EE-554: OPTIMAL CONTROL THEORY

TERM PROJECT REPORT

AHMET EMRAH DEMİRCAN

1123413

Contents

[Term Project Context: 3](#_Toc379550689)

[TCP Mathematical Model and Problem Definition: 3](#_Toc379550690)

[The Practical Work Done: 7](#_Toc379550691)

[Tests and Comparison 11](#_Toc379550692)

[Test 1: GA Test and PID Control Parameter Training 12](#_Toc379550693)

[Test 2: GA Test and PID Control Parameter Training 12](#_Toc379550694)

[Test 3: Comparison of AQM Mechanisms in long time intervals 13](#_Toc379550695)

[Test 4: Comparison of AQM Mechanisms in short time intervals 15](#_Toc379550696)

[Achievements: 17](#_Toc379550697)

[References: 18](#_Toc379550698)

[Figure 1: TCP AQM Closed Loop Control System 4](#_Toc379550699)

[Figure 2: PID Controller 6](#_Toc379550700)

[Figure 3: TCP Active Queue Management Simulator Graphical User Interface 8](#_Toc379550701)

[Figure 4: Binary decomposition of control parameters on the chromosome 10](#_Toc379550702)

[Figure 5: GA Objective Function 11](#_Toc379550703)

[Figure 6: Test 1: GA Test and PID Control Parameter Training in 50 seconds interval 12](#_Toc379550704)

[Figure 7: Test 2: GA Test and PID Control Parameter Training in 1000 seconds interval 13](#_Toc379550705)

[Figure 8: Test 3: Simulation of NONE AQM in 1000 seconds interval 14](#_Toc379550706)

[Figure 9: Test 3: Simulation of RED AQM in 1000 seconds interval 15](#_Toc379550707)

[Figure 10: Test 4: Simulation of NONE AQM in 50 seconds interval 16](#_Toc379550708)

[Figure 11: Test 4: Simulation of RED AQM in 50 seconds interval 17](#_Toc379550709)

# Term Project Context:

The paper investigated in this project is “GA-based PID active queue management control design for a class of TCP communication networks”. The scope of the paper is proposal of a proportional-integral-derivative (PID) controller parameterized by genetic algorithm and used as an active queue manager (AQM) for TCP flows in between IP routers.

TCP\IP networks are connected to each other by limited buffered IP routers which may suffer from congestion in prime time traffic conditions. Congestion control plays an important role to decrease link usage demands while actively interrupting packets either by dropping or labeling. This congestion control can be effectively applied on connection oriented protocols with flow control mechanisms.

Unsurprisingly, the main candidate for the AQM workspace is Transmission Control Protocol (TCP) in the IP world. This protocol can respond to dropped\lost segments or explicit congestion notification labels by decreasing the traffic rate on the source node. Thus, AQM enabled routers can control output link usages by just interrupting TCP flows when needed. The main objective of the concept is to decrease buffer overruns, control delay constraints and optimize link throughputs.

There are several congestion control schemes developed to improve the performance of the communication networks. Many of these works have been dependent on a fluid based mathematical model of TCP communications developed in the context of stochastic theory. The paper also discusses its proposition based on this proven mathematical model and tries to visualize the results later on a network simulator environment called NS2.

The context of this term project is bounded by the fluid based mathematical model of TCP and the performance evaluation of the proposed GA-based PID AQM controller with respect to another very famous AQM controller scheme called Random Early Detection (RED).

# TCP Mathematical Model and Problem Definition:

In the paper, a window based nonlinear fluid-flow dynamic model for TCP networks is considered. This model defines coupled non-linear differential equations reflecting the dynamics of TCP accurately with the average TCP window size and the average queue length.

First of all, a few basics about TCP communication should be noted. TCP flows transmit frames based on a sliding window technique where each bulk transmission is limited by the TCP window size. The window size of each and every flow also designates the instantaneous average queue length needed on the egress link buffers. Each TCP connection names a flow of whose transmission rate is directly dependent on the instantaneous window size. The window size is additively increased if egress flow is acknowledged by the recipient on the ingress link and on the other hand multiplicatively decreased in the event of an acknowledgement timeout which means that lately transmitted frames have not been received by the end node, or have been lost or timed out. Thus the mathematical model should express this behavior of TCP in its structure.

