Master's Thesis

An Efficient Packet Scheduling Scheme to Reduce Power Consumption of Mobile Devices

Young Deok Park (박 영 덕)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2014

모바일 단말의 파워 소비 절감을 위한 효율적인 패킷 스케줄링 기법

An Efficient Packet Scheduling Scheme to Reduce Power Consumption of Mobile Devices

An Efficient Packet Scheduling Scheme to Reduce Power Consumption of Mobile Devices

by

Young Deok Park
Department of Computer Science and Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Computer Science and Engineering

Pohang, Korea

10. 22. 2013

Approved by

____

Hwangjun Song

Academic Advisor

An Efficient Packet Scheduling Scheme to Reduce Power Consumption of Mobile Devices

Young Deok Park

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master's degree from POSTECH

10. 22. 2013

Committee Chair Hwangjun Song (Seal)

Member Young-Joo Suh (Seal)

Member Chansu Yu (Seal)

MCSE 박영덕, Young Deok Park

20120596 An Efficient Packet Scheduling Scheme to Reduce Power Consumption of Mobile Devices,
모바일 단말의 파워 소비 절감을 위함 효율적인 패킷 스케줄링 프로토콜
Department of Computer Science and Engineering, 2014, 25p, Advisor: Hwangjun Song, Text in English

ABSTRACT

IEEE 802.11 power saving mode mechanism is introduced for reducing power consumption of mobile devices such as smartphone. The mechanism reduces power consumption by switching WiFi interface of mobile device between active-state and sleep-state alternately. Since the power consumption in active-state is significantly higher than that in sleep-state, switching between both states should be done carefully i.e., low priority packets such as delay tolerant packet should not trigger switching to active-state to increase power saving effect. Delay tolerant packets, however, also affect the behavior of the power saving mode mechanism since the mechanism is operated within the MAC layer. Thus, much power is consumed by delay tolerant traffic. To overcome this issue, we propose an efficient packet scheduling scheme, called EPS. EPS separates delay tolerant packets from delay sensitive packets and transmits delay tolerant packets during an additional active time of mobile device's WiFi interface. Thus, the delay tolerant packets never affect the behavior of power saving mode mechanism. Proposed EPS improves power saving performance without significant transmission delay under common application's traffic patterns.

Contents

List of Figures

List of Tables

I. Introduction

WiFi interface is a significant contributor to battery draining on mobile devices such as a smartphone. Especially, when the WiFi interface is in active-state, 15 times more power is consumed compared with sleep-state [1]. Accordingly, IEEE 802.11 standard introduced Power Saving Mode (PSM) mechanism for reducing power conservation [2]. IEEE 802.11 PSM mechanism can be divided into Static PSM (PSM-S) and Adaptive PSM (PSM-A).

In the PSM-S, client keeps WiFi interface sleep-state when there is no packet to receive. Thus, PSM-S can provide much power saving effect compared with Constantly Awake Mode (CAM) where client always keeps WiFi interface active-state regardless of presence of the packet to receive. Although the PSM-S provides great power saving effect, it causes significant transmission delay impermissible on the delay sensitive services such as web-browsing [3]. Consequently, PSM-A mechanism has been adopted by most commercial smartphones. A smartphone with PSM-A alternately switches between PSM-S and CAM. More specifically, smartphone goes to active-state at every beacon listening interval to receive the beacon message; otherwise, it remains in the sleep-state until next beacon listening interval i.e., smartphone acts as PSM-S until next beacon listening interval. In addition, whenever receiving the data packet, smartphone stays in active-state for a fixed time called tail-time to receive following packets without delay i.e., smartphone stays in CAM during tail-time.

Since the power consumption in the active-state is significantly higher than that in the sleep-state, switching from PSM-S to CAM should be done carefully i.e., certain packets such as delay tolerant packets should not trigger switching to CAM for power saving. Since the PSM mechanism is operated on the MAC layer, however, we expect that switching from PSM-S to CAM may be done recklessly. To verify our assumption, we conducted several controlled experiments with commercial smartphones such as iPhone5. The results of controlled experiments shows that tested all smartphones rashly react against receiving packets even though the packets have unreachable destination port number. Clearly, this thoughtless reaction leads to waste of precious battery power. It will be described in Section IV. Several previous researches also pointed out that such a reckless behavior of PSM mechanism as well as too long tail-time significantly degrades power saving performance [4] [5] [6] [7]. Unfortunately, previously proposed schemes demand modification of client side which makes deployment to be hard. To make things worse, continuous and expensive processes for proposed schemes on client side might result in another power consumption of client itself.

In this thesis, we propose an efficient packet scheduling scheme called EPS. EPS aims to reduce power consumption by preventing it from triggering switching from PSM-S to CAM without significant transmission delay. EPS separates delay tolerant traffics from delay sensitive traffics and transmits delay tolerant traffics during an additional active time of commercial WiFi interface immediately before going to sleep-state. Since commercial APs cannot recognize the presence of this additional active time, we call it "hidden tail-time". Unlike previous approaches, EPS does not require any modifications of the client side and any changes to the IEEE 802.11 protocol. Thus, EPS is more deployable, and we do not need to worry about power consumption squandered by additional operations on the client side. Implemented EPS prototype reduces power consumption caused by delay tolerant traffic by up to 95% without significant delay and without any negative effect to delay sensitive traffic under common applications' traffic patterns.

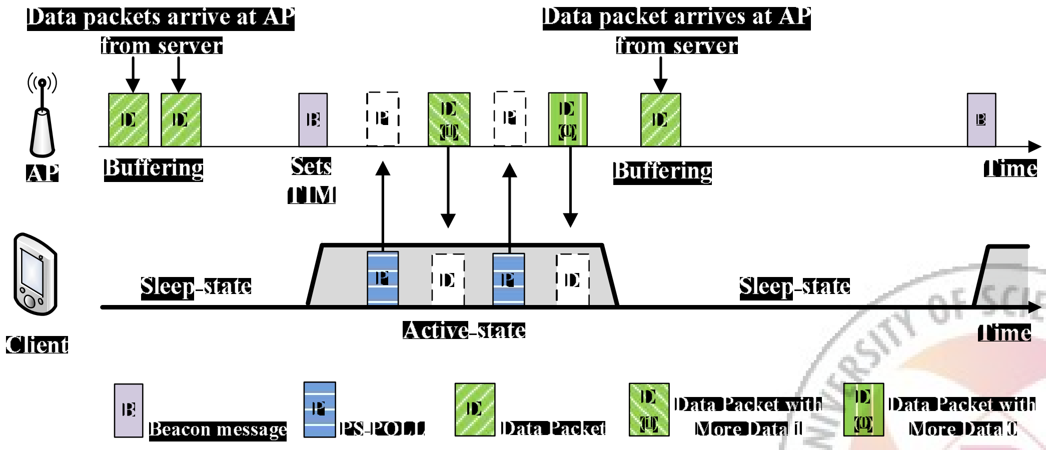
The remainder of this thesis is organized as follows. In Section II and Section III, we discuss background and related work, respectively. We present our motivation in Section IV. In Section V, we propose an efficient packet scheduling scheme to reduce power consumption of mobile device, EPS. Performance of the implemented prototype is evaluated in section VI. Finally, we conclude this thesis in section VII.

