Internet congestion control

Internet Congestion Control

Paul van Dongen

DCT 2004.05

Traineeship report

Coach(es): Y. Chait

Supervisor: M. Steinbuch

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Technology Group

Eindhoven, February, 2004

Abstract

The analytical treatment of communication between two computers or 'end systems' has been recently proposed. Analytical models have progressed to the point that conventional control theories are now implementable. Modelling, however, inherently means simplification. Reasons for simplification can be for computational ease or estimation that a certain influence will be minimal.

Improving network behavior can be sought in the currently implemented controller schemes used in the router of the network. When end systems communicate throughput is limited by bandwidth available in the network. An end system does not know what is availability and needs to probe the network. The router needs to actively set boundaries by means of congestion control. Once controllers are derived they will be tested by simulations and then experiments for validation.

It will be shown using non-linear design techniques, improvement upon the current situation can be achieved. The system however has a lot of unknowns. The control scheme is shown to still be applicable, but with trade-off. These limitations will be investigated.

Contents

Chapter 1

Introduction

Communication in a network environment is the main topic in this discussion. In network communication computers send information to each other by means of routers and a sending protocol. This sending protocol however has no knowledge about the network properties. To ensure communication and especially to ensure a Quality of Service (QoS) this behavior needs to be controlled. Devising new control schemes is currently of great interest. A reference controller used in this report is a PI control scheme (Proportional-Integral) because of its significant improvement of its predecessors RED (Random Early Detection). Cisco Systems manufactures routers and its current schemes are based upon RED schemes. WRED (Weigthed RED) is Cisco's implementation of RED for standard Internet operating system platforms [1]. Reference [1] also states if the WRED is not configured, the router uses the cruder default packet drop mechanism called tail drop. Tail drop means if the router is congested the rest of the ingoing information is discarded. This work will represent background and an attempt at the improvement of the current controller. Linear and non linear control techniques will be applied. This Chapter will provide information to understand the basics about the system dynamics that will be presented in Chapter 2. The following two subsections will provide historical and network background information.

1.1 Historical Background

In the 1960s computers were developed to the extent that the question arose how to link computers to each other. The world's dominant network then was the telephone network. This is a circuit switching network, meaning the network established an end-to-end circuit between two hosts. During communication the link is reserved for this communication. Since traffic generated between computers would most likely be 'bursty', having periods of activity and inactivity a packet switching method was developed. It is more efficient and more robust then the circuit variant [2].

The first packet switching network was the result of ARPA (Advanced Research Project Agency). In 1969 the first interface message processor was installed at UCLA Berkeley. At the end of 1969 three more nodes were installed and the ancestor of internet contained four nodes or hosts. By 1972 the ARPAnet consisted of 15 nodes. Around this time additional packet switching networks came to be. In 1973 Robert Metcalfe's Ph.D. thesis presented the principles of Ethernet. This led to an increase in local area networks (LAN's) working over small distances based upon the ethernet protocol. Great effort was put into interconnecting these networks, from this pioneer work the term 'internetting' was derived. This gave birth to the predecessors of the main internet protocols (ALOHA,TCP,IP and UDP [2]). By the end of the 1980s the number of hosts connected to the public internet would reach 100,000.

In the 1990s the world wide web was released, bringing Internet into homes and businesses of millions of people across the globe. The web contained four key components, being initial versions of HTML, HTTP, a web server and a browser. As time progressed better browsers were developed with GUI interfaces (Netscape, Microsoft) and Internet kept expanding and expanding to what it is today.

1.2 Network Background

The Internet is a very well-known result of the research that started once computer networking was first conceived. It is a concept that is embedded in the life of the majority of people on the globe. Its roots lie in the networks whose principles are presented in this section. The reader is referred to [2] for more details.

The Internet is a world-wide computer network that connects millions of computers. Desktop PC's, Unix-based workstation and servers that store and transmit information such as email messages and web pages are all on this network. In Internet jargon they are all referred to as end- or host systems. Browsing the Internet and using email services means using HTTP, SMTP, POP3 and IMAP, which are network applications. These all run on the end systems. End systems are not usually connected directly to each other through a single communication link (coaxial cable, copper wire, fiber optics or a radio spectrum). Usually a router links end systems indirectly, routers accepts information arriving on the incoming communication link and forwards it to one of its outgoing links. The way information travels from the sending end system via a series of links and routers is called the *route* or *path* through the network. The transmission protocol of Internet uses a packet switching technique which entails that no dedicated path is provided for communicating end systems. This means a path, or a part of it, is shared among users or end systems.

Because of the many different users some guidelines are needed, the Internet's principal protocols are collectively known as TCP/IP and [2] describes their function in great detail. In a nutshell, network protocols play a big part in the communication between end systems. They define the format and the order of messages exchanged between two or more communicating end systems. They also define the actions taken when transmitting and receiving of messages and other events.

To understand the way computers communicate, a simple communication situation is now sketched. If someone wants to visit a website the end system requests a communication request' message to the server via routers. The message travels the route and (hopefully) reaches the server end system. If the server has enough resources (bandwidth) and is not down it sends a connection reply. The sender then sends the page information by a *get* message. The final message will be the server with the web document. So communication is made possible trough the routers.

Ending this introduction the protocol UDP (User Datagram Protocol) is mentioned. There is a difference in TCP (Transmission Control Protocol) and UDP transmission. TCP is a reliable transport, meaning it guarantees every information packet will arrive. Packet loss will result in having to resend packets, resulting in additional communication and timing schemes. UDP is not a reliable way of communication and does not guarantee all information is delivered. If a program is using UDP as transmission protocol the communication will go faster for it is a connectionless way of communication. Reference [3] contains research about internet traffic and it states that in 1998 95 % of all Internet traffic is TCP based. UDP is used for real time video and voice services. On Internet Live video streams are well-known. A recent application for calling over the Internet is Skype [4]. It is freeware and the audio quality this software provides is very good. (It uses P2P (peer-to-peer) communication and basically works like MSN messenger. More information can be found on [4]). Reference [5] (dated 1997) however describes a number of additional applications like video conferencing. It also refers to a survey that mentioned internet phone will be an application that will drive the implementation of better networking schemes that can cope with the changing network behavior and applications. This is mentioned to state the ever changing nature of computer networks. It also mentions the effects on TCP/IP in the past ten years. Ever increasing demands on the internet have led to a number of changes in its working principles.

Current routers across the world are however still working and have been working despite this increase of other applications. The reason is because the routers now are over provisioned. Internet traffic reports can be found online. For instance [6] is a reference on which routers on every continent are updated. Its time delay is presented and an index is presented on performance. The index itself is a relative measure of performance based upon previous time delays achieved by the router. Also the number of dropped packages (and percentage) is presented and as can be seen this is very low. Basically if a router is running (not down) it shows 0 % drop of packets. Low package drop levels with simple tail drop schemes imply over provisioned routers.

The rest of this report will start with Chapter 2 where the mathematical model of the system will be presented. Chapter 3 will deal with the current implementation of the AQM (Active Queue Management) controller for a single congested link network and its background. Also two other linear approaches will be mentioned here. In Chapter 4 a non-linear technique will be presented. This non linear technique however requires full state feedback. For this an observer will be needed which will be mentioned in 5. In Chapter 6 NS-2 (Network Simulator version 2.0 [7]) will be introduced and used to validate one controller by performing experiments.

Chapter 2

Single Congested Link Network

In this Chapter a mathematical model for the dynamics of a single congested link network will be described.

2.1 Network Representation

The simplified computer network is presented in Figure 2.1.

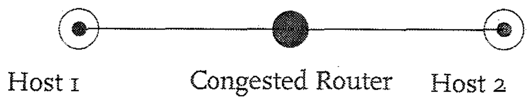


Figure 2.1: Schematic representation of single link single user network

It consists of one router connecting two end systems, for instance a personal computer and a webserver. Under TCP, a sender has authority to set its transmission rate using a 'window flow-control' mechanism. The sender sends a number of packets which represent a number of bytes (in this discussion 1 packet averaged is equal to 500 bytes). The window size is the number of packets the sender will or can send. A dail up telephone line has a maximum bandwidth of about 56.6 [Kbytes/s], a telephone line can send as many packets per second as the line can handle. The sender continuously probes the networks available bandwidth and increases its window size to get the maximum of the network resource. For each successful end-to-end packet transmission TCP increases the senders window size. Conversely, TCP cuts the window whenever a senders packet does not reach the receiver. Such packet losses affect network performance by decreasing the senders effective transmission rate, because lost packets need re-sending for else a file might not be complete. This however introduces delay. By itself, TCP has no information of network mechanisms contributing to packet loss, it does not know when the router is congested or close to it. Thus, routers must assume a role in network management to prevent congestion by preemptively signaling TCP rather than have it react to unreceived packets upon router congestion.

Packet dropping can be noticed with a timer mechanism. When a packet is send, a timer is started and stopped when the acceptance confirmation of the router is received. If a stated 'maximum time' is exceeded the packet is thought of as lost in the network.

Slow start is the process of probing the network and increases window size exponential (2^n). Slow start continuous until the window size exceeds a set threshold. Then congestion avoidance comes into play which basically changes exponential growth to linear growth by juts adding 1 packet to window size until a loss is detected. When a time out is noticed the threshold is set to the half of the current window size and the window size is set to 1. This is often referred to as Tahoe algorithm. Reno is a variant of Tahoe that incorporates a fast recovery mechanism, which means it cancels out slow start due to timeouts. There are other mechanism implemented for those the reader now is referred to [2]. For completion there are several TCP source algorithms that exist, for instance Tahoe, Reno and Vegas. They all have different algorithms yielding specific window size versus time graphs.

2.2 Mathematical Background

2.2.1 Equations

The dynamics of a TCP network is a discrete event. Recently a fluid-flow model for TCP was developed in [8] so that conventional control techniques can be applied to it.

When ignoring the slow start phase it shows TCP essentially increases its window size by 1 every round trip time (additive increase) when the network path is not congested, and decreases its window size by a factor 2 each round trip time a packet loss is detected. When assuming slow start occurs very fast this describes the situation well. The set back to window size is 1 is quickly recovered to the new threshold from which the growth is linear again. This is called an **additive-increase, multiplicative-decrease** (AIMD) algorithm, given by the following equations

$$\dot{W} = \frac{1}{R(t)} - \frac{W(t)}{2}\frac{W(t-R(t))}{R(t-R(t))}p(t-R(t)) \tag{2.1}$$

$$\dot{q} = \begin{cases} -C + \frac{NW(t)}{R(t)} & \text{if } q > 0 \\ max\left(0, -C + \frac{NW(t)}{R(t)}\right) & \text{if } q = 0 \end{cases}, \tag{2.2}$$

where the following holds:

- W = average TCP window size [packets];

- q = average queue length [packets];
- $R(q(t))$ = round trip time [s];
- C = link capacity [packets/s];
- T_p = propagation delay [s];
- N = load factor (# TCP sessions) [-];
- p = probability of packet mark [-].

The round trip time is defined as

$$R(t) = \frac{q(t)}{C} + T_p.$$

The queue length q and window size W are positive and bounded quantities; i.e., $q \in [0,q_{max}]$ and $W \in [0,W_{max}]$. Also, the marking probability is bounded: $p \in [0,1]$. These differential equations are illustrated in the block diagram of Figure 2.2 which shows TCP window-control (left) and queue dynamics (right). We now consider an AQM scheme in which the probability of packet marking depends on actual queue length.

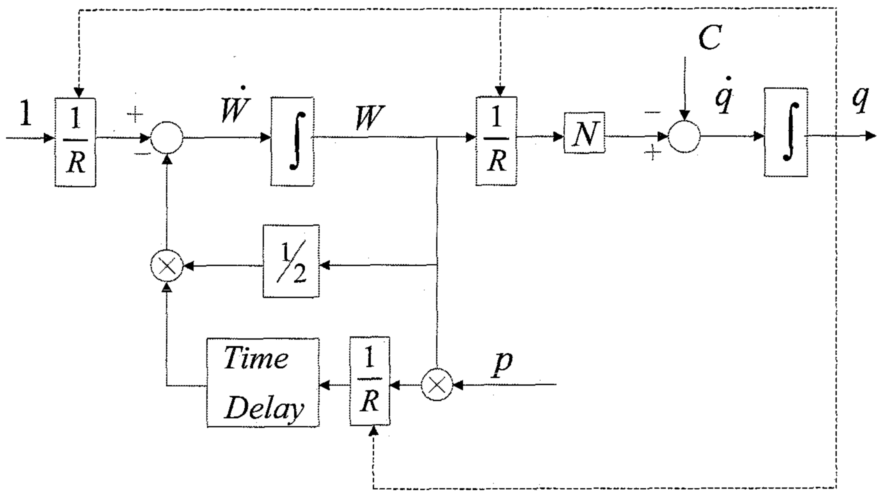


Figure 2.2: Block diagram of a TCP connection

2.2.2 Parameters

C is the link capacity. A link can sustain 15 mega bit per seconds. A packet in this discussion is set to be averaged at 500 bytes. This can be found in [9]. Conversion bits to bytes has a factor 8 yielding 500 [bytes/packets] $\cdot$ 8 [bit/byte] $\cdot$ C [packets/s] = $15 \cdot 10^6$ [bits/sec]: $C = 3750$ [packets/s]. Propagation delay is a transportation time. The order is about $1 \cdot 10^2$ [ms] as can be seen in [9]. In this particular case it is set to 200 [ms].

The router buffer limit is set to 800 packets as is also done in ([9]). It is not wise to have AQM try to achieve high steady state queue lengths because it yields a high time delay and thus a lower effective throughput (maximum throughput is end system send rate). 800 packets is a maximum where a much lower current queue length is to be aimed for. A filled buffer is worse then an empty one, for this will mean packet loss, having to resend packets, severely reducing effective throughput. An empty buffer means under utilizing a link which is a lesser evil, but not to be achieved.