Let’s return back to the model. The model is expressed by the following coupled non-linear differential equations [1]:

|  |  |
| --- | --- |
|  | 1.a |
|  | 1.b |

Where:

* w is the average TCP window size in packets
* q is the instantaneous queue length in packets
* is the propagation delay in seconds
* is the transmission round trip time where
* is link capacity in packets per second
* is the number of TCP connections
* is the packet dropping probability (this is the output of the AQM to the TCP model as an input, and this value controls the decrease rate of the window size as seen on the second half of equation 1.a)

Considering equation 1.a, it can easily be seen that the rate of change of the window size w is increasing with a decrease on (left hand size) and decreasing with a doubled multiplicative rate based on the round trip delayed w and q components and the packet drop probability input. This behavior is called the additive increase multiplicative decrease (AIMD) congestion control mechanism of TCP.

The closed loop control system, AQM, is deployed to solve the congestion problem of TCP communications (see Figure 1: TCP AQM Closed Loop Control System).

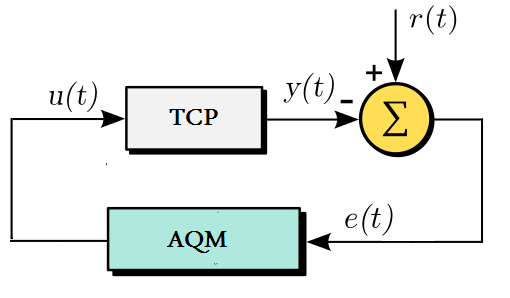


Figure : TCP AQM Closed Loop Control System

The TCP process is controlled by the AQM module with the given u(t) input. Looking back to the equation 1.a u(t) is matched with the component. The reference signal and the output of the TCP process are used as an error feedback to the AQM module. e(t) is the difference between the desired queue length and the average queue length at time t. Figure 1 will be taken as a reference for the PID controller from now on. Recall that, AIMD congestion control mechanism modeled on the TCP process works in with the AQM module cooperatively. In real world, AQM is processed in the egress link router by packet drops or explicit congestion notification labeling and the congestion response is given by the TCP flow’s endpoint somewhere behind the routing network. This modeling is quite a good way to analyze and work on AQM mechanisms without any simulation process or experimental setup.

The authors of the paper attack the TCP congestion avoidance problem by the assumption of a saturated TCP demand input on the egress link. Thus, the packet drop probability input is defined as the saturated input of the differential equations system as follows:

|  |  |
| --- | --- |
| Where, | 2 |

Thus coupled differential equation set becomes:

|  |  |
| --- | --- |
|  | 3.a |
|  | 3.b |

The saturated input of the differential equation system is the AQM output as stated previously. This project includes the evaluation of the differential equations system on a time period with a given link rate/capacity, egress link buffer length etc. parameters according to 3 different AQM schemes presented:

* **NONE**: This a name I have given where no AQM is processed but a buffer limit constraint is introduced to the system as a saturated input as follows:

|  |  |
| --- | --- |
| q is the instantaneous average queue length at time t  *queueSize* is the egress link buffer size | 4 |

* **RED**: Random Early Detection scheme is a mechanism controlling the packet drop probability given the average queue length real time value or an estimation of that. RED works according to buffer threshold levels defined by the user and is defined as follows:

|  |  |
| --- | --- |
| q is the instantaneous average queue length at time t  *qMin, qMax and* areconfigurable parameters | 5 |

* **PID**: The drop probability functions given in equations 4 and 5 are straightforward by means of using instantaneous average queue size on the egress link. What is different in the PID controller is that the response of the controller is depending on the current error rate , previous errors and the error derivative. The rates of these inputs on the drop probability output are dependent on , and parameters which are calculated through an offline optimization process using genetic algorithms (see Figure 2). Proportional-Integral-Derivative controller is defined as follows:

|  |  |
| --- | --- |
| q is the instantaneous average queue length at time t  , are configurable parameters | 6 |

