TCP Congestion Control Algorithms Performance in 3G networks with moving client

MACIEJ ROSTAŃSKI a  PIOTR PIKIEWICZ a

a Academy of Business in Dabrowa Gornicza
{mrostanskilppikiewicz}@wsb.edu.pl

Abstract: Presented article focuses on improving performance of the TCP/IP connection in specific condition - connection between the data server and client downloading data, using mobile (cellular) network as an Internet connection method, while driving. A way to affect mechanisms of transport layer, and achieve better performance, is method described as changing TCP’s Congestion Control Algorithm (CCA), which is responsible for congestion window behaviour. Today’s TCP flavours are presented. For experimental research, topology is created, and test scenarios are being discussed. Methodology and tools are presented, as well as comparison of different CCA performance in realized test runs. Presented research leads to conclusion there is a field of study on cwnd behaviour in 3G network while using a family of congestion control algorithms designed for fast networks, since those get better results than CCAs designed for wireless networks. The results and conclusions of this article address the infrastructure level of a typical, modern, european, urban region.

Keywords: Congestion Control, TCP, CuBIC, Veno, Reno, Throughput, UMTS, 3G

1. Introduction

Presented article focuses on improving performance of the TCP/IP connection in specific condition - connection between the data server and client downloading data, using mobile (cellular) network as an Internet connection method. Performance and similar parameters of a connection using GSM / WCDMA technologies as an Internet access method is a subject widely researched. The variety of conditions and factors affecting performance of such connection is often a topic of scientific reports - for example [1], [2] or [3].

Problem, addressed in this article, is (in spite of fact that conclusions of similar research are useful for mobile network operators nad vendors), they present little
value to the end-user - who doesn’t have the capability to alter any parameters of 3G infrastructure he uses.

There is, however, a way to affect mechanisms of transport layer, and achieve better performance. Method described in this article focuses on changing TCP’s Congestion Control Algorithm (CCA), which is responsible for congestion window behaviour. Server administrator can easily recompile and merge any CCA (even his own) into Linux kernel and change CCAs using Linux kernel parameters.

2. TCP State of art

During data transfer, every packet is exposed to phenomena slowing down or interrupting its travel through computer network. This slowing down, in the case of TCP segments in particular, may be caused by transmission channel congestion or interferences observable in wireless networks. In TCP protocol, mechanisms regulating sending data rate in dependence of network state, were introduced as a solution to network congestion problem.

First such modification was so called Tahoe [4] algorithm, including slow-start, congestion-avoidance and fast-retransmission mechanisms. In TCP Reno [5] algorithm, besides Tahoe modifications, an algorithm of Fast-Recovery of lost packets was introduced [6], [7]. Fast-Recovery uses fast retransmission mechanism, in which multiple acknowledges of the same packet indicate packet loss. So called New Reno algorithm [8] is capable of Selective Acknowledgments (SACK), allowing TCP protocol to continue fast retransmission procedure without Slow-Start, even after getting only partial acknowledgments (allowing for some ACK to arrive later).

Today there are many known versions of TCP algorithms, altering TCP window size in different manner, that are in majority modifications of Reno mechanism, but focusing on different problems of modern networks - some address specific wireless/satellite networks issues, other, for example, deal with TCP problems on Long Fat Networks. Algorithms BIC TCP (Binary Increase Congestion Control)[9] and CuBic [10], designed for use with networks with large BDP (Bandwidth-Delay Product), distinguish themselves with window growth function, specific around link saturation point. Mentioned CCAs include methods of new TCP window value designation, which cause no exaggerative growth, which in turn leads to maintaining optimal data transfer rate longer comparing to other algorithms. CuBIC Algorithm is used by default in modern Linux kernel.

Another approach is presented by Vegas algorithm [2] - Vegas tries to accomplish better bandwidth saturation, avoiding retransmissions, with appropriate flow reduction. Algorithm predicts (or tries to predict) congestion before it happens and
reduces packet sending rate, hoping to reduce packet delay, and, in effect, increase performance. This is known as proactive behaviour. Around 2003, Veno algorithm was proposed [11] - an attempt to merge the advantages of Reno and Vegas algorithms. Veno algorithm relies on Vegas proactive mechanism only to identify, whether specific packet loss incident is caused by network congestion or is it effect of random wireless environment issues. Congestion window regulation is made similar to Reno behaviour.