2.3 Linearized Model

System linearization is presented here since it will be needed in deriving linear controllers. In [10] and [11] the linear systems are derived (slightly different, same end result).

The following assumptions are made:

- $N(t) \equiv N; C(t) \equiv C$
- $R(q(t)) \equiv R(q)$; meaning a constant for queuelength is chosen yielding a constant round trip time
- time delay argument in window control dynamics is disregarded which means $W(t)W(t-R)$ can be written as $W(t)^2$ as seen and explained in ([11])
- The implicit time delay dependance R on itself ($R(t-R)$) in the window control dynamics disregarded as seen and explained in ([11])

The operating point when taking (W_0, q_0, p_0) is defined as follows

$$\dot{W} = 0 \Rightarrow W_0^2 p_0 = 2$$
$$\dot{q} = 0 \Rightarrow W_0 = \frac{R_0 C}{N}; R_0 = \frac{q_0}{C} + T_p \quad .$$

Linearizing about this equilibrium

$$\partial \dot{W}(t) = -\frac{2N}{R_0^2 C}\partial W(t) - \frac{R_0 C^2}{2N^2}\partial p(t - R_0) \tag{2.3}$$

$$\partial \dot{q}(t) = \frac{N}{R_0}\partial W(t) - \frac{1}{R_0}\partial q(t), \tag{2.4}$$

where

$$\begin{aligned}\partial W &= W - W_0\\ \partial q &= q - q_0\\ \partial p &= p - p_0.\end{aligned}$$

Going to Laplace domain yields the following block representation and equations.

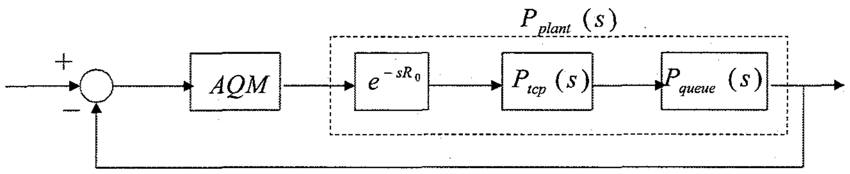


Figure 2.3: Block diagram of a linearized TCP connection

This system has two unique poles as can be seen clearly in the following Equations

$$\begin{aligned}P_{tcp}(s) &= \frac{\frac{R_0C^2}{2N^2}}{s+\frac{2N}{R_o^2C}},\\ P_{queue}(s) &= \frac{\frac{N}{R_o}}{s+\frac{1}{R_0}}.\end{aligned}$$

The plant dynamics can be written as second order system with a time delay as seen in the following Equation

$$\begin{aligned}P_{plant}(s) &= \frac{\frac{C^2}{2N}e^{-sR_0}}{\left(s+\frac{2N}{R_o^2C}\right)\cdot\left(s+\frac{1}{R_o}\right)} = \frac{e^{-sR_0}}{Ms^2+Bs+K},\\ M &= \frac{2N}{C^2}; B = \frac{2NR_oC+(2N)^2}{R_0^2C^3}; K = \frac{(2N)^2}{(R_0C)^3}.\end{aligned} \tag{2.5}$$

To get a feeling for this system the bode diagram of the system (Equation (2.5)) for different system parameters is plotted. Robustness to parameter deviation will be a part of validating the assumptions made. Its sensitivity for chosen variations is shown in Figures 2.4 and 2.5. The choice for the parameters will be explained later in this discussion. For now it can be said that changing these parameter in this way severely changes total systems behavior.

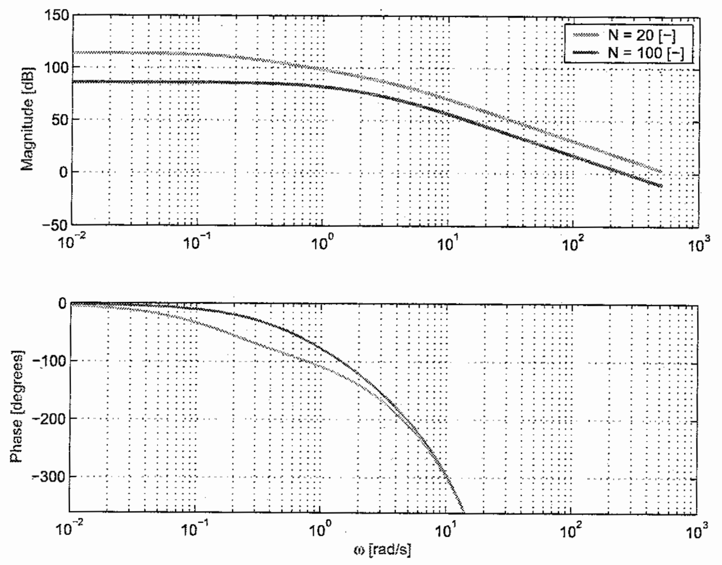


Figure 2.4: bode diagram of the linearized system for varying N; $T_p = 0.2$

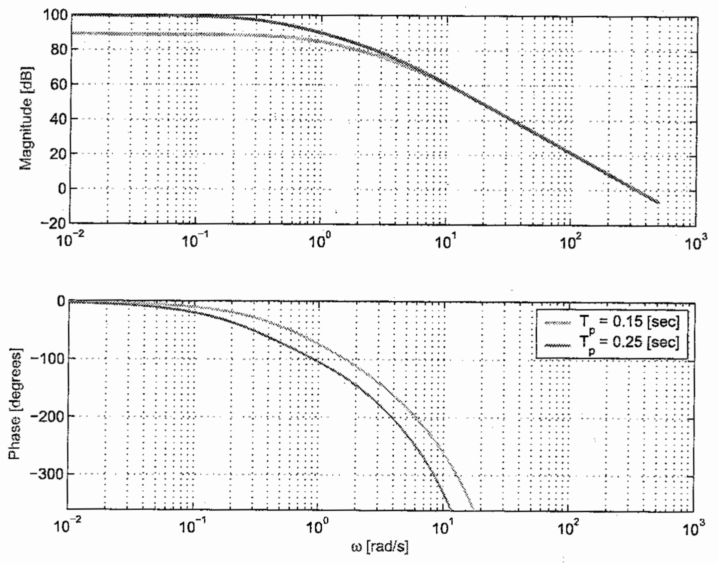


Figure 2.5: bode diagram of the linearized system for varying T_p; $N = 60$

Chapter 3

Linear Controllers for Single Congested Link Network

This Chapter describes a number of controllers that were derived based upon the system as presented in Section 2.2. First, however, some background is given which explains choices made for these controllers. All controllers will be based upon the linearized system and then tested in simulation with the non-linear system. Chapter 6 describes validation methods for these simulations using discrete event software.

3.1 Control Goals

Our goal in designing new controllers is to improve on the state-of-the-art, PI controller [10]. In general one is interested in its performance figures.

1. Steady state behavior: $q = q_{ref}$.
2. Transient behavior: fast response.
3. Robustness: Maintaining the first two items in spite of varying network parameters N and T_p.
4. Increase feasible domain of robustness for N and T_p.

The first means that in the steady state behavior little deviation is allowed from the setpoint or desired value. This immediately implies the application of a noticeable integral action in the controller [12]. The second item involves bandwidth and stability margins. For the third the Sensitivity theorem, mentioned in [12], can be used. When evaluating the sensitivity of a system its robustness can be compared to an upper limit that is considered acceptable. The fourth item will be a goal to be achieved. When implemented the new controller(s) will be tested for feasible domain.

To show robustness of controllers the load and propagation delay are varied in a domain. A choice for the variation is $N \in$ [20,100] which is common when reviewing for instance [10]. For the linear system if the low frequent gain is validated ($\frac{(CR_0)^3}{(2N)^2}$), this gives about a 30 [dB] difference in this range of N, which is very high compared to 'normal' systems. This is validated for C = 3750 [packets/sec] and $R_0 = \frac{q_0}{C} + T_p = 0.246[s]$. T_p is also changed because this has no fixed value. The range used is $T_p \in$ [0.15,0.25] which is chosen to be around the value chosen in [9].

3.2 Current Implemented Controller

We now evaluate the performance of the PI design, then proceed with designing several competing designs. The PI controller has the following equation

$$C_{PI}(s) = P\left(1 + \frac{K_i}{s}\right), \tag{3.1}$$

where

$$\begin{aligned} P &= 1.818 \cdot 10^{-5}[-], \\ K_i &= 0.53[-]. \end{aligned}$$

In Figures 3.1 through 3.4 the system with controller are presented. The background behind the given constants is given in [10]. This paper ([10]) is in fact a description of a successful attempt in improving an older version of AQM which used a RED controller [8] & [11]. Without going into too much details of the RED controller, RED is a type of control which gives the drop probability p based upon an averaging filter and a packet-marking profile. The profile is a function that consists of three subsequent linear functions. This function needs the average queue length as input thus the averaging filter is needed. This is all described with proper background in [8]. The idea is that before reaching the limit in queue length the controller starts dropping packets randomly because this will result in time outs at the sender, which will result in a the window size W to be altered before the queue is full and the actual tail drop occurs. For stability reasons however, mentioned in [10] and [11], the bandwidth is relatively low. The controller in Equation (3.1) improves this bandwidth with a factor 10 to 0.53 [rad/s].

The properties of the current controller are given in the Figures 3.1 - 3.4. Key is the delay which restricts the bandwidth of the controller system. As can be seen in Figure 3.3 the sensitivity function remains under 2 [dB] where 6 [dB] is a generally accepted robustness margin. Moreover the time delay that shows in the sensitivity function as oscillations around the 0 [dB] line are very small. This can be seen in Figure 3.3.

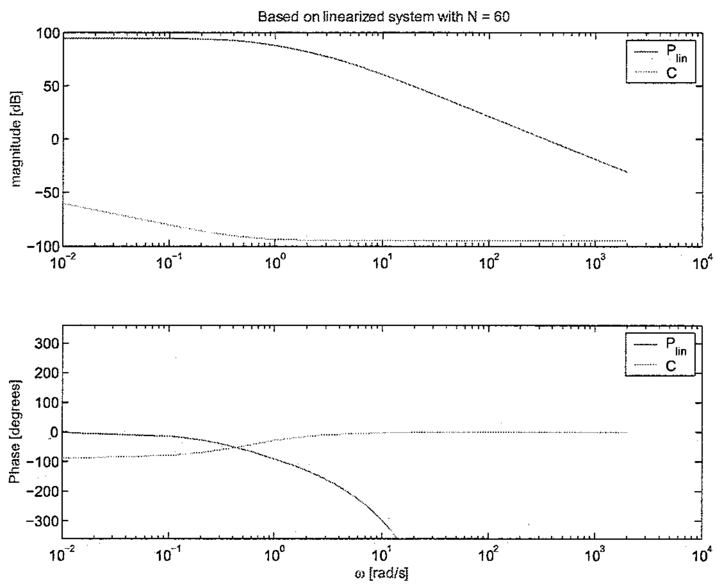


Figure 3.1: frequency response function system and PI; N = 60

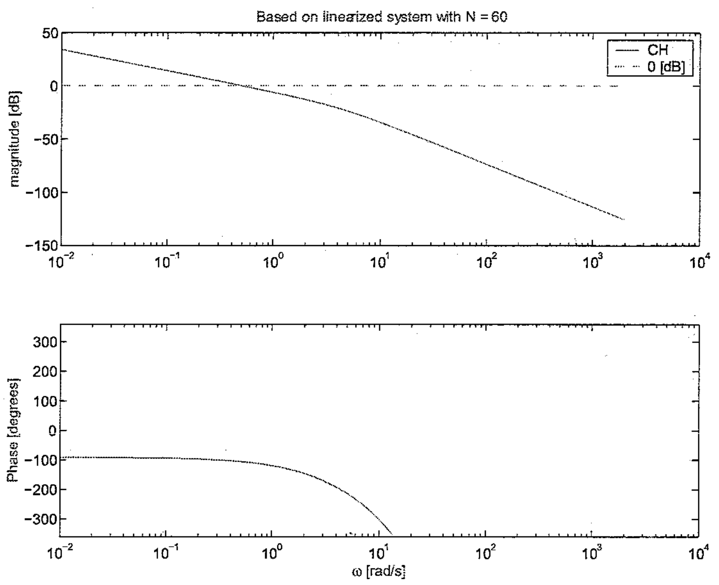


Figure 3.2: frequency response function open loop

Control techniques are applied in attempt to try and improve this controller as will be explained in Section 3.3. This is attempted for the PI controller, which

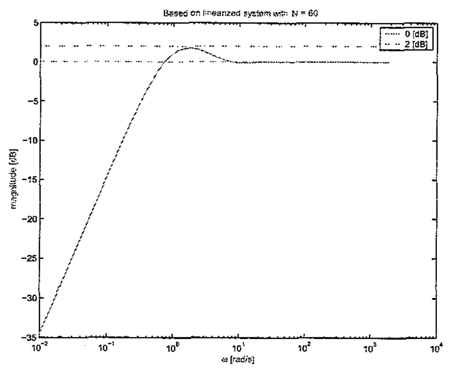


Figure 3.3: Sensitivity function (robustness)

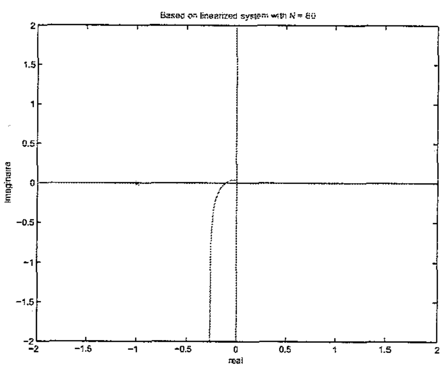


Figure 3.4: Nyquist diagram (stability)

is seen as the best controller for this system because it betters the RED controller. It is possible to try and squeeze performance of this controller scheme. The trade-offs between robustness and speed of the system can be varied by 'playing' with the sensitivity function (with proportional gain and integral pole as fine tune parameters). However because two engineers can have a different opinion which is more important, pushing trade-off is not used as method of improving the controller. The results the current controller has on controlling queue length to go to its reference value can be seen in figure 3.5.

