INVESTIGATION OF ALGORITHMS FOR PREVENTION OF CONGESTION IN COMPUTER NETWORKS AT LEVEL OF TCP PROTOCOL USING MININET NETWORK EMULATOR

PLAMEN D. ROZINOV AND STEFAN S. DIMITROV

This article examines the performance and values of network parameters when applying different algorithms to control network congestion. The Mininet network emulator is used, which creates software-defined networks that give us the opportunity to conduct various experiments with the network and manipulate its parameters. The conclusion is made for the best algorithm to prevent network congestion. An estimate of the relative value of the error was made, which shows that in Mininet deviations in the values can be observed with a larger number of hosts in the network.

Keywords: software-defined networks, network congestion prevention algorithms, Mininet network emulator

CCS Concepts:
- Networks~Network performance evaluation~Network performance analysis;
- Networks~Network performance evaluation~Network measurement;
- Networks~Network performance evaluation~Network experimentation;
- Networks~Network performance evaluation~Network simulations

1. Research

1.1. Introduction

This article examines the performance and values of network parameters when applying different algorithms to control network congestion. The Mininet network emulator was used, which creates software-defined networks that give us the opportunity to conduct various experiments with the network and to manipulate its parameters [16].

The network congestion prevention algorithms used in the TCP protocol are used by many Internet-based applications. Their main purpose is to prevent more
data from being sent than the network can transmit. This prevents network conges-
tion. Algorithms react differently to network load, but what they have in common
is that they all have the principle of preventing network congestion [28].

The most commonly used algorithms to prevent network congestion are BBR
(Bottleneck Bandwidth and Round-Trip Propagation Time), BIC (Binary Increase
Congestion Control), CUBIC - successor to the BIC algorithm, DCTCP (TCP Con-
gestion Control for Data Centers), HighSpeed TCP, HTCP (Hamilton TCP), Hybla,
Illinois, LP (TCP Low Priority), Reno, Scalable, Vegas, Veno – Combination of Reno
and Vegas, Westood, Yeah (Yet Another High-speed TCP) [2–6, 9, 11, 12, 15, 17–24,
28].

The study takes into account the values of network parameters influencing the
behavior of the TCP protocol such as sent data over time, Initcwnd (Initial Conges-
tion Window), Cwnd (Congestion Window), MSS (Maximum Segment Size), MTU
(Maximum Transmission Unit), repeated Retransmits, RTT (Round-Trip Time),
Jitter – big difference in delay (Delay Variance), RTT Variance – RTT variability in
time, Throughput – time interval bandwidth [1, 7, 8, 10, 13, 14, 20, 25, 27].

1.2. Realization

1.2.1. Creating a network

A software-defined network topology with minimum parameters is created –
two hosts, one switch and one controller using the sudo mn command.

1.2.2. Setting bandwidth

In order to recreate the operation of Ethernet cable category 6a, which is widely
used in practice, network bandwidth is limited by Linux Traffic Control. A maximum
value of 10 Gbit/s is set for both hosts and the switch with the command:

```
sudo tc qdisc add dev h1-eth0 root handle 1:
tbf rate 10gbit burst 5000000 limit 15000000
```

1.2.3. Checking algorithm for network overload control

In order to check the currently set algorithm for controlling network congestion,
we need to use the `sysctl net.ipv4.tcp_congestion_control` command.

1.2.4. Setting an algorithm

The command helps us to set an algorithm for controlling congestion in the
network

```
sysctl -w net.ipv4.tcp_congestion_control=congestion_control_algorithm
```

1.2.5. Network performance and network parameters

To measure network performance and measure the values of individual network
parameters, we use the network tool `iperf3`. In the experiments, host h2 acts as a
server and host h1 as a client. Hosts and the switch are set to the same network
congestion control algorithm. With the command: `iperf3 -s` the host h2 is started in the role of server and with the command: `iperf3 -c 10.0.0.2` a client of host h1 is started to `iperf3` server – host h2.

