ECEN 689 – Real-Time Wireless Networks: Project 2 Scheduling for Uplink Transmissions with Point Coordination Function

Ping-Chun Hsieh
lleyfede@tamu.edu

Tao Zhao
alick@tamu.edu

Dongni Han
handongni2015@tamu.edu

Due on 4/1

Terminology

In our report, we use AP or "server" to denote the WiFi access point, and "client" to denote the terminal device such as a mobile phone, a tablet, and so on. Throughout our simulation, we let node 0 be the server, and the other nodes be the clients. The basic time unit for packet transmission is called *slot*, which should be greater than a round-trip time (RTT). For real-time traffic, we group an integer number (denoted by T) of slots into an *interval*, which is the relative deadline of the packets.

1 Background

We consider a wireless network of one AP and K clients, where $K \geq 1$. We consider uplink transmission with point coordination function (PCF). In PCF mode, the AP has full control over channel allocation. In the beginning of each interval, each client generates a random number of packets and waits for channel access that is fully controlled by the AP. Since the packets are generated and queued on the client's side, the AP needs to collect the queue information of the clients to make scheduling decisions.

In this project we consider a baseline policy, where the AP collects the queue information by polling each client one by one for the number of data packets it generates. After all clients are polled, the AP knows the number of packets at each client, then schedules one of the clients in each time slot for actual uplink data transmission.

2 Implementation in NS-2

Because there is no PCF functions available in 802.11 Mac module in ns-2, and we prefer to make minimal changes to the 802.11 Mac implementation itself, we implement the PCF functions with the baseline policy in the application layer, where we define the `hdr_swifi` packet header and the `SWiFiAgent` class.

In the beginning of simulation, one node becomes the AP by issuing the `server` command. For example, `$sw(0) server` makes the zeroth node the server. Client registration is done by the `register` command.

For instance, `$sw_(0) register 1 0.5` registers node 1 as a client with channel reliability 0.5.[1]

In the beginning of each interval, the AP calls `boi`, which stands for "beginning of interval", to reset its internal state and packet counters. Each client generates a random number of packets from a uniform distribution, and uses the `pour` command to make it available from Tcl to C++.

In each time slot, the AP executes the `poll` command. Based on its state, it either sends a certain type of poll packet, or idle. In the beginning of each interval, the AP is in the `SWiFi_POLL_NUM` state, and calls `scheduleRoundRobin(false)` to find the next client to poll the number of new packets in the order of initial registration. Here `false` means once all clients have been scheduled exactly once, no one will be scheduled again, instead of scheduling from the first one repeatedly. The AP sends a `SWiFi_PKT_POLL_NUM` packet to the target client.

When the client receives the `SWiFi_PKT_POLL_NUM` packet, it replies a `SWiFi_PKT_NUM_UL` packet with its number of packets in the packet header. The number (denoted by X_n) is the number of newly generated packets in this interval for real-time traffic, while it is the number of total generated packets up to now for non-real-time traffic. When the AP receives the packet, it acquires the number, and calculates the queue length of the client. For real-time traffic, the queue length is the same as the received number. For non-real-time traffic, in order to derive the real queue length of each client, the AP needs to keep track of the total number of received packets (denoted by Y_n) from each client n, and then calculates the queue length as $(X_n - Y_n)$ for each client n. If either the transmission of `SWiFi_PKT_POLL_NUM` or `SWiFi_PKT_NUM_UL` fails, the AP will retransmit the `SWiFi_PKT_POLL_NUM` packet again to the same client in the next slot.

After all clients have been polled for their numbers of packets, the `SWiFi_POLL_NUM` ends and the AP goes into the `SWiFi_POLL_DATA` state, where actual data packets are polled. In each slot, the AP applies the Max-Weight policy, which schedules the client that maximizes the product of its queue length and the channel reliability in our implementation.