3.3 Other Linear Controllers

The bandwidth is limited because of the time delay. This time delay will be the bottle neck of the linear control techniques because once the phase drops to far even very aggressive controllers cannot compensate for this phenomena which decreases proportional to frequency. Controllers can somewhat help bandwidth by adding phase to the open loop behavior. This will allow the crossover frequency to be pushed further keeping stability (about) the same but increasing system response. The two controllers derived will be explained and tested in the following two subsections. Both will focus on increasing phase. Adding derivative action will add a pole thus a maximum of 90 degrees of phase. The lowpass filter is included to eliminate high frequent amplification which is needed in most experimental environments.

3.3.1 PID Controller

The system can be written in the form of a mass-spring-damper system as seen in Equation (2.5). A PID controller can be written to exactly cancel out the systems two poles. In [10] the same technique is used to cancel the 'bottle-

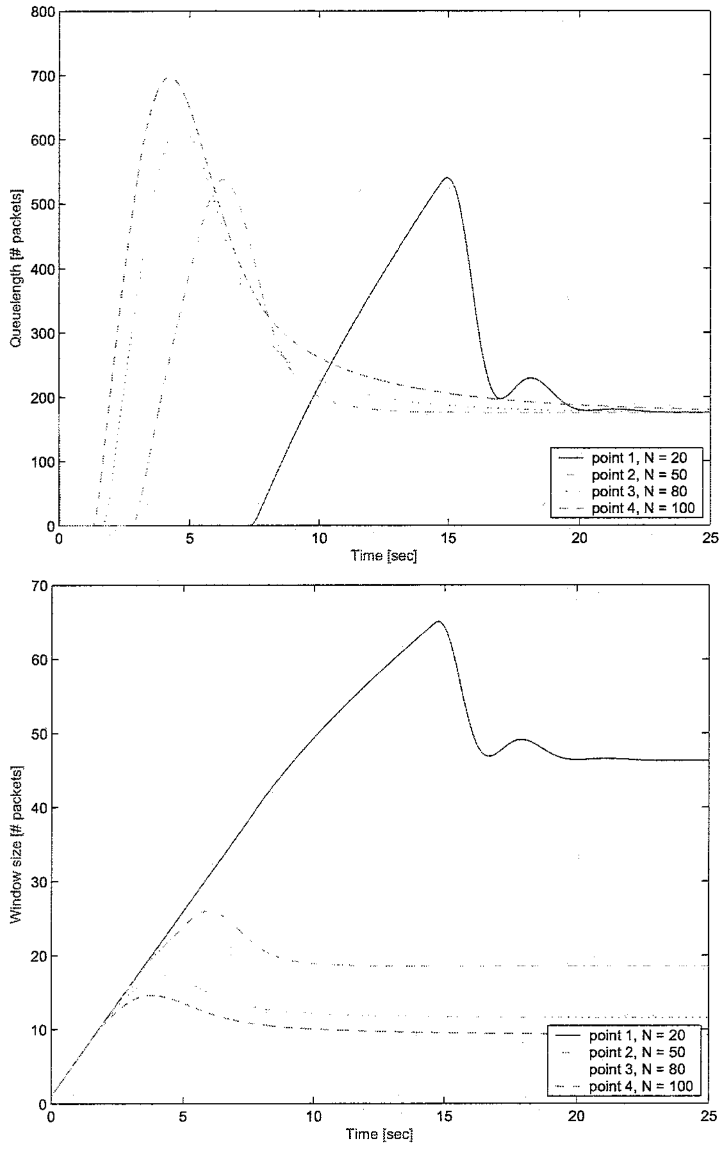


Figure 3.5: top: queue length PI controller; bottom: window size PI controller

neck' pole by choosing the PI controller pole equal to that pole. Here this is taken one step further. The PID controller is given by

$$
\begin{aligned}
PID(s) \; &= P\left(K_d s + 1 + \frac{K_i}{s}\right) \\
&= \frac{(PK_d s^2 + Ps + PK_i)}{s} = \frac{Ms^2 + Bs + K}{s} \quad ,
\end{aligned}
\tag{3.2}
$$

Chosing the PID controller like this will fix all PID control parameters when solving the three equations with three unknowns, since for fixed N and T_p, M, B and K are known. A lowpass filter however has to be added because of the resulting closed loop transfer function ($L(s) = PID(s) \cdot P(s)$;Equation (3.3)). If the PID controller would exactly cancel out the system then no pole can be chosen to influence the bandwidth of the system. Adding the lowpass filter enables this again

$$\begin{aligned} L(s) &= \frac{e^{-sR_o}}{Ms^2+Bs+K} \cdot \left(\frac{Ms^2+Bs+K}{s(\frac{1}{\tau}s+1)} \right) \\ &= \left(\frac{e^{-sR_o}}{s(\frac{1}{\tau}s+1)} \right) \end{aligned} \qquad . \tag{3.3}$$

Now the only parameter left to choose is τ and this should reflect the desired bandwidth <u>and</u> the cross over frequency. For placing bandwidth as function of τ the following Equation holds

$$|L(jw_c)| = \left| \frac{e^{-jw_cR_o}}{jw_c(\frac{1}{\tau}jw_c+1)} \right| = 1; \tau = \sqrt{\frac{w_c^2}{\left(\frac{1}{w_c^2}-1\right)}}. \tag{3.4}$$

When solving Equation (3.4) it will yield that the cross over frequency is limited at $w_c \rightarrow 1$. The relationship between τ and w_c shows $w_c \rightarrow 1$ for $\tau \rightarrow \infty$ and also that w_c cannot be larger then 1. Important to see is that if w_c is respectively 0.9 and 0.99 τ equals 1.85 and 6.49. τ does not need to approach ∞ to let w_c approach 1, the function is asymptotic.

For setting phase the following Equation holds

$$\begin{aligned} \angle L(jw_c) &= \angle \frac{e^{-jw_cR_o}}{jw_c(\frac{1}{\tau}jw_c+1)} = -w_cR_o \cdot \frac{180}{\pi} - 90^o - arctan(\frac{w_c}{\tau}) \\ &\approx -14^o - 90^o - arctan(\tau^{-1}) \approx -104^o \end{aligned} \qquad . \tag{3.5}$$

This means that the highest possible phase is achieved when τ is chosen as high as possible since $arctan(\tau^{-1}) \rightarrow 0$ for $\tau \rightarrow \infty$. However for small values $arctan(x) \approx x$. This means with the single parameter τ the system can be given a w_c of almost 1 [rad/s], and a phase margin of about 76 [degrees]. This is about the same phase margin the PI controller of [10] and double the cross over frequency. Figures 3.6 - 3.9 depict the controller.

Compared with the PI, this controller has a higher bandwidth of factor 2 with about the same phase margin and similar robustness. This should reflect in the simulation by a modest improvement in speed and better behavior for varying load. The simulation has the load N varying between 20 and 100. The results depicted in Figure 3.10 have the controller with fixed N = 60. The PI controller was created with this nominal value of N in mind therefore its only

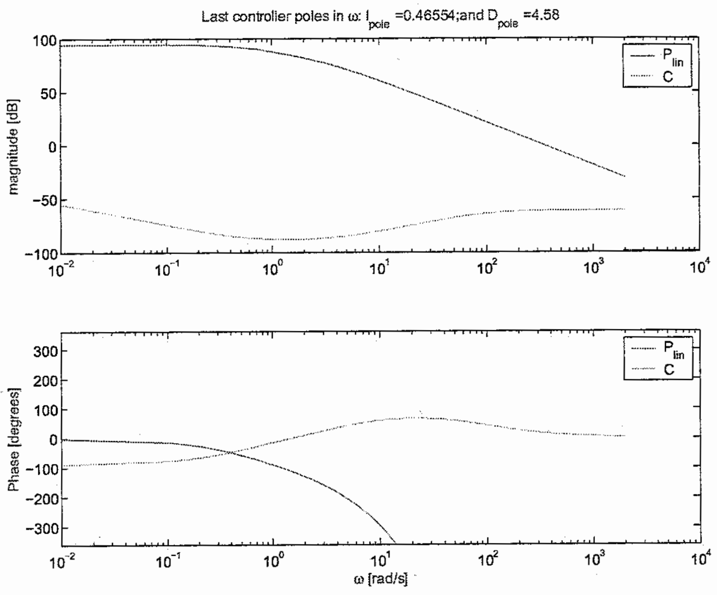


Figure 3.6: frequency response function system and PID; N = 60

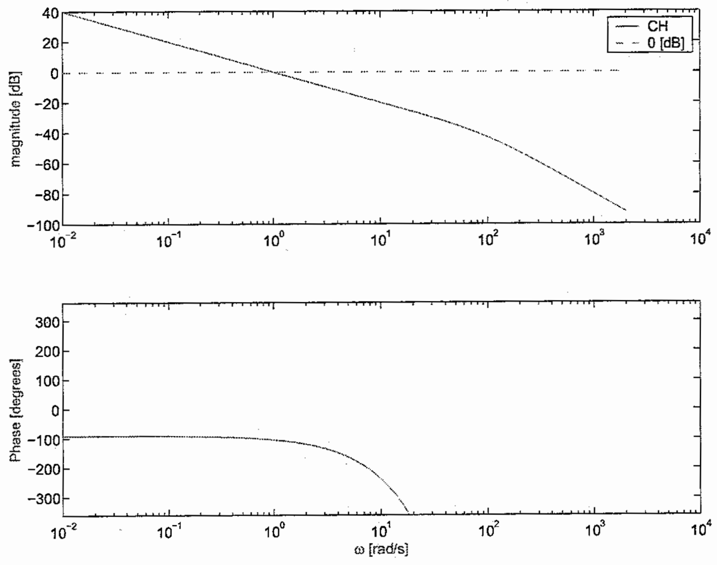


Figure 3.7: frequency response function open loop

fair to do the same. Robustness influences with regard to varying load can be assessed. In the constants M, B and K as seen in Equation (2.5), the load N

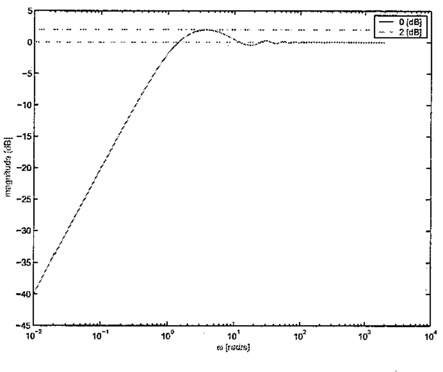


Figure 3.8: Sensitivity (robustness)

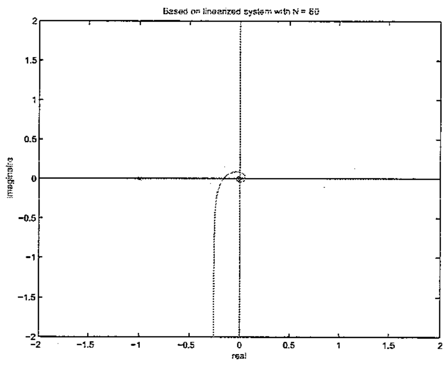


Figure 3.9: Nyquist (stability)

is in the numerator. When the simulation load is smaller then the load in the controller this increases bandwidth. This could result in stability issues. For higher simulation loads the system will therefore become more sluggish. The results seen in Figure 3.10 are as expected. The improvements shown are not ground breaking. But worth mentioning.

3.3.2 PI with Stretched Notch- and Low Pass Filter

A stretched notch filter is the same as a notch filter used to filter out a specific frequency, the only difference is that in its transfer function

$$NF(s) = \left(\frac{\frac{1}{\omega_1^2}s^2 + \frac{2\beta_1}{\omega_1}s + 1}{\frac{1}{\omega_2^2}s^2 + \frac{2\beta_2}{\omega_2}s + 1)} \right),$$

the two frequencies (ω_1 and ω_2) are not chosen equal. Adding a 'stretched' notch filter to the current PI controller can add phase because two zeros are added yielding up to a +2 slope in bode diagram which is the same as 180 degrees of phase advantage between ω_1 and ω_2. Low pass is added to prevent high frequent amplification. This controller has too many variables design analytically so some choices for this design have to be made. All choices will be supported so without control background they will seem logical. The controller is of the following form

$$C_{PInotch}(s) = P\left(1 + \frac{K_i}{s}\right) \cdot \left(\frac{\frac{1}{\omega_1^2}s^2 + \frac{2\beta_1}{\omega_1}s + 1}{\frac{1}{\omega_2^2}s^2 + \frac{2\beta_2}{\omega_2}s + 1)} \right) \cdot \left(\frac{1}{\frac{s}{\tau} + 1}\right).$$

The PI controller part in this is chosen the same as that of [10] for its basis is cancelling out a bottleneck pole. Normally a notch filter is used to filter or amplify frequencies, the factor β_1/β_2 can be chosen to filter out a frequency (like a