1.2.6. Preparation of results

From each measurement with `iperf3` a json file is created with the command:

`iperf3 -c 10.0.0.2 -J> congestion_control_algorithm.json`

Then, with the help of the `plot_iperf.sh` script and the

`plot_iperf.sh congestion_algorithm.json`

command, graphs of the individual network parameters are prepared, and for each network parameter there is a separate graph with a separate file in PDF format.

2. Results

2.1. Sent data over time
CUBIC

Sent data over time

HYBLA

Sent data over time

ILLINOIS

Sent data over time

LP

Sent data over time

HTCP

Sent data over time

SCALABLE

Sent data over time

VEGAS

Sent data over time

VENO

Sent data over time
2.2. Congestion window size (CWND)
2.3. Maximum transmission unit (MTU)

The MTU has a fixed value of 1500 bytes and no change in values, regardless of the applied network congestion control algorithm.

2.4. Retransmits

Retransmissions do not occur on the network, regardless of the network congestion control algorithm.

2.5. Round-trip time (RTT)
2.6. Round-trip time variance (RTT Variance)
2.7. Throughput

- **SCALABLE**
  - RTT Var over time

- **VEGAS**
  - RTT Var over time

- **VENO**
  - RTT Var over time

- **WESTOOD**
  - RTT Var over time

- **YEAH**
  - RTT Var over time

- **BBR**
  - Throughput over time

- **BIC**
  - Throughput over time
2.8. Summarized results

Summarized results are shown in Table 1 and Diagrams 1–5.

3. Conclusions

3.1. Congestion windows size

The highest number of packets that can be transmitted for 1 MSS was reported at Highspeed, followed by LP and Illinois. They are followed by Veno, Yeah and Westood. The number of packages is the smallest in HTCP, and in Vegas they are a little more, but without much difference. The algorithms BIC, Cubic, DCTCP, Hybla, Reno, Scalable generate a relatively small number of packages.
Table 1. Summarized results

<table>
<thead>
<tr>
<th></th>
<th>CWND [number of packets]</th>
<th>MTU [bytes]</th>
<th>RTT [ms]</th>
<th>RTT VAR [ms]</th>
<th>THROUGHPUT [megabits per second]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBR</td>
<td>552.62</td>
<td>1500</td>
<td>82.7</td>
<td>75.6</td>
<td>1150.45</td>
</tr>
<tr>
<td>BIC</td>
<td>2311.71</td>
<td>1500</td>
<td>1285.8</td>
<td>770.4</td>
<td>1181.17</td>
</tr>
<tr>
<td>CUBIC</td>
<td>1276.47</td>
<td>1500</td>
<td>391.9</td>
<td>156.1</td>
<td>918.96</td>
</tr>
<tr>
<td>DCTCP</td>
<td>1174.55</td>
<td>1500</td>
<td>1016.5</td>
<td>168.5</td>
<td>1174.55</td>
</tr>
<tr>
<td>HIGHSPEED</td>
<td>23415.46</td>
<td>1500</td>
<td>1202.2</td>
<td>608</td>
<td>1130.42</td>
</tr>
<tr>
<td>HTCP</td>
<td>296.95</td>
<td>1500</td>
<td>218.4</td>
<td>293.5</td>
<td>835.85</td>
</tr>
<tr>
<td>HYBLA</td>
<td>3268.89</td>
<td>1500</td>
<td>695.3</td>
<td>246.3</td>
<td>1048.23</td>
</tr>
<tr>
<td>ILLINOIS</td>
<td>18433.01</td>
<td>1500</td>
<td>701</td>
<td>267.3</td>
<td>1038.58</td>
</tr>
<tr>
<td>LP</td>
<td>20780.21</td>
<td>1500</td>
<td>916.3</td>
<td>572.9</td>
<td>1134.15</td>
</tr>
<tr>
<td>RENO</td>
<td>835.53</td>
<td>1500</td>
<td>99.6</td>
<td>77.3</td>
<td>835.53</td>
</tr>
<tr>
<td>SCALABLE</td>
<td>866.65</td>
<td>1500</td>
<td>378.9</td>
<td>427.8</td>
<td>866.65</td>
</tr>
<tr>
<td>VEGAS</td>
<td>306.85</td>
<td>1500</td>
<td>132.4</td>
<td>137.2</td>
<td>801.95</td>
</tr>
<tr>
<td>VENO</td>
<td>12083.16</td>
<td>1500</td>
<td>765.45</td>
<td>48.5</td>
<td>765.45</td>
</tr>
<tr>
<td>WESTOOD</td>
<td>6876.59</td>
<td>1500</td>
<td>752.17</td>
<td>339.5</td>
<td>752.17</td>
</tr>
<tr>
<td>YEAH</td>
<td>7906.02</td>
<td>1500</td>
<td>228.5</td>
<td>275.5</td>
<td>820.89</td>
</tr>
</tbody>
</table>

