VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Li Jie

# Performance Evaluation of Different TCP Congestion Control Schemes in 4G System

Information Technology

2013

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

## ABSTRACT

| | |
|---|---|
| Author | Li Jie |
| Title | Performance Evaluation of Different TCP Congestion Control Schemes in 4G System |
| University | Vaasa University of Applied Sciences |
| Year | 2013 |
| Language | English |
| Pages | 58 |
| Name of Supervisor | Gao Chao |

TCP has variable congestion control schemes used both in wired and wireless network, such as Cubic, Reno, Vegas, Westwood and WinSock. As most cellular systems have already migrated to 3G and 4G is emerging from the horizon, it would be meaningful to know which scheme performs the best in 4G wireless network. This thesis aims at testing TCP communication performance by using different congestion control schemes in 4G system and helping mobile industries and application developers to select a correct scheme. Different FTP methods were used. In total, 1200 samples were collected by network monitors. They are analyzed and processed using Microsoft Excel. The statistics analysis results indicate that WinSock scheme performs the best in the case of downlink and Westwood behaves the best when the network condition is not good.

CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ACK | Acknowledge |
| AMTS | Advanced Mobile Telephone System |
| BDP | Bandwidth Delay Product |
| CDMA | Code Division Multiple Access |
| ECN | Explicit Congestion Notification |
| FDMA | Frequency Division Multiple Access |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile |
| GSM EDGE | Mobile Enhanced Data Rates for GSM Evolution |
| HSDPA | High-speed Downlink Packet Access |
| HSUPA | High-speed Uplink Packet Access |
| iDEN | Integrated Digital Enhanced Network |
| IMTS | Improved Mobile Telephone Service |
| IS-95 | Interim Standard 95 |
| LTE | Long Term Evolution |
| MTS | Mobile Telephone Service |
| PDC | Personal Digital Cellular |
| PTT | Push to Talk |
| RTT | Round-trip Time |
| TDMA | Time Division Multiple Access |
| UMTS | Universal Telecommunication System |
| VOAS | Vaasan Opiskelija Asuntosäätiö, Vaasa Student Housing |
| W-CDMA | Wideband-CDMA |
| WiMAX | Worldwide Interoperability for Microwave Access |

## LIST OF FIGURES

## LIST OF TABLES

# 1. INTRODUCTION

## 1.1 The Evolution of Cellular Network

In late 1970s, the first cellular communication system was established in Japan and the concept of cellular was put forward. From then on, cellular wireless network came into use and opened its market. Cellular wireless network is a mobile network with a dedicated range of radio frequencies in a certain area named cell. The dedicated frequencies in one cell are different from those in its neighborhoods so as to avoid bandwidth interference. Every cell contains at least one base station working as a transceiver. It allows roaming from one cell to another, namely handover.

The evolutions of cellular network generations, from 0G to 4G, contribute to a faster modern world. The analog telecommunication was the only choice after it was introduced in 1980s. It was used in MTS (Mobile Telephone Service), AMTS (Advanced Mobile Telephone System), and IMTS (Improved Mobile Telephone Service), PTT (Push to Talk). FDMA (Frequency Division Multiple Access) is a technique used by all the 1st generation cellular systems. The speed of 1G is in the range of 28Kbit/s to 56Kbit/s with the actual download speeds from 2.9KB/s to 5.6KB/s.

The 2G (second generation) digital replaced 1G in the early 1990s. In 2G, the voice during a call is digitally encoded. There are several main 2G standard either TDMA-based (Time Division Multiple Access) or CDMA-based (Code Division Multiple Access): GSM (Global System for Mobile) (TDMA), IS-95 (Interim Standard 95) (CDMA), PDC (Personal Digital Cellular) (TDMA), iDEN (Integrated Digital Enhanced Network) (TDMA). 2G is the predecessor of 2.5G (second and half generation). In 2.5G, a packet-switched domain is added to the circuit switched domain. [1] GPRS (General Packet Radio Service) is the main utility method at the speed from 56Kbits/s to 115Kbits/s.

2G is the antecedent to 3G. 3G technologies enable network operators to offer users a wider range of more advanced services while achieve greater network capacity through improved spectral efficiency. [1] On the opposite of IEEE 802.11 (Wi-Fi and WLAN), 3G covers wide area with high-rate. The speed is up to 14.4Mbit/s in the downlink and up to 5.8Mbit/s in the uplink. W-CDMA (Wideband-CDMA), UMTS (Universal Telecommunication System), GSM EDGE (Mobile Enhanced Data Rates for GSM Evolution), CDMA 2000 and WiMAX (Worldwide Interoperability for Microwave Access) are main standards in 3G. 3.5G includes HSDPA (High-speed Downlink Packet Access) up to 8-10Mbit/s in the downlink. 3.75G is HSUPA (High-speed Uplink Packet Access) up to 1.4Mbit/s in the uplink.

In 2006, pre-4G system WiMAX came out after the emerging of 3G for four years. Three years later, LTE (Long Term Evolution) was on the road. LTE is a technology that is reverse compatible. When the device goes outside the coverage of a 4G network, it automatically switches on to the fastest available alternative, which is usually the 3G network.[2] The speed is about 100Mbit/s for mobile users and 1G in maximum for fixed devices. Moreover, 4G unifies cellular and wireless local area networks, and introduces new routing techniques. It provides efficient solutions for sharing dedicated frequency bands, increased mobility and bandwidth capacities. [3]

## 1.2 Motivations for Conducting This Project

In recent years, as huge expenditure and large labors have been spent in setting up and maintaining new 4G wireless communication system, the performance improvement is of great importance. One of the main solutions to improve the performance is improving TCP performance. [3] Therefore, selecting an appropriate TCP congestion control scheme imposes the throughput of network to a large extent. In addition, a correct scheme benefits the performance of mobile devices and applications.

In today’s wired networks, packet loss is primarily caused by congestion in intermediates. It is known that TCP congestion control schemes are designed for wired networks. In Section 2.3, several congestion control algorithms are described in details.

In wireless networks, transmission and reception errors become significant reasons of packet loss. At the same time, congestion is still a possible reason. The utility of TCP congestion control scheme in 4G wireless network might be an interesting topic. Whether TCP congestion control schemes optimize the 4G performance and which scheme performs the best are pending issues. These need inspecting and verification. As reported in previous research, the performance of traditional transport protocols such as TCP degrades significantly over a wireless link. [4] Before testing the schemes in 4G network, collecting some conclusions under 3G system will be helpful. Authors in [5] have proved that (i) the considered TCP congestion control algorithms (New Reno, BIC, Westwood+) performed similarly both in downlink and uplink scenarios; (ii) the UMTS (Universal Telecommunication System) uplink channel did not exhibit any remarkable issues, providing good channel utilization and very low number of timeouts and packet retransmissions; (iii) a very high number of timeouts has been observed in our measurements in the case of downlink channel that does not seem to be caused by congestion. [5]

According to wide review, I found that most of the works were conducted based on simulations of 4G wireless networks rather than live 4G, such as [3], [6], [7], [8], [10], [12]. However, wireless in reality will be affected by variable factors, such as obstacles and radio interference. Besides, no comparison of TCP congestion control algorithms between Linux and Windows was discussed before. Upon the consideration of the untruthfulness, this paper is going to talk about the performance of five TCP congestion control schemes under real 4G wireless network in Vaasa, Finland.

## 1.3 Outline of the Rest of the Thesis

This paper proceeds as follows. The Chapter 1 introduces the evolution of cellular networks, motivation and the outline of this project. The second chapter gives descriptions of five TCP congestion control schemes. Chapter 3 focuses on the solution and procedures in detail. Chapter 4 illustrates all the results both and discusses the limitations and future directions. At last, Chapter 5 draws conclusions.

# 2. BACKGROUND OF TCP CONGESTION CONTROL

Congestion control is a set of behaviors determined by algorithms that each TCP implements in an attempt to prevent the network from being overwhelmed by too large an aggregate offered traffic load. [15]

## 2.1 Differences between Wireless and Wired Channels

The signal or data can be transmitted through channels. There are two types of channels. One is wire-based, transmitting signal by twisted-pair wire, coaxial cable and optic fiber cable. This is called wired channel. The other type is wireless channel. It allows information transmits from one device to another device without the wires or cables, but normally over the radio channel. It outweighs wired communication in extensibility to remote area or satellites and roaming freedom.

TCP was originally designed for fixed end-systems and fixed/wired networks. TCP assumes congestion if packets are dropped. The performance of an unchanged TCP degrades severely if don’t customize the scheme according to situations. However, TCP cannot be changed fundamentally due to the large base of installations in the fixed network. Mobility of TCP has to remain compatible. [16]

When it comes to the wireless network, TCP’s performance is different. In reality, packet loss is typically due to bit errors due to wireless channel impairments, handoffs, possibly congestion. Bursts of errors are due to low signal strength or noise. These errors lead to more than one packet lost in channel. This is likely to be detected as a timeout. In addition, delay is often very long, because round-trip time can be very long and variable and timeout mechanisms may not work well. Furthermore, asymmetric links cause delayed ACKs (Acknowledge) in the forward or reverse direction, which can limit throughput in the other direction. [16]

Table 1 and Table 2 are about the advantages and disadvantages of wireless and wired channels: [17], [18], [20]

| Wired Channel | |
|---|---|
| Advantages | Disadvantages |
| Cheap in equipment and maintenance | Affected by moisture |
| Better security | Affected by noise generated by machinery and magnetic |
| High speed | Possible disorganized |
| High QoS (Quality of Services) | Adding more computers to a wired network may result in unexpected expense if you run out of connections on your network and could slow down the network |
| Avoid interference from other wireless signal | |

**Table 1 The Advantages and Disadvantages of Wired Channel**

| Wireless Channel | |
|---|---|
| Advantages | Disadvantages |
| Convenient, flexible | Vulnerable to interference (microwave), or obstructions, like walls |
| Neat and, clean, no untidy cables | Expensive |
| | Reliability is not good. Once one major section breaks down, the whole network will be affected. |

**Table 2 The Advantages and Disadvantages of Wireless Channel**

## 2.2 The Usability of TCP in the Wired Network

In the Internet protocol suite, commonly known as TCP/IP, a four layered standard, Transport Layer is the third layer, responding for the end-to-end data transfer by transmitting data from its upper layer to a remote device. [13] Figure 1 shows the four layer TCP/IP models.

| | |
|---|---|
| Application Layer | Application Layer |
| Presentation Layer | |
| Session Layer | |
| Transport Layer | Transport Layer |
| Network Layer | Internet Layer |
| Datalink Layer | Network Access Layer |
| Physical Layer | |

**Figure 1 Four Layer TCP/IP Models [13]**

TCP (Transmission Control Protocol) is the most widely used transport layer protocol, providing connection-oriented reliable delivery, congestion control, flow control, retransmission and error detection.[14] Most of the user application protocols use TCP, for example Telnet, FTP and HTTP.

Connection-oriented reliable delivery is guaranteed by connection establishment and termination. In order to keep the reliability and packets in order when they arrive, TCP protocol breaks the file into a number of packets and attaches a sequence number to each packet before transmitting. The confirmation of ACK (acknowledge) from its peer transport layer is compulsory to confirm a successful transmission. Flow control capabilities take effects between devices. They utilize a special sliding window acknowledgement system to keep optimizing transmitting and receiving speed and address other issues. [19] The congestion window size will be adapted to the condition of packet loss automatically. Retransmission plays a key role in TCP. TCP designs the sequence numbers which are attached to an amount of segments after dividing a big file before transmission. An ACK from receiver demonstrates the packet is delivered successfully. Retransmission will be launched when unacknowledged packets are detected. Figure 2 shows how packets are transmitted.

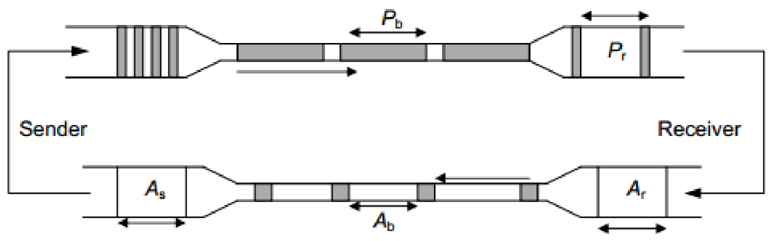


**Figure 2   Sender and Receiver [15]**

Sender transmits packets ($P_b$) to receiver. When receiver gets the packet ($P_r$), it will generate an ACK ($A_r$) to Sender. The arrival of ACK ($A_s$) triggers sender to transmit a new packet, providing an "ACK clock". This system is called "self-clocked" [15]

## 2.3 The Usability of TCP Congestion Control Algorithms in the Wired Network

In wired network, when a host sends a file composed of hundreds of packets to another host in remote area through cables, the packets will pass through numerous intermediates. Every intermediate has a buffer with certain size. If the connection becomes so busy that buffers are saturated, the speed will be reduced. The packet may even drop if buffers are overflowed. The delay of arriving packets and packet loss that are caused by huge traffic in the networks are called congestion [3], [8].

When congestion increases enormously on the network, segments would be delayed or dropped. This will lead to retransmission, even timeout. At the same time, TCP retransmits the lost packets. Consequently, the network will suffer from increasing burdens and finally is exhausted if no congestion control algorithms handle it. Performance decreases tremendously, resulting in congestion collapse. [19]

Therefore, TCP congestion control schemes are developed to deal with congestions in wired network. TCP congestion control comes into force when congestion happens. These schemes aim at adjusting the transmission rate to a more appropriate one to avoid the network congestion. In the very early TCP standard, RFC 793, there is little discussion about TCP congestion control schemes due to implementation problems in discovery congestion. After several upgrades, the author in [21] explained the schemes mentioned in RFC 793 in details. In [22], several new congestion control algorithms come out. They are known as TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

Congestion Control Scheme decreases the transmitting speed when congestion happens or is about to happen. The most difficult part is how to detect the underlying overload to avoid congestion in advance, how to slow down the sending rate and when to increase the sending rate again. There are many variants of TCP congestion control schemes. Taking the operating system hosting TCP/IP stacks into consideration, some different and veiled schemes from public notice are still exiting. For example, Linux system is open source so TCP congestion control schemes can be modified by commands. On the other aspect, Windows system is not open source so that TCP congestion control scheme can’t be known by customers.