Using such algorithms as Veno or Vegas should bring good results in wireless networks, where packet loss probability is much higher compared to wired networks. In addition to those, specifically for application in wireless networks, where potential packet loss is a result of transmission error rather than network congestion, and network load is highly dynamic nature, Westwood algorithm [12] was also designed. In Westwood algorithm, the stream of acknowledgments is analyzed for estimation of Eligible Rate value, used in turn for congestion window optimization and accurate setting up \textit{ssthresh} value in case of packet loss event discovery. Important difference between Westwood and Reno is a method of congestion window reduction. Westwood, contrary to Reno (which halves the \textit{cwnd} - value of congestion window, after congestion event), uses information of last available bandwidth with window reduction calculation.

Another CCA worth mentioning, YeAH algorithm[13], is quite fair with competing Reno flows, and doesn’t cause significant performance loss in case of random packet loss event. YeAH assumes cooperation of two phases - ”quick” phase, when \textit{cwnd} is increased following procedures like in STCP [14], and ”slow”, during which algorithm YeAH behaves like Reno algorithm. The state of algorithm phases is conditioned on predicted packet number in log queue.

3. 3G technologies

As forementioned, the performance of the connection to the Internet via cellular/mobile networks is a subject of vast scientific research, specifically for technologies in scope of this paper, articles like [15], [16], [17]. Some reason for this, and an important fact is that while using mobile terminal for data connection with the Internet, different cellular data transfer technologies may be used. This is also dependable of the distance between base stations and client, base stations and/or client capabilities, radio conditions etc. [18],[19]. Therefore, typical 3G cellular network offers its clients a gradable performance, deploying specific technologies for infrastructure of crowded cities rather than suburban region, or rural areas, downgrading when necessary to 2.5G or even 2G technologies.

In table 1 we placed main cellular access technologies with their key perfor-
Table 1. Cellular data connection comparison

<table>
<thead>
<tr>
<th>Technologies (generation)</th>
<th>Available bandwidth / throughput at end-user</th>
<th>Observable packet loss (end-to-end Internet conn.)</th>
<th>Latency (RTT values)</th>
<th>Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPRS (2.5G) [22], [25]</td>
<td>Very Low (115kbps / 40kbps)</td>
<td>up to 10%, due to timeouts and delay spikes</td>
<td>Bad (up to 2000ms)</td>
<td>High</td>
</tr>
<tr>
<td>EDGE (2.5G) [26],[27]</td>
<td>Mediocre (474kbps / 100-130kbps)</td>
<td>up to 10%, due to timeouts and delay spikes</td>
<td>Bad (up to 1500ms)</td>
<td>Medium</td>
</tr>
<tr>
<td>UMTS (3G) [27]</td>
<td>High (2Mbps / 300kbpss)</td>
<td>low (sometimes timeouts due to poor radio conditions)</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>UMTS - HSDPA (3.5G) [28], [29]</td>
<td>Relatively very high (1.5Mbps-4Mbps / 900kbps )</td>
<td>very low, omittable</td>
<td>Good (20-100ms)</td>
<td>Low</td>
</tr>
</tbody>
</table>

...performance metrics, judging from user and transport layer perspective. As described in [20], [21] we concentrate on RTT, throughput, packet loss and jitter - those are the key factors of a TCP connection performance [22]. In research presented herein, mostly UMTS technology was available during test trials.

One must note that the classification herein (Table 1) is not in any way comprehensive - we concentrate on the packet-switched technologies, and GSM-originated (deployed mostly european-wide). More information was presented in our recent papers, e.g. [22].

In addition, one must have in mind that the scope of this paper is limited - discussed technologies are continuously advancing, especially 3.5G for example such as HSPA+, MIMO HSDPA [23] and there are first commercial realizations of 4G family, based on LTE technology. Good presentation of all 3GPP standards may be found on [24].

4. Experimental setup

4.1. Methodology and tools

As the thesis of the article claims, effective throughput depends among others of applied congestion control algorithm at the sender side and may be significant in wireless 3G network. So, the test topology should realize following scenario: (1) The server (sender side) should be capable of using many pluggable CCAs, (2) Mobile client should be able to download data using 3G connection with the server, (3) The case is, there should be disturbances caused by moving client.

Figure 1 shows created topology.

As a traffic generator, nuttcp application was used - known and widely deployed in Solaris and Linux systems measurement application [31]. Nuttcp is capable of creating data transfer in any direction (client-to-server or server-to-client) which makes it very useful in researched network, as there is no possibility of creating connection with open ports (server) at the mobile operator (client) side.
Transfer tests were measured at the mobile station side, recorded using Windows XP and Wireshark, open source network analyzer, that permits analysis of network traffic as well.

Congestion window has to be observed on the sender-side (server). For this, in Linux system, TCP Probe module was used. Module is capable of saving TCP connection state, cwnd and ssthresh values, and packet sequence numbers in observed connection.