The AP sends a `SWiFi_PKT_POLL_DATA` packet to the target client to provide access to the channel. Note that we put the expected packet ID in the header of `SWiFi_PKT_POLL_DATA` packets so that the target client knows which packet to transmit and the previous packets are delivered once it receives the `SWiFi_PKT_POLL_DATA` packet. This is a better solution than an additional application-layer ACK from the AP because the latter needs more time and is still unreliable when the channel is bad. The scheduled client's data packet is of type `SWiFi_PKT_DATA_UL`. If the AP receives the data packet successfully, the AP will decrease the queue length by 1. If either the transmission of data polling or the data packet fails, the AP will stay with the same schedule in the next slot (suppose the next slot is not the end of the interval).

If the AP already receives all the packets available in the current interval, it will go to the idle state `SWiFi_POLL_IDLE` until the start of the next interval. No packet is transmitted in this state. On the other hand, for real-time traffic, if some packets are not delivered within the interval, they will be dropped.

Table 1 summarizes the four packet types used in our implementation, and Figure 1 shows their packet structures. Table 2 summarizes the states of the AP in PCF mode, and state transitions are illustrated in Figure 2.

3 Simulation Results

Throughout the simulation, we consider a fully-symmetric network with one AP and 2 clients. We use the shadowing module as the wireless channel. The parameters of the channel are summarized in Table 3. The transmitter power level is 1 W. The channel reliability can be changed by varying the distance between the

[1] We use a lookup table to map the distance between the server and the client to the channel reliability based on the result of reliability measurement in project 1.

Table 1: Packet types.

Packet Type	Meaning
`SWiFi_PKT_POLL_NUM`	Poll number of packets in uplink
`SWiFi_PKT_POLL_DATA`	Poll data packet transmission in uplink
`SWiFi_PKT_NUM_UL`	Packet in uplink that carries number of packets at client
`SWiFi_PKT_DATA_UL`	Data packet in uplink (client to AP)

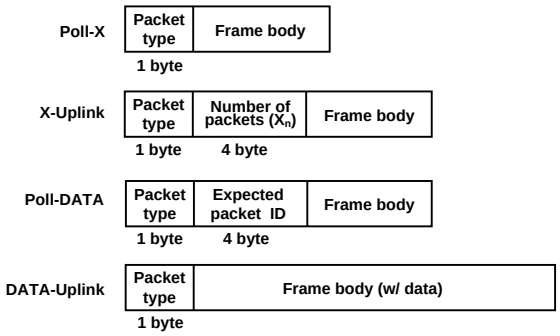

Figure 1: Packet structures.

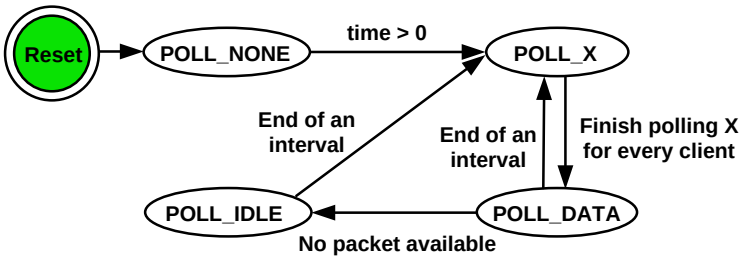

Figure 2: State transitions of the AP.

Table 2: Poll states.

Poll State	Meaning
`SWiFi_POLL_NONE`	Uninitialized
`SWiFi_POLL_NUM`	Poll the number of packets
`SWiFi_POLL_DATA`	Poll a data packet
`SWiFi_POLL_IDLE`	Idle

Table 3: Parameters of the wireless channel.

Item	Value
Path loss exponent	2.0
Shadowing deviation	4.0 dB
Reference distance	1.0 m

AP and the clients. In our simulation, we set the distance between the AP and a client to be 1 m when creating a reliable channel. For an unreliable channel with $p_n \approx 0.57$, the corresponding distance is 1000 m.

Table 4: Parameters of the 802.11b MAC.