Before TCP congestion control algorithm addresses the problem, congestion should be detected or predicted in some ways. This is always implemented by measuring delays, network-supported ECN (Explicit Congestion Notification) or lost packets, in other word, more ACKs of previous packets that have been sent. [15] Whenever the congestion or coming congestion is detected, sending speed will be slowed down under window control mechanism. The estimate value of the network available capacity is called the congestion window, written more compactly as simply $cwnd$. [15] As far as the sender’s window size $w$ is the minimum of the receiver’s window ($rwnd$) and the congestion window ($cwnd$), overload and packet dropping will be avoided. This can be expressed as hereinafter:

$$w = \min(cwnd, rwnd) \tag{1}$$

Then, it comes to two main algorithms of TCP: Slow Start and Congestion Avoidance, first mentioned in [15]. Under the principle of packet conservation and ACK clocking (see section 1.2), these two schemes work alone but will switch back and forth between each other.

### 2.3.1 Slow Start

The Slow Start is launched when establishing a new connection, when a retransmission time is expired or TCP has gone idle for some time. [15] It will switch to Congestion Avoidance after a packet loss within retransmission time. The main objective is probing a specific $cwnd$ exponentially before exploring for more available bandwidth under steady state in Congestion Avoidance. The Congestion Avoidance phase avoids congesting the network with an inappropriately large burst of data [21]. This growth seems quite "fast" (increasing as an exponential function) but is still "slower" than what TCP would do if it were allowed to send immediately a window of packets equal in size to the receiver's advertised window. (Recall that W is still never allowed to exceed $awnd$.) [15] For simplicity, initial window ($W$) is one SMSS (Sender Maximum Segment Size, the value actually refers to the maximum amount of data that a segment can hold, not including TCP headers [19]), although it's allowed to be up to three or four segments. [21]

$$SMSS = min\big(receiver's\ MSS, path\ MTU\ (Maximum\ Transmission\ Unit)\big) \tag{2}$$

When no packets drop and every segment is ACKed, $W$ will increase one SMSS according to every valid ACK. Hence, the $W$ will increase to 2 and two SMSS will be sent next time. After $n$ round-trip, $W = 2^n$. That is $n = \log_2 W$. Obviously, the $cwnd$ will rise quickly to send large amount of packets, which overloads the capacity of network finally. Once a packet drops, TCP switches to Congestion Avoidance. At the same time, $ssthresh$ ($slow\ start\ thresh$) is set to the half of the current $cwnd$ value and $cwnd$ is set to $ssthresh$. Therefore, $sthresh$ always dynamically records the last best of $W$ without packet loss.

$$ssthresh = max(\frac{flight\ size}{2}, 2 * SMSS) \qquad (3)$$

$flight\ size$ is the amount of outstanding data in the network. [21]

The larger one between $cwnd$ and $ssthresh$ decides which algorithm should be chosen. When $cwnd$ < $ssthresh$, Slow Start is chosen. When $cwnd$ > $ssthresh$, Congestion Avoidance is chosen. When $cwnd = ssthresh$, either can be used. [15]

## 2.3.2 Congestion Avoidance

Once the Slow Start is achieved, there is always the possibility that more network capacities may become available. [15] If the $cwnd$ still increments exponentially, the available bandwidth will be saturated immediately and causes packet dropping. Congestion Avoidance adopts sub-linear increment rather than exponential increment. The $cwnd$ will be added maximum one $SMSS$ when receiving a valid and non-duplicate ACK, normally a slight growth. The $cwnd$ will change according to this formula:

$$cwnd_{t+1} = cwnd_t + SMSS * SMSS/cwnd_t \qquad (4)$$

Since TCP uses the integer arithmetic, this formula will yields 0 if the $cwnd$ is larger than $SMSS * SMSS$. When this happens, the result will be rounded up to 1 Byte. [21]

## 2.3.3 Fast Retransmit

Fast Retransmit avoids the expiration of retransmission time when a segment is lost. As we all know, ACK will inform the sender to send the packet with a sequence number that receiver wants to receive next. As long as there is reordering of segments, or packet loss, the receiver will send a duplicate ACK. [21] One or two duplicate ACKs can’t guarantee packet loss. [14] If the sender receives three or more ACK with the same sequence number, it strongly demonstrates packet loss. The sender will immediately retransmit the packet with the required sequence number. [19]

### 2.3.4 Fast Recovery

After executing the fast transmit, TCP doesn't come back to the Slow Start, instead, Congestion Avoidance. This process is governed by Fast Recovery algorithm until a non-duplicate ACK arrives. [21] It is an improvement that allows high throughput under moderate congestion, especially for large windows. [15] It's not efficient to return $cwnd$ to one segment by adopting Slow Start. So, the current phase switches to Fast Recovery. In Fast Recovery, the $ssthresh$ will be set to half of the current $cwnd$ and then $cwnd$ will be set to $ssthresh$ plus 3* $SMSS$. [15] Then, $cwnd$ will be incremented by 1 $SMSS$ when receiving a duplicate ACK in order to state the flowing packets that have left the network. [15], [21] When a valid ACK (non-duplicate) arrives, indicating TCP is recovered, $cwnd$ is set to $ssthresh$ again.

### 2.3.5 Reno

A single algorithm was not enough to address complex congestion issues, so combined algorithms came out. The first congestion control algorithm was introduced in the late 1980s. It was first used in the 4.2 release of UC Berkeley's version of UNIX, called Tahoe or the Berkeley Software Distribution (BSD UNIX). [15] It is composed of Slow Start, Congestion Avoidance and Fast Retransmit. One problem of Tahoe is that whenever retransmission happens, TCP will go back to the Slow Start. This will cause underutilizing of available bandwidth.

In order to handle the underutilizing of bandwidth, Reno was introduced. Reno is the offspring of Tahoe, namely, 4.3 version BSD. Compared to Tahoe, Reno has Fast Recovery. $Cwnd$ will be set to latest $ssthresh$ instead of 1 $SMSS$. TCP Reno became very popular and was called "standard TCP." [15] Figure 3 shows the procedure of Reno. Figure 4 illustrates how $cwnd$ behaves when Reno is in different states.

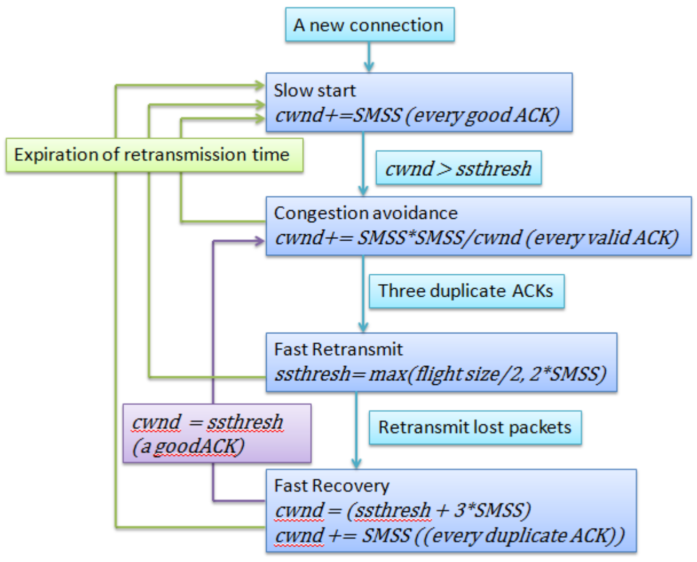

A new connection
Slow start
cwnd+=SMSS (every good ACK)
Expiration of retransmission time
cwnd > ssthresh
Congestion avoidance
cwnd+= SMSS*SMSS/cwnd (every valid ACK)
Three duplicate ACKs
Fast Retransmit
ssthresh= max(flight size/2, 2*SMSS)
cwnd = ssthresh (a good ACK)
Retransmit lost packets
Fast Recovery
cwnd = (ssthresh + 3*SMSS)
cwnd += SMSS ((every duplicate ACK))

**Figure 3 The procedure of Reno**

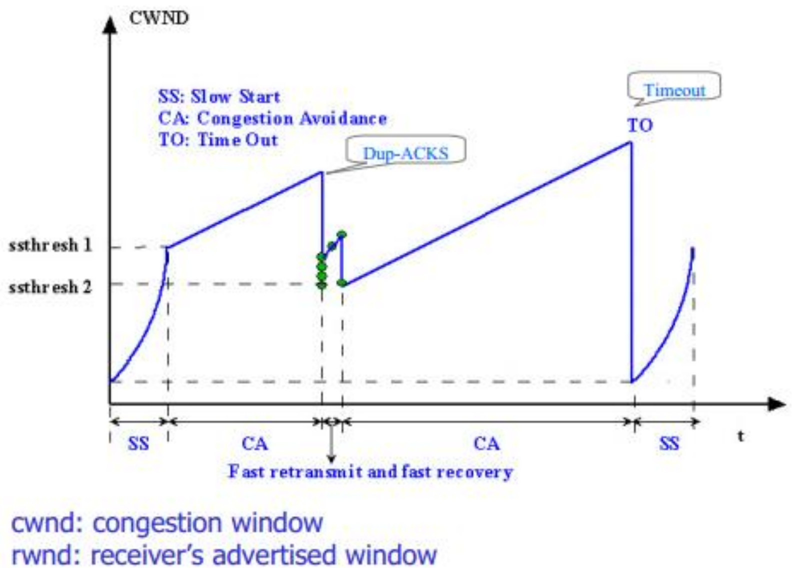


**Figure 4 *Cwnd* of Reno [23]**

### 2.3.6 Cubic

When it comes to the High-Speed Network, normally, 10G Ethernet and such, Reno is not an efficient TCP congestion control algorithm anymore. It is impossible for Reno to increase *cwnd* large enough to take the most use of available bandwidth in a short time. This leads to the release of BIC-TCP in version 2.6.8 of Linux kernels. BIC provides linear RTT (round-trip time) fairness in high-bandwidth network.

Aiming at removing the over aggressiveness of BIC, Cubic uses a particular cubic function to control $cwnd$. Cubic function describes a line containing two shapes, convex in the former part and concave in the latter part. The formula is as follows:

$$W(t) = C(t-K)^3 + W_{max} \qquad (5)$$

$C$ is a constant parameter (default 0.4), $t$ is the elapsed time since the last window decreases. It is used to guarantee the RTT fairness. And $K$ is the time period the function takes to increase $W$ to $W_{max}$ when there is no loss event. [15] When there is Fast Retransmit, new $cwnd$ and $ssthresh$ are set to $\beta * cwnd$, $\beta$ is the multiplicative decrease constant (default 0.8), and $W_{max}$ is set to $cwnd$. [15] The next objective $cwnd$ is $W(t+RTT)$. Cubic helps $cwnd$ increase quickly, stay large. Figure 5 shows how $cwnd$ behaves.

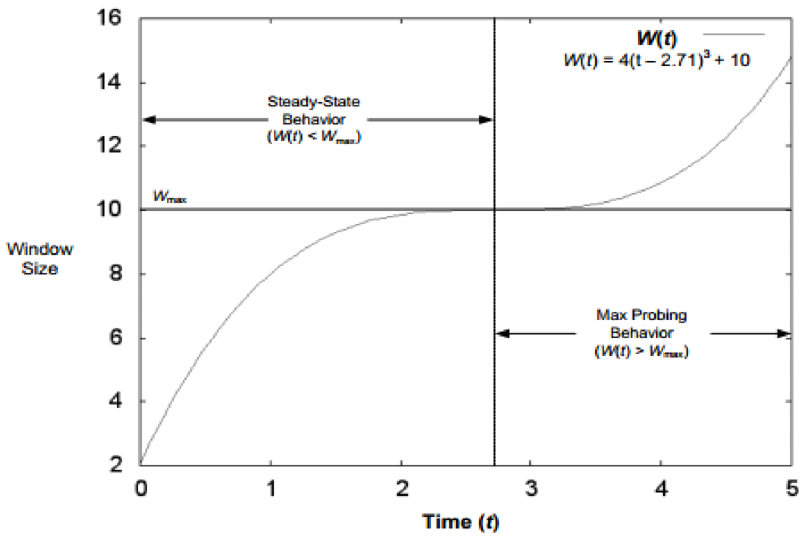


**Figure 5 C*wnd* of Cubic**

### 2.3.7 Vegas

In the above two algorithms, the congestions are always indicated by the ACK, packet loss or the expiration of retransmission time. Another index may indicate the congestion is RTT. The next two TCP congestion control schemes are based on the RTT. They may provide a more effective way for wireless network.

The first delay-based algorithm is Vegas, born in 1994. It is supported by Linux, but not set as the default. [15] Vegas fixes a certain problem in Reno by taking the advantage of RTT. In Reno, the increase of $cwnd$ won't stop unless there is packet loss. In a different way, Vegas evaluates optimizing $cwnd$ by monitoring the throughput ($cwnd/RTT_{min}$). A range of throughput, from $\alpha$ to $\beta$, is set. If the deviation between real throughput and expected throughput is within the range, no change happens. If the deviation is less than $\alpha$, $cwnd$ increases. If the deviation is larger than $\beta$, $cwnd$ decreases. This is called AIAD (additive increase/ additive decrease).The algorithm will adjust $cwnd$ to the better throughput. It doesn't perform well under all the conditions. [24]

### 2.3.8 Westwood

Westwood focuses on RTT to estimate the effective bandwidth (ERE, eligible rate estimate) of the channel, in a similar way to Vegas. The difference is that Westwood uses a variable measurement interval to calculate the bandwidth based on ACK arrivals. [15] The interval will be small if congestion is low, vice versa. When a packet drops, Westwood calculates a new BDP (Bandwidth delay product, $ERE * RTT_{min}$) and assigns it to $ssthresh$ rather than half the $ssthresh$.