II. Background

2.1 IEEE 802.11 power saving mode mechanism

According to the power mode of WiFi interface, state of client can be divided into two states: active-state and sleep-state. In case of active-state, client keeps power mode of WiFi interface active with consuming high power to receive the data packet. In contrast, client consumes very low power but cannot receive any packet in the sleep-state. Consequently, AP should buffer the packets which arrive from server while client is in the sleep-state. At every beacon interval, AP informs client whether or not the client has packets intended for it by using the Traffic Indication Map (TIM) within beacon message.

A client adopting PSM-S switches from sleep-state to active-state at every beacon listening interval to receive beacon message. After receiving the beacon message, client checks the TIM to lean the presence of buffered packets for it. As we can see in the Fig. 1, if TIM is set, client sends PS-POLL message to receive buffered packet. If there are more packets buffered, AP sets MORE DATA bit within transmitted packet, so that client sends PS-POLL message again to receive additional packets. Otherwise, client is allowed to immediately go back to sleep-state. Thus, PSM-S is better than CAM in terms of power saving since the client remains in the sleep-state when there is no outstanding packet at AP.



However, client suffers from significant transmission delay impermissible in the delay sensitive service such as web-browsing since transmission of packets is synchronized with beacon listening interval.

To alleviate significant delay of PSM-S, PSM-A uses tail-time, an additional waiting time for incoming packets. Fig. 2 shows an illustrative example of PSM-A. In the figure, client switches to active-state after sending null data frame with disabled PwrMgt bit (null_active frame) if there is buffered packet for it. After switching its mode to active-state, client keeps the same state until the tail-time is finished (i.e., CAM) to receive following packets without delay. Whenever client receives the packet from AP during the tail-time, client starts a new tail-time. As a result, the duration of CAM is extended as much as tail-time whenever client receives a data packet. After the tail-time is finished, client sends null data frame with enabled PwrMgt bit (null_sleep frame) to inform AP that it will go to sleep-state soon. In fact, client does not go to sleep-state immediately after sending null_sleep frame, but rather additionally keeps active-state for a certain time called "hidden tail-time" to mitigate packet retransmissions caused by contentions with background packets. We will describe hidden tail-time concretely in the following subsection.

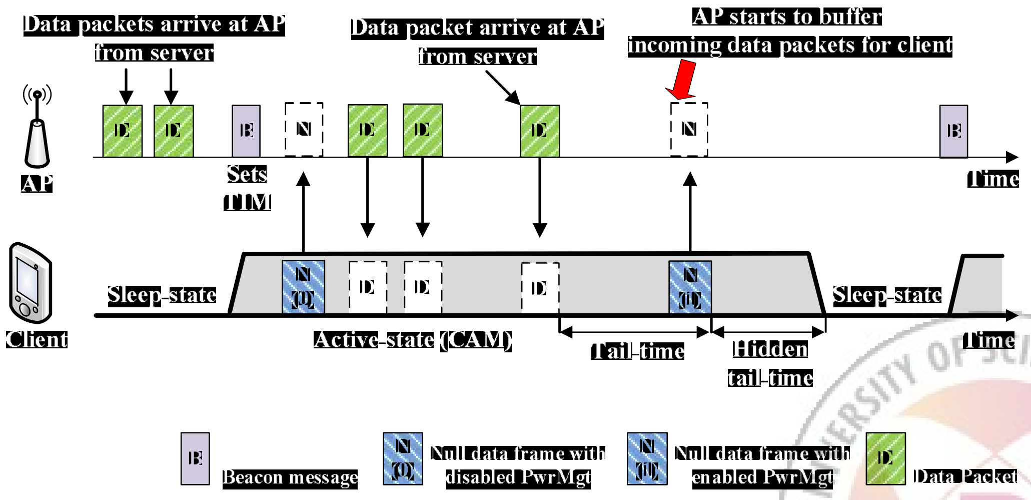

Data packets arrive at AP from server
Data packet arrive at AP from server
AP starts to buffer incoming data packets for client
AP
Sets TIM
Time
Client
Sleep-state
Active-state (CAM)
Tail-time
Hidden tail-time
Sleep-state
Time
Beacon message
Null data frame with disabled PwrMgt
Null data frame with enabled PwrMgt
Data Packet

Fig. 2. Illustrative example of PSM-A.

In the EPS, delay tolerant packets are transmitted without switching CAM like the PSM-S, except for synchronizing with hidden tail-time duration instead of synchronizing beacon listening interval. In contrast, delay sensitive packets are transmitted in the same way to PSM-A.

2.2 Hidden tail-time

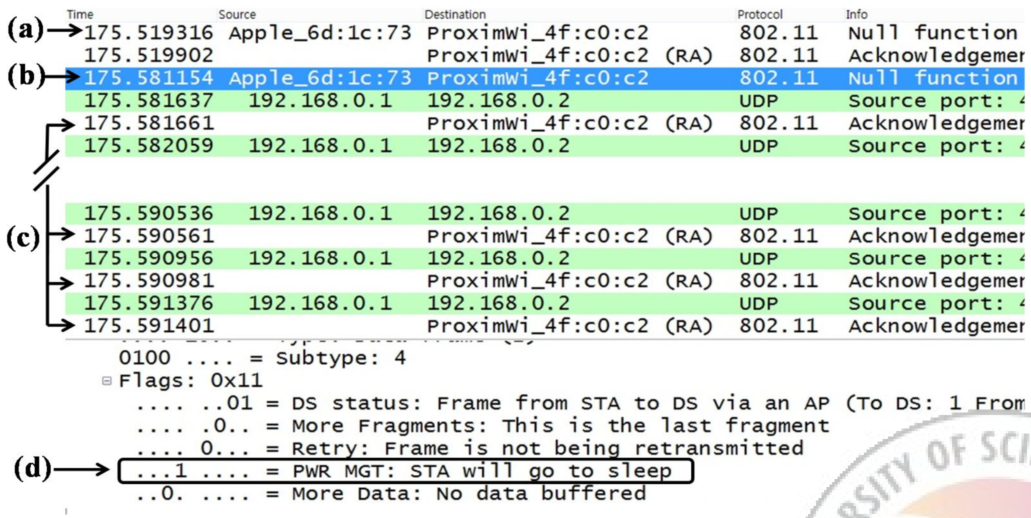
Immediately after sending null_sleep frame, client keeps additional active time called hidden tail-time. To the best of our knowledge, there is no commercial AP which recognizes the presence of hidden tail-time. The reason is that AP regards client is really in sleep-state when the null_sleep frame is received, but client has hidden tail-time just after sending null_sleep frame.