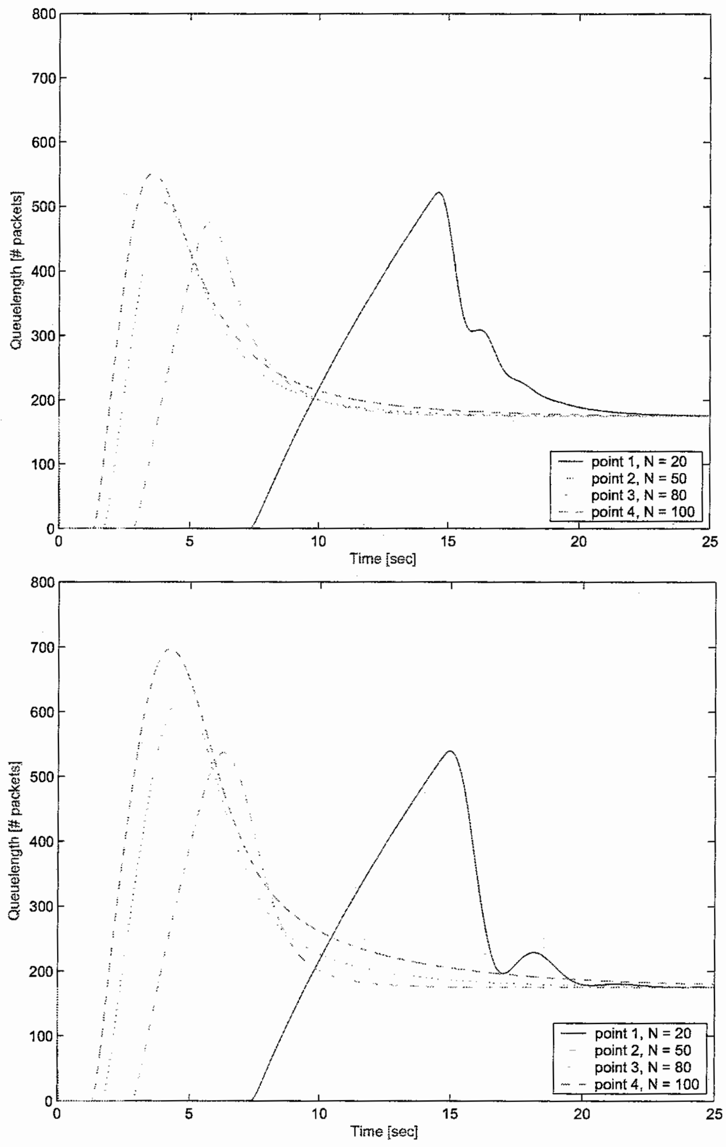


Figure 3.10: top: PID controller; bottom: PI controller

resonance). It will make the system much less sensitive for that frequency. For this, ω_1 has to equal ω_2, when these are not equal some phase advantage can be given as shown in Figure 3.11. In this Figure $\beta_1 = 0.7$ and $\beta_2 = 1$. When these are chosen a stretched Notch filter has the smooth shape as seen in Figure 3.11. Otherwise around the first pole first a dip would occur in magnitude and at the second pole an overshoot would occur.

The frequencies are limited because of the strength of these two poles in the numerator of the loop transfer function. The delay severely limits the usefulness of this part of the controller. Having the first pole very low, or the second high, would result in a high bandwidth (order 10^1), which will lead to instability because of delay. The goal is to try to push the bandwidth to go above 1.0 [rad/s] since this is the currently achieved bandwidth of the PID controller (Section 3.3.1). However chances are robustness will suffer because of the limited usefulness of the stretched notch filter. Designing the rest of the controller the bandwidth was able to be placed at 1.6 [rad/s] when choosing the poles ω_1 at 0.5 [rad/s] and ω_2 at 2.0 [rad/s]. The low pass pole had to be placed at 1.0 [rad/s] to ensure this bandwidth.

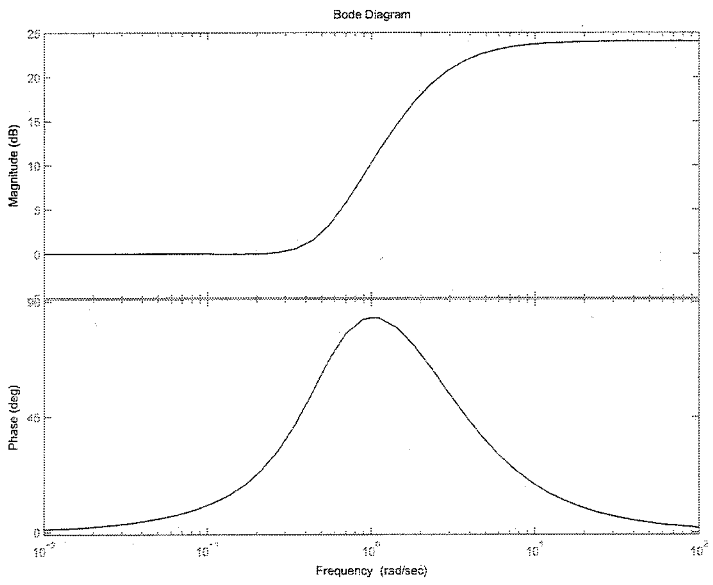


Figure 3.11: Notch filter bode diagram

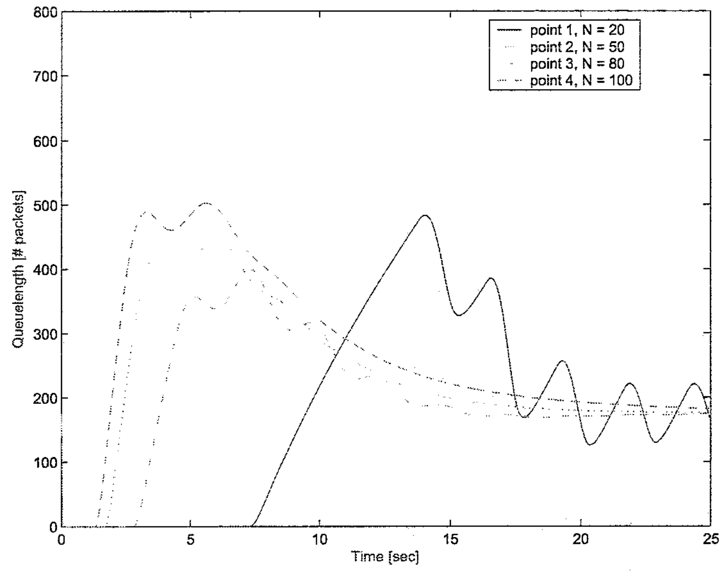
When applying this controller to control queue length the results are depicted in Figure 3.12 and robustness suffers because at $N = 20$ this system oscillates severely. The conclusion drawn from this controller is that other ways can be devised to push trade-off. Still the time delay will limit linear applications and therefore a non-linear approach will be sought. This controller will not be used in comparison in this discussion, since the PID controller is shown to be better.

3.4 Feed forward

A well-known technique in control is the use of feed forward. The basic idea is to invert the plant and insert this signal behind the normal controller, ideally cancelling plant dynamics. The linearized system can be written in the form mass-spring damper system as seen in Equation (2.5). The queue length does not have an set path but a setpoint. The queue length is best controlled as low as possible because of the time delay it brings. It is a constant value yielding $\dot{q}_{ref} = \ddot{q}_{ref} = 0$. Then of the three parameters (M, B and K) only K can then be used as feed forward. However, the spring factor K cannot be considered constant. The load factor N is not known and varies. If N could be estimated feed forward would be possible, however, because of N's nature this is unlikely. In [13] this is seen to be difficult as can be seen in the figures showing the difference in the real and estimated load.

3.5 Linear & Adaptive Techniques

Although not presented in this discussion, adaptive techniques improve the behavior of linear controllers. As mentioned in [13] the systems parameters variations cause a robustness choice for a controller based upon certain nominal system parameters. Meaning a worst-case scenario is chosen in for instance load and T_p a controller is derived based upon this scenario. Other scenarios result in lower bandwidth thus worse performance. [14] proposes a Self tuning PI controller (S-PI) where the pole and gain of the controller are based upon the current drop/mark probability. This showed an improved results to the fixed PI design. [13] proposes an External Exited Adaptive Loop (EEAL) algorithm which roughly estimated N. This gives better results, the overall results are the same but much faster. However, these techniques have been tested and tried in earlier discussions. A more interesting approach is to try a new technique and 'see what happens'. This is done next.



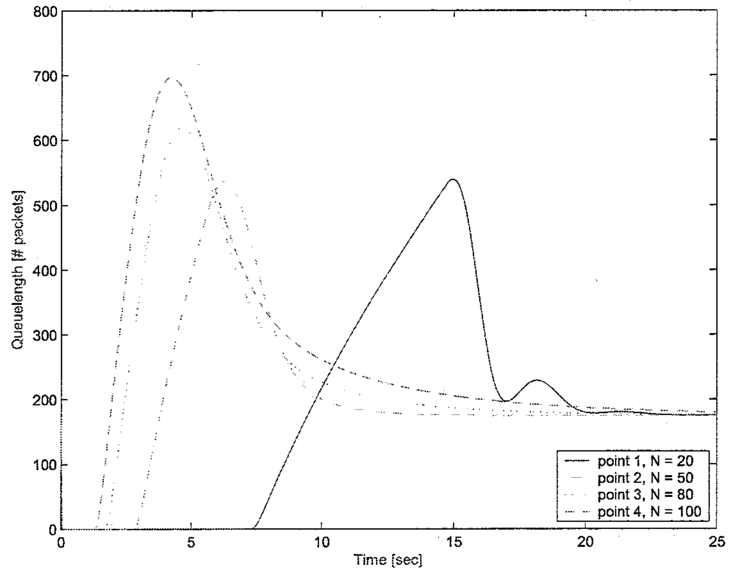


Figure 3.12: Top: PI with notch and low pass filter, Bottom: PI

Chapter 4

Non-linear Controller for Single Congested Link network

4.1 Non-linear Controller

The approach used in deriving a non-linear controller is based on full state feedback linearization. For background and a more generic view of controlling systems based upon this approach [15] and for a more detailed background [16] should be examined. First the theory of the controller will be explained. What is very important to realize is that the window size is not known at the router. This means that the window size, which is a state, is not available for feedback. When applying full state feedback an observer will have to be devised.

In this Chapter first the assumption is made that full states are available. If it can be shown this method provides profound improvement upon the current controlling method this controller will by extended by an observer and additional experiments.

4.1.1 Theory of Input-output Feedback Linearization

Input-output feedback linearization can be seen as a method to find a static state feedback control law ψ, such that the closed loop system has a linear input output behavior. This means the relationship between y and u is non-linear and between y and v is linear. In [17] this is explained in greater detail and Figure 4.1 gives a representation of this.

Consider the non linear system

$$\begin{aligned} \dot{x} &= f(x) + g(x) \cdot u \\ y &= h(x) \end{aligned} \quad , \tag{4.1}$$

where x denotes the state of the system, u denotes the input *without* time delay and $f(x)$ and/or $g(x)$ are non-linear functions in x and the output is $h(x)$. Important is that the time delay is disregarded for controller implementation.

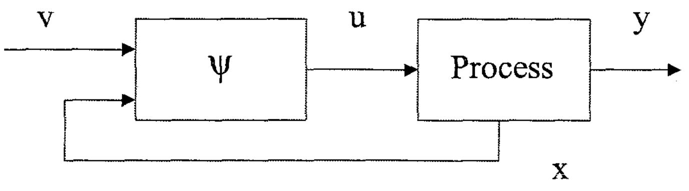


Figure 4.1: Non-linear control law schematic, x is system state, u is non-linear input, v is linear feedback in terms of output

The relative degree of a system is 'that' time derivative of the system output that contains the input explicitly. Thus k is relative degree when y^k depends explicitly on u. $\dot{y}$ is as follows

$$\dot{y} = \frac{dy}{dt} = \frac{\delta h(x)}{\delta x} \cdot \dot{x} = \frac{\delta h}{\delta x} \cdot f(x) + \frac{\delta h}{\delta x} \cdot g(x) \cdot u,$$

and if $\dot{y}$ depends on u explicitly, relative degree k equals 1. If it does not relative degree is higher and the next step in finding relative degree is computing $\ddot{y}$. This is seen in the following Equation

$$\ddot{y} = \frac{\delta}{\delta x}\left(\frac{\delta h}{\delta x} f(x)\right) \cdot \dot{x}.$$

If $\ddot{y}$ is computed and it depends on u relative degree is 2, else the next time derivative of the output has to be computed. When going up to relative degree 4, which is common for many mechanical systems, this search for k will become obscure. The Lie derivative simplifies notation

$$L_f h(x) = \frac{\delta h}{\delta x} f(x).$$

When applied to the second order derivative it yields the following equation for the general case

$$\ddot{y} = L_f^2 h(x) + L_g L_f h(x) u.$$

The general case is that all the time derivatives of the output are independent of the input *except* the k^{th} derivative. This means the $y, \dot{y}, \ldots, y^{k-1}$ are not a function of the input. This yields the following

$$L_g h(x) = L_g L_f h(x) = ... = L_g L_{k-2} = 0; L_g L_{k-1} \neq 0.$$

For the simplest case where the relative degree k equal to the number of states (n) a stabilizing feedback control law is easy to derive. When relative degree is smaller then the number of states this method still holds. For this case the reader is referred to [16]. Introduce the following coordinate change to the so called Byrnes-Isidori Normal form

$$\begin{aligned}
&y = h(x) = z_1\\
&\dot{y} = L_f h(x) = z_2\\
&\ddot{y} = L_f^2 h(x) = z_3\\
&\vdots\\
&y^{k-1} = L_f^{k-1} h(x) = z_k \quad .
\end{aligned}$$

This is a new form or state and it has the following useful property

$$\begin{aligned}
&\dot{z}_1 = z_2\\
&\dot{z}_2 = z_3\\
&\vdots\\
&\dot{z}_{k-1} = z_k \quad ,
\end{aligned}$$

with the final time derivative to be desired equal to a stabilizing feedback v, this will in turn stabilize the system

$$\dot{z}_k = y^k = L_f^k h(x) + L_g L_f^{k-1} h(x) u \equiv v. \tag{4.2}$$

In order to have Equation (4.2) to hold, meaning $\dot{z}_k = v$, a controlling feedback u is proposed as seen in the following Equation

$$u = (L_g L_f^{k-1} h(x))^{-1} \cdot (-L_f^k h(x) + v).$$

The stabilizing feedback v is then to be derived to stabilize $\dot{z}_k$ for it will stabilize all normal form coordinates. This will in turn stabilize the original system locally or globally as will be proven in the following section. What remains is to prove that y converges to a desired value $y_{desired}$. To see this, define

$$v = -a_{k-1} y^{(k-1)} - \ldots - a_1 \dot{y} - a_0 (y - y_d), \tag{4.3}$$

and with $e = y - y_{desired}$ the following holds

$$e^k + a_{k-1} e^{(k-1)} + \ldots + a_1 \dot{e} - a_0 e = 0.$$

By choosing $a_0, \ldots, a_{k-1}$ so that

$$\lambda^k + a_{k-1} \lambda^{(k-1)} + \ldots + a_0 = 0, \tag{4.4}$$

has all roots in the open left-half plan it yields $e(t) \to 0$ for $t \to \infty$.