Westwood is available since version 2.6.13. Its authors claim that Westwood is especially good for wireless links or other situations where the packet loss may have nothing to do with congestion. [24]

### 2.3.9 WinSock

Windows is not open source, so no information about WinSock is available.

## 2.4 Command Lines in Linux Used to Change TCP Congestion Control Schemes

Linux kernel supports pluggable Congestion Avoidance modules since version 2.6.18. [15], [24] To get a list of congestion control algorithms in the kernel, this command should be typed in:

```
sysctl net.ipv4.tcp_available_congestion_control
```

This command usually outputs the default algorithms. [25]

In order to modify the congestion control algorithm, namely, tuning, this command should be used:

```
sudo /sbin/sysctl -w net.ipv4.tcp_congestion_control=Vegas
```

Definitely, you must type in the superuser (sudo) password for every first time modifying.

# 3. METHODOLOGY

## 3.1 Proposed Solution

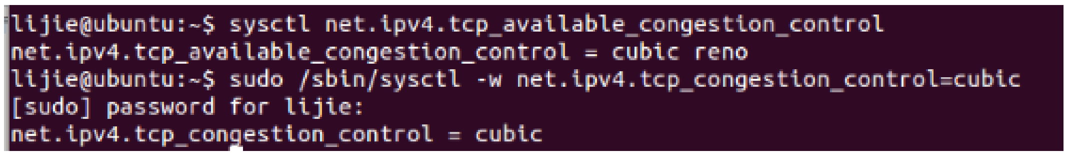
Aiming at modifying variable congestion control algorithms, Version 12.10 Linux Kernel is installed. There are many congestion control algorithms available in Linux currently, major four are Cubic, Reno, Vegas and Westwood. Windows TCP is implemented as part in WinSock, so the scheme in Windows can be called “WinSock”. Totally, five algorithms will be discussed.

TCP performance evaluation of different congestion control algorithms can be viewed from transmitting rates. A network monitor is needed to generate speed plots. From the pictures of flow waves, the time durations of downloading and uploading a file can be recorded. Therefore, average flow speeds can be worked out. In order to record trustful transmission speed, quantitative research method is selected. Statistics is the mainstream solution.

Due to the fact that the 4G wireless channel is asymmetrical, the transmission rates of downloading and uploading are different. Hence, both directions should be considered.

In order to find the performance differences under different scenarios, data from three different places and four fixed periods in a day will be obtained.

Figure 6 shows how the modifying and confirmation are implemented in this way under Linux:



```
lijie@ubuntu:~$ sysctl net.ipv4.tcp_available_congestion_control
net.ipv4.tcp_available_congestion_control = cubic reno
lijie@ubuntu:~$ sudo /sbin/sysctl -w net.ipv4.tcp_congestion_control=cubic
[sudo] password for lijie:
net.ipv4.tcp_congestion_control = cubic
```

**Figure 6 Method of modifying algorithm**

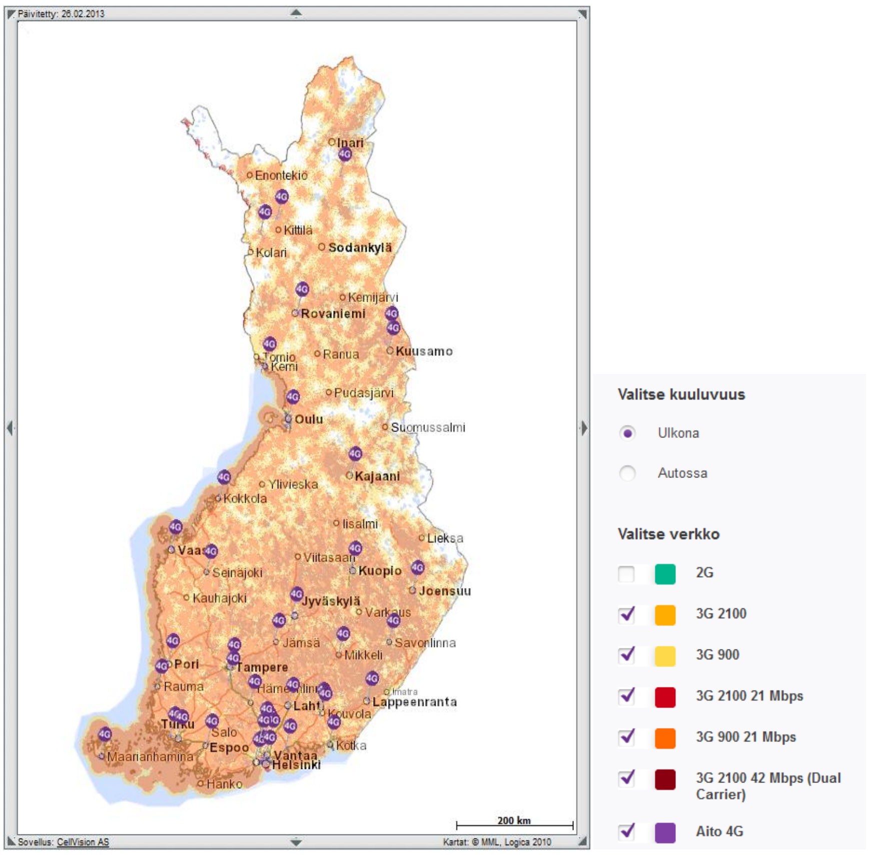
Then, the raw data will be processed in EXCEL to get these values: 1. Average speeds in fixed periods; 2. Average speeds of different schemes. Finally, compare and analyze the speeds to draw conclusions.

In short, downlink’s and uplink’s average speeds of five distinctive schemes will be computed at three different scenarios during four periods for ten times on different working days.

## 3.2 Testing Environment

TeliaSonera launched first 4G commercially in Stockholm and Oslo in December 2009. [9] One year later, it firstly launched the 4G network in Finland in December 2010. Sonera provides a 3G and 4G combined network covering 95% area in Finland. [26] Figure 7 shows the Sonera’s 4G coverage in Finland. The 4G wireless services support up to 100Mbit/s in downlink and 50Mbit/s in uplink. [27] Under normal circumstances, the speed is 20-80 Mbit/s. [11]

The Sonera website released that the 4G wireless network was available in Vaasa on May 22nd, 2012. [28] Figure 8 shows the Sonera’s 4G coverage in Vaasa.



Valitse kuuluvuus

[● Ulkona]

○ Autossa

Valitse verkko

☐ 🟩 2G

✓ 🟧 3G 2100

✓ 🟨 3G 900

✓ 🟥 3G 2100 21 Mbps

✓ 🟧 3G 900 21 Mbps

✓ 🟥 3G 2100 42 Mbps (Dual Carrier)

✓ 🟪 Aito 4G

**Figure 7 Sonera’s 4G System coverage in Finland**

## 3.3 Sample and Data Collection

### 3.3.1 Place Selection

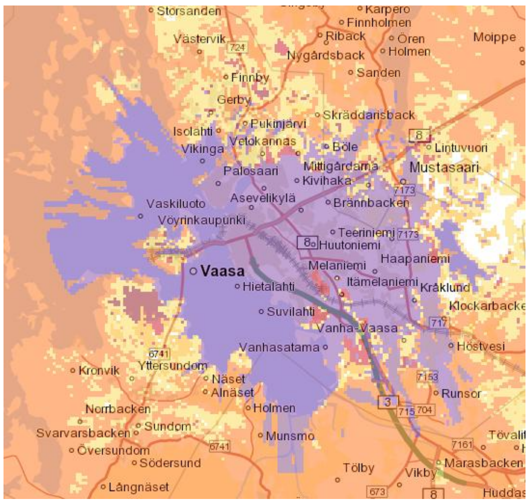


**Figure 8 Sonera's 4G network coverage in Vaasa**

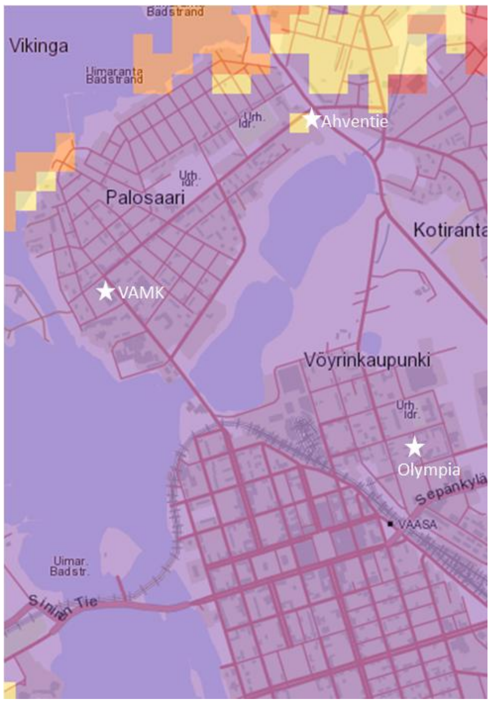


**Figure 9 Three places**

Based on the 4G map, it's obvious that city center has strong power of 4G. Hence, VOAS (Vaasan Opiskelija Asuntosäätiö, Vaasa Student Housing) is selected. According to Figure 9, the place where I live, Building P at Ahventie is the edge of 4G. It is supposed that the edge of 4G may be a good place to test distinctive algorithms. The algorithms may display their congestion control abilities to the utmost. Another possible reason is that the weak-power place doesn't influence the performance of schemes too much. In order to refine the settings in Linux at this school, this place is also selected.

### 3.3.2 Time Selection

In order to make the evaluation more persuasive, tests were performed only on working days, and the same time periods were used for each test setup. The testing task of ten samples can be divided into three to four days with two to four times per day. Table 3 shows the fixed periods:

| | |
|---|---|
| Early morning | 6:00-7:30 (VAMK: 7:00-8:30) |
| Forenoon | 9:30-11:00 |
| Afternoon | 14:30-16:00 |
| Night | 19:30-21:00 (VAMK:18:30-20:00) |

**Table 3 The fixed periods**

## 3.4 Tools and Methodology

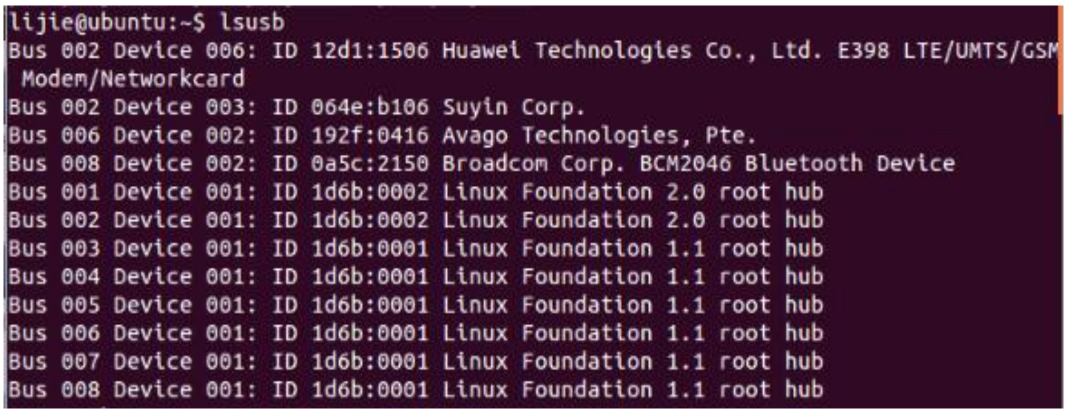
### 3.4.1 Huawei 4G Modem

VAMK friendly subscribes a mobile broadband internet stick, Huawei E392, and a mobile broadband package. Figure 10 shows the modem I use. According to [30], this modem can be used in Windows 7 and Linux operating systems. This modem can also connect to all the different networks, including 2G, 3G and 4G.

**Figure 10 Huawei E392 4G modem**

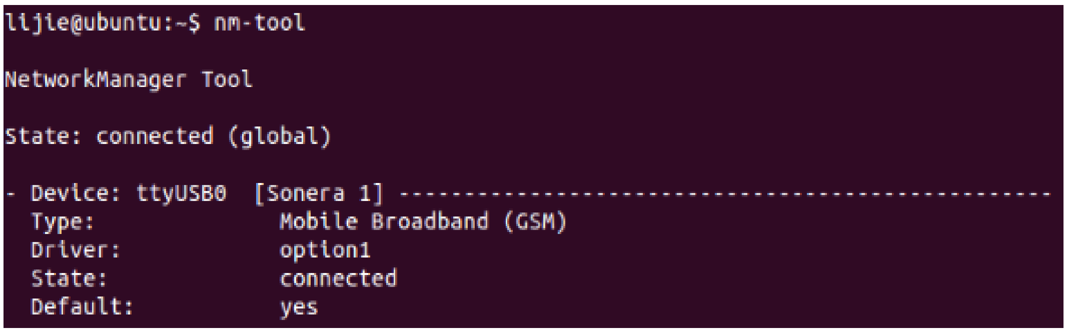
Before using the modem, it's necessary to install its driver. Then, it will connect to the primary network.

