterprise Wi-Fi networks. In *Proc of USENIX NSDI*, pages 73–88, 2008.

[27] Rohan Murty, Jitendra Padhye, Alec Wolman, and Matt Welsh. Dyson: an architecture for extensible wireless LANs. In *Proc. of USENIX ATC*, 2010.

[28] B.W. Parkinson and J.J. Spilker. *The Global Positioning System: Theory and Applications*. Number v. 1 in Progress in Astronautics and Aeronautics. 1996.

[29] G. Pei and T.R. Henderson. Validation of OFDM error rate model in ns-3. *Boeing Research Technology*, pages 1–15, 2010.

[30] G. Pei and V. S. A. Kumar. Distributed link scheduling under the physical interference model. In *Proc. of INFOCOM*, March 2012.

[31] Hariharan Rahul, Haitham Hassanieh, and Dina Katabi. SourceSync: a distributed wireless architecture for exploiting sender diversity. In *Proc of the ACM SIGCOMM*, pages 171–182, 2010.

[32] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wirel. Netw.*, 5:81–94, March 1999.

[33] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *Proc. of ACM SIGCOMM*, pages 51–62, 2006.

[34] Jiho Ryu, Changhee Joo, Ted Taekyoung Kwon, Ness B. Shroff, and Yanghee Choi. Distributed SINR based scheduling algorithm for multi-hop wireless networks. In *Proc. of MSWIM*, pages 376–380, 2010.

[35] D. Saha, A. Dutta, D. Grunwald, and D. Sicker. PHY aided MAC - a new paradigm. In *Proc. of IEEE INFOCOM*, pages 2986–2990, 2009.

[36] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. CSMA/CN: Carrier sense multiple access with collision notification. In *Proc. of ACM MOBICOM*, pages 25–36, 2010.

[37] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. No time to countdown: migrating backoff to the frequency domain. In *Proc of the ACM MobiCom*, pages 241–252, 2011.

[38] Vivek Shrivastava, Nabeel Ahmed, Shravan Rayanchu, Suman Banerjee, Srinivasan Keshav, Konstantina Papagiannaki, and Arunesh Mishra. CENTAUR: realizing the full potential of centralized wlans through a hybrid data path. In *Proc. of ACM MOBICOM*, pages 297–308, 2009.

[39] Mythili Vutukuru, Kyle Jamieson, and Hari Balakrishnan. Harnessing exposed terminals in wireless networks. In *Proc. of USENIX NSDI*, pages 59–72, Berkeley, CA, USA, 2008.

[40] Kaishun Wu, Haoyu Tan, Yunhuai Liu, Jin Zhang, Qian Zhang, and Lionel Ni. Side channel: bits over interference. In *Proc. of ACM MOBICOM*, pages 13–24, 2010.

[41] Xinyu Zhang and Kang G. Shin. E-MiLi: Energy-minimizing idle listening in wireless networks. In *Proc. of ACM Mobicom*, pages 205–216, 2011.

[42] S. Zhou and Z. Zhang. cMAC: A centralized MAC protocol for high speed wireless LANs. In *Proc. of IEEE GLOBECOM*, pages 1–6, 2011.