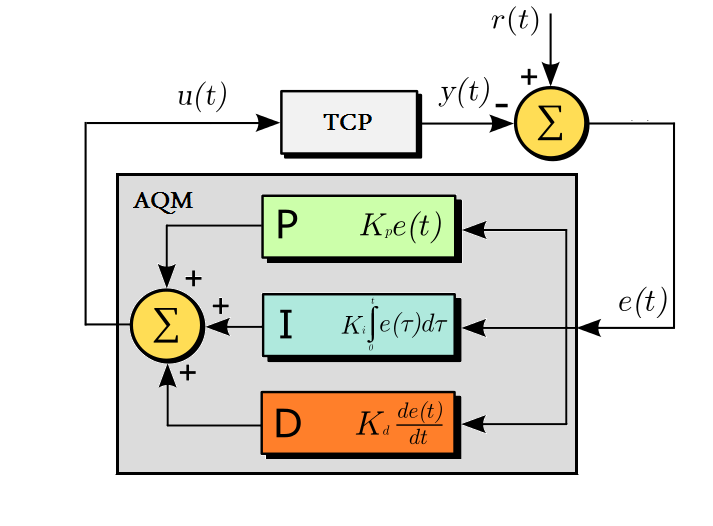


Figure : PID Controller

To sum up, equations 4, 5, 6 are used in equation 3.a for the evaluation of the instantaneous TCP flow window sizes. The average queue component in equation 3.b is also dependent on drop probability functions via the window size parameters in the differential equation. Of course, the input of the AQM module, which is the error rate , can be defined in many ways [4]:

|  |  |
| --- | --- |
|  | 7 |
|  | 8 |
|  | 9 |
|  | 10 |
|  | 11 |

The error component of the problem is selected as the Integral Absolute Error (IAE) in the paper. I also used this error as the objective function in the chromosome cost evaluation (see equation 12).

|  |  |
| --- | --- |
|  | 12 |

Solving the differential equations over a time interval by using 3 previously mentioned AQM methods will yield to the aims of this project work. If there are N flows given by the user, the coupled differential equations will be extracted as 1 average queue differential equation and N window size differential equations. The main complexity in the solution is the usage of older window sizes and average queue size (through and components in eq. 3.a). So, N+1 equations, will be used to solve the whole system.

# The Practical Work Done:

Firstly, I have developed a set of Matlab files to implement a solution for the TCP model with 3 different AQM schemes and user provided inputs. Secondly, I have used the Matlab simulation environment to test the AQM methods and Genetic Algorithm based PID optimization process. And finally, I have compared my results with the similar simulations made by the authors in the paper. Below, implementation related information is given for the instructor. And in the next section an explanation of the tests and comparisons will be told also.

The Matlab implementation files are as follows:

* aqm\_sim.m
* none.m
* pid.m
* pidGenetic.m
* queue.m
* queueOutput.m
* red.m
* simulateNone.m
* simulatePID.m
* simulateRED.m
* tcp.fig
* tcp.m

To be more precise and to explain the internal functionality of the Matlab files, a classification can be made as below:

* **Graphical User Interface (tcp.fig, tcp.m):** This module is responsible for providing an interface for the user to start simulations and train PID controller (see Figure 3).

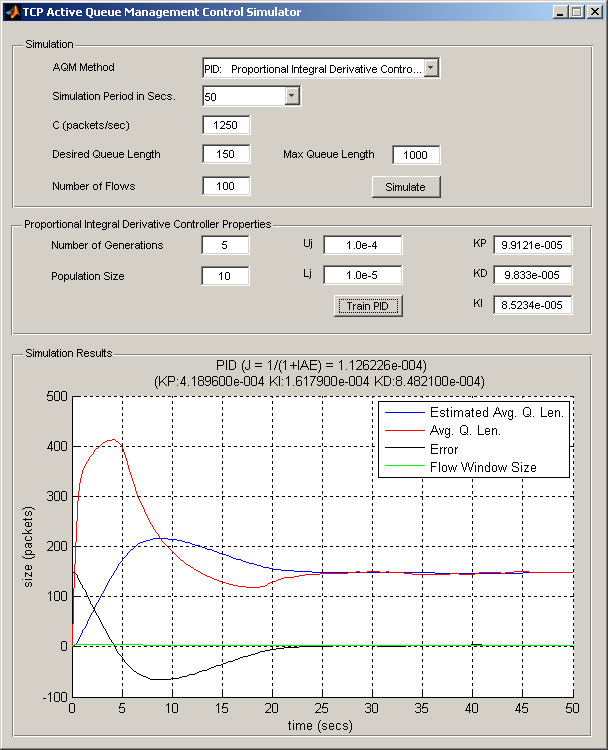


Figure : TCP Active Queue Management Simulator Graphical User Interface

The GUI includes many controls for user interaction. Uppermost panel is for simulation purposes and the middle panel is for GA based PID control parameter training purpose. The simulation results are figured out in the lowermost panel as seen in Figure 3. The user interface controls are as follows:

* + **Simulation Panel:**
    - **AQM Method:** User can select from 3 aforementioned active queue management controllers via this dropdown box.
    - **Simulation Period in Seconds:** User can select from hardcoded simulation period values via this dropdown box.
    - **C (packets/sec):** This edit box is used for defining the capacity of the egress link in packets/sec unit.
    - **Desired Queue Length:** User can define the desired queue level of the egress link via this edit box. This queue level together with the maximum queue length value is used for error calculation and directly affects the AQM method’s drop probability.
    - **Maximum Queue Length:** User can define the maximum queue length via this edit box. This is the hard limit for the queue on the egress link. Desired queue length on the other side is used for sharing a proportional amount of this maximum level for TCP traffic. This value is directly used by the NONE AQM method to limit the traffic at the queue limits.
    - **Number of Flows:** This edit box is used to define the number of flows on the coupled differential equations as defined in equation 3.b.
    - **Simulate:** This button starts a simulation with the given parameters in the simulation panel.
  + **Proportional Integral Derivative Controller Properties Panel:**
    - **Number of Generations:** This edit box is used to define the number of generations in the GA optimization heuristic.
    - **Population Size:** The number of chromosomes in one generation can be defined via this edit box.
    - **Uj:** This value is related to the encoding scheme of the generations as proposed in the paper (see Genetic Algorithm explanation below). The value entered in this edit box limits the value of PID control variables , and to an upper bound.
    - **Lj:** This value is related to the encoding scheme of the generations as proposed in the paper (see Genetic Algorithm explanation below). The value entered in this edit box limits the value of PID control variables , and to a lower bound.
    - **KP:** This edit box can be used to manually change the controller variable of the PID controller, but it can also be updated after a training process for PID optimization (see Genetic Algorithm explanation below).
    - **KD:** This edit box can be used to manually change the controller variable of the PID controller, but it can also be updated after a training process for PID optimization (see Genetic Algorithm explanation below).
    - **KI:** This edit box can be used to manually change the controller variable of the PID controller, but it can also be updated after a training process for PID optimization (see Genetic Algorithm explanation below).
    - **Train PID:** This button starts a PID control variable training session via genetic algorithm. Both the values defined in the PID controller properties and the simulation panels are used in the training process.
  + **Simulation Results Panel:** The output of the simulation process is given in the axes on this panel. The axes shows the estimated average queue size, average queue size, instantaneous window size and the instantaneous error rate with respect to time.
* **AQM Simulation (aqm\_sim.m):** This Matlab file is responsible for the solution of the aforementioned coupled differential equations over a given time interval [t0,tf]. The solution is dependent on the AQM method selected on the GUI from the AQM Method dropdown box. The differential equations are solved through an ODE23 solver given the differential equations defined in the queue function defined in queue.m. Since, the differential equation solutions are dependent on the historical error data in the PID method, the solver is accompanied by an output function defined in the queueOutput.m file. Each internal iteration of the ODE23 solver, processing the differential equations, saves the output values to global workspace. The output values saved from the previous iterations are used in the next iteration of the ODE23 function callback.
* **Queue Differential Equations (queue.m, queueOutput.m):** These Matlab files are responsible for the definition of the differential equations and the iterative output handling of the solutions achieved. The differential equations in queue.m are fed by the previous results achieved from the queueOutput.m function.
* **NONE AQM (none.m):** This Matlab file includes the node function handling the drop probability processing as defined in equation 4.
* **RED AQM (red.m):** This Matlab file includes the random early detection functionality and handles the drop probability processing as defined in equation 5.
* **PID (pid.m):** Proportional-Integral-Derivative controller functionality is handled in this file. What is different from the above 2 functions is that this function gets previous error values from the outputs of the previous queue differential equation iterations. But working on quite a large historical data may slow down the procedure and here in this work I have limited the error data to last 50 entries for the integral evaluation in the component of the PID controller.
* **Simulation Wrappers (simulateNone.m, simulateRED.m, simulatePID.m):** These wrappers are responsible for running the AQM Simulation module with right parameters according to 3 different AQM schemes. When the user pushes the Simulation button on the GUI, one of these wrappers are called according toe the AQM Method dropdown box selection. All the parameter read from the GUI, simulation period, link capacity, minimum and maximum buffer lengths and the number of flows are passed to the wrappers to be handled at the AQM simulation module level.
* **Genetic Algorithm (pidGenetic.m):** This module includes necessary functions to optimize a PID controller heuristically through changing , and control parameters. The paper designates a way to run the genetic algorithm step by step and the process is handled the same in this module.  
  The algorithm works on the generation of populations composed of chromosomes defining the , and values as defined in the paper. Each , and values are encoded into a binary string of constant length (see Figure 4).