Item	Value
Data rate	11 Mb/s
Basic rate	1 Mb/s
PLCP data rate	1 Mb/s
Preamble length	144 bits
Slot time	20 μs
SIFS	10 μs

For the medium access control (MAC) layer, we use the 802.11 module built in ns-2. Following the IEEE 802.11b standard, the MAC layer parameters are chosen as in Table 4. Besides, the automatic retry function built in 802.11 is disabled, i.e. `ShortRetryLimit_` and `LongRetryLimit_` are set to be 0 in the Tcl domain, to avoid channel congestion due to uncontrollable retransmission in the MAC layer.

Throughout the simulation, a slot is chosen to be 10 ms, which is much longer than a RTT (≈ 1.63 ms), to provide enough margin for packet delivery. We consider random packet arrivals. The number of packets generated in an interval follows discrete uniform distribution that ranges between 0 and N_{max}.

To evaluate the performance of the baseline policy, We use the average number of delivered packets in an interval as the main performance metric. It corresponds to the timely-throughput for real-time traffic, and is simply the throughput for non-real-time traffic. The final result is the average of 10 runs under each set of parameters.

3.1 Network Capacity

We first study the network capacity with different packet generation rates under the baseline policy. Let $T = 10$ and the channels be reliable, i.e. $p_1 = p_2 = 1$. We measure the average network throughput (denoted by $\bar{R}$) with different N_{max}.

In the non-real-time scenario, Figure 3 shows the system throughput for N_{max} between 1 and 12. We can see that the maximum achievable throughput is 8, less than T. This is because the AP has to collect the queue information in the polling phase, which lasts for 2 slots in every interval. Therefore, polling indeed reduces the network capacity. Moreover, the dashed line shows the expected network throughput if there is no randomness in packet generation. Thus, for non-real-time traffic, randomness in packet generation does not affect the network throughput.

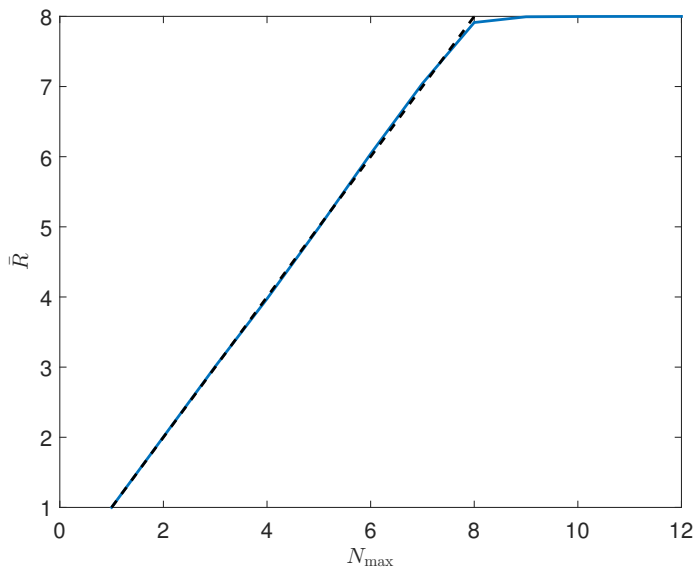

Figure 3: Network throughput versus N_{max} with non-real-time traffic

For real-time traffic, the network throughput is shown in Figure 4. Compared to the non-real-time case, the network throughput with real-time traffic is smaller when $N_{max} \geq 5$. This is because packets may expire when the total packets generated in an interval is larger than 8, which is the length of the scheduling phase. Therefore, in addition to polling overhead, packet deadline can further reduce the network throughput.

3.2 Choice of Interval Length

We study the how the choice of T affects the network throughput. Choose $N_{max} = 2$ and $p_1 = p_2 = 0.57$. Figure 5 shows the network throughput with different interval length in the non-real-time scenario. The total throughput is very low when $T = 4$ because the polling overhead is particularly significant for small T. Moreover, since larger T implies more available slots to the scheduling phase, the system throughput should increase with interval length.