Hidden tail-time aims to mitigate packet retransmissions caused by contentions with background packets. Every packet that arrives at AP while client is CAM is enqueued at end of main transmission queue and transmitted. However, the packet in main transmission queue could not be delivered in time due to the contention with other background packets when there are many buffered background packets [1]. Consider simple scenario where client does not have hidden tail-time. If packets for a client, which is in CAM, arrive at AP from server immediately before end of the client's tail-time; i.e., the end of the client's tail-time is imminent. AP just enqueues the packets to main transmission queue, since client is still in CAM. As mentioned above, however, the transmission of packets might be delayed due to the contention with other packets if there are many background packets. At the same time, client goes to the sleep, because there is no packet received by end of its own tail-time, though the packets intended for it are still buffered in the main transmission queue.

As a result, the late packets are eventually sent while the client is sleeping and the packets incur the maximum retransmissions before being dropped. In fact, AP cannot control this unwanted retransmissions. In contrast, when hidden tail-time is used, this retransmission problem can be mitigated since client does not immediately go to sleep-state after sending null_sleep frame, but rather client additionally waits for a hidden tail-time.

To measure hidden tail-time of commercial smartphones, we modified Madwifi driver [9] on AP for controlling packet transmission as follows. AP repetitively transmits the several packets to the smartphone until smartphone sends null_active frame. After receiving the null_active frame from smartphone, AP stops to transmit the packets so that the smartphone, after waiting for its own tail-time, sends null_sleep_frame to notify AP that it will go to sleep state soon. Immediately after receiving the null_sleep frame from smartphone, AP resumes transmitting packets repetitively. Clearly, if the smartphone responds with ACK frame for receiving packet even after sending null_sleep frame, we can know that it means the smartphone is actually in the active-state; i.e., the smartphones is in the hidden tail-time duration. We can also know that if there is no ACK frame any more, it means the smartphone actually goes to sleep-state; i.e., the hidden tail-time is finished.

Fig. 3 shows the packets captured during experiments of hidden tail-time measurement



```
Time        Source            Destination                 Protocol  Info
175.519316  Apple_6d:1c:73    ProximWi_4f:c0:c2           802.11    Null function
175.519902                    ProximWi_4f:c0:c2 (RA)      802.11    Acknowledgemer
175.581154  Apple_6d:1c:73    ProximWi_4f:c0:c2           802.11    Null function
175.581637    192.168.0.1     192.168.0.2                 UDP       Source port: 4
175.581661                    ProximWi_4f:c0:c2 (RA)      802.11    Acknowledgemer
175.582059    192.168.0.1     192.168.0.2                 UDP       Source port: 4

175.590536    192.168.0.1     192.168.0.2                 UDP       Source port: 4
175.590561                    ProximWi_4f:c0:c2 (RA)      802.11    Acknowledgemer
175.590956    192.168.0.1     192.168.0.2                 UDP       Source port: 4
175.590981                    ProximWi_4f:c0:c2 (RA)      802.11    Acknowledgemer
175.591376    192.168.0.1     192.168.0.2                 UDP       Source port: 4
175.591401                    ProximWi_4f:c0:c2 (RA)      802.11    Acknowledgemer

  0100 .... = Subtype: 4
⊟ Flags: 0x11
    .... ..01 = DS status: Frame from STA to DS via an AP (To DS: 1 From
    .... .0.. = More Fragments: This is the last fragment
    .... 0... = Retry: Frame is not being retransmitted
    ...1 .... = PWR MGT: STA will go to sleep
    ..0. .... = More Data: No data buffered
```

Fig. 3. Captured packets under the hidden tail-time measurement.

(a) Null_active frame. (b) Null sleep frame. (c) ACK frames after sending null_sleep frame. (d) Flag of null sleep frame.

with iPhone 5. In the figure, null_sleep frame (b) is sent after about 60ms from null_active frame (a), which means finish of tail-time and start of hidden tail-time. In addition, (d) shows the flag of captured null_sleep frame. Especially, we can know that smartphone actually responds to received packets by ACK frames even after smartphone sends null_sleep frame (c). We observe that after about 10ms, there is no ACK frame for received packet any more. Thus, we can calculate hidden tail-time of smartphone by subtracting receiving time of null_sleep frame (b) from receiving time of last ACK frame. The results of measurements are shown in the table 1.

Interestingly, there is no any extension of active-time even if smartphone receives the packet from AP during hidden tail-time. This feature of hidden tail-time provides opportunity for improve PSM performance without any modification of client side. In contrast, as mentioned section 2.2, whenever smartphone receives the packet from AP during the tail-time, the smartphone restarts the tail-time resulting in extension of active-time.

Table 1. Measured tail-time (TT) and hidden tail-time (HT).

Smartphone	WiFi chipset	TT (ms)	HT (ms)
Galaxy A	SWB T30	300	15
Galaxy S	BCM 4329	300	15
Galaxy S3	BCM 4334	330	15
iPhone 3GS	BCM 4325	60~70	10
iPhone 4	BCM 4329	60~70	10
iPhone 5	BCM 4335	60~70	10

III. Related Work

There are several researches which take similar approaches to EPS. TailTheft [4] utilizes wasted tail-time of cellular interface such as 3G for delay tolerant packets. TailTheft provides APIs to application developers so that they indicate whether or not their application's traffic could be deferred. STPM [5] also uses "hint" given by application developer to determine whether given application is delay tolerant or not. Unfortunately, it is impractical to rely on developers to determine priority of applications since there are few developers who want to set their application to be low priority.