4.1.2 Stability

Proof

It is proven in Section 4.1.1 that error dynamics goes to zero for time going to infinity. This means that $x_2 \equiv q$ approaches a constant thus $\dot{x_2}$ goes to zero. In turn this means that $\dot{x_1}$ goes to zero, where $x_1 \equiv W$. As can be seen when evaluating the second part of the system (in Equation (4.7) state is represented with $\underline{x}$).

Actual stability analysis is proving the system is stable in the sense of Lyapunov. For the exact formulation of stability in the sense of Lyapunov the reader is referred to [16]. Since $n = k = 2$ the system has a linear feedback v (in terms of y and its time derivative), proving stability should prove a simple exercise. The normal form is as follows

$$\begin{aligned}
z_1 &= h(x) = y = x_2 \\
z_2 &= L_f(x) = \dot{y} = \dot{x_2} \\
\dot{z_1} &= z_2 \\
\dot{z_2} &= v = -a_1 z_2 - a_0(z_1 - z_{1desired}) \quad .
\end{aligned}$$

Since this system does not have the state $\underline{\dot{z}}$ equal zero for $(z_1, z_2) = (0,0)$ a coordinate change is proposed for further Lyapunov analysis. This is transform is stated as follows

$$\begin{aligned}
r_1 &\equiv z_1 - z_{1desired} \\
r_2 &\equiv z_2 \\
\dot{r_1} &= r_2 \\
\dot{r_2} &= -a_1 r_2 - a_0 r_1 \quad .
\end{aligned}$$

Because this is a linear system the Lyapunov equation can be used to derive a suitable candidate function. Here a simple quadratic Lyapunov candidate function V is chosen with a_0,$a_1 > 0$

$$\begin{aligned}
V &= \frac{1}{2}a_0 r_1^2 + \frac{1}{2}r_2^2 \\
\dot{V} &= -a_1 r_2^2 \quad .
\end{aligned}$$

This means $\dot{V} \leq 0$ for all r_1 and for $r_2 = 0$. To further prove $\underline{\dot{r}}$ equal zero for $(r_1, r_2) = (0,0)$ LaSalle's Invariance principle is needed. This is also described in [16] for a more complete description.

Consider a set $\mathbb{Q} = \{x|V(x) < v\}$ and assume $\dot{V} \leq 0$ in $\mathbb{Q}$. Let set $\mathbb{S} = \{x|\dot{V}(x) = 0\}$. Let $\mathbb{M}$ be the largest invariant set in $\mathbb{S}$. All solutions starting in $\mathbb{Q}$ tend to $\mathbb{M}$ for t $\rightarrow \infty$. Here choose $\mathbb{Q}$ equal to $\mathbb{R}$, meaning the entire two dimensional space spanned by r_1 and r_2. Meaning $\mathbb{R} = \{r|V(x) < \infty\}$. Here $\mathbb{S} = \{(r_1, r_2)|r_2 = 0\}$ and the largest invariant set is (0,0) since $\dot{V} = -a_1 r_2^2$. This means $\mathbb{M} = \{(r_1, r_2)|(0,0)\}$ and thus this proves asymptotic stability for

$(r_1, r_2) = (0, 0)$. Which enables it to state the original system to be asymptotically stable. Because $k = n$ there is no zerodynamics. This is supported by the next section where the systems normal form is proven to have diffeomorphismic properties to the original system.

Diffeomorphism [16]

Definition 4.1 *A map f: $X \mapsto Y$ is said to be a diffeomorphism if f is a homeomorphism (i.e., a one-to-one continuous map with continuous inverse) and if both f and f^{-1} are smooth.*

The sets X and Y are said to be diffeomorphic if there exists a diffeomorphism between them. As also can be seen in [16] diffeomorphismic properties between the two coordinate systems ($\underline{x}$ and $\underline{z}$) can be shown. This will shows that when stability is proven for one coordinate system it therefore also goes for the other. When the following Equation

$$\frac{\partial \underline{z}}{\partial x} = \begin{bmatrix} 0 & 1 \\ \frac{N}{(\frac{x_2}{C}+T_p)} & \frac{-Nx_1}{C(x_2+T_pC)^2} \end{bmatrix}, \tag{4.5}$$

has full rank it is said the sets $\underline{x}$ and $\underline{z}$ are diffeomorphic. Here $\underline{z}$ is the proposed normal form and $\underline{x}$ is the original coordinate system. It can easily be seen the above relation is nonsingular. It even proves a global diffeomorphism since full rank is guaranteed for all $\underline{x}$.

4.1.3 Stability for Fixed Controller

A stabilizing non-linear control law for this system can be derived based on full state knowledge (x_1 and x_2) and constant load (N) and propagation delay (T_p). Controllers without estimating techniques or updating schemes are however based on constant parameters. It will now have to be shown that this non-linear controller works, or under what limitations this controller will work since the system parameters are not constant. First the derived controller will be presented. For simplicity Equations (2.1) and (2.2) will be presented as equations (4.6) and (4.7)

$$\dot{x_1} = \frac{1}{\frac{x_2}{C}+T_p} - \frac{x_1^2}{2(\frac{x_2}{C}+T_p)} u(t - \left(\frac{x_2}{C}+T_p\right)) \tag{4.6}$$

$$\dot{x_2} = \begin{cases} -C + \frac{Nx_1}{\frac{x_2}{C}+T_p} & \text{if } x_2 > 0 \\ max\left(0, -C + \frac{Nx_1}{\frac{x_2}{C}+T_p}\right) & \text{if } x_2 = 0 \end{cases}. \tag{4.7}$$

The state variable $\underline{x} = [x_1, x_2]$ represents window size and queue length respectively ($[W, q]$). If the theory presented in Subsection 4.1.3 is used it can be easily shown this system has a relative degree of 2 which is equal to the number of states of this system. This is seen in here

$$\begin{aligned}\dot{y} &= L_f h(x) + L_g h(x)u = -C + \frac{Nx_1}{\frac{x_2}{C} + T_p} \\ \ddot{y} &= L_f^2 h(x) + L_g L_f h(x)u \\ &= \frac{N}{(\frac{x_2}{C} + T_p)^2} + \left(\frac{Nx_1}{(x_2 + T_pC)^2} - \frac{N^2x_1^2}{(x_2 + T_pC)^3}\right) - \frac{Nx_1^2}{2(\frac{x_2}{C} + T_p)^2}u \\ &= \alpha + \beta + \gamma u \end{aligned} \quad (4.8)$$

Important here is that in Equation 4.8 the time delay in the input is disregarded for simplicity. If relative degree was one (in the general case not equal to n) then additional 'tools of the trade' will have to be used to define a working controller as is explained in [15]. This means the non-linear controller can be derived as explained. The stabilizing feedback for this controller is shown here

$$v = -a_1\dot{y} - a_0(y - y_d) = -a_1\dot{x_2} - a_0(x_2 - x_{2reference}), \quad (4.9)$$

with the poles to be chosen. Faster poles can result in better behavior but there is a limit. They are used as gains and pushing them to high may end up creating a much too sensitive system which oscillates around the reference. Choosing these poles will be done while simulating for a trade-off will be apparent.

The second derivative of the output y, contains the actual controlling feedback u. The α, β and γ are introduced for simplicity. A stabilizing controller is given below

$$u = \frac{1}{\gamma}(-\alpha - \beta + v), \quad (4.10)$$

when v is defined as seen in Equation (4.9). We need now to show that this will provide $\ddot{y} = v$ under the constant parameters. To simplify, define the following terms

$$\begin{aligned}\alpha_o &= \frac{N_o}{(\frac{x_2}{C} + T_{po})^2} \\ \beta_o &= \frac{N_ox_1}{(x_2 + T_{po}C)^2} - \frac{N_o^2x_1^2}{(x_2 + T_{po}C)^3} \\ \gamma_o &= \frac{N_ox_1^2}{2(\frac{x_2}{C} + T_{po})^2},\end{aligned} \quad (4.11)$$

which when plugged into Equation (4.10) yields the control law

$$u = \frac{1}{\gamma_o}(-\alpha_o - \beta_o + v). \quad (4.12)$$

Evaluating $\ddot{y}$ from Equations (4.8) and (4.12) gives

$$\ddot{y} = \underbrace{\alpha - \frac{\gamma}{\gamma_o}\alpha_o}_{part1} + \underbrace{\beta - \frac{\gamma}{\gamma_o}\beta_o}_{part2} + \underbrace{\frac{\gamma}{\gamma_o}}_{part3} v. \tag{4.13}$$

To gain further understanding the three parts will be examined separately. Expanding part 1 reveals that it can be disregarded as seen in the following

$$\alpha - \frac{\gamma}{\gamma_o}\alpha_o =$$
$$\frac{N}{(\frac{x2}{C} + T_p)^2} - \left(\frac{N}{N_o} \cdot \frac{(\frac{x2}{C} + T_{po})^2}{(\frac{x2}{C} + T_p)^2}\right)\left(\frac{N_o}{(\frac{x2}{C} + T_{po})^2}\right) = 0.$$

Similarly, the second part can be expanded to

$$\begin{aligned}\beta - \frac{\gamma}{\gamma_o}\beta_o &= \left(\frac{Nx_1}{(x_2 + T_pC)^2} - \frac{N^2x_1^2}{C(x_2 + T_pC)^3}\right) - \\ &\left(\frac{\frac{N}{C^2}(x_2 + T_{po}C)^2}{\frac{N_o}{C^2}(x_2 + T_pC)^2}\right) \cdot \left(\frac{N_ox_1}{(x_2 + T_{po}C)^2} - \frac{N_o^2x_1^2}{C(x_2 + T_{po}C)^3}\right) = \\ &(0) - \left(\frac{Nx_1^2}{C(x_2 + T_pC)^2}\right) \cdot \left(\frac{-N}{C(\frac{x2}{C} + T_p)} + \frac{N_o}{C(\frac{x2}{C} + T_{po})}\right).\end{aligned} \tag{4.14}$$

The term: $f = (\frac{-N}{RC} + \frac{N_o}{R_oC})$ can be shown to be negligible. For this the simulation bounds for N and T_p will have to be used. In the simulations with the linear controller $N \in [20, 100], R \in [0.2, 0.4]$ were used. Here the range of R is used and is the result of the constant T_p (=0.2 [s]) and the domain on the queue $q \in [0, 800]$. Varying T_p could be chosen as well, the result will show the low influence this entire second part has, meaning taking T_p or R does not matter. For reasons to be explained N_o will be fixed at $N_{max} = 100$ and R_o at $R_{min} = 0.2$. The above mentioned Function f is numerically evaluated as can be seen in Figure 4.2, the edge extreme points are shown beneath.

- $f(N, R) : f(20, 0.2) = 0.11$ [-]
- $f(N, R) : f(20, 0.4) = 0.12$ [-]
- $f(N, R) : f(100, 0.2) = 0!$ [-]
- $f(N, R) : f(100, 0.4) = 0.07$ [-]

When evaluating the multiplicative term for 'normal' simulation parameters it shows the term has order 10^{-7}. These simulation parameters will be explained later, taking ($\{x1, x2, C, N, T_p\}$ = {50, 100, 3750, 60, 0.2}). When trying to achieve zero 10^{-8} can be considered negligible, also because v contains $\dot{x}_2$ and $(x_2 - x_{2desired})$ multiplied by the poles. So when looking at Equation (4.14)

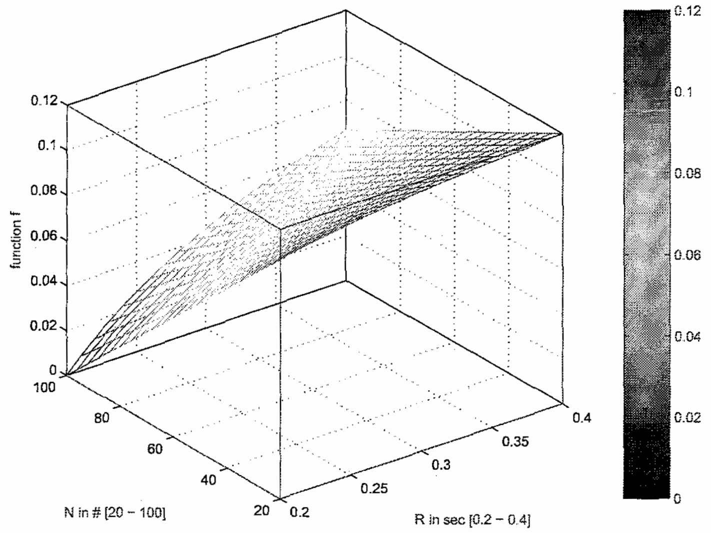


Figure 4.2: Function f evaluated over $N \in [20, 100]$ and $R \in [0.2, 0.4]$

this shows the term that remains when evaluating $\beta - \frac{\gamma}{\gamma_o}\beta$ is small. The multiplicative term which remains is very small because of the size of C, and is even further diminished by this term between the brackets.