Figure 11 and 12 show what the terminal displays in Linux if the modem is connected to the 4G.



```
lijie@ubuntu:~$ lsusb
Bus 002 Device 006: ID 12d1:1506 Huawei Technologies Co., Ltd. E398 LTE/UMTS/GSM
 Modem/Networkcard
Bus 002 Device 003: ID 064e:b106 Suyin Corp.
Bus 006 Device 002: ID 192f:0416 Avago Technologies, Pte.
Bus 008 Device 002: ID 0a5c:2150 Broadcom Corp. BCM2046 Bluetooth Device
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 008 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

**Figure 11   The terminal displaying with modem plugged 1**



```
lijie@ubuntu:~$ nm-tool

NetworkManager Tool

State: connected (global)

- Device: ttyUSB0  [Sonera 1] -----------------------------------------------
  Type:              Mobile Broadband (GSM)
  Driver:            option1
  State:             connected
  Default:           yes
```

**Figure 12   The terminal displaying with modem plugged 2**

### 3.4.2 Download File and Upload File

After tens of testing, appropriate file sizes were found. In the case of downlink, the file less than 100MB is too small to tell the differences among distinctive schemes and the file larger than 400MB is beyond the window of plotter and also time-wasting. The uplink case has similar situations. The result is that 250MB-400MB file size for downlink and 20MB-30MB file size for uplink are most appropriate. These files with right sizes guarantee the time long enough to get a comparatively accurate average speeds and are proper for the plotter to display. As a result, a 293MB file and a 21.46MB file are selected.

Tests were conducted in this order: Ahventie, Olympia, and VAMK. At the very beginning, the VAMK's server was used as SFTP server. The reason I chose VAMK's server is that using domestic server in Finland is much more stable in wired section so we can focus on wireless link only. Considering web-based service would be a better way, I came up an idea to use gmail to send files. Unfortunately, after uploading tens of times, the google server closed my email-box because of incomprehensible frequent uploading. What is worse, the testing result may be affected by the google server performance. So I abandoned this idea.

WinScp Terminal in Windows and SecPanel Terminal in Linux were used as clients. Both are terminal-based clients. It is found that there is a deviation of four and six times of uplink speeds between Linux and Window at Ahventie and Olympia respectively. It was surmised that maybe different SFTP clients cause serious differences in two operating systems. I searched for about dozens of clients and servers, but I didn't find any available terminal-based clients that can be used in both systems. However, it is feasible to know how different the various SFTP clients perform if they are in a certain operating system. Therefore, some tests among diversified SFTP clients in Windows were conducted at Ahventie. The result shows that there is difference existing, but not to a large extent.

And before conducting tests at VAMK, it's figured out a better way to avoid the influence of different SFTP clients in different systems, that is Funet FileSender. This method is web-based service for sending large files. [31]

Funet is the shortage of Finnish University and Research Network. It is a backbone network providing Internet connection for Finnish universities and polytechnics as well as other research facilities. [32] Funet FileSender allows you to upload files with size up to 50 GB. The uploaded file will not be available after the expiration of a certain period. This period depends on the sender's preference. The Funet server will send a link to receiver to download. Therefore sending and receiving are possible without installing additional programs. [31] It can be accessed to the service window by choosing VAMK and typing in student number and password. The URL is https://filesender.funet.fi.

At the end of this part, Table 4 shows selected servers and clients. Table 5 shows the FTP clients in the additional tests.

| Place | Time | SFTP Server/Downloading | | SFTP Client/Uploading | |
|---|---|---|---|---|---|
| | | Linux | Windows | Linux | Windows |
| A | 10 | VAMK Server | VAMK Server | SecPanel | WinScp |
| O | 10 | VAMK Server | VAMK Server | SecPanel | WinScp |
| V | 10 | Funet Server | Funet Server | Funet Client | Funet Client |

**Table 4 The servers and clients selected**

| Place | Time | Client in WinSock/Uploading |
|---|---|---|
| A | 2 | WebDrive, WinScp, FileZilla, Funet |

**Table 5 The clients in the extra tests**

### 3.4.3 KNemo and Computing

There is traffic plotter function attached when plugging the modem in Windows.

KNemo is a network monitor in Linux. It displays an animated icon in the systray for every network interface. Besides, it provides a few interesting functions, such as traffic plotter, daily, monthly and yearly traffic statistics for each interface. It displays two directions' traffics, incoming and outgoing, through the same interface in one plot.

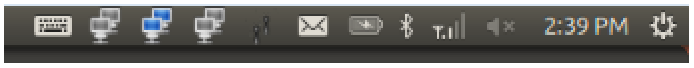


**Figure 13 KNemo icon in Linux**

PPP interface is selected here. The blue icon in Figure 13 shows the PPP interface. Once connected, the Show Traffic Plotter function will be available. The orange line represents the uplink, and the blue line represents the downlink.

The explanation of how to use plotter to calculate the speed rate is displayed.

Figure 14 is caught after two files uploaded to Funet server in the afternoon at VAMK. TCP congestion control scheme is Reno.

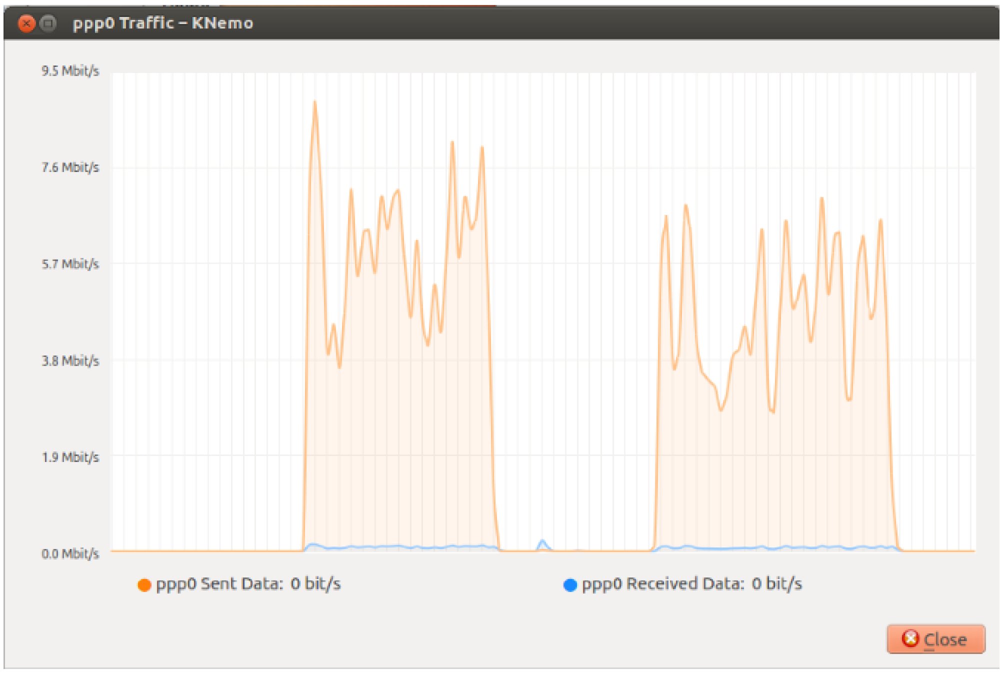


**Figure 14 Reno on March 13th in the afternoon at VAMK**

We just focus on the left plot. The file's size is 21.46MB.

Every grid stands for two second. There are 16 and half grids in the picture.

$$16 * 2 + 1 = 33s$$

Hence, the time duration is 33 second totally.

The speed is

$$21.46MB/33s = 0.65MB/s = 0.65MB/s * 8bits/Byte = 5.2Mb/s$$

# 4. RESULTS AND DISCUSSION

Totally 1200 samples were collected. As reported in Chapter 3, EXCEL was used to do some computes based on raw data. Due to the space limitation, all the data can't be exhibited, but the average value, comparison and analysis will be displayed.

## 4.1 Average Speeds in Four Periods

In this section, all the speeds under five schemes are summed up and averaged. The average values will be compared vertically and horizontally.