SAPSM [6] use machine leaning algorithm to determine priority of applications without any information given by application developer. In [7] authors introduce "packing" scheme to reduce power consumption caused by delay tolerant traffic. However, they require client side modification that makes deployment to be hard. To make things worse, they need additional applications continually collect statistics from kernel, or successively monitor network activity. As a result, these additional processes and running application on client side can bring about another power consumption of client itself. In contrast, EPS is implemented within AP without any modification of client side. Thus, it is more deployable, and there is no power consumption caused by additional operations on client side.

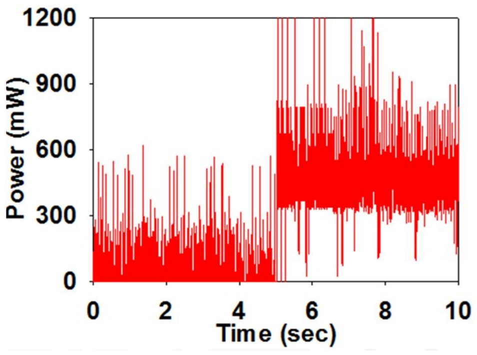
Many other researches have been going on to reduce the energy consumption of mobile device based on PSM-S and PSM-A. The authors of [3] propose BSD algorithm which provides bounded delay while minimizing energy by adaptively adjusting sleep duration of client. In [1], authors observe the unfairness problem owing to competition between PSM packets and background traffic. To resolve this problem, they propose NAPman that uses energy aware scheduling guaranteeing time-fairness of the packets. Another approach [8] reduces energy consumption of client by making the client sleep state during packets inter-arrival time. In PSM-throttling [10], the authors reduce the energy consumption of client by shaping the busty traffic.

IV. Motivation

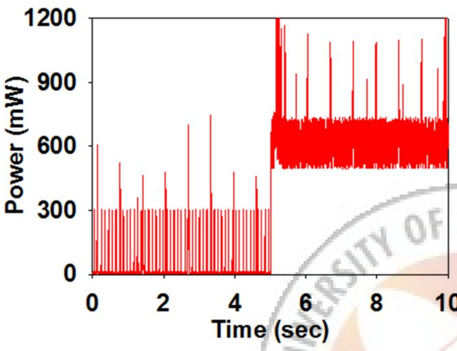
To study inability of existing PSM mechanism, we conducted several controlled experiments with various commercial smartphones. A methodology for the experiment is as follows. We configured an AP, and we generated UDP packets to the smartphone at 100-pkts/sec from 5 second. The generated UDP packets have unreachable destination port number (we call it unimportant packet). We measured power consumption of smartphones and analyzed traffic between AP and smartphone.

We used the packet sniffing tool, Wireshrk [11] to capture traffic between the AP and the smartphones. Note that behavior of PSM mechanism can be observed by examining 802.11 management frames. Especially, null data frame is good enough as an indicator of switching between CAM and PSM-S. Although this scenario unlikely arises under common condition, it is good example to study how unimportant packets impact behavior of PSM mechanism.

In fig. 4, we can see the smartphones' power plot is drastically increased from 5 second at which unimportant packets was generated. In addition, we observed the null_active frame around 5 second. Judging from fact that null_active frame is observed around 5 second, and the unimportant packets are generated from the same time, the drastic increase of power from 5 second is evidence that unimportant packets make smartphone to switch from PSM-



(a) iPhone 4



(b) Galaxy S

Fig. 4. Power plots of commercial smartphones under the unimportant packets.

S to CAM. This unwanted situation happens because PSM mechanism is implemented within WiFi chipset responsible for only MAC layer. Thus, every received packet equally affects behavior of PSM mechanism i.e., unimportant packet triggers switching to CAM. This situation, of course, will incur when delay tolerant packets are received. Recently, similar experiments have been conducted with more diverse smartphones and it is shown that tested all mobile devices are susceptible to any received packet regardless of traffic type [6].

Taken all together, we can confirm that most commercial implementations of PSM mechanism do not act carefully for power saving, but it is aggressively triggered by every receiving packet including delay tolerant packet.

It is clear that transmission of packet for a high delay tolerant service should use a strategy optimized for reducing power consumption. In contrast, transmission of packet for delay sensitive service should use strategy optimized for minimizing delay. Thus, if we can allow only delay sensitive packets to trigger switching to CAM, and allow delay tolerant packets to be somewhat delayed without triggering switching to CAM, the power consumption might be reduced without any negative impact on user experiment. We will present our solution to address this challenge in the following section.

V. Proposed Scheme

5.1 Measuring hidden tail-time

To exploit hidden tail-time properly, AP has to estimate hidden tail-time duration of each smartphone in the manner of automation. Fig. 5 shows schematic representation of online measuring hidden tail-time process.

After a new smartphone associates with AP, AP pretends there is buffered packet for it by setting TIM bit to make the smartphone switch to active-state. Since there is no received packet, the smartphone, after waiting for its own tail-time, sends null_sleep frame to notify AP that it will go to sleep-state soon. Immediately after receiving null_sleep frame, AP repetitively sends the small probe packet (1-byte packet with 1ms interval in this thesis) until there is no ACK frames any more. As mentioned in section III, as long as smartphone sends the ACK frames in a response to received probe packet, we can know that the smartphone is actually in the active-state. Finally, the hidden tail-time (HT) is calculated from flowing equation.

$$HT = T_{last\_ACK} - T_{null\_sleep} \tag{1}$$

where T_{null_sleep} is the receiving time of the null_sleep frame, and T_{last_ACK} is the receiving time of the last ACK frame for probe packet.

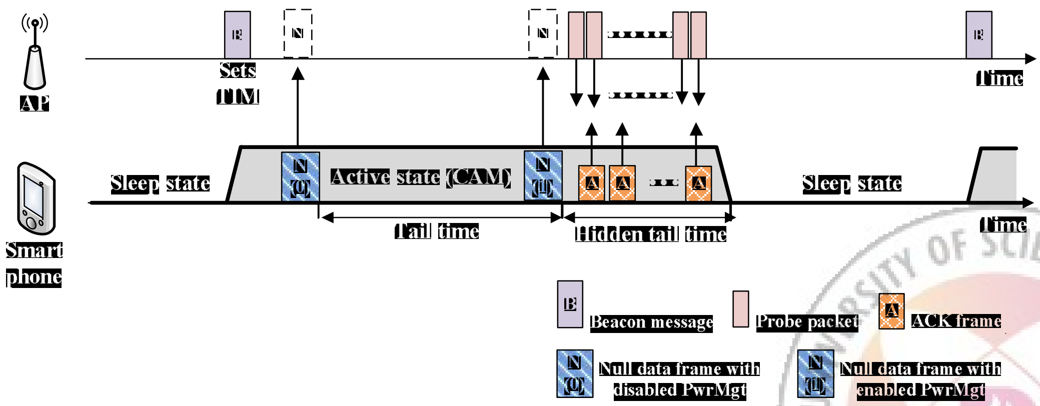


Fig. 5. Schematic representation of measuring hidden tail-time.