Figure : Binary decomposition of control parameters on the chromosome

There are M1, M2 and M3 number of bits on each K value. In this project, each control parameter is composed of 10 genes on the chromosome. In other words, M1 = M2 = M3 = 10 which is also consistent with the authors’ work. The evaluation of the values of control parameters through the binary strings is another issue which has also been shown in the paper. The method used in the decoding is based on a minimum-maximum value bounded calculation as defined below:

|  |  |
| --- | --- |
|  | 13 |

is the decimal value of the 10 bits long binary number and and denote the upper and lower boundary of search range for the K values. and are used in **evaluate\_k** function and is configurable from the GUI. Selection of an upper and lower limit for the K values may help the genetic algorithm and better and precise results may be achieved.

At the end of the GA process, , and control parameters are updated in the GUI PID controller properties panel KP, KI, KD edit boxes.

The GA algorithm uses all of the parameters listed in the GUI. But the main parameters in concern are the number of generations, population size, Uj and Uk values. It must be expressed that the standard GA algorithm works as usual but the cost function of each chromosome is evaluated by simulating the TCP coupled differential equation system with the given parameters in the simulation panel. The GA cost evaluation mechanism uses the following objective function as define in the paper:



Figure : GA Objective Function

As seen on the objective function taken from the paper, the objective function given in equation 12 (Integral Absolute Error) has been normalized to get better results. This time the objective function becomes to be maximized throughout the optimization process.

# Tests and Comparison

This project also includes the investigation of the proposed AQM mechanisms in the practical work context. The tests made on the Matlab implementation can be classified as below:

* Testing the Genetic Algorithm method proposed in the paper: The tests for the first class are made in 2 steps. First the PID controller is tuned by pushing the train PID button, and secondly pushing the simulation button to see the results.
* Testing AQM mechanisms, comparing them within each other and to the foundings written on the paper: The tests for the second class, i.e. AQM testing, requires only to select an AQM method from the related dropdown box and then push the simulate button.

The tests are made with the default values of the controls in the GUI. A first time user shall only trig the simulate and train PID buttons to get an opinion about the general context. The test results will be shown on the graph with sufficient information. The user can repeat this test by selecting the AQM methods one by one and pushing the simulate button. As a next level, simulation periods can be increased to see if there is any degradation in the method.

**Note: The GUI should be opened from the Matlab console by typing tcp.fig and then pressing enter. The initialization of the figure by double-clicking on tcp.fig from the current folder view does not work.**

## Test 1: GA Test and PID Control Parameter Training

This test is triggered with the values shown in Figure 6. The number of generations and the population size has been kept low on purpose. Also, the simulation period has been kept 50 seconds to make the GA iterations faster. Because each chromosome in each generation is simulated for the given period and the total time spent in training increases as the simulation period increases.