Then the remaining equation for the second derivative of the output will be the following Equation

$$\ddot{y} = (\alpha - \frac{\gamma}{\gamma_o}\alpha) + (\beta - \frac{\gamma}{\gamma_o}\beta) + \frac{\gamma}{\gamma_o}v \approx (0) + (0) + \frac{\gamma}{\gamma_o}v.$$

Now the third part will be evaluated. What has to hold is that $\frac{\gamma}{\gamma_o} > 0$, because else the poles of the system will flip into the right half-plane which will yield instability. The following holds

$$\frac{\gamma}{\gamma_o} = \frac{N}{N_o}\frac{(\frac{x_2}{C} + T_{po})^2}{(\frac{x_2}{C} + T_p)^2}. \tag{4.15}$$

Because the parameters are all positive in light of what the parameters represent pole flipping will not occur. As long as $\frac{\gamma}{\gamma_o}v > 10^{-8}$ this system remains stable. The problem that can be seen when looking at Equation (4.15), is that this factor will increase the poles used making the controller much too sensitive or too desensitize for variations. Therefore, limitations will be placed by choosing $N_o \approx N_{max}$ and $T_{po} \approx T_{pmin}$. The resulting scheme will be as seen in Figure 4.3, with non-linear AQM controller as seen in Equations (4.11) and (4.12).

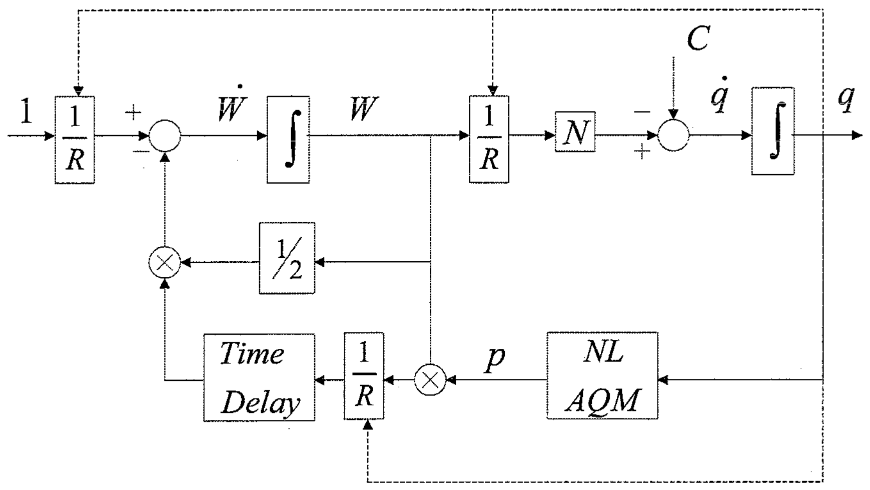


Figure 4.3: Block diagram of a TCP connection

4.2 Non-linear Controller Parameters

The controller derived still has unchosen poles. The chosen constants for simulation shown in Table 4.1, deviations from these will be mentioned per experiment. The propagation delay here is considered to be a constant value as also done in [10]. In experiment 4 in Section 4.3, the first comparable experiment, this will also be made variable to see if the controller can cope with this compared to the PI. In the first three experiments the T_p is considered to be constant and known. Although propagation delay is not constant in actuality, it does not differ as much as the load constant. Its influence is seen in Figure 2.5, there a variation of 100 [ms] is used which is very high as can be seen in varies sources like [8].

Table 4.1: Choice for other constants used for simulation

N_o	T_{po}	C	T_p
100	0.2	3750	0.2

Table 4.2: Controllers used for simulation

Non-linear	Poles	$a1$ and $a0$
controller 1	5,5	10,25
controller 2	4,4	8,16
controller 3	2,2	4,4
controller 4	7,7	14,49

Experiment 1: Controller 1 from Table 4.2

The first controller is chosen to have relative high poles. As can be seen in Figure 4.4 the controller stabilizes quite well for the simulations with low N. For high N the term $\frac{\gamma}{\gamma_{con}} \approx 1$ and as seen the controller is very sensitive as seen for N=100. Disregarding the oscillations the overshoot seen in PI and PID is gone and the time at which steady is reached is lower.

Removing the oscillations could be done by increasing N_o well above N_{max} because the 'resulting poles' would be lower for N = 100. As can be seen in figure 4.5 this indeed is the case. The trade-off here is that it lowers the poles for the other cases as well. This could be seen by a more sluggish responses. Again noted the overshoot not worth mentioning and still is faster compared to PI and PID as seen in Figure 3.10. Another way is to try other poles in the controller (see following experiment).

Experiment 2: Controller 2 from Table 4.2

The results are shown in Figure 4.6. With lower poles and N_o = 100 the results are almost identical to those of Figure 4.5. This is logical because in both situations the poles were lowered. It implies that if the maximum of N cannot be estimated very good needed to choose N_o, the poles can be altered to still yield good performance!

Experiment 3: Controller 3 from Table 4.2

The third controller shows the expected behavior for relatively low poles and is shown in Figure 4.7. The approach towards the setpoint is smooth, but more time is needed because the system is sluggish. This is considered worse then the PI controller results. A little static however is not a problem because of the nature of NS-2 simulator, which will be explained in Chapter 6.

In conclusion, it can be said the second controller performs best and is therefore chosen. But more important the trends when changing the parameters N_0 and the poles (resulting in a_0 and a_1) are made clear.

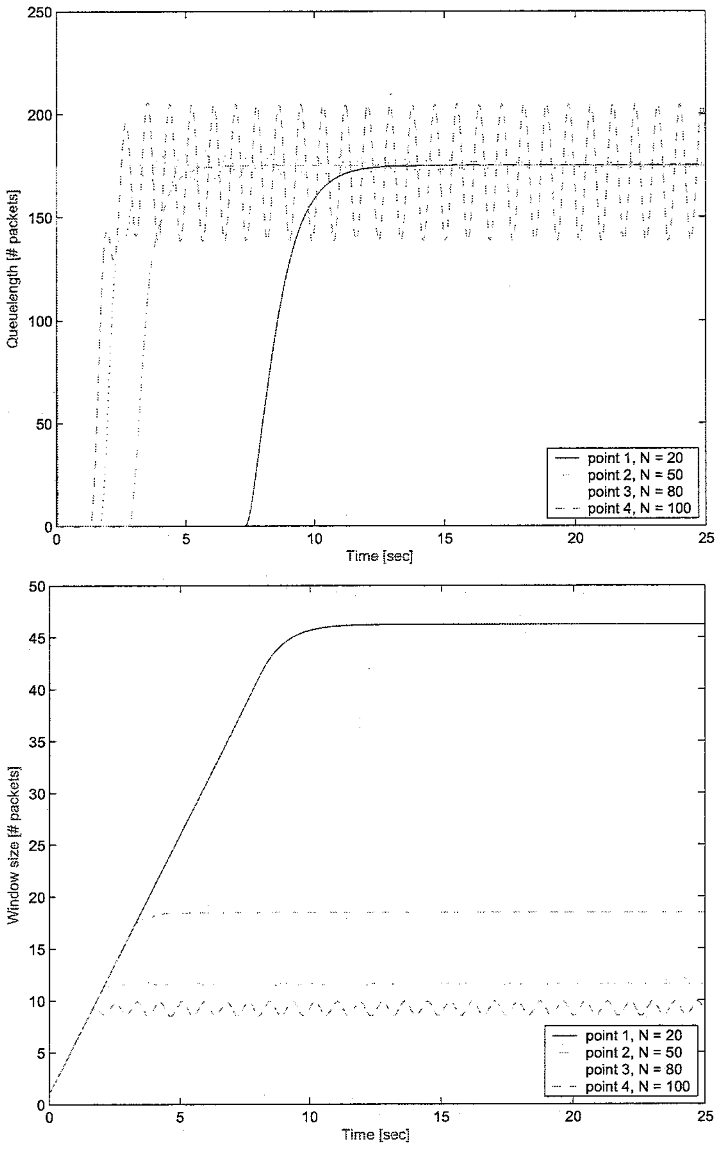
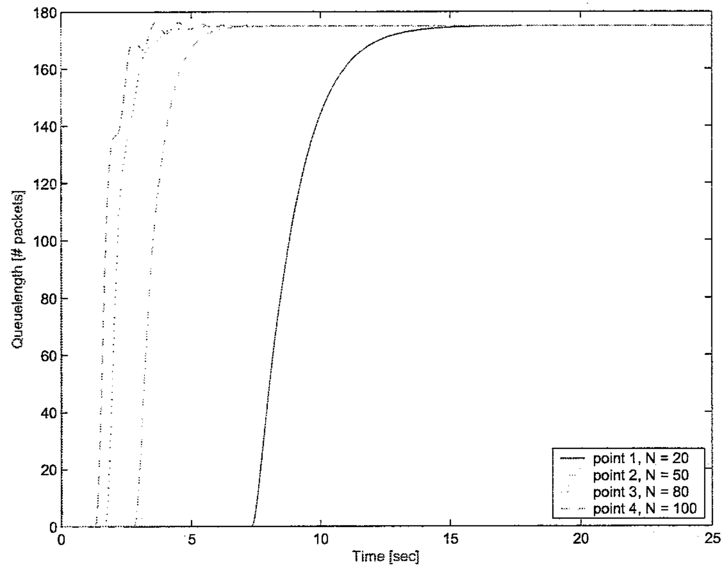


Figure 4.4: top: queue length non-linear controller 1; bottom: window size non linear controller 1

4.3 PI-, PID- & Non-linear Controller Comparison

First the full state feedback controller is tested. For readability and avoiding a surplus of figures these are placed in Appendix A. The figures of the experiment types are kept in the main text because of their significance.



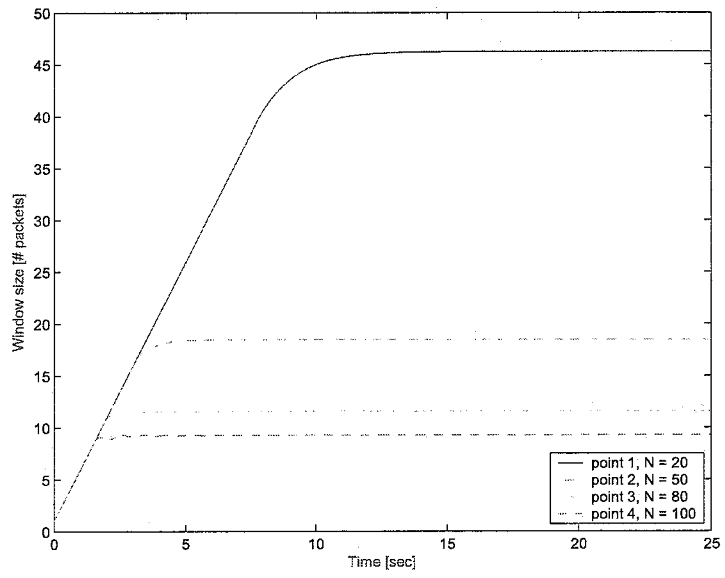


Figure 4.5: $N_0 = 150$ instead of 100. top: queue length non-linear controller 1; bottom: window size non linear controller 1

4.3.1 Experiments

In experiment 4 the propagation delay is varied. Propagation delay is defined as the time delay introduced by the duration to propagate a packet from end system to the destination (router, other end system, etc.). The propagation speed

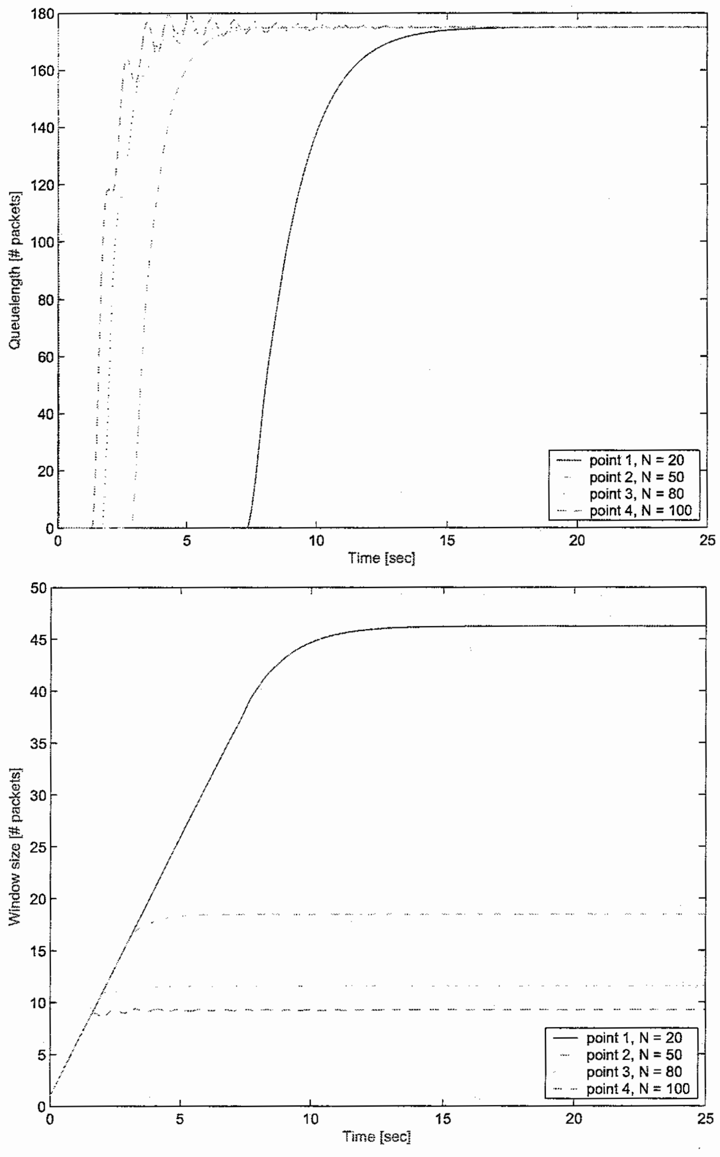
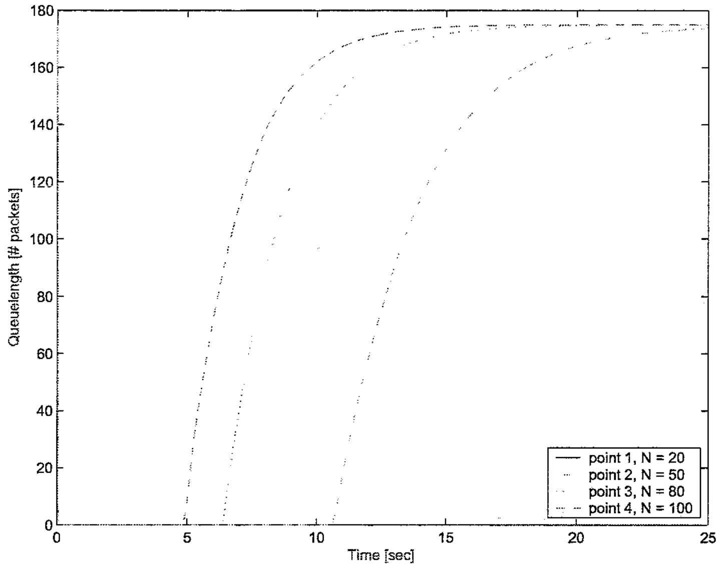