Figure 6 shows the network throughput with different interval length with real-time traffic. Compared to the non-real-time case, the throughput with real-time traffic is even lower due to possible packet expiration. In other words, real-time traffic requires larger T so that the effect of the expired packets can be mitigated. Besides, Figure 6 serves as a useful reference for system design. For example, suppose both clients require a delivery ratio of 0.6. From Figure 6, we need to choose $T \geq 8$ to meet the requirement.

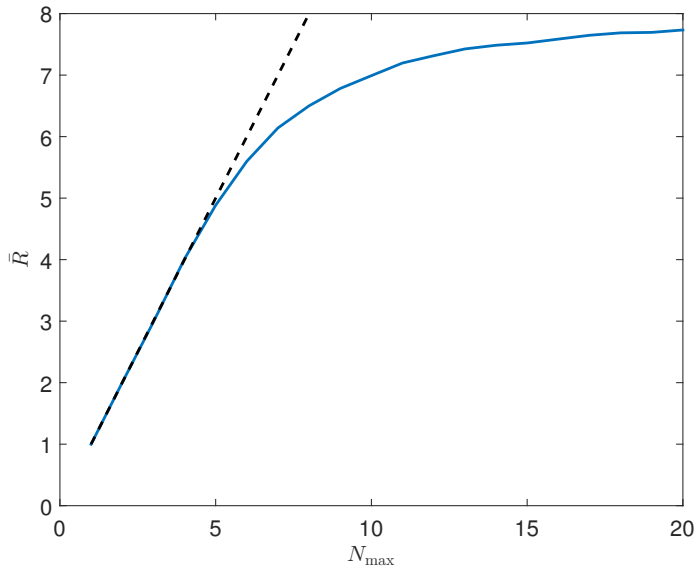

Figure 4: Network throughput versus N_{max} with real-time traffic

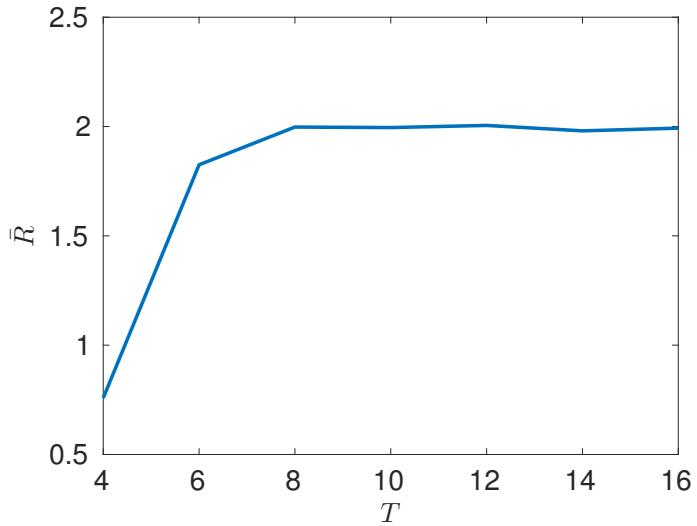

Figure 5: Network throughput versus interval length with non-real-time traffic.

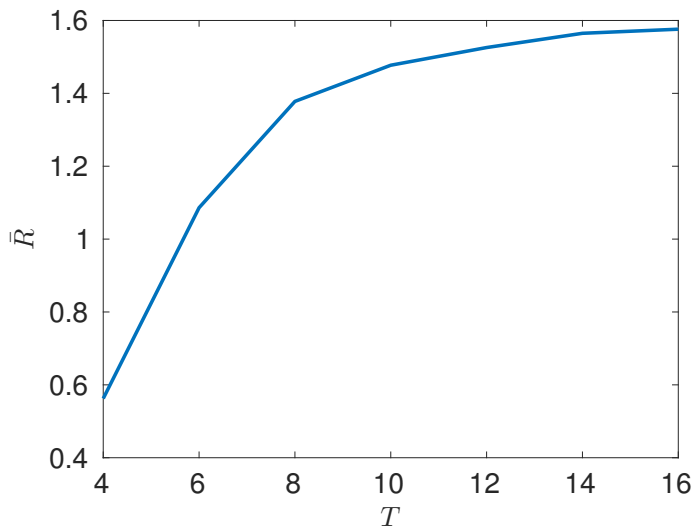

Figure 6: Network throughput versus interval length with real-time traffic.