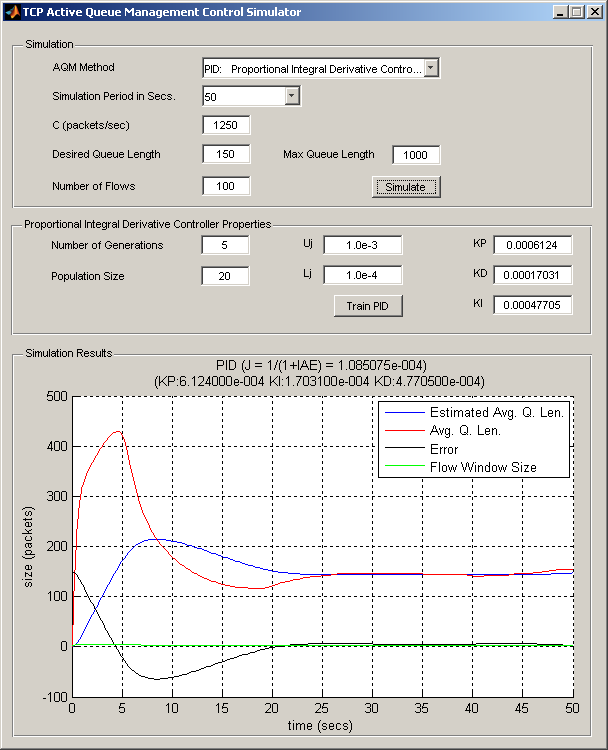


Figure : Test 1: GA Test and PID Control Parameter Training in 50 seconds interval

At the end of the training process, , and values were updated and then simulate button was pushed to see the performance of the optimized PID controller. It can easily be seen that the error line reaches zero, and the average queue lengths stabilize at the desired queue length level.

## Test 2: GA Test and PID Control Parameter Training

This test is made to see the performance of the tuned PID controller of the previous test on a larger period of simulation time. As can be seen from the following figure, the performance of the PID controller is stabilized in [0,1000] seconds time interval.

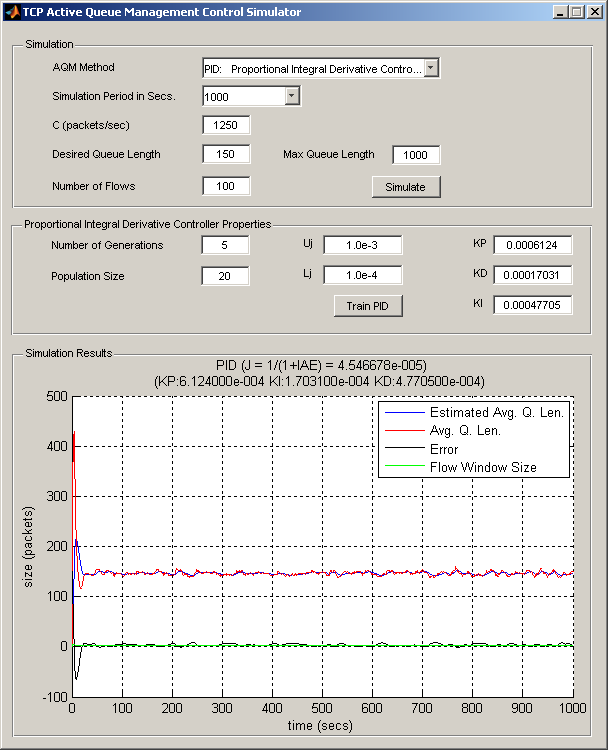


Figure : Test 2: GA Test and PID Control Parameter Training in 1000 seconds interval

## Test 3: Comparison of AQM Mechanisms in long time intervals

The configuration from the previous test is kept constant and the other AQM mechanisms are simulated in this test. Note that, first NONE method (see Figure 8) and then RED method is simulated (Figure 9). The PID method is not simulated again as it has been already tested in Figure 7. Comparing those simulation results from the graphs have shown that PID works better than the RED mechanism as there are a lot of unstable scattering in between queue margin levels defined at the RED mechanism (Queue levels for RED was defined to be max:170 packets, min:150 packets and pmax:0.1). As told in the paper RED is very sensitive to the given working parameters on different network loads. Since it has been designed in as an ad-hoc architecture, it needs to be configured well.

On the other hand, NONE AQM method has done nothing well in the test and was far away from being successful as its only aim is to limit the traffic at the hard limits of the maximum queue length. Thus it needs not any comparison with RED and PID.