Figure 4.6: top: queue length non-linear controller 2; bottom: window size non linear controller 2

depends on the sort of medium used and will be about constant. but not entirely. In the first three experiments as mentioned this was kept constant. The range of T_p will be chosen [0.15,0.25] and will be uniformly randomly chosen in this interval. This range is pretty extreme as can be seen when reviewing [2]. The variation in T_p was put below the bandwidth of the controllers to keep it



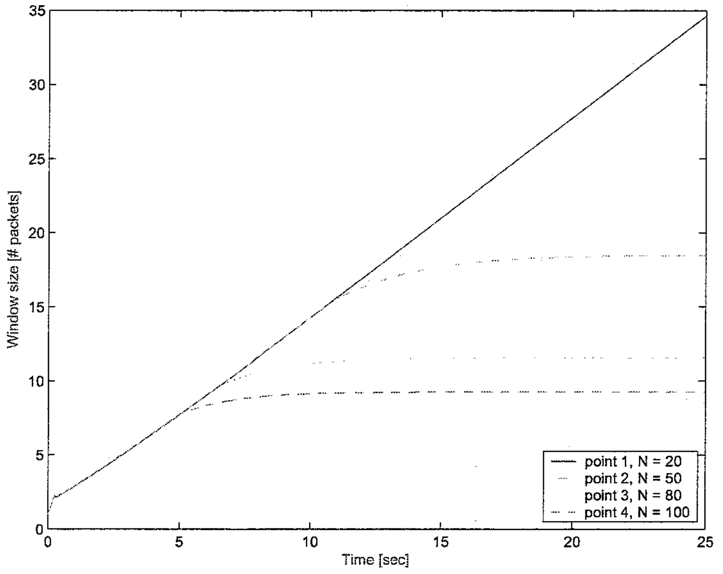


Figure 4.7: left: queue length non-linear controller 3; right: window size non-linear controller 3

'fair'. A frequency of 1 [Hz] was used and as can be seen in Figure A.1 the PI is unable to overcome this obstacle. PID improves upon the PI as said earlier (Figure A.2). The non-linear controller clearly the best results as when comparing Figures A.1 and A.3.

Experiments 5 and 6 validate the controller under changing load in time as is the case in real life. Load varies tremendously in real life since internet users across the world in different time zones log on and of at random. Routers across the world simply have to adjust. First an input without noise will be tested (experiment 5). Then some noise will be added to this (experiment 6, 1 [Hz] and standard deviation of 2.5 [-]) . Both inputs for load are shown in Figure 4.8. Without adding noise the non linear controller again performs best

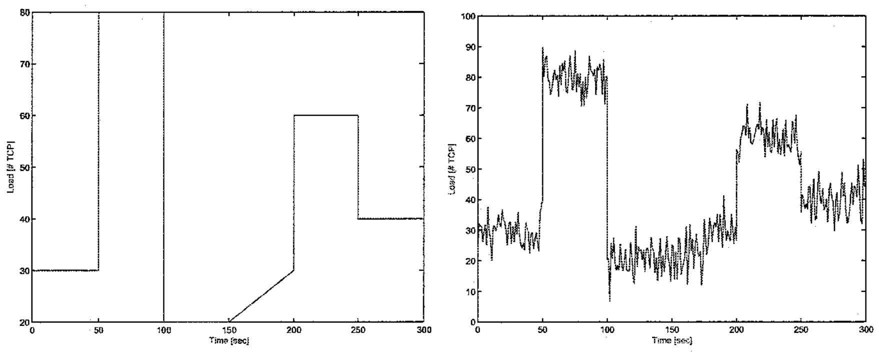


Figure 4.8: left: reference load without noise (left) and with noise (right), 1 [Hz] and standard deviation 2.5 [-]

as seen when comparing Figures A.4-A.6. The settling times are long therefore all controllers are able to deal with this experiment. Drops to zero queue length could be avoided by increasing the desired queue length however, this will increase time delay and reducing effective send rate of the sender. The PID controller compared to the PI does perform better, but as stated earlier nothing ground breaking. When adding noise (experiment 6) the non-linear controller also has some serious trouble. Controller 4 form Table 4.2 was tested for it has the emphasis not on $\dot{x}_2$ and queue length is changing very rapidly. The non-linear controller with controller 2, perform best out of the four compared in Figures A.7 till A.10. The non-linear controller does improve, but it is not 'by far' better then the PI controller. Because the PID controller is not ground breaking better then PI, from now it is not mentioned and depicted any more in these experiments.

Experiment 7 will combine experiments 4 and 6 by varying T_p and N at the same time. The results are seen in Figure A.11. T_p again is varied uniformly in range [0.15,0.25]. For the controller the T_{po} was not lowered to 0.15 as stated in Section 4.1.3 to avoid changing two things at once. Again a slight improvement is seen with the non-linear scheme. It has to be said that the chosen conditions are though to control.

Experiments 8 and 9 are both like experiments 5 and 6. The profile for N however is changed. Less time is given to settle per change in load. To avoid the number of times the boundary $q = 0$ is reached, which introduces non-linear effects, the minimum load is raised to 30 [-]. The references are depicted in

Figure 4.9. Figure A.12 shows the non-linear controller to be much better.

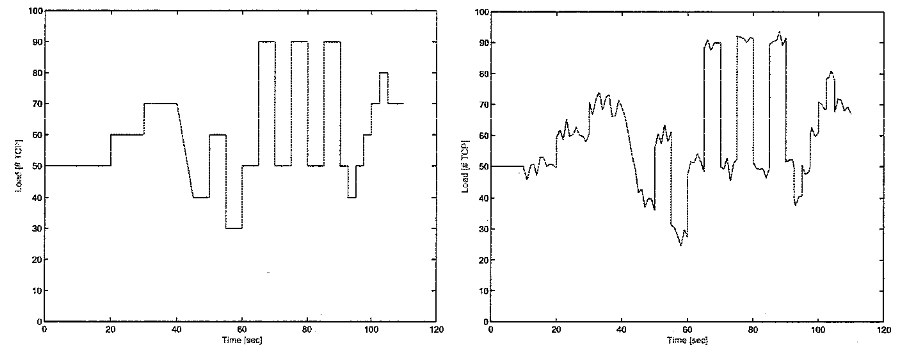


Figure 4.9: exp. 9 and 10: left: reference load without noise (left) and with noise (right)

Figure A.13 shows experiment 9, which is experiment 9 with noise added. Gaussian noise on load has frequency 1 [Hz] and standard deviation 2 [-]. The non-linear controller performs better.

In experiment 10 the frequency of the load variation is raised to 5 [Hz]. This is above the crossover frequency of the PI controller. The non-linear controller, as seen in Figure A.14, can cope better with this.

Since N is never constant but always varying experiment 11 was run with N a constant with noise added. The noise added was gaussian with standard deviation of 2.0 [-] with a frequency of 5 [Hz]. The results speak for themselves as seen in Figure A.15. For higher values of N, because of the factor N_o divided by N, the non-linear controller gives some oscillations.

4.4 Robustness for varying N and T_p

Concluding these experiments it suffices to say the non-linear controller outperforms the PI (and PID) controller. What remains now is to give a domain in which both controllers will be usable. It will be shown this type of control severely improves system behavior and the controller will be fitted with an observer for full state feedback. This will be done in Chapter 5.

4.4.1 With Respect to Load

The PI controller will have stability issues for low values of load because of the high low frequent gain ($\frac{(RC)^3}{(2N)^2}$). For high load the systems high frequent gain is lower which means lower bandwidth resulting in a sluggish system (considering rise and settling time). The non-linear system dynamics seems slower for lower N, meaning system dynamics starts later for lower N. This is the same for PI because load times the rate ($N \cdot \frac{W(t)}{R(t)}$) has to become large enough to exceed the links capacity and start filling the router buffer. For high N the systems starts to oscillate because of the artificial raising of the poles. Because of the nature of NS-2, which will be explained in Chapter 6, oscillations will be allowed in a limited range. Figure 4.10 shows for low N the PI controller has

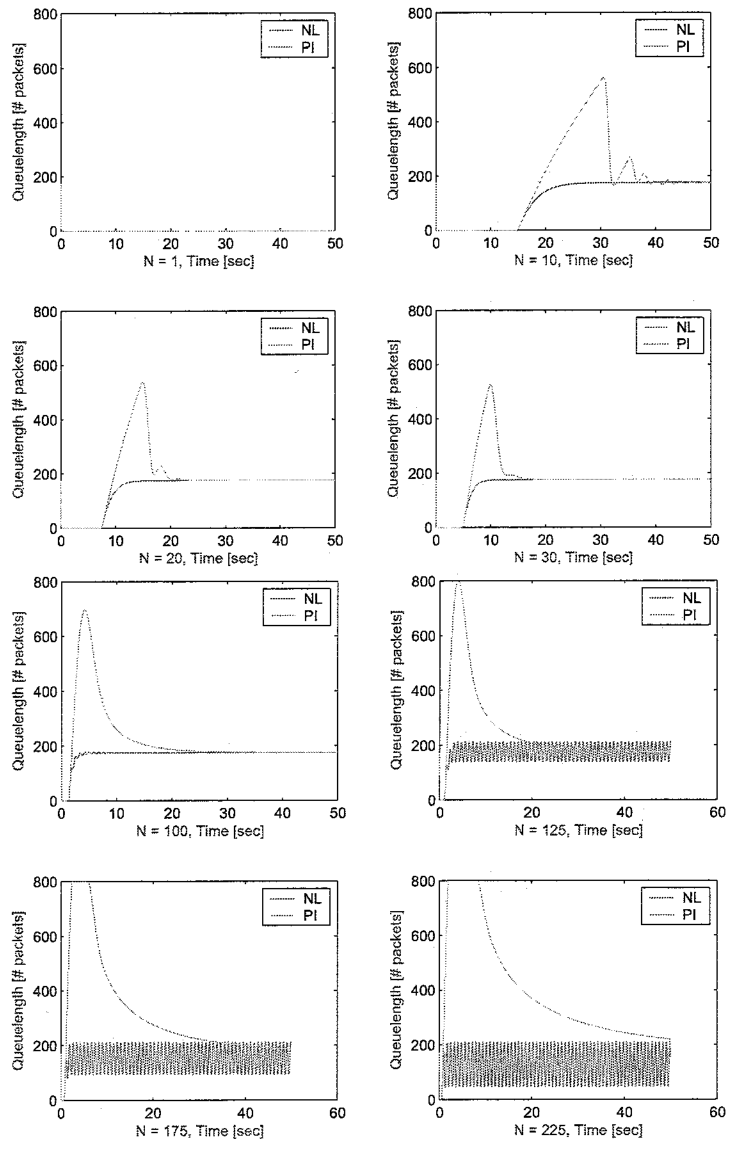


Figure 4.10: Lower load bounds (top); Upper load bounds (bottom)

abrupt changes. This is caused by the probability which hits the zero bound, a nonlinearity. The lowest N for this PI is taken to be about 20. For the non-linear controller this bound is 1 (keeping it a real number) as can be seen in Appendix B Figure B.1. The high bound for the PI is taken when the overshoot reaches the buffer limit, because tail drop is to be avoided at all time. As can be seen for $N = 125$ this occurs. Moreover the system is quite sluggish for this value. For the non-linear controller the upper value is set to 175. Here the oscillation around steady state is about 100 [packets] (q_{max} minus q_{min}).

4.4.2 With Respect to Propagation Delay

When looking at the simulation results one has to take under consideration that a very low propagation delay logically fills the router much quicker so system dynamics starts earlier. For low T_p the low frequent gain is relatively low ($\frac{(RC)^3}{(2N)^2}$), for high values of T_p stability issues will arise. As can be seen in Figure 4.11 for low T_p the system becomes sluggish. The system poles both increase, but the crossover frequency goes down (for $T_p = 0.1$ [s] the resulting crossover frequency is 0.11 [rad/s]).

Figure 4.11 shows for very low T_p the system with non-linear controller reaches a lower steady state then is stated with certain oscillation. This oscillation and lower value of steady state is due to the fact the non-linear controller cannot work efficiently. The bound on probability is reached, adding nonlinearity to the system resulting in oscillations. When this upper boundary on p is not reached (increasing T_p) this is not seen. Because however limited oscillations are not a real issue because of the application to NS-2, this is still regarded as acceptable. NS-2 simulations could prove otherwise.