*The case of downlink flows (all are web-based)*

| Time Period | Ahventie | Olympia | VAMK |
|---|---|---|---|
| 6:00-7:30/Early morning | 23,28159 | 21,21891 | 25,4563 |
| 9:00-10:30/Forenoon | 22,188554 | 21,14571 | 25,81134 |
| 2:30-4:00/Afternoon | 22,15197 | 20,28833 | 25,09461 |
| 7:30-9:00/Evening | 22,137504 | 20,20011 | 24,62392 |

**Table 6 Average downlink speeds in fixed periods at different places**

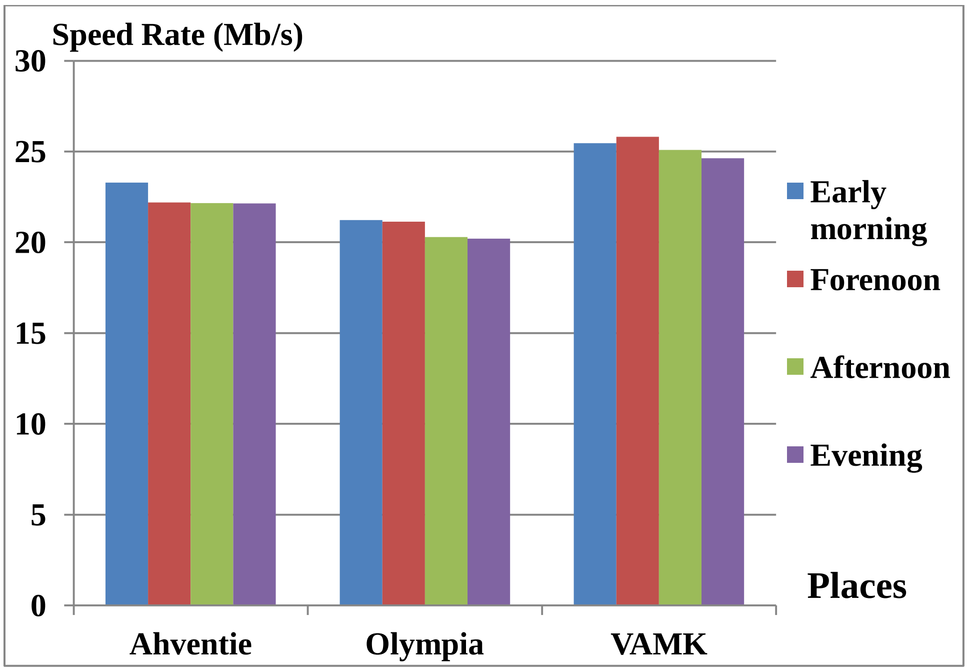


**Figure 15 Average downlink speeds in fixed periods at different places**

According to Table 6 and Figure 15, the descending order of the speeds of different periods is: early morning, forenoon, afternoon and evening. Occasionally, the speed in the forenoon at VAMK was higher than that in the early morning, but shortly apart. From the common sense we know that in the early morning, the channels are left more vacancies, resulting in higher speeds. With the increasing of working people, the channels become more occupied. It can be seen that the performance of TCP congestion control schemes decreases slightly while the channels turn into saturating gradually.

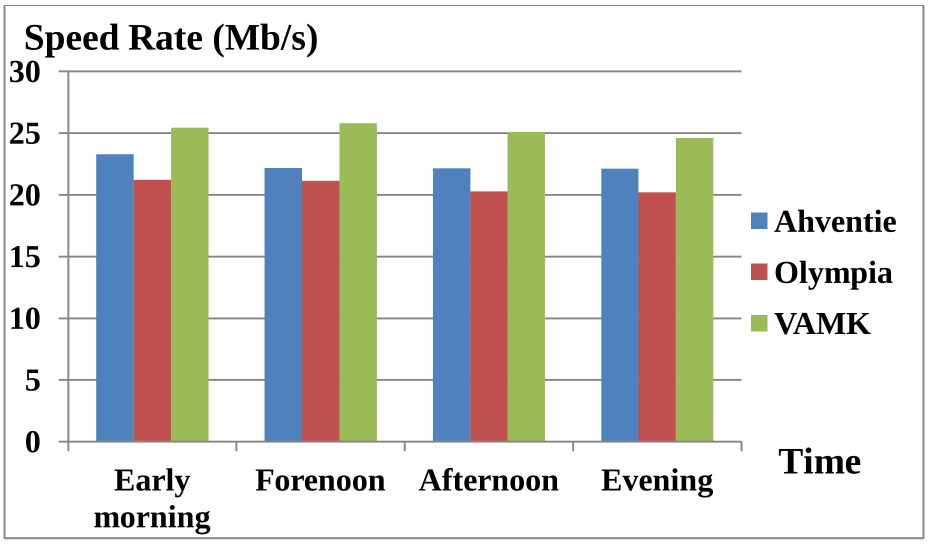


**Figure 16 Average downlink speeds at different places in different periods**

When comparing horizontally, it’s found in Figure 16 that the speeds at VAMK are higher than those at Ahventie, followed by Olympia. Since the same download method is used, web-based method, we only need to take the network conditions into consideration. VAMK and Olympia are located in the area with strong 4G signal whereas Ahventie is situated on the edge of 4G network with less power. In theory, the speed in Olympia should be higher than that at Ahventie. But the reality is the opposite. As we know, the transmitting direction of the wireless signal will influence the downlink performance. Based on this, it is surmised that it is the window’s orientation that causes poor performance. The window of testing room may be toward to the antenna spreading out direction, which is to the back of antenna.

*The case of uplink flows (Ahventie and Olympia are terminal-based, VAMK is web-based.)*

| Time Period | Ahventie | Olympia | VAMK |
|---|---|---|---|
| Terminal | ScePanel, WinScp<br>Terminal-based | ScePanel, WinScp<br>Terminal-based | Funet<br>Web-based |
| 6:00-7:30/Early morning | 4,015784257 | 2,948151728 | 6,34918087 |
| 9:00-10:30/Forenoon | 3,25398923 | 2,895059725 | 6,110073478 |
| 2:30-4:00/Afternoon | 3,280852614 | 2,827395022 | 5,945433731 |
| 7:30-9:00/Evening | 3,435531199 | 2,744786468 | 5,766039394 |

**Table 7 Average uplink speeds in fixed periods at different places**

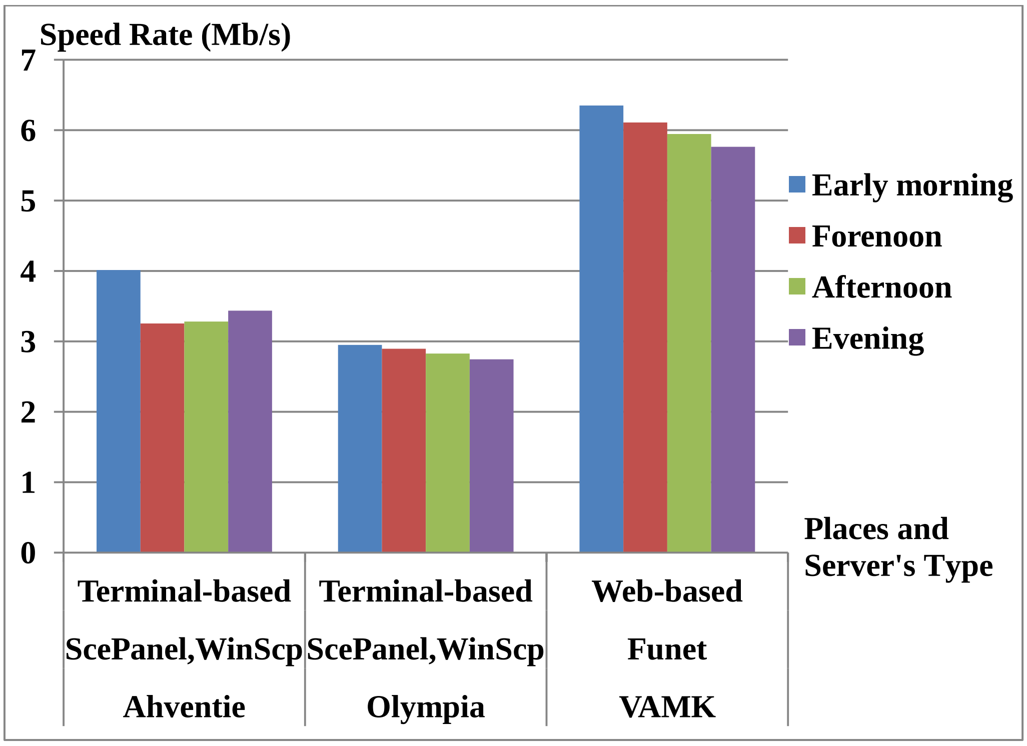


**Figure 17 Average uplink speeds in fixed periods at different places**

In Table 7 and Figure 17, the descending order of speeds is the same as in downlink. Unexpectedly, at Ahventie the average speed in the evening was slightly higher than that in the forenoon and afternoon. It's proved that the performance of TCP congestion control schemes descends slightly whereas the channels turn into saturating consistently.

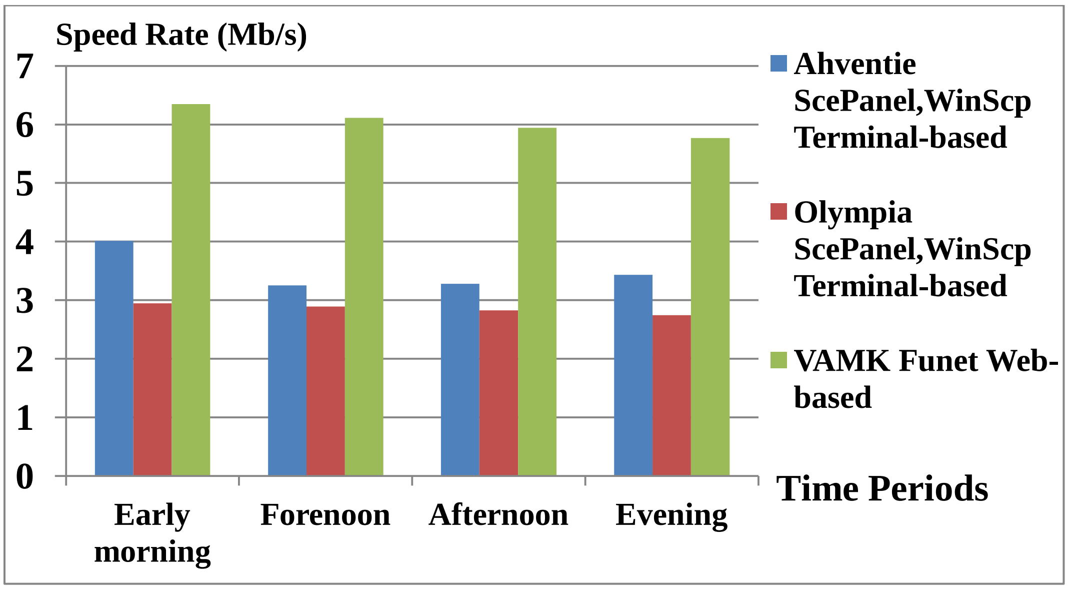


**Figure 18 Average uplink speeds at different places in fixed periods**

When it comes to Figure 18, the comparison should be separated: Ahventie versus Olympia, and VAMK alone. Under normal circumstances, the upload speeds in different scenarios would be a small disparity below or above each other. The result is that the speed at VAMK is over two times of that at Ahventie. The reason is related to uplink channels. The web-based uplink helps generate higher speed than the uplink channel through SFTP client terminal.

## 4.2 Average Speeds of Five Schemes

*The case of downlink flows (all are web-based)*

| | | Cubic | Reno | Vegas | Westwood | WinSock |
|---|---|---|---|---|---|---|
| #Average (bits/s) | Ahventie | 21,39 | 21,63 | 22,19 | 23,21 | 23,93 |
| | Olympia | 20,80 | 20,54 | 20,85 | 21,14 | 20,20 |
| | VAMK | 23,08 | 24,01 | 23,17 | 22,58 | 33,38 |
| #Order | Ahventie | WinSock | Westwood | Vegas | Reno | Cubic |
| | Olympia | Westwood | Vegas | Cubic | Reno | WinSock |
| | VAMK | WinSock | Reno | Vegas | Cubic | Westwood |

**Table 8 Average downlink speeds of five schemes at different places**

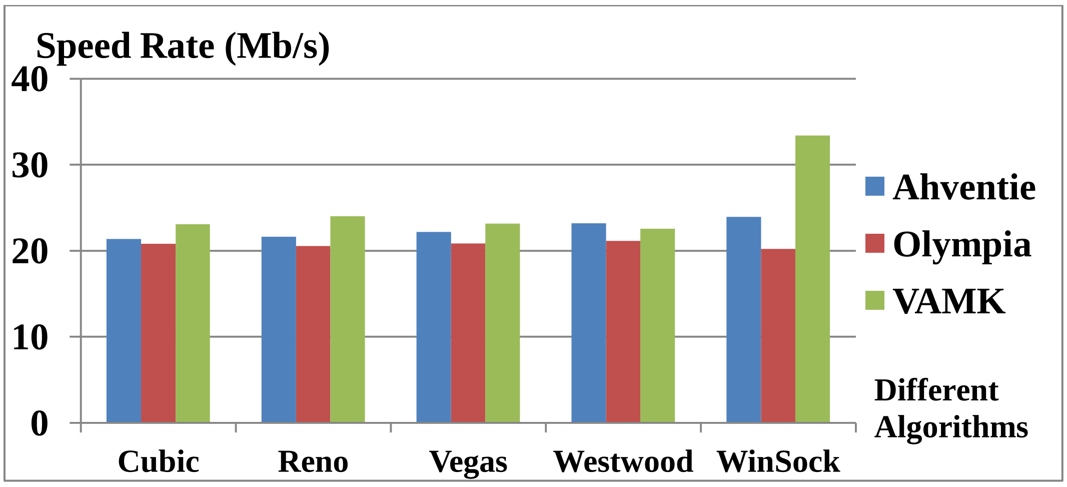


**Figure 19 Average downlink speeds of five algorithms**

As shown in Table 8 and Figure 19 that, at VAMK, the wireless condition is fabulous while at Olympia, the network is not good. Generally, VAMK > Ahventie > Olympia. All the algorithms perform best at VAMK except for Westwood.

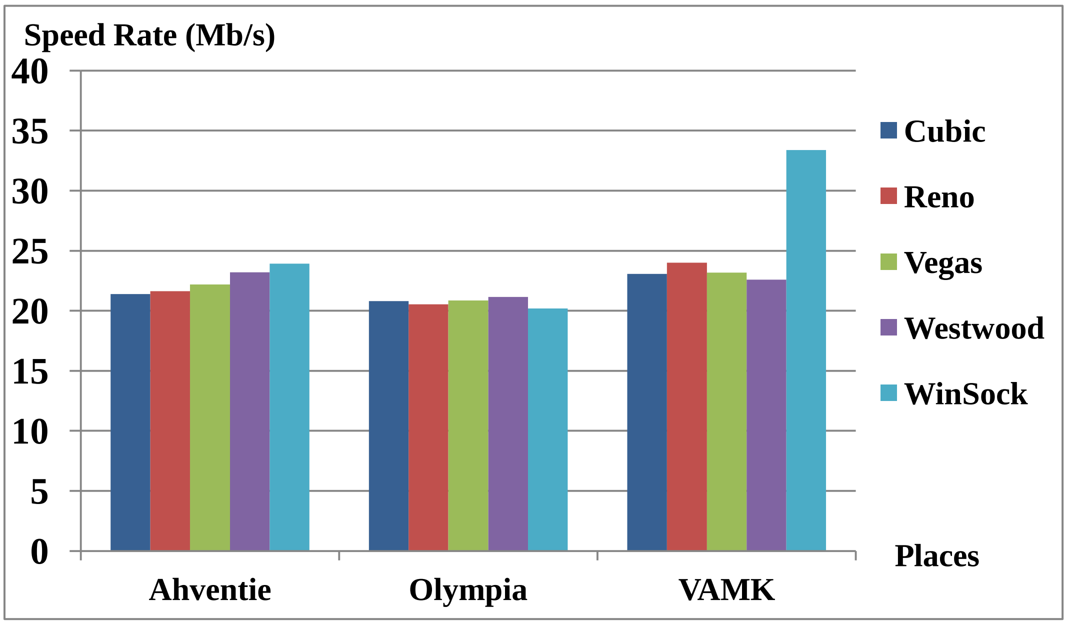


**Figure 20 Average downlink speeds of five algorithms at different places**

In Figure 20, we find that WinSock performs the best at VAMK and Ahventie. Because all these downlinks are web-based, all the algorithms are comparable, it can be concluded that WinSock has advantages under good 4G network, but not in bad 4G network.

Westwood performs better when the wireless condition is not good. As described in Chapter 2, its authors claim that Westwood is especially good for wireless links or other situations where the loss of packet may have nothing to do with congestion. [24] In the bad wireless place, Olympia, the packet loss may have more things to do with errors rather than congestion. Therefore, Westwood plays the best among other algorithms.

It also can be seen that the descending orders at VAMK and Olympia are nearly the opposite. It shows the contrary performances in good and bad networks.

Giving a more specific look at Figure 20, we can find that Vegas always behave slightly better than Cubic. If we refer to the theory part in Chapter 2, this phenomenon is not hard to understand. Cubic is based on the ACKs while Vegas is based on delay, or say RTT. The latter provides a more effective way for wireless network. As we all know, typically in wireless networks, packet loss is often due to transmission errors. [17], [18], [20] For Cubic, the $W_{max}$ will be set to $cwnd$, a less size when fast retransmission occurs. Therefore, Cubic can't always take the advantage of cubic function. In contrast, Vegas always evaluates optimizing $cwnd$ based on RTT in different ways. This makes a contribution to faster speed in wireless.

*The case of uplink flows (Ahventie and Olympia are terminal-based, VAMK is web-based.)*

| | | Cubic | Reno | Vegas | Westwood | WinSock |