It should be noted that the first response time of the PID controller compared to the RED controller is much better. Because, RED limited the average queue length to 320 packets and, PID limited the average queue length to 200 packets at startup. Actually, all those results achieved in this test are consistent with the comparisons made in different studies.

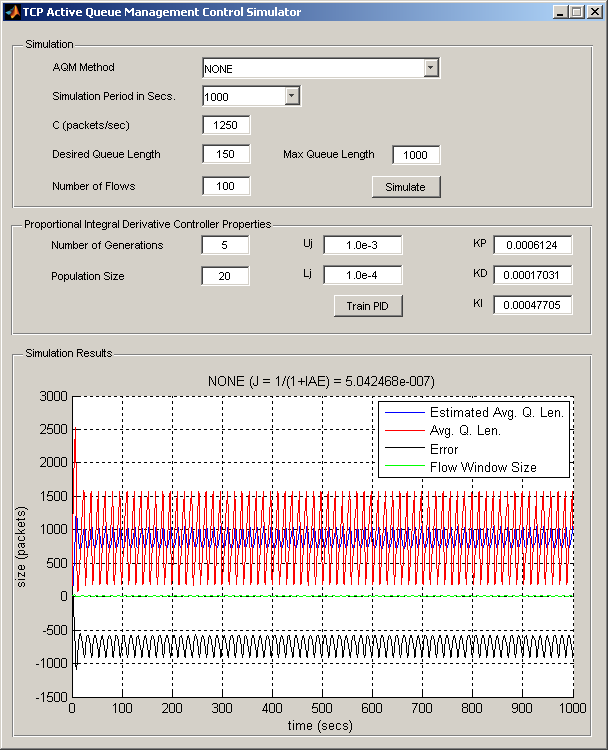


Figure : Test 3: Simulation of NONE AQM in 1000 seconds interval

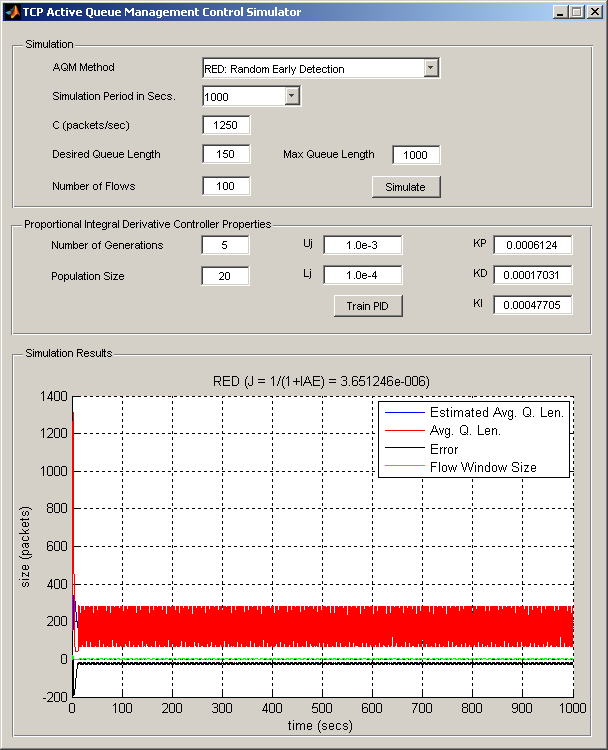


Figure : Test 3: Simulation of RED AQM in 1000 seconds interval

## Test 4: Comparison of AQM Mechanisms in short time intervals

The configuration from the previous test has only been changed by means of the period value. Now the simulation time is 50 seconds. Note that, first NONE method (see Figure 10) and then RED method is simulated (see Figure 11). The PID method is not simulated again as it has been already tested in Figure 6. Here in this test, the simulation results are also nearly the same with the previous test made with a much longer time interval. NONE AQM is again needs not to be compared to the other mechanisms. It should again be noted that the first response time of the PID controller compared to the RED controller is much better. Because, RED limited the average queue length to 320 packets and, PID limited the average queue length to 200 packets at startup. Actually, these results are again consistent with the comparisons made in different studies.