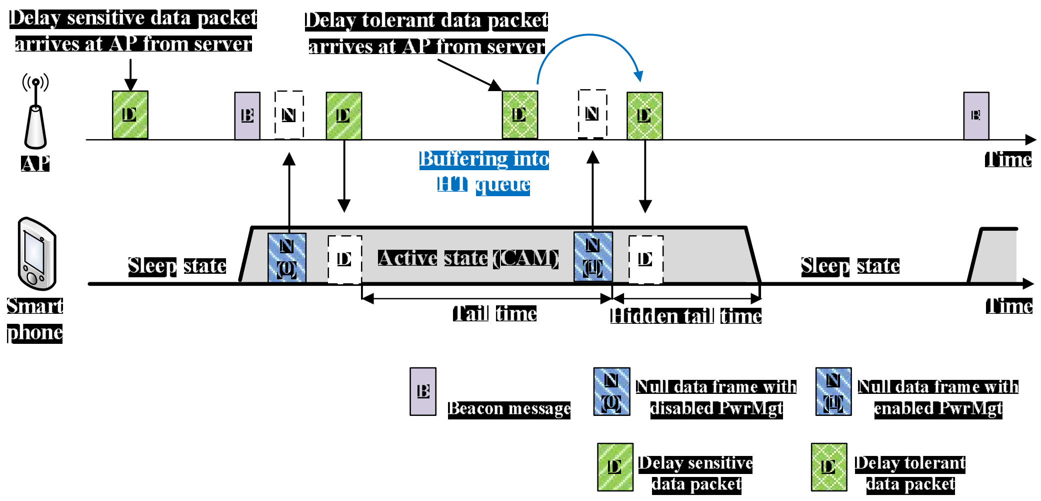

Delay sensitive data packet arrives at AP from server
Delay tolerant data packet arrives at AP from server
AP
Buffering into HT queue
Time
Smart phone
Sleep state
Active state (CAM)
Tail time
Hidden tail time
Sleep state
Time
Beacon message
Null data frame with disabled PwrMgt
Null data frame with enabled PwrMgt
Delay sensitive data packet
Delay tolerant data packet

Fig. 6. Illustrative example of transmission for delay tolerant packet.

Note that since the hidden-tail time is fixed value, these measuring processes are executed just one time only when the smartphone associates with AP. Thus, an overhead caused by sending small probe packets during short time is actually negligible. To reduce overhead caused by frequent handoff, AP maintains MAC address and hidden tail-time information tuples for a minute even after smartphone disassociates. This information is reused when same smartphone associates again.

5.2 Transmission of delay tolerant packets

To prevent delay tolerant packets from triggering switching from PSM-S to CAM or extending active time, AP buffers them in the high priority queue named hidden tail queue (HT queue) for each smartphone, and, in turn, AP transmits them only when the smartphone is in the hidden tail-time interval. Fig. 6 shows the illustrative example of transmission for delay tolerant packet.

In the figure, we assume the initial state of smartphone is sleep-state. If delay sensitive packet arrives at AP from server, it is processed as in the PSM-A for reducing transmission delay. In contrast, when delay tolerant packet arrives, it is buffered in the HT queue.

Whenever hidden tail-time of the smartphone is started, the buffered delay tolerant packets in the HT queue are transmitted during hidden tail-time.

In addition, since the HT queue is assigned to higher priority than the main transmission queue, the buffered packet can go ahead with other packets buffered in main transmission queue. Thus, delay tolerant packets can be transmitted during hidden tail-time without any contention with other packets buffered in main transmission queue. As mentioned in Section II, hidden tail-time does not incur any extension of active-time even if smartphone receives the packet from the AP. Therefore, delay tolerant packets never impact the power saving mechanism i.e., they never trigger switching to CAM, and they never extend active time.

To realize ESP, we have to address queue overflow problem on HT queue. Once the AP determines to buffer the delay tolerant packets in the HT queue, the queue is built up, and eventually becomes full. Accordingly, we exploit TCP Zero Window (ZW) message to let sender to stop to generate traffic. More specifically, when the queue length is excesses the top-threshold value (60 in this thesis), AP issues the ZW message to the server. AP, in turn, issues Non-Zero Window message to allow server to resume the transmission whenever the queue length become below the bottom threshold value (30 in this thesis).

The challenge then is how can AP generate the ZW message? Actually, AP cannot generate the ZW message since the responsibility of the ZW message generation is belonging to the transport layer. To deal with this challenge, we use the similar way to M-TCP [12] i.e., AP duplicates the TCP ACK packet of the client and keeps it to use when the ZW message is needed.

Unfortunately, to use ZW message for freezing sender might bring degrading throughput and eventually increase of task completion time. A large increase in task completion time can affect the user experience of an application, even if the traffic can be considered delay tolerant. Accordingly, we use resuming scheme to mitigate elongated task completion time. In this scheme, if cumulative delay caused by freezing of sender excesses 20 seconds, AP just transmit all received packets as in PSM-A to prevent from further degrading throughput. Similarly, if there is no hidden tail-time for 10 consecutive seconds (e.g., there is no network activity for 10 seconds), we assume that there is only delay tolerant traffic, so that there is no chance to transmit buffered delay tolerant packets. Accordingly, AP also just transmit all buffered in HT queue and incoming packets as in PSM-A to avoid degradation of throughput.

5.3 Adaptive hidden tail-time window

EPS maintains hidden tail-time window (HTwnd), 15 as an initial value and 30 as a maximum value. Basically, the number of packets AP transmits during hidden tail-time is up to HTwnd value. If the packets are transmitted successfully up to given HTwnd value during a hidden tail-time, HTwnd value is increased by 1 to increase the number of packets transmitted during next hidden tail-time. In contrast, when a packet incurs seven retransmissions during hidden tail-time, AP assumes that the channel condition is not enough to transmit every packet buffered in HT queue.

Accordingly, AP stops to transmit the packets and the packet which incurs retransmissions is re-enqueued at the front of HT queue instead of being discarded. At the same time, the HTwnd is decreased by one to reduce the number of packets transmitted during next hidden tail-time. Thus, the number of packets transmitted is adjusted according to the channel condition. Note that the HTwnd is also decreased in the situation mentioned in section II, where delay sensitive packet can be transmitted through the hidden tail-time due to the contention with other delay sensitive packets. Thus, to exploit hidden tail-time for delay tolerant traffic does not affect transmission of delay sensitive traffic.