|---|---|---|---|---|---|---|
| #Average (bits/s) | Ahventie | 1,60 | 1,60 | 1,60 | 1,61 | 11,05 |
| | Olympia | 1,67 | 1,72 | 1,71 | 1,63 | 7,51 |
| | VAMK | 5,27 | 4,78 | 2,29 | 4,36 | 13,48 |
| #Order | Ahventie | WinSock | Westwood | Reno | Vegas | Cubic |
| | Olympia | WinSock | Reno | Vegas | Cubic | Westwood |
| | VAMK | WinSock | Cubic | Reno | Westwood | Vegas |

**Table 9 Average uplink speeds of five schemes at different places**

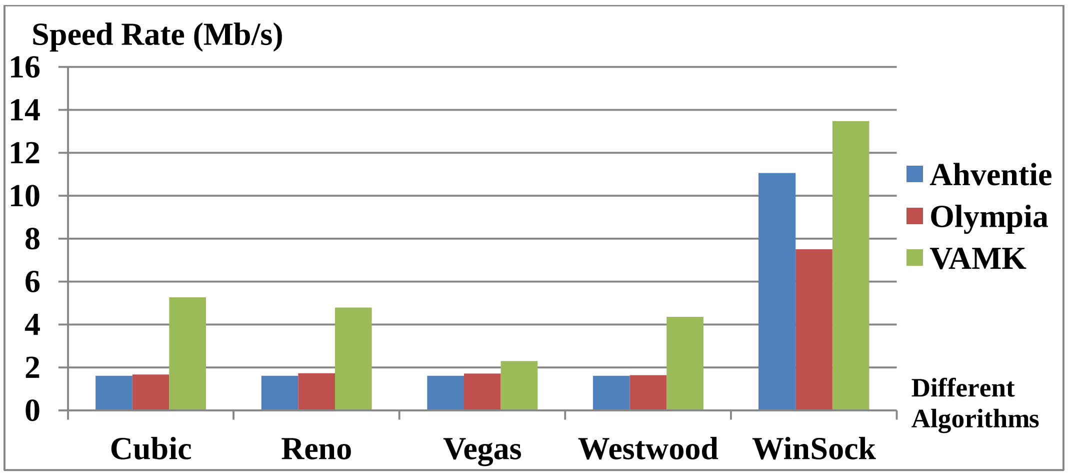


**Figure 21 Average uplink speeds of five algorithms**

As exhibited in Table 9 and Figure 21, WinSock has the overwhelming superiority in uplink among all the other algorithms. The speed of WinSock is nearly 7 times, 4.5 times and 3 times of that in Linux at Ahventie, Olympia and VAMK respectively. The deviations are too high. Some tests are needed to find out whether WinSock really make overwhelming congestion controlling abilities in the uplink.

WinSock works the best at VAMK, then Ahventie, and finally Olympia. Maybe it is because the Olympia's room window doesn't direct toward the base station, it works worse with long delays. It is surmised that WinSock performs better if delays are smaller.

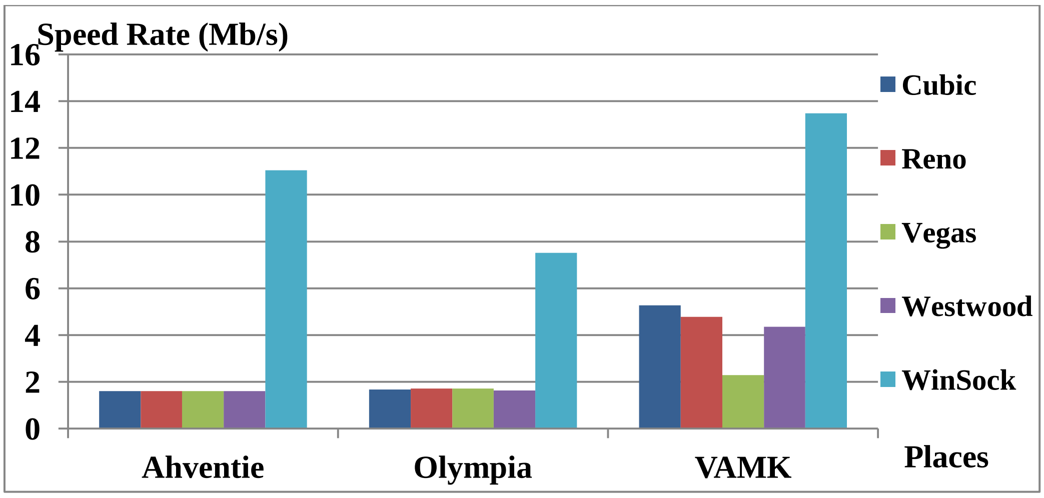


**Figure 22 Average uplink speeds of five algorithms at different places**

In the case of uploading in Figure 22, two different uplink channels are used. In SFTP terminal-based uplink, Cubic, Reno, Vegas and Westwood behave similarly when the network power is not very strong or stable.

In the web-based uploading, one noteworthy thing is that the speed of Vegas at VAMK is almost half of speeds of other schemes in Linux. Vegas' uplink performance is totally different from that in downlink. A closer look at the screenshot of Vegas at VAMK shows that its plot is completely different.

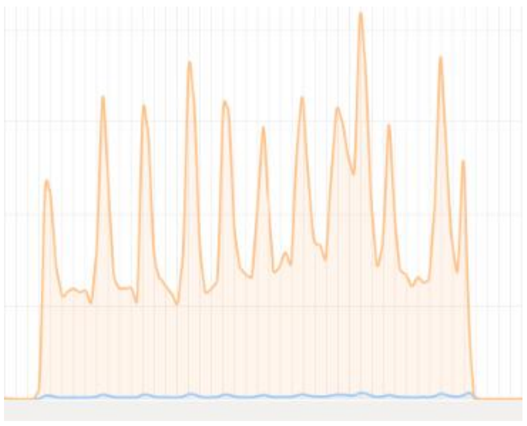


**Figure 23 Vegas on March 13th in the afternoon at VAMK**

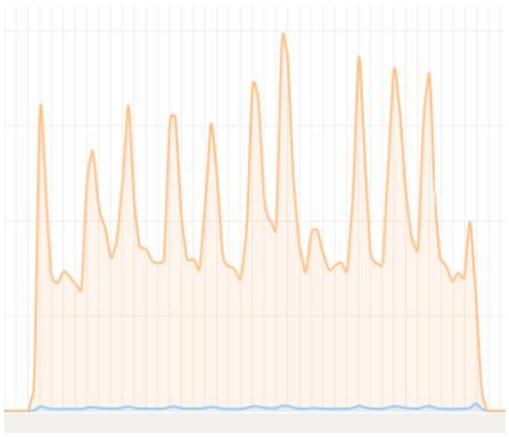


**Figure 24 Vegas on March 15th in the forenoon at VAMK**

Figure 23 and Figure 24 shows the Vegas' speed flows. Figure 24 presents a regular-shaped wave. There are several sharps followed by small fluctuations. At the beginning, the speed soars to two to four times of the average speed. After reaching the highest point, it plummets to the average speed and experiences a marginal and quick fluctuation. This process repeats again and again until the end of uploading.

Vegas is implemented in AIAD (additive increase/ additive decrease). This algorithm will change $cwnd$ to achieve better throughput. There is a range as the reference, which will help decide whether $cwnd$ should increase or decrease. If the deviation is less than $\alpha$, $cwnd$ increases. If the deviation is larger than $\beta$, $cwnd$ decreases. [24]

We can see from the plot that, the flow is always jumping. It is surmised that the fluctuation is caused by two reasons. The first is the not-well-designed range, parameters $\alpha, \beta$. For an example, the deviation is less than $\alpha$, the $cwnd$ should increase. The $cwnd$ increases so fast to the top point that the comparison between $cwnd$ and $\beta$ doesn't implement.

The other reason is that under considered circumstances, Vegas can be “fooled” into believing that the forward-direction delay is higher than it really is. [15] As we all know that the paths in the two directions of a connection may be different and have different states of congestion. [15] Although the forward direction's congestion doesn't exit, the ACKs may arrive at the sender with a relatively high delay if there is heavy congestion in the reverse direction. As described in the Chapter 2, Vegas estimates $cwnd$ by measuring RTT. The long delay can cause Vegas to decrease $cwnd$ misleadingly. This hypothesis has been proved in [15], significant traffic in the reverse direction can even cause the ACK clock to be significantly perturbed.

As what described in [24], Vegas doesn't perform well under all the conditions. Upon the analysis above, it is convinced that Vegas doesn't perform well in web-based uplink in good 4G network.

In order to make sure whether WinSock actually make overwhelming performance in the uplink, the following tests were conducted. The comparison is between Cubic and WinSock at VAMK through Ethernet in the case of uplink.

| Cubic | WinSock |
|---|---|
| 56.24Mb/s | 46.4Mb/s |

**Table 10 Comparison between uplink speeds of Cubic and WinSock**

Table 10 shows that the speed of Cubic is higher than that of WinSock. Therefore, WinSock just has better abilities in controlling the modem in the case of uplink.

## 4.3 Different FTP Methods in Uplink

Considering that different SFTP clients may lead to distinctive performance of TCP congestion control schemes, tests among different SFTP clients in Windows were conducted at Ahventie. Except for WinScp and Funet, another two SFTP clients were used, WebDrive and FileZilla. The same file was used in the forenoon and evening for two times each. Table 11 and Table 12 show the results:

| Type | Average (bits/s) | Type of SFTP |
|---|---|---|
| WebDrive | 8,809895 | SFTP |
| WinScp | 7,478485 | SFTP |
| FileZilla | 12,73451 | SFTP |
| Funet | 10,09882 | FTP, web-based |

**Table 11 Uplink speeds of WinSock, in the forenoon, at Ahventie**

| Type | Average (bits/s) | Type of SFTP |
|---|---|---|
| WebDrive | 8,175238 | SFTP |
| WinScp | 7,010267 | SFTP |
| FileZilla | 9,818301 | SFTP |
| Funet | 9,060889 | FTP, web-based |

**Table 12 Uplink speeds of WinSock, in the evening, at Ahventie**

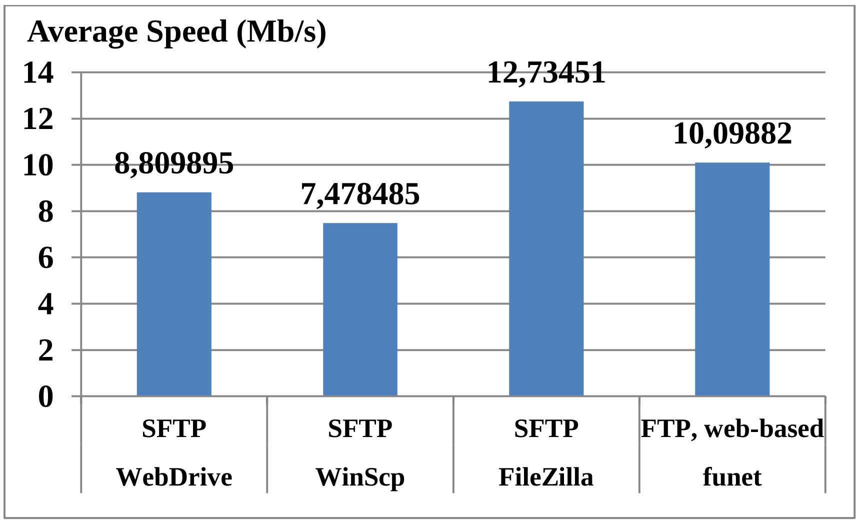


**Figure 25 Uplink speeds of WinSock, in the forenoon, at Ahventie**

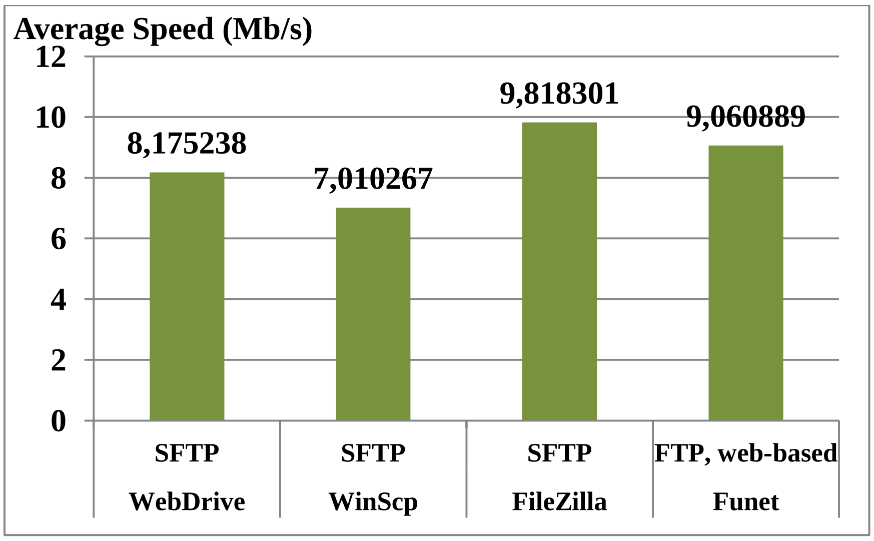


**Figure 26 Uplink speeds of WinSock, in the evening, at Ahventie**

As illustrated in Figure 25 and Figure 26, the differences among four FTP methods show that they will influence the performance of TCP congestion control algorithms. The FileZilla behaves the best. And the deviations among them are not remarkable. Recite that it was Funet and WinScp that have been used at VAMK and Ahventie respectively. The uplink average speeds through Funet are about 1.3 times of that through WinScp. The uplink average speeds at VAMK are about 2 times of that at Ahventie. Therefore, one conclusion comes out that TCP congestion control algorithms performance in good 4G network is better than that under not good 4G network.