4.2. Topology setup

The testbed topology consists of three key elements (see Fig. 1): wireless (cellular) client with 3G capable modem, the Internet server, and a car. Wireless client, connected to the 3G operator is put in the car, allowing testing while driving. Regardless of the setup topology, some moving client conditions had to be put up to express typical moving client behavior. We propose a simple example of recreating good, average and bad conditions, as follows.

4.2.1. Good conditions case

In order to create good conditions scenario, cellular client was placed in a slow cruising car. To ensure there are no roaming events during tests, cruising path
was carefully traced in vicinity of an operator access point. Speed did not exceed
30kmph, and there were frequent stops.

4.2.2. Average conditions case

Creating an average conditions scenario we assumed there should be roaming
events involved; speed of movement should vary between 20-60kmph and there will
be stops, e.g. on the red lights. In other words, case would be about transferring
data in average city traffic. The path of test was therefore placed between cities of
Dabrowa Gornicza and Sosnowiec, mainly an urban area.

4.2.3. Bad conditions case

Naturally bad conditions case is about data transfer during fast travel - test in-
volves a highway run (80-100 kmph). In this research, the case was tested between
Dabrowa Gornicza and Katowice cities in Silesia metropolis region.

Naturally, one of the main issues is the effect of 3G infrastructure level in tested
areas on our research (this includes an effects of roaming, attenuation, disturbances
and unknown quality of service rules at the operator network). This research does
not address the problem of base stations localization and configuration, though.
Suffice to say is that one has to have in mind that results and conclusions of this
article address the infrastructure level of a typical, modern, european, urban region.

5. Results and conclusions

Tests were conducted with few distinct congestion control algorithms (Reno,
CuBIC, Veno, Westwood) and analyzed for (1) congestion window behaviour,
(2)RTT values, (3) achieved throughput with given CCA. Results are given as fol-
lows.

5.1. Congestion window behaviour

As expected, CCAs try to achieve maximum allowable throughput very
quickly. Such action is shown on Fig. 2 for three algorithms. CuBIC’s specific
growth function near saturation point effect can be seen. Worth noting is, algo-

rithms predestined for wireless networks (such as Westwood or Veno) achieve stu-
ration point longer than aggresive CCAs, like CuBIC, when beginning with slow-
start phase.
5.2. RTT values

Round-Trip Time values are consistent for every CCA; average RTT is low - around 20\textit{ms}. A small, but observable jitter exists. This is very important for any TCP CCA since cwnd may be altered by algorithm once every RTT occurrence. Also, timeout values are derived from RTT.

5.3. Throughput comparison

For clearance, throughput achieved using CCAs diagrams were split onto three exemplary comparisons.

Fig.3 shows throughput achieved in similar operating conditions and time by using Westwood, and later, CuBIC algorithm. Even disregarding radio problems around 55th second for Westwood, it is clear that CuBIC achieves better throughput in any circumstance.

As shown on Fig. 4, comparing Veno to Cubic shows interesting difference. On this figure, first 30 seconds is bad radio conditions period, and other half is good conditions period. Under good conditions scenario, both algorithms perform similar and achieve similar results. The difference lies in bad conditions handling. Veno, using proactive mechanisms, predicts problems and stays within lower cwnd values. This causes uninterrupted transfer (with little jitter), but is not as effective as aggressive behaviour of CuBIC.

When comparing CuBIC to Reno results, it becomes obvious that in this scenario, CuBIC modifications perform very well. CuBIC achieves better performance
with this type of connection. Reno doesn’t handle RTT jitter very well.

Main problems of experimental research are:

  a) tests involving moving (driving) are hard to reproduce in identical manner - the only solution seems to be performing statistically big number of test runs, which leads to another problem:

  b) 3-minute test requires around 10MB of data transfer, which leads to notifiable cost of research,
c) there is a possibility of many concurrent users in researched area, which will lead to lower maximum throughput allowed by an operator. Tests then have to be repeated to be comparable to other results.

Nevertheless, conducted tests lead to interesting conclusions.

Unlike technologies of lower mobile generations (2, 2.5G), scenario of Internet access using 3G connection delivers IP parameters similar to wired connection - low RTT, high throughput, almost no packet loss probability. Therefore, we observe better performance using congestion control algorithms designed with Long Fat Networks in mind - especially CuBIC. Wireless - friendly CCAs, as Westwood or Veno, do not improve throughput at all in this circumstances.

Presented research leads to conclusion there is a field of study on cwnd behaviour in 3G network while using a family of congestion control algorithms designed for fast networks - such as HS-TCP, Hamilton, YeAH. Also, should trials be consistent with observation, detailed study may be produced.

References