5.4 Identifying delay tolerant packet

For the EPS to be viable in practice, we should address a challenge: How can we identify whether given packet is the delay tolerant packet or not? We use “well known port number” for it. For example, if the packet has source port number 21, it is packet related to FTP. In this thesis, we assume that the method using well known port number works well. Unfortunately, since many applications does not use well known port due to the some reason; e.g., fire-wall issues as well as security issues, it is necessary to use more efficient way to identify delay tolerant packet. Thanks to the many previous researches related to the traffic identification on the network [13] [14] [15] [16] [17], the accuracy of identifying the delay tolerant packet most likely is increased. We leave dealing with such enhancement opportunity as a future work.

VI. Performance Evaluation

To validate performance of EPS, we evaluated the EPS prototype in practice. In first scenario, we assume most simple case that an AP serves single client. In second scenario, we assumed more general case that an AP serves multiple clients. In both scenarios, we assume that there is mixed traffic of delay sensitive traffic and delay tolerant traffic. For example, a user browse website while updating application or downloading file from remote server. Since Android already provides multi-tasking function and iOS also has started to adopt it from iOS7, this scenario will be more prevalent. Note that we mainly focused on power consumption caused by delay tolerant traffic since EPS aims for reducing power consumption caused by delay tolerant traffic while never affecting delay sensitive traffic.

6.1 Experimental method

We implemented EPS prototype within AP based on MadWiFi wireless network driver for supporting 802 11g. iPhone5 (iOS 6) was used as a client. The phone has about 60ms tail- time and 10ms hidden tail-time. To measure power consumption of only WiFi interface, we turned off screen and disabled all other communication interfaces such as 3G and Bluetooth. We disassembled the phone and connected it to a Monsoon Power Monitor [18] to measure accurate power consumption (Fig. 7). To generate realistic traffic, we collected, by using tcpdump [19], packet sizes and packet arrival intervals of several common applications (table 2). Finally, we replayed generating traffics from AP to smartphone according to the collected packet sizes and intervals.

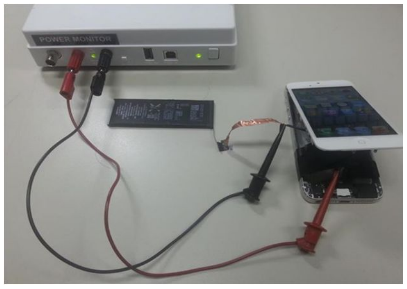


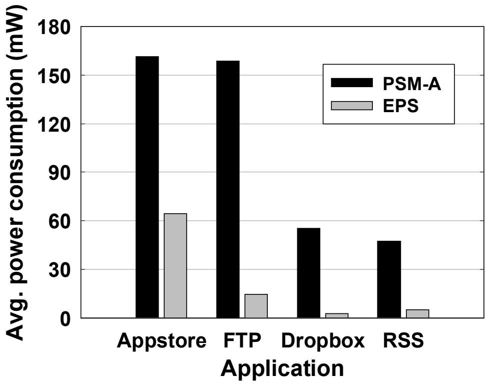
Fig. 7. Experimental setup with iphone5 and external power measuring tool.

Table 2. Information of used common applications.

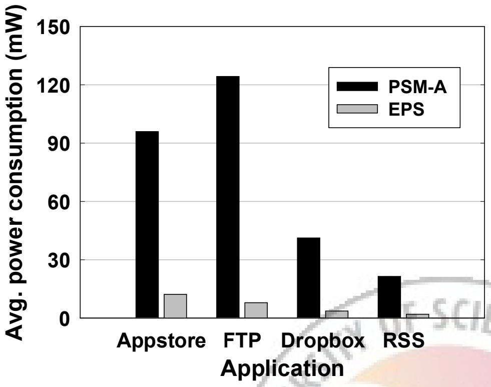
Traffic type	Application	Other information
Delay sensitive	Youtube	Music video streaming (4'50")
	Web-browsing	http://www.cnn.com
Delay tolerant	FTP	Downloading a 10-Mbyte file
	Dropbox	Downloading a 10-Mbyte file
	RSS reader	Downloading a 3-Mbyte xml update-file
	Updating application	Downloading a 16-Mbyte update-file from appstore

6.2 Evaluation result

Fig. 8 shows the power consumptions of EPS versus PSM-A. In the figure, we use "RSS" to indicate RSS reader application, and "appstore" to indicate to update application from Apple appstore. When web-browsing is used as a delay sensitive application (Fig 8a), the average power consumption of smartphone with PSM-A is 161.3 mW, 158.6 mW, 55.4 mW and 47.3 mW for each delay tolerant application respectively. In contrast, the average power consumption of smartphone with EPS is 64.4 mW, 14.6 mW, 2.8 mW, and 5.1 mW for each delay tolerant application i.e., EPS improves power saving performance up to about 60%, 91%, 95% and 89% respectively. We observed that resuming scheme was triggered in appstore case, so that more power was consumed than other application cases.



(a) Delay sensitive application
: Web-browsing



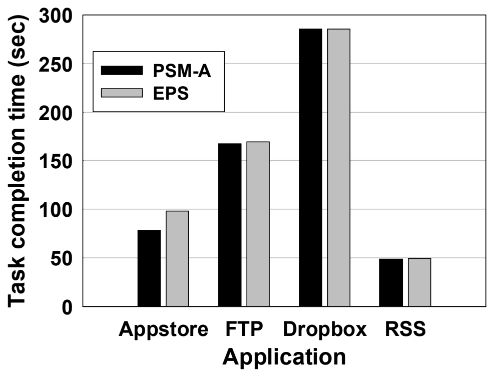
(b) Delay sensitive application
: Video streaming

Fig. 8. Average power consumption for delay tolerant applications.

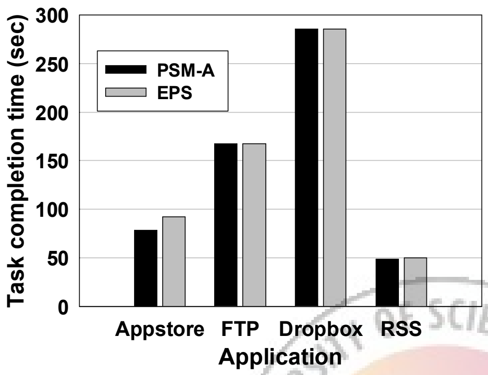
When video streaming is used as a delay sensitive application (Fig 8b), similarly to web-browsing case, power consumption is significantly reduced from 95.8 mW, 124.3 mW, 41.3 mW, and 21.3 mW to 12.2 mW, 7.9 mW, 3.4 mW, and 1.9 mW i.e., power saving performance is improved as much as about 87%, 94%, 92% and 91% for each application.