Diagram 1. Measured values of Cwnd [number of packets]
Diagram 2. Measured values of MTU [bytes]

Diagram 3. Measured values of RTT [milliseconds]
Diagram 4. Measured values of RTT Variance [milliseconds]

Diagram 5. Measured values of throughput [megabits per second]
3.2. Maximum transmission unit

The MTU value is fixed at 1500 bytes and does not change during the measurement period, regardless of the algorithm used.

3.3. RTT and RTT variance

The highest value is for RTT and RTT Variance using the BIC algorithm, followed by Highspeed, DCTCP and LP. The Hybla, Illinois, Veno, and Westood algorithms generate similar RTT values. The lowest value is on BBR, followed by Reno and Vegas. Yeah and HTCP have similar values, as do Cubic and Scalable.

3.4. Throughput

The best throughput values were achieved with the BIC algorithm, followed by DCTCP and BBR. In fourth and fifth place respectively are LP and Highspeed, with the difference between them is minimal. Then there are Hybla and Illinois. The lowest values were measured in Westood, followed by Veno. Cubic achieves about 920 Mbps of bandwidth. Vegas, Yeah, Reno, HTCP and Scalable report bandwidth in the 800–870 Mbps range.

3.5. Summary

The algorithms that managed to provide the highest values of network bandwidth are BIC, DCTCP, BBR, LP and Highspeed. Unfortunately, BIC, DCTCP, LP, and Highspeed are among the algorithms that generate the highest values of Round Trip Time and Round Trip Time Variance. Only BBR is an exception, as it achieves the lowest values for Round Trip Time and is in second place for the lowest values for Round Trip Time Variance. This is because BBR purposefully reduces the size of congestion window size, which reduces the likelihood of network congestion. In addition, unlike other algorithms that recognize when a network is congested according to the number of lost packets and low bandwidth values, ie. when congestion is already a fact, BBR is a model-based algorithm and manages to “predict” congestion as it builds its own network model based on bandwidth values and Round-Trip-Time at which the network performs best. BBR is supported on Linux. It is used by default on Google and Youtube as a replacement for Cubic, as Cubic has a longer latency, lower download speeds and data upload. This is due to the fact that Cubic’s congestion window size values are independent of Round-Trip-Time. As a result of the present study, BBR is the algorithm that performs best among the most commonly used algorithms to prevent network congestion.

3.6. Possible deviations when using Mininet

Studies have shown that in Mininet, deviations in RTT values can be observed in five, six or seven hosts. This means that the average RTT values extend over a wide range of values and Mininet cannot obtain accurate and consistent results. Therefore, the current study has fewer hosts to avoid this type of deviation. It is also
not desirable for the number of hosts to be too large, as packet loss may occur due to failed processor cycles performing network emulation. Because the emulator must run all OpenFlow switches, hosts, and network programs in real time, CPU usage increases significantly as the number of switches on the network increases. This inevitably increases the latency of network packets and a large number of packets remain waiting in the buffer memory, from where they are sent to the recipient. But if the buffer memory is full, the “release” of packets from it begins. To calculate the relative value of the error, we need to divide the number of failed attempts by the total number of attempts made. This attempt failed, in which we have at least 20% of unsuccessfully sent pings [26].

REFERENCES