3.3 Network Size and Polling Overhead

We study the effect of network size on the polling overhead. Let $T = 10$, $N_{max} = 2$, and channel reliability $p_1 = p_2 = 0.57$. Figure 7 shows the network throughput with non-real-traffic traffic for different network size. When K grows from 1 to 3, the total throughput increases since the total traffic is not saturated yet. As K exceeds 3, the network throughput drops significantly because the polling phase takes most of the time in an interval. Figure 8 shows a similar pattern for the real-time case.

In both scenarios, the network performance degrades severely with more clients. Hence, the baseline policy is not suitable for large networks due to the polling overhead.

4 Conclusion

The baseline policy employs a simple polling scheme to collect queue information of the clients. Accordingly, the AP can be work-conserving in the scheduling phase. However, the channel utilization for data packets can be low due to the polling overhead. The overhead can degrade the network performance even more severely when the number of clients is large or the interval length is small. To mitigate this effect, we definitely need a smarter policy to accommodate uplink transmission with PCF.

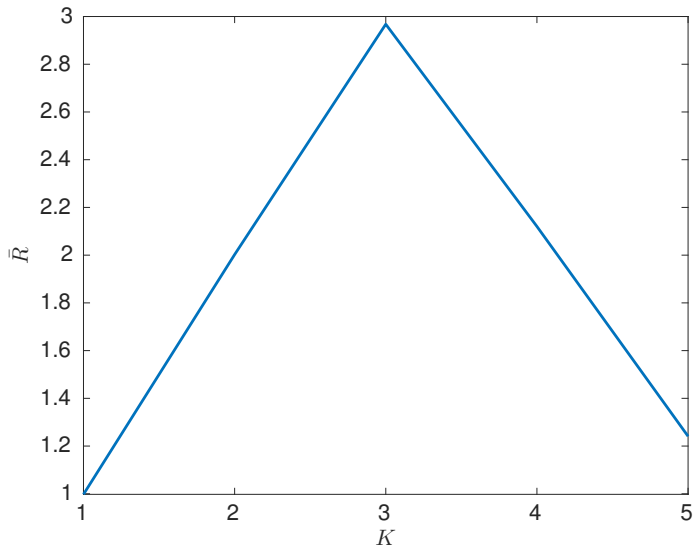

Figure 7: Network throughput versus network size with non-real-time traffic.

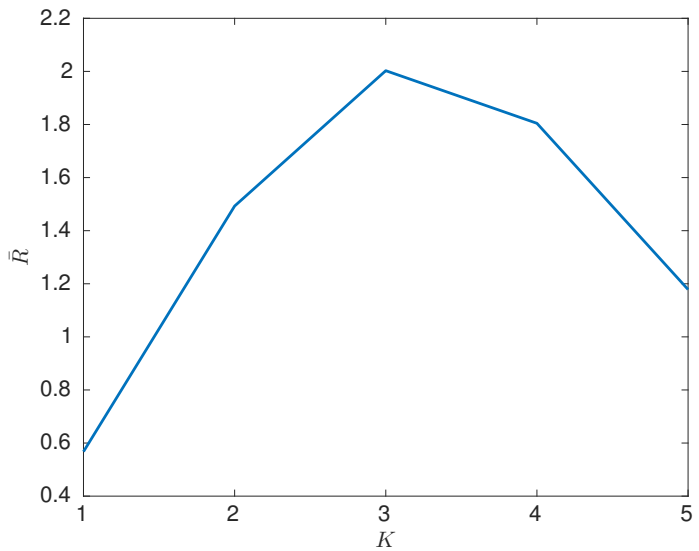

Figure 8: Network throughput versus network size with real-time traffic.