Task completion time is also important factor to affect user experiment of application. Fig. 9 shows completion time of each delay tolerant application of EPS and PSM-A. In Fig. 9a and Fig 9b, three applications; FTP, Dropbox, and RSS are never delayed, but appstore case are delayed from about 78seconds to about 98 seconds (web-browsing) and to about 92 seconds (video streaming) respectively. Note that 20 seconds delay is guaranteed by resuming scheme.

These enhancements of power saving performance without significant delay can be explained as follows. Whenever the packet arrives at AP, PSM-A just transmits the packet regardless of whether it is delay tolerant packet or not. Even if the smartphone is in sleep-state, AP with PSM-A makes it to go to active-state at next beacon interval through the TIM. Thus, delay tolerant packets eventually affect the behavior of PSM mechanism. In contrast, EPS prevents the delay tolerant packets from impacting on the PSM mechanism. Thus, EPS



(a) Delay sensitive application
: Web-browsing



(b) Delay sensitive application
: Video streaming

Fig. 9. Task completion time of delay tolerant applications.

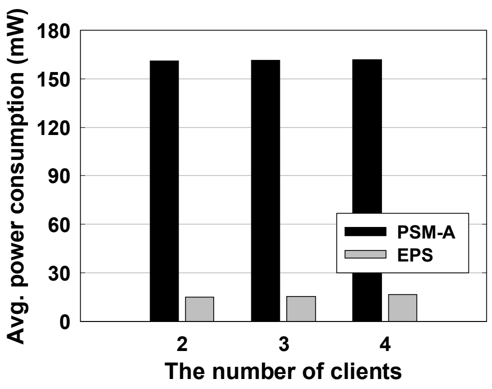


Fig. 10. Average power consumption with varying the number of clients.

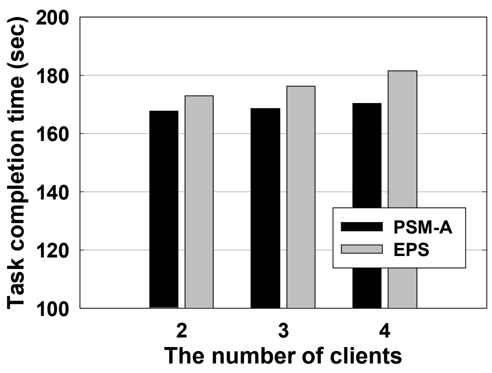


Fig. 11. Task completion time with varying the number of clients.

can achieve high power saving effect.

In general, an AP serves multiple clients rather than only single client. Accordingly, we assume there are multiple smartphones from two to four associating the AP. We can expect that when the hidden tail-times of multiple smartphones occur at exactly same time, the number of packets transmitted by AP during given hidden tail-time is reduced because AP cannot transmit packet to multiple clients at the same time.

Fig. 10 shows the result of average power consumption. We used web-browsing and FTP as a delay sensitive and delay tolerant application respectively. The average power consumption of smartphone with PSM-A is 160.8 mW, 161.2 mW, and 161.7 mW with varying the number of clients. In contrast, the average power consumption of smartphone with EPS is 15 mW 15.5 mW, 16.7 mW with varying the number of clients; i.e., power saving performance is improved as much as about 91%, 90%, and 89% with varying the number of clients. Fig. 11 shows the task completion time. The more the number of smartphones is, the longer task completion time is, but not significantly. The task completion time is increased by 5.2 seconds, 7.7 seconds, and 11.3 seconds with varying the number of clients. Although the task completion time is elongated, some delay of the delay tolerant application does not largely impact on user experiments since its small delay can be acceptable to users.

The reason there is no significant degradation of performance is that hidden tail-time seldom, if ever, occurs at exactly same time. The traffic pattern is very different on each application and its user behavior. Thus, hidden tail-time also occurs as a various patterns. Further, each WiFi chipset has different tail-time duration and hidden tail-time duration (Note that, hidden tail-time occurs following tail-time). As a result, hidden tail-time independently occurs at each smartphone, and the hidden tail-time is rarely overlapped to each other.

VII. Conclusion

Power conservation is a general concern for mobile devices. Through the experiments with off the shelf smartphones, we find the inability of existing PSM that all received packets equally affect behavior of the PSM mechanism. As a result, this inability leads to degradation of the power saving performance. In addition, we discover some interesting additional active time of commercial smartphones called hidden tail-time. To overcome inability of existing PSM mechanism, we have proposed a new packet scheduling scheme, EPS. EPS reduces power consumption caused by delay tolerant traffic by exploiting hidden tail-time. Evaluation results in practice shows that proposed EPS significantly reduces power consumption caused by delay tolerant traffic without significant transmission delay under common application’s traffic patterns. As a future work, we plan to implement accurate traffic identification technique, and to design more intelligent adaptive hidden tail-time window.

요 약 문

IEEE 802.11 power saving mode 기법은 스마트폰과 같은 모바일 단말의 파워 소비 절감을 위해 제안되었다. 이 기법은 모바일 단말의 WiFi 인터페이스의 상태를 액티브 상태와 슬립 상태간에 반복적인 변경을 통해 모바일 단말의 파워 소비를 줄인다. 액티브 상태에서는 슬립 상태에 비해 상당히 많은 양의 파워가 소비되기 때문에, 양 상태간에 전환은 신중히 이루어져야 한다. 즉, 소비 파워 절감 효과를 증가시키기 위해서 지연 내성 패킷과 같은 우선순위가 낮은 패킷들은 액티브 상태로의 전환을 야기시키지 말아야 한다. 하지만, 해당 기법은 매체 접근 제어 계층에서 동작하기 때문에 지연 내성 패킷 역시 해당 기법의 동작에 영향을 미친다. 이러한 문제를 해결하기 위해서, 본 학위 논문에서는 EPS 라 불리는 효율적인 패킷 스케줄링 기법을 제안한다. EPS 는 지연 내성 패킷을 지연에 민감한 패킷들과 분리 후, 이를 모바일 단말의 WiFi 인터페이스가 갖는 추가적인 액티브 시간동안 전송한다. 그 결과, 지연 내성 패킷들은 power saving mode 기법의 동작에 영향을 미치지 않는다. 상용 어플리케이션들의 트래픽 패턴상에서 진행된 실험 결과, EPS 는 기존의 power saving mode 기법에 비해 큰 지연 없이 더 나은 성능을 보였다.