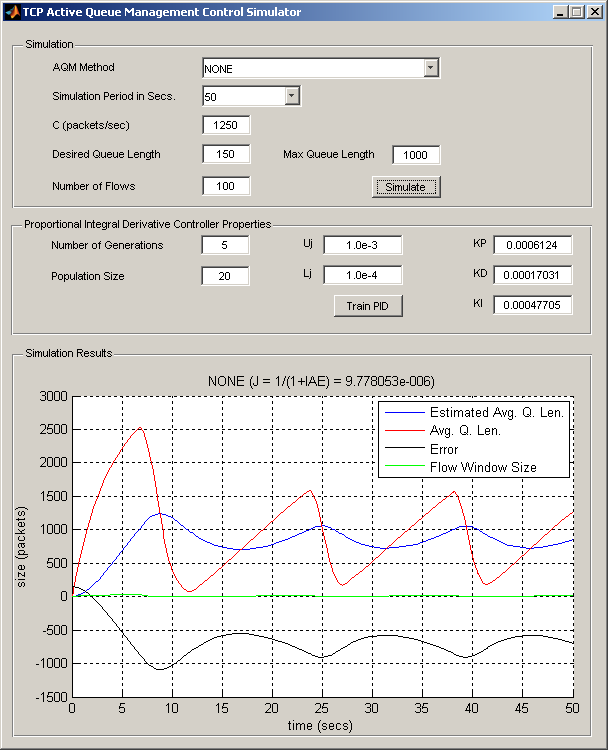


Figure : Test 4: Simulation of NONE AQM in 50 seconds interval

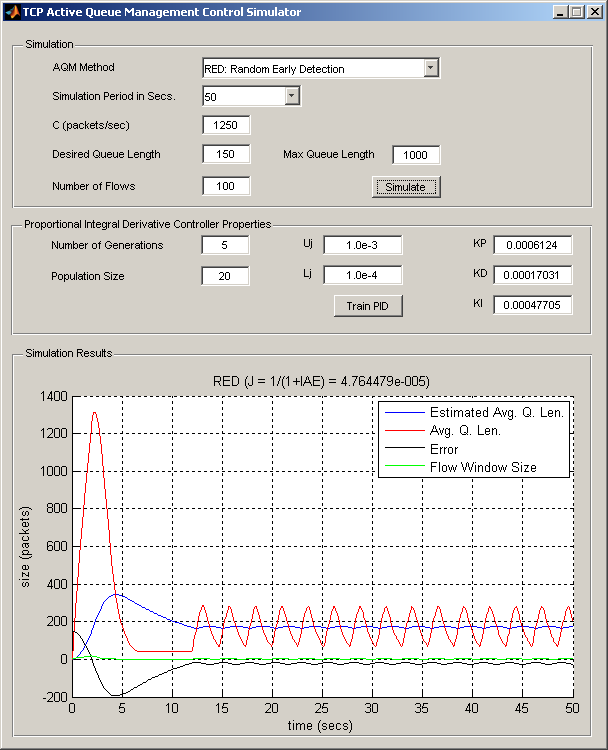


Figure : Test 4: Simulation of RED AQM in 50 seconds interval

# Achievements:

This term project included a detailed examination of an active queue management system for TCP flows. Thereby, mathematical modeling of the problem, the AQM mechanisms and the optimizations of the AQM mechanism parameters have been researched through different academic works.

Taking the main article as a base, the methods and mathematical models explained in the other references were also studied and also used in the practical work. As there have been too many research activities done in this subject, this work has only been able to include a small subset of it. But, indeed, this has been a good experience to learn about stochastic fluid based TCP communication modeling and its linkage to the AQM congestion control mechanisms. As the results found in this project were similar to those of the research activities, it has also been good to achieve the term project goals.

# References:

[1] GA-based PID active queue management control design for a class of TCP communication networks, Chang-Kuo Chen, Hang-Hong Kuo, Jun-Juh Yan, Teh-Lu Liao, 2009

[2] Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED, Vishal Misra, Wei-Bo Gong, Don Towsley, 2000

[3] GA-Based PI-PD Controller for TCP Routers, R. K. Sundaram and P. K. Padhy, 2013

[4] PID Parameters Optimization by Using Genetic Algorithm, Andri Mirzal, Shinichiro Yoshii, Masashi Furukawa, 2013

[5] Design of PID Controllers for TCP/AQM Wireless Networks, Teresa Alvarez, 2012