## 4.4 Traffic Plot

Except for the comparisons of average speeds, the speed waves could be another interesting thing to discuss about. KNemo was used to record the traffic plot at Ahventie and VAMK. When tests in Olympia started, there were some problems with the plotter so that traffic plots of Olympia could not be recorded. The speed flows show more details. The analysis of speed waves will be based on the theory mentioned in Chapter 2.

*The case of uplink flows (at VAMK)*

*Cubic*

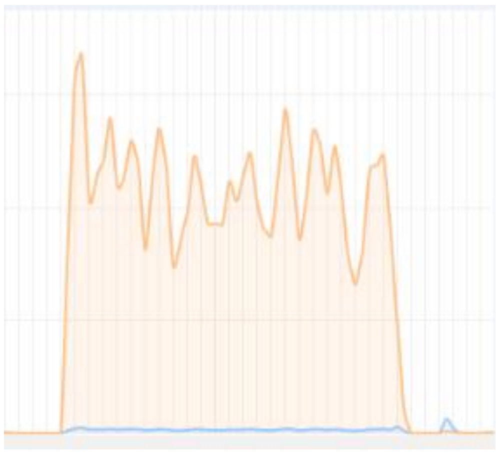


**Figure 27 Cubic on March 26th in the forenoon at VAMK**

There are some sharps in Figure 27. The reason is the unstable wireless network. Some packets dropped so Cubic needs to handle it. The function of $W$ is $W(t) = C(t-K)^3 + W_{max}$. This function will increase $W$ to $W_{max}$ when there is no further loss event. Even though there is packet loss, function with third power helps $cwnd$ grow large quickly and stay large. That’s the reason why Cubic performs well.

*Reno*

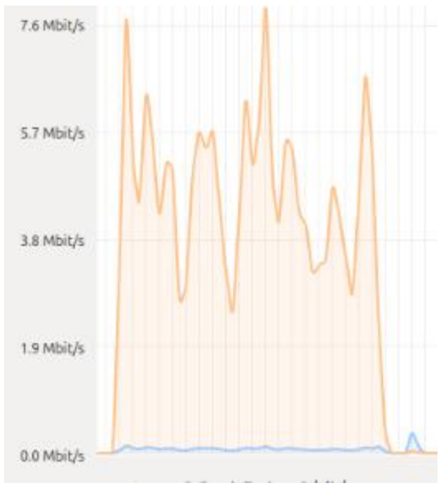


**Figure 28 Reno on March 12th in the forenoon at VAMK**

When it comes to Reno, it’s always found that the bottom points are closer to the bottom line. What’s more, if we compare Figure 28 with Figure 27, we can find that Reno’s sharps are longer. On the grounds that Reno will cause *cwnd* start from Slow Start stage if there is expiration of retransmission time. This makes Reno take longer time to recover the *cwnd* back to larger size. Thus, the amplitudes of sharps are always the longest among the five algorithms.

*Vegas*

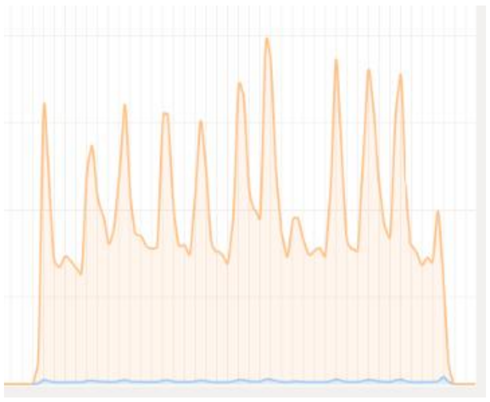


**Figure 29 Vegas on March 15th in the forenoon at VAMK**

As explained previously in section 4.2, Figure 29 presents a regular-shaped wave. There are several sharps followed by small fluctuations.

*Westwood*

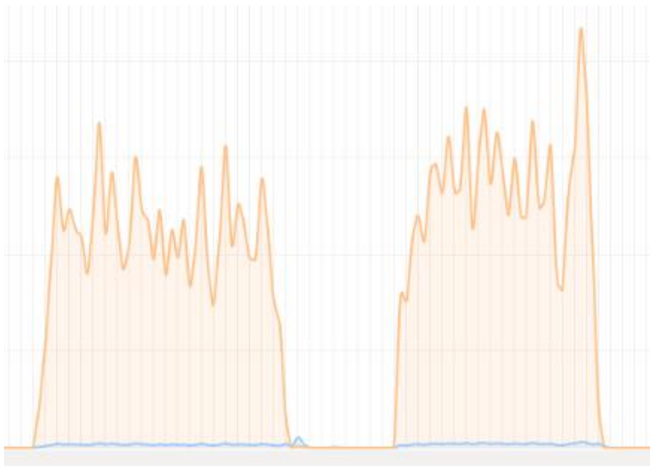


**Figure 30 Westwood on March 12th in the evening at VAMK**

Compared to Cubic and Reno, Westwood has more little sharps. In Figure 30, the amplitudes of sharps tend to be shorter. The reason is that Westwood uses estimation to anticipate $cwnd$ better. As mentioned in Chapter 2, when packet drops, Westwood calculates a new BDP $(ERE * RTT_{min})$ and assigns it to $ssthresh$ rather than half of the $ssthresh$. [15] Estimation avoids tremendous changes in $cwnd$. This contributes to flatter speed wave.

*WinSock*

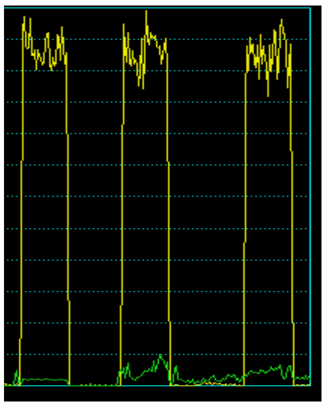


**Figure 31 WinSock on March 4th in the morning at VAMK**

The speed wave in Figure 31 is not flat as well. But there is no big fluctuation. The speed is always around the average line.

*The case of downlink flows (at VAMK)*

Then, the downlink plots at VAMK (web-based) are shown. The KNemo will adjust the wave to the window size of plot to show the complete wave, therefore the amplitude are not fixed.

*Cubic*

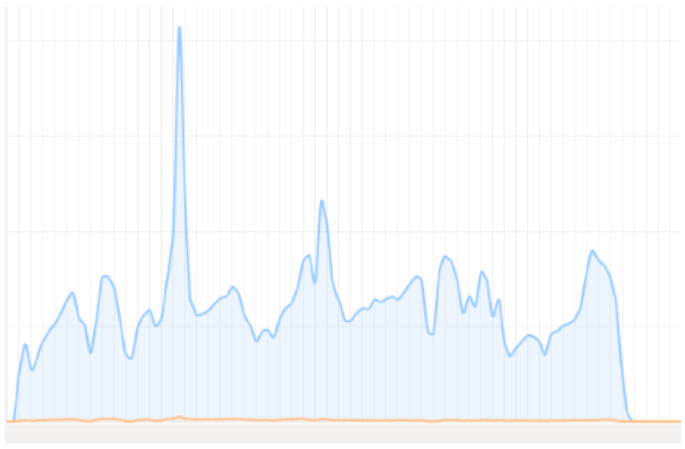


**Figure 32 Cubic on March 14th in the early morning at VAMK**

*Reno*

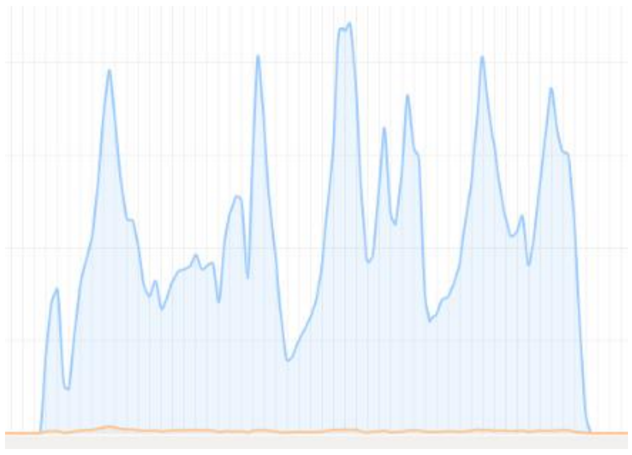


**Figure 33 Reno on March 14th in the evening at VAMK**

*Vegas*

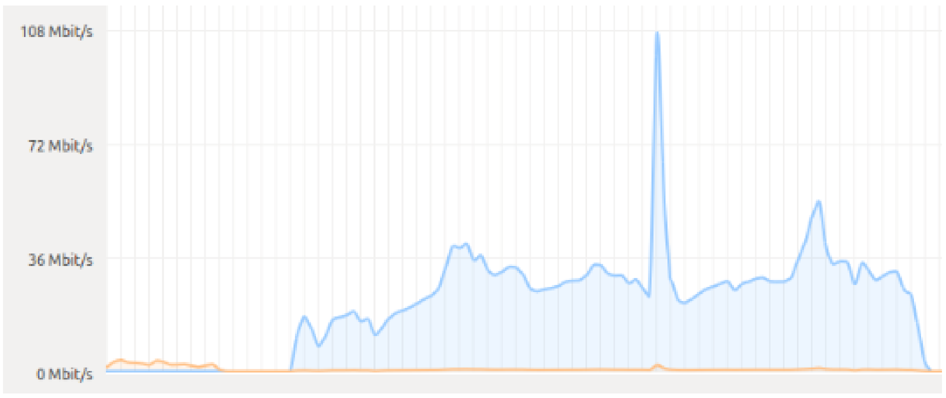


**Figure 34 Vegas in the March 12th in the forenoon, at VAMK**

*Westwood*

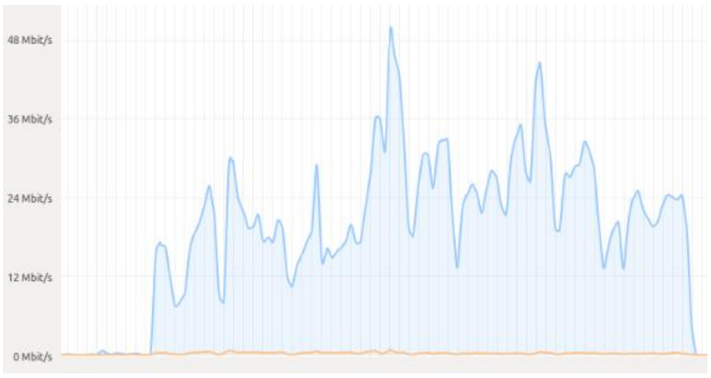


**Figure 35 Westwood on March 14th in the early morning at VAMK**

*WinSock*

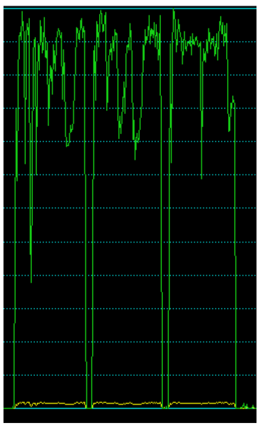


**Figure 36 WinSock on March 11th in the afternoon at VAMK**

Figure 32 to Figure 35 demonstrate that the speed flows in Linux are always fluctuating even under good 4G network. In Figure 36 we can see that WinSock is more stable. Consequently, the speed flows always fluctuate in downlink. There are possibilities that a high speed occurs. In Figure 34, the top speed in Vegas is about 106Mbit/s.

*The case of downlink flows (at Ahventie)*

Then, the downlink plots at Ahventie (web-based) are shown.

*Cubic*

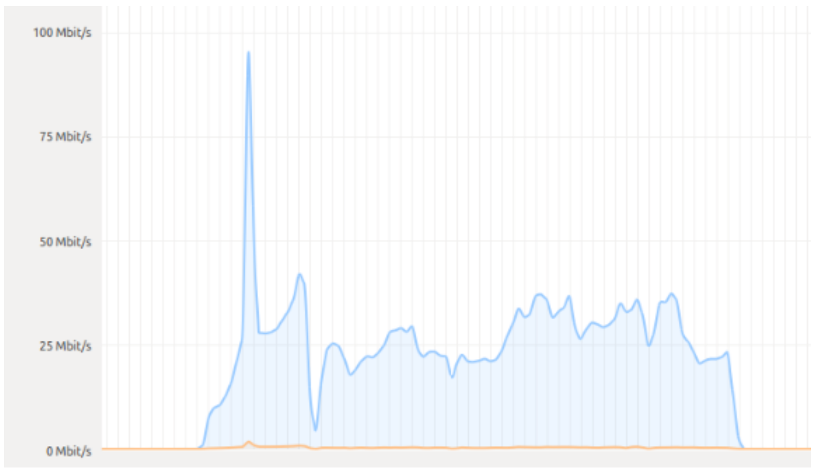


**Figure 37 Cubic on Feb 26th in the early morning at Ahventie**

*Reno*

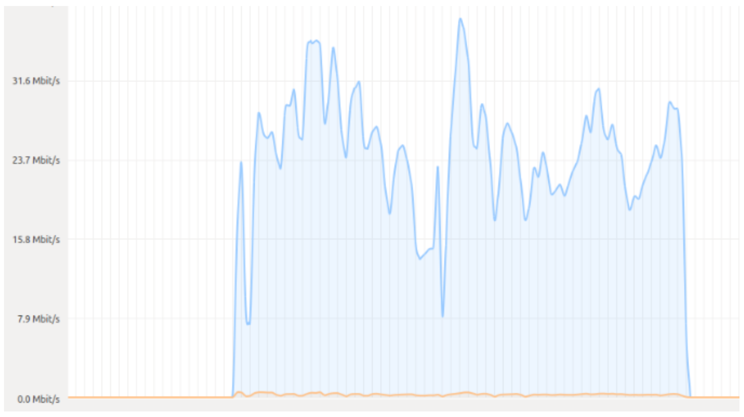


**Figure 38 Reno on Feb 24th in the evening at Ahventie**

*Vegas*

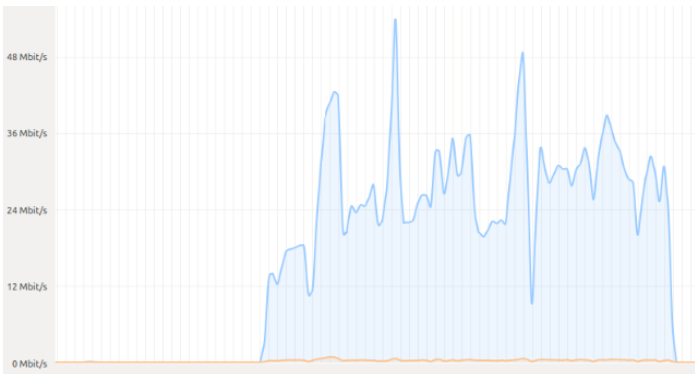


**Figure 39 Vegas on Feb 26th in the early morning at Ahventie**

*Westwood*

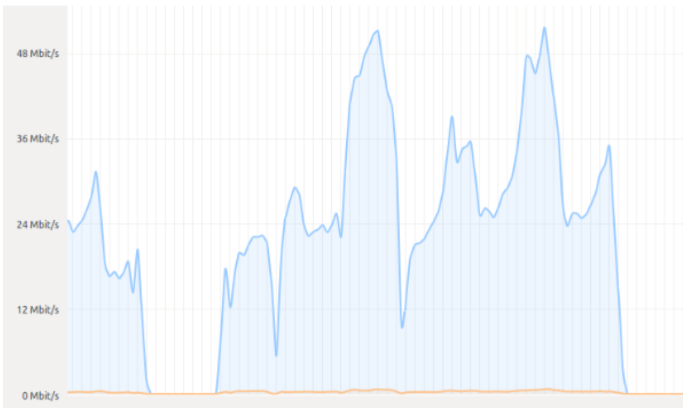


**Figure 40 Westwood on Feb 26th in the early morning at Ahventie**

*WinSock*

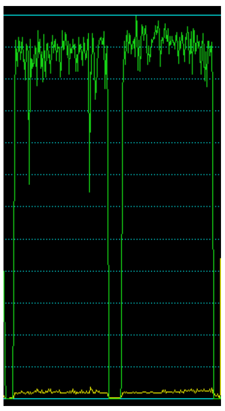


**Figure 41 WinSock on March 5th in the morning at Ahventie**

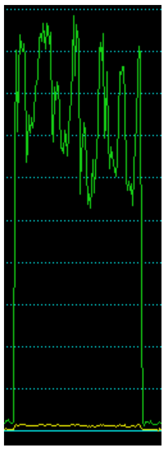


**Figure 42 WinSock on March 4th in the evening at Ahventie**

Figure 37 to Figure 42 show that in the downlink side, TCP congestion control algorithms don’t have specific regulations in not good 4G network. WinSock behaves well normally, but there will be occasionally big fluctuations because of the unstable wireless, like the third flow. And sometimes, there is high speed occurs. Any algorithm has the possibility. In Figure 43, the top speed of Reno is about 240Mbit/s.

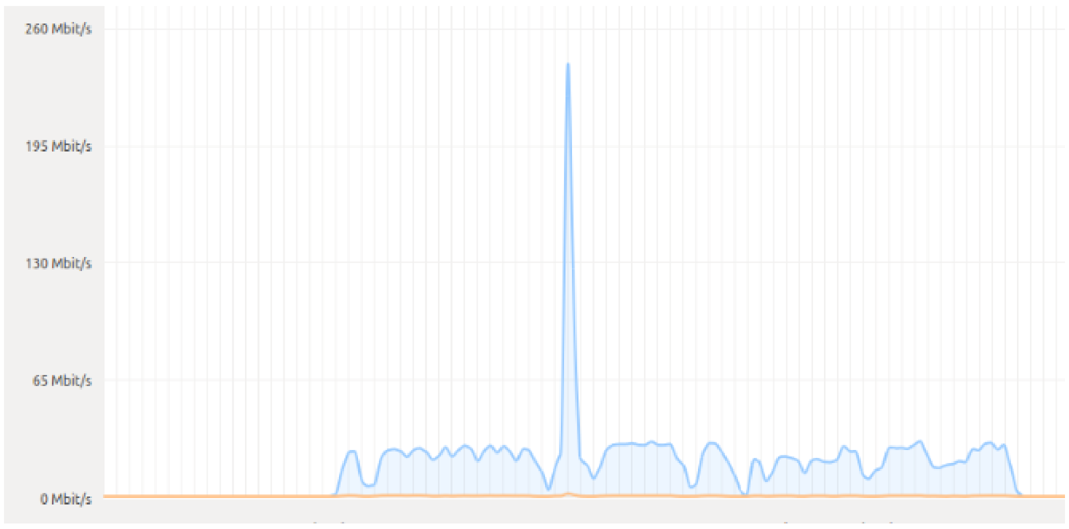


**Figure 43 Reno on Feb 25th in the early morning at Ahventie**

*The case of uplink flows (at Ahventie)*

From now on, I show the uplink plots at Ahventie (SecPanel, WinScp, not web-based).

*Cubic*

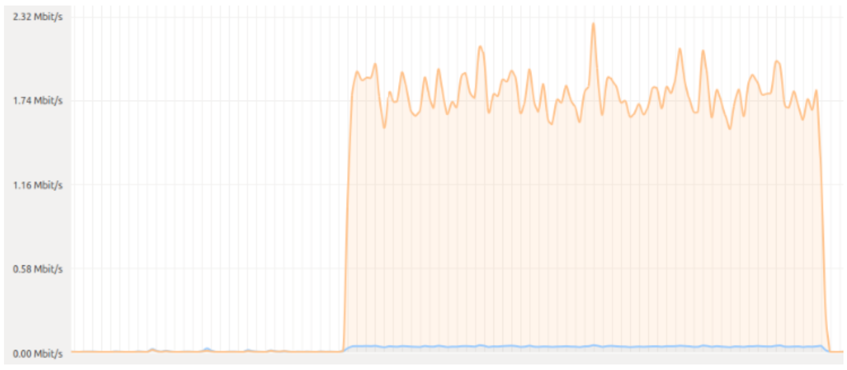


**Figure 44   Cubic on Feb 25th in the early morning at Ahventie**

*Reno*

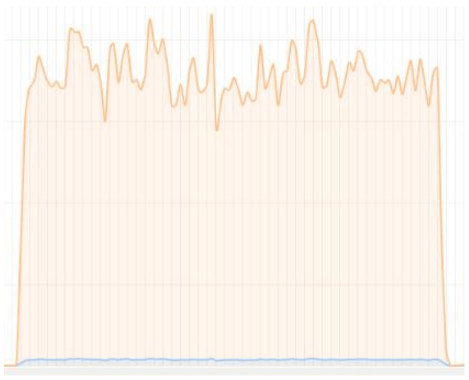


**Figure 45   Reno on Feb 26th in the forenoon at Ahventie**

*Vegas*

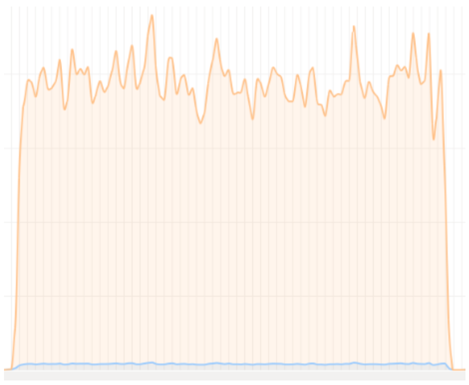


**Figure 46   Vegas on Feb 26th in the forenoon at Ahventie**

*Westwood*

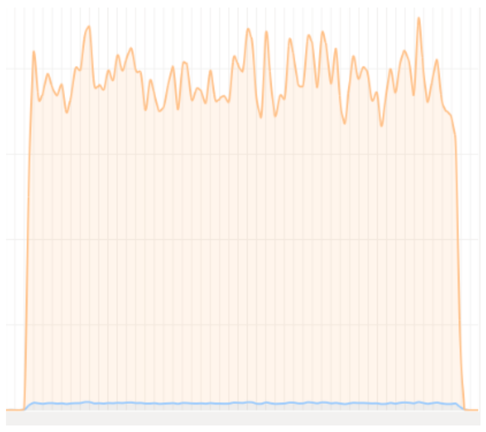


**Figure 47 Westwood on Feb 26th in the forenoon at Ahventie**

*WinSock*

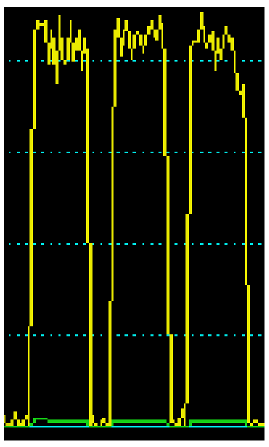


**Figure 48 WinSock on March 4th in the forenoon at Ahventie**

From Figure 44 to Figure 48 we can see that the uplinks with SecPanel and WinScp clients are stable. And there is no big difference of the waves among different algorithms. Sometimes, there is high big sharp as well. In Figure 49, the top speed is up to 3.92Mbit/s.

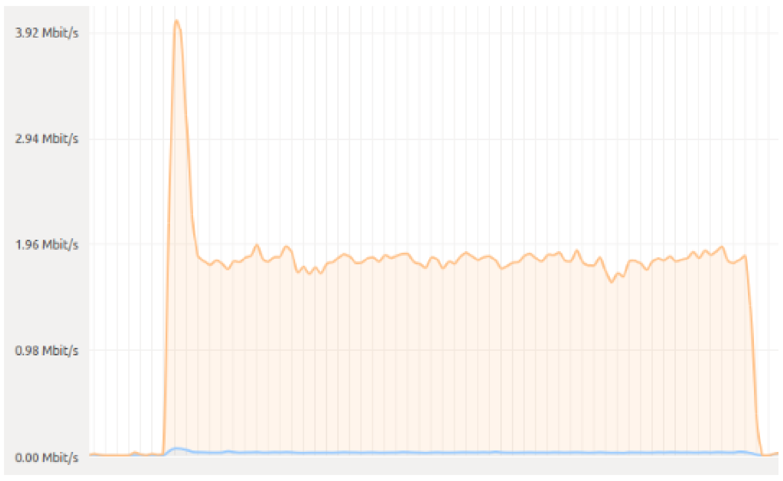


**Figure 49 Cubic on Feb 25th in the early morning at Ahventie**

## 4.5 Limitations and Implications for Future Researches

Limited to time and locations, it's not realistic to conduct more tests in other places. During this whole process, there were still some factors influencing the accuracy of results, such as the weather, the direction of antenna. Restricted to the same reason, only one sample file was used. Another limitation is that the locations of base stations are unknown. This is a big obstacle to choose appropriate places.