REFERENCES

1. E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu, “NAPman: Network-Assisted Power Management for WiFi Devices,” Proc. ACM MobiSys, 2010

2. IEEE 802.11-2007, “IEEE Standard Part 11: Wireless LAN Medium Access Control and Physical Layer specifications, Jun. 2007.

3. R. Krashinsky and H. Balakrishnan, “Minimizing Energy for Wireless Web Access with Bounded Slowdown,” Proc. ACM MobiCom, 2002.

4. H. Liu, Y. Zhang, Y. Zhou, “TailTheft: leveraging the wasted time for saving energy in cellular communications," Proc. MobiArch, 2011.

5. M. Anand, E. B. Nightingale, and J. Flinn, “Self-Tuning Wireless Network Power Management.” Proc. ACM MobiCom, 2003.

6. J. Pyles, Xin Qi, G. Zhou, M. Keally, and Xue Liu, “SAPSM: Smart Adaptive 802.11 PSM for smartphones,” Proc. ACM UbiComp, 2012.

7. J. Li, J. Xiao, James Hong, and R.Boutaba, “Application-Centric Wi-Fi Energy Management on Smart Phone,” Proc. IEEE APNOMS, 2012.

8. J Liu and L. Zhong, “Micro power management of active 802.11 interfaces,” Proc. ACM MobiSys 2008.

9. MadWiFi project official webpage, http://madwifi-project.org

10. E. Tan, L. Guo, S. Chen, and X Zhang, "PSM-throttling: Minimizing Energy Consumption for Bulk Data Communications in WLANs," Proc. IEEE ICNP 2007.

11. Wireshark official webpage, http://www.wireshark.org.

12. K., Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," Proc. ACM SIGCOMM, 1997.

13. T. Karagiannis, D.. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," Proc. ACM SIGCOMM, 2005.

14. L. Bernaille, R. Teixeira, and K. Salamatian, "Early Application Identification," Proc. ACM CoNEXT, 2006.

15. M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," Proc. ACM SIGCOMM 2007.

16. Guown Xie, Marios Iliofotou, Ram Keralapura, Michalis Falouitsos, and Antonio Nicci, "SubFlow: Towards Practical Flow-level Traffic Classification," Proc. IEEE INFOCOM, 2012.

17. J. Zhang, Y. Xiang, Y. Wang, W. Zhow, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," IEEE Trans. Parallel

Distrib. Syst., vol. 24, no. 1, pp. 104-117, Jan. 2013.

18. Monsoon solutions inc., http://www.msoon.com.

19. Tcpdump officail webpage, http://www.tcpdump.org

감사의 글

주위 모든 분들의 관심과 도움으로 무사히 석사과정 졸업을 앞두게 되었습니다. 이 글을 통해 짧게나마 고마움을 표현하고자 합니다.

먼저 세상에서 가장 소중한 우리가족에게 감사함을 전합니다. 많은 연세에도 불구하고, 철없는 아들 뒷바라지 해주시느라 여전히 고생하시는 부모님 생각을 하니 많이 속상하고 죄송해요. 아직은 제가 해드릴 수 있는 게 없어서 너무 속상하지만, 꼭 성공해서 못 해드린 거 모두 해 드릴께요. 평소에 부끄러워서 표현을 잘 못했지만 진심으로 감사하고 사랑합니다. 아빠, 엄마 항상 건강하셔야 되요. 철없는 동생 뒤에서 챙겨주느라 고생하는 누나! 항상 미안하고 고맙고 사랑해. 민지야, 삼촌이 평소에 시간없다고 못 놀아줘서 미안해. 그래도 삼촌이 항상 우리 민지 사랑한다!

서영주 교수님 감사합니다. 교수님의 따뜻한 관심과 배려로 낯선 포항에서 잘 적응하고 즐겁게 지낼 수 있었습니다. 또한, 교수님의 정성 어린 조언 및 지도 덕에 한 단계 더 발전 할 수 있었습니다. 이점 다시 한번 감사드리며, 박사과정에 진학해서도 계속 발전하는 모습 보여드리도록 하겠습니다. 더불어, 바쁜 와중에 논문 심사를 위해 시간 내어주신 유찬수 교수님, 송황준 교수님께도 감사의 말씀을 전합니다.

2 년동안 연구실에서 같이 지내온 연구실 식구들에게도 고마움을 전합니다. 비록 몇 달 보진 못했지만, 짧은 기간에도 많은 관심 가져주신 동욱이형, 감사합니다. 보고 싶은 정윤이형, 형 덕분에 많이 배우고 연구실에 잘 적응할 수 있었습니다. 항상 큰 힘이 되던 격려와 조언도 감사드립니다. 맏형으로서, 랩장으로서 연구실을 이끌고 있는 경학이형 감사합니다. 연구실 초기부터 많은걸

알려주고 도움 준 솔선수범 재필 선배, 보고 있으면 기분이 좋아지는 상욱씨, 조용하지만 정감이 가는 석성씨 모두 감사합니다. 나이 많은 동기 기분 맞춰주느라 고생 많이 하는 우중이, 즐거운 연구실 만들어주는 귀여운 재국이, 논문쓰면서 같이 고생도 많이 하고 항상 옆에서 기분 좋게 만들어주는 은근 듬직한 효련이 모두 감사합니다. Chenglong and Yuepeng, thank you for sharing happiness. 항상 말 잘 듣고 잘 따라와주는 사랑하는 후배들인 훈훈한 승호, 믿음가는 성중이, 똑똑한 하림이, 귀여운 시영이 모두들 고맙습니다. 앞으로도 지금까지처럼 즐거운 연구실 생활 하길 기대합니다.

마지막으로 멀리 있지만 항상 묵묵히 응원해주는 우리 작전 패밀리분들과, 같이 있으면 즐겁고 힘이나는 소중한 내친구들, 멋진 변호사가 될 우리의 자랑 민혁이, 고등학교 때부터 롤 모델이었던 잘생긴 진화, 우리의 엘리트 유머감각 남다른 남준이, 꿈을 이뤄서 더 멋진 특공대 화신이 그리고, 동생이지만 형 같고 듬직해서 더 좋은 건우에게도 감사하다는 말씀 전합니다.

더욱 노력해서 소중한 모든분들의 기대에 부응하도록 하겠습니다.

감사합니다.