According to the limitations, selections of places and window's directions should be considered carefully. Besides, variable sizes can provide more accurate results. In addition, more places could be considered, and number of samples in each place could be larger.

In the future research, I recommend to focus on the Data-Link Layer and Physical Layer, or more parameters in Transport Layer. Data-Link Layer and Physical Layer are responsible for interconnecting hosts or nodes in the network logically and physically. The investigation on these two layers will clarify the congestion control schemes more. The second is about the other parameters, such as RTT, throughput, buffer sizes and the information in the packets.

## 5. CONCLUSION

In this paper, five TCP congestion control algorithms are considered, Cubic, Reno, Vegas, Westwood and WinSock. Tests are conducted in downlink case and uplink case in three different scenarios. The results point out that in both cases of uplink and downlink: (1) Comparing with a bad 4G network, all the schemes perform better in good 4G network; (2) The performance of TCP congestion control algorithms decreases slightly whereas the channels become saturating gradually. In the case of downlink: (1) Westwood performs the best when the network condition is not good; (2) Vegas behaves better than Cubic. In the case of uplink: (1) The performance through web-based FTP outweighs that through terminal-based FTP; (2 )Vegas performs badly in web-based FTP; (3)WinSock has better abilities in controlling the modem; (4)Different FTP methods don't perform the same.

For the mobile industries and application developer, they can select Westwood and avoid using Vegas in uplink in web-based FTP in 4G system.

## REFERENCES

[1] Mudit Ratana Bhalla
Generations of Mobile Wireless Technology, A Survey, International Journal of Computer Applications (0975 – 8887) Volume 5– No.4, August 2010

[2] Fast 4G networks set up in cities. Accessed 2.3.2013
http://www.hs.fi/english/article/Fast+4G+networks+set+up+in+cities/1329103688495

[3] Ghassan A. Abed, Mahamod Ismail, Kasmiran Jumari
“Integrated Approaches to Enhance TCP Performance over 4G wireless Network”, 2012 IEEE Symposium on Computers and Information 2012

[4]Christine E.Jones, Krishna M.Sivalingam, Prathima Agrawal and Jyh Cheng Chen
“A Survey of Energy Efficient Network Protocols for Wireless Networks”, Wireless Network, 7343-7358, 2001, 2001 Kluwer Academic Publishers

[5] De Cicco, Luca, and Saverio Mascolo. "TCP congestion control over 3G communication systems: an experimental evaluation of new Reno, BIC and westwood+." *Next Generation Teletraffic and Wired/Wireless Advanced Networking.* Springer Berlin Heidelberg, 2007.73-85.

[6] Abed, Ghassan A., Mahamod Ismail, and Kasmiran Jumari. "A Survey on Performance of Congestion Control Mechanisms for Standard TCP Versions."*Australian Journal of Basic and Applied Sciences* 5, no. 12 (2011): 1345-1352. ISSN 1991-8178

[7] Ghassan A.Abed, Mahamod Ismail, and Kasmiran Jumari
“Behavior of cwnd for TCP Source Variants over Parameters of LTE Networks”, Information Technology Journal 10 (3): 663-668, 2011, ISSN1812-5638

[8] Ghazaleh, Hosam, and Muhanna Muhanna. "Enhancement of throughput time using MS-TCP transport layer protocol for 4G mobiles." In *Systems, Signals and Devices, 2008. IEEE SSD 2008. 5th International Multi-Conference on*, pp. 1-5. IEEE, 2008.

[9] TeliaSonera first to launch 4G in Finland, Accessed 2.3.2013
http://www.teliasonera.com/en/newsroom/press-releases/2010/11/teliasonera-first-to-launch-4g-in-finland/

[10] Yung-Chih Chen, Erich M.Nahum, Richard J.Gibbens, Don Towsley, Yeon-sup Lim
“Characterizing 4G and 3G Networks Supporting Mobility with Multi-Path TCP”, UMass Amherst Technical Report: UM-CS-2012-022, 2012

[11] Elisa brings 4G speed to the market, Accessed 2.3.2013
http://www.goodnewsfinland.com/archive/news/elisa-brings-4g-speed-to-the-market/

[12] Palazzi, Claudio E. "Evolution Toward 4G Communication System." 2008

[13] Comparison between TCP/IP and OSI
http://www.omnisecu.com/tcpip/tcpip-model.htm

[14] Parziale, Lydia, David T. Britt, Chuck Davis, Jason Forrester, Wei Liu, Carolyn Matthews, and Nicolas Rosselot. *TCP/IP Tutorial and Technical Overview*. IBM International Technical Support Organization, 2006.

[15] Kevin R.Fall, W.Richard Stevens, "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Welsey, 1994

[16] Mateti Prabhaker, "Mobile TCP CEG436 Mobile Computing"

[17] Wired vs. Wireless, Accessed 11.3.2013
http://www.skullbox.net/wiredvswireless.php

[18] A comparison of wired and wireless communication technology essay, Accessed 11.3.2013
http://www.americanessays.com/study-aids/free-essays/communications/compair-wired-and-wireless-communication.php

[19] Kozierok, Charles M. *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Pr, 2005.

[20] Wired vs Wireless Networking, Accessed 11.3.2013
http://compnetworking.about.com/cs/homenetworking/a/homewiredless_3.htm

[21] Allman M.: RFC 5681- TCP Congestion Control

[22]Stevens W.: RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms

[23]Zeng Wanggang, "Improving TCP Performance over Wireless Networks"

[24]Pluggable Congestion Avoidance Modules, Accessed 11.3.2013
http://lwn.net/Articles/128681/

[25] Linux Tuning, Accessed 11.3.2013
http://fasterdata.es.net/host-tuning/linux/

[26] Sonera. Accessed 11.3.2013
http://www.sonera.fi/kauppiaat/ajankohtaista/4/soneran-aito-4g-kaikkien-kokeiltavissa/

[27] Soneran 4G. Se aito 4G. Accessed 11.3.2013
http://www.sonera.fi/nettiyhteydet/liikkeelle/4g+langaton+laajakaista

[28] Sonera Kauppa Rewell Center, VAASA, Accessed 11.3.2013
http://www.sonera.fi/kauppiaat/rewell-center/ajankohtaista/2/4g-nyt-vaasassa

[29] Huawei E392 (4G-nettitikku), Accessed 11.3.2013
http://www5.sonera.fi/ohjeet/Huawei_E392_(4G-nettitikku)

[30] Huawei E392 4G Modem Quick Start Guide

[31] 5.6 Using Funet FileSender to share and transport files, Accessed 11.3.2013
http://datakeskus.csc.fi/5.6-using-funet-filesender-to-share-and-transport-files

[32] Brief History of Funet, Accessed 11.3.2013
http://en.wikipedia.org/wiki/FUNET