The upper boundary for the PI controller with regard to T_p is set to 0.55 [s]. There the queue buffer limit is reached. The oscillation is high because the probability keeps hitting the zero boundary for a long time. For $T_p = 0.45$ [s] this bound is hit once and the system stabilizes after that (Figure 4.11). The non-linear system has no real boundary. In Appendix B Figure B.1 a simulation is shown with $T_p = 5$ [s] which still stabilizes! Because of the high propagation delay however the system dynamics starts late in time.

Summarized Table 4.3 gives an overview of the feasible domains of the non-linear and PI controller.

Table 4.3: Controllers summary

	PI controller	Non linear controller
N_{min}	20	1
N_{max}	125	175
T_{pmin}	0.01	0.01
T_{pmax}	0.55	> 5

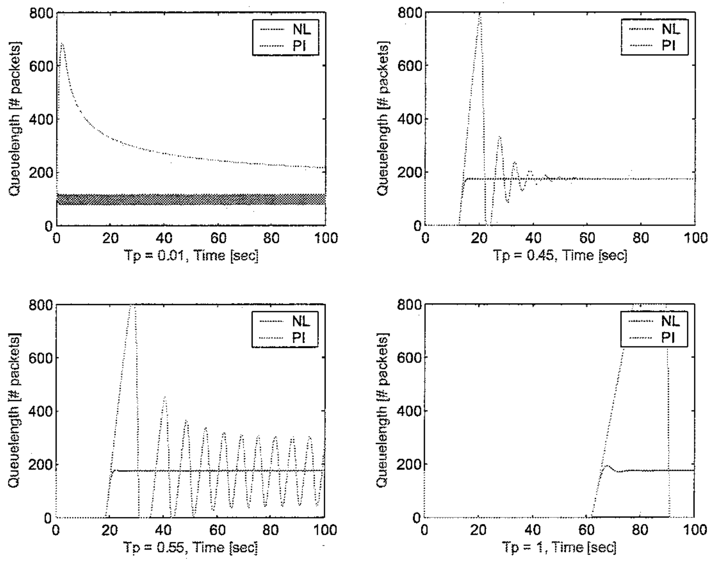


Figure 4.11: T_p bounds

Chapter 5

Adding Observer

Since full state back is required for this controller an observer will be derived to estimate W, the window size. The difficulty with the internet congestion problem is that the router needs to be simple. It can only be based upon queue length, its time derivative, and a reference value for queue length. It needs to be able to avoid congestion and provide an as stable as possible queue under varying circumstances. This is attempted in this Chapter.

5.1 Linear Observer

Consider the non-linear system described by the following Equation

$$\begin{aligned} \dot{x} &= f(x) + g(x) \cdot u \\ y &= h(x) \quad . \end{aligned} \tag{5.1}$$

Because of $g(x)$ in Equation 5.1 the system cannot be written in non-linear observer canonical form (NOCF). A linear observer is based on the linearized system, which can be written as

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \quad . \end{aligned}$$

Then an observer for the system is taken as

$$\dot{\widehat{x}} = A\widehat{x} + Bu + K(Cx - C\widehat{x}).$$

Observer error is defined as $e = x - \widehat{x}$ which will yield for observer error dynamics Equation

$$\dot{e} = (A - KC)e = A_{obs}e. \tag{5.2}$$

The error dynamics should be asymptotically stable to have the observer error $e(t) \rightarrow 0$ for $t \rightarrow \infty$. Then estimated state will converge to real state x. A sufficient condition for the existence of a $(n,1)$-vector K that results in a matrix that is asymptotically stable A_{obs} is that the system satisfies the observability rank test [16]. This means the following should hold where n is the number of states of the system

$$rank \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{pmatrix} = n. \tag{5.3}$$

In this case the linearized system can be described by the following equation, which is a state space description in x of Equations (2.3) and (2.4)

$$\begin{aligned} \dot{\tilde{x}} = & \begin{bmatrix} \frac{-2N}{R_0^2 C} & 0 \\ \frac{N}{R_0} & \frac{-1}{R_0} \end{bmatrix} \tilde{x} + \begin{bmatrix} \frac{-R_0 C^2}{2N^2} \\ 0 \end{bmatrix} \tilde{u} \\ y = & \begin{bmatrix} 0 & 1 \end{bmatrix} \tilde{x} \quad . \end{aligned} \tag{5.4}$$

Here

$$\begin{aligned} R_0 &= \frac{x_{20}}{C} + T_p \\ \tilde{x}_1 &= x_1 - x_{10} \\ \tilde{x}_2 &= x_2 - x_{20} \\ \tilde{u} &= u - u_0, \end{aligned}$$

and this yields the equilibrium point in $\tilde{x}$ to be $(\tilde{x}_1, \tilde{x}_2) = (0,0)$ which equals the original system equilibrium $(x_1, x_2) = (x_{10}, x_{20}) = (W_0, q_0)$ as described earlier in Section 4.1.2. In the linearization seen in Equation 5.4 the time delay in the input is disregarded for simplicity. The observability rank test is seen in the following equation

$$\text{rank}\begin{bmatrix} C \\ CA \end{bmatrix} = rank \begin{bmatrix} 0 & 1 \\ \frac{N}{R_0} & \frac{-1}{R_0} \end{bmatrix} = 2 = n. \tag{5.5}$$

Observer dynamics will then become

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}).$$

A representation of this scheme is given in Figure 5.1. Important in the observer equation is the comparison between $y - \hat{y}$. Here y denotes the non-linear systems output, which is equal to $x_2 = q$. However $\hat{y}$ is the output of the observer in original coordinate system. This means the difference between $\tilde{x}$ and x has to be taken along for this observer to work.

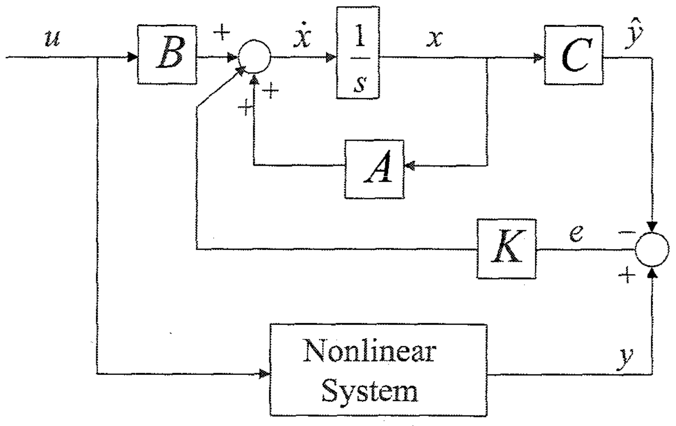


Figure 5.1: Observer schematic

5.2 Observer Implemented

In choosing the observer poles the observer is immediately implemented in the non-linear controller. With this it is meant the estimated state x_1 which represents estimated window size is used as signal for x_1 in the non-linear controller. If state estimate for window size converges to actual window size then this complete controlling scheme is based upon just q, $\dot{q}$ and $q_{desired}$.

In Figure 5.2 you can see the results of a simulation. This simulation was run at $N = 60$ and $T_p = 0.2$. The linear observer is also set with these constants. Experience with the non-linear controllers poles and value for N_o for the controller showed that convergence can be assured by tuning both the chosen nominal value of the load (N_o) for the observer and the chosen observer poles. The results of observing of the states by the combined observer and controller, which will be termed $NL2$ in the rest of this discussion, are depicted in Figure 5.2. The observer poles are in vector K, and are chosen in such a way the matrix A_{obs} from Equation (5.2) has desired eigenvalues in the left half plane. By means of Ackerman computation (acker.m) these observer gains are computed. When choosing the eigenvalues λ_1, λ_2 = -3,-2 it results in k_1 = 0.0150, k_2 = 0.4200. For λ_1, λ_2 = -10,-9 it results in k_1 = 0.3301, k_2 = 14.4200. For ease the algorithm acker.m is used, because changing the poles would result in having to compute the eigenvalues with variable k_1 and k_2. The drawback is that because C has rank 1 no two equal poles can be chosen. A maximum multiplicity of rank C transposed can be used. As will be shown this does not have a bad influence on estimation. As can be seen both observers have convergence. Still higher poles result in a better behavior and for when the load is going to be varied the higher of the two would be preferred.

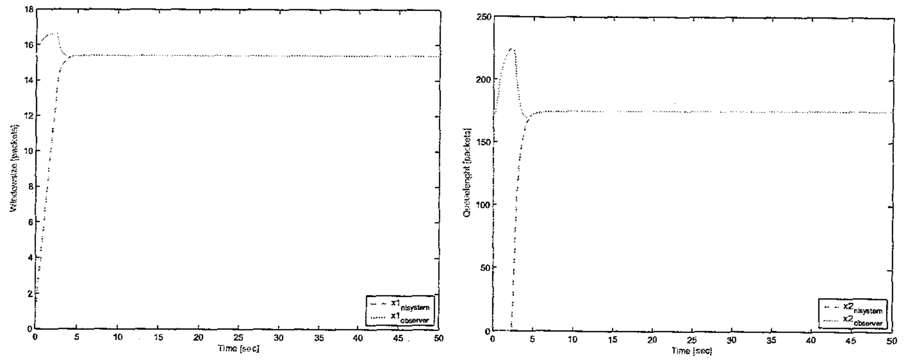


Figure 5.2: Observer poles -3 and -2

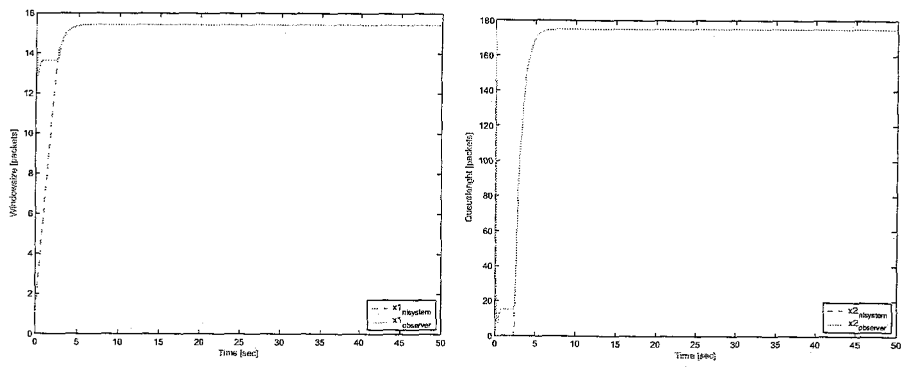


Figure 5.3: Observer poles -10 and -9

5.3 Experiments

The experiment 4 through 12 as described in Section 4.3 will be depicted in Appendix C and described here. Experiment 4 has varying T_p uniformly in the same range of [0.15,0.25]. Figures C.2 and C.3 show what happens when the constant N chosen for the observer part is chosen. Respectively they are chosen 60 and 20. Choosing it equal to 100 showed worse results. As can be seen the observer added raises the settling queue length. For very low nominal load (N_o = 20) chosen for observer this raising occur fast. For now this constant is kept at 60. The queue length is more stable through NL_2 then PI, but the reference value is not exactly 175. The robustness comes at the cost of a raised 'steady state' queue length.

Experiment 5 and 6 had the load on a profile as seen in Figure 4.8. The results shown in Figures C.4 - C.7. The most striking thing is the sensitivity to the load variation. This again changes the queue length where the controller is aiming for. It does handle the noise much better. It can be said having more knowledge about the load variation seems crucial to really outclass PI.

Experiment 7 through 12 were all run for comparison. However the steady state deviation occurs in all of them which is why they are not depicted in Appendix C. In short it can be said the controller has steady state deviation but the is better equipped (more aggressive) to handle to noise.

5.4 Steady State Deviation

This type of control is very different from PI control, and has as it seems its own trade-offs. Knowing the nature and causes of these trade-offs is key in comparison to PI control. When evaluating the simulation results with the non-linear controller with added observer, it is evident that it estimates the state x_1 so that $N_{obs} \cdot x_1 \approx N_{actual} \cdot x_{1actual}$. This results also in x_2 to be estimates wrong, however that is no problem since actual queue length is known. To investigate the deviation in steady state queue length a simulation is run with actual load set to 100 [-] and observer and controller load both set to 60 [-]. All states were evaluated and then the equations evaluated. The reason the steady state queue length does not go to 175 [packets] is because the wrong estimation of state x_1 needs a different queue length to guarantee $\ddot{y}$ goes to zero

$$\ddot{y} = \underbrace{\alpha - \frac{\gamma}{\gamma_o}\alpha_o}_{part1} \underbrace{+\beta - \frac{\gamma}{\gamma_o}\beta_o}_{part2} + \underbrace{\frac{\gamma}{\gamma_o}}_{part3} v. \tag{5.6}$$

In the exact feedback linearized system $\ddot{y}$ would go to zero for v going to zero which means steady state feedback is the same is the desired one. However because the observer has trouble estimating the real x_1 state, part 1 and part 2 are *not* equal to zero. (The value of part 2 is negligible here also for the same reasons mentioned earlier in Section 4.1.3). The fact that α_0 is based on the estimates window size results in this term remaining. To still guarantee $\ddot{y}$ will go to zero $(q - q_{reference})$ must remain a value. For N_{actual} higher then N_{obs} the window size is estimated to low resulting in part 1 to be positive needing part 3 to be negative. Because in part 3 there is a negative sign before the queue error$(-a0 \cdot (q - q_{reference}))$ q will be estimated to high. Thus for N_{actual} lower then N_{obs} this will result a lower queue length at steady state then desired. This is conform to what is seen in all simulations.

The fact that a steady state error is a direct result of a 'wrong' N could be useful. Under the assumption that $N_{obs} \cdot x_1 \equiv N_{actual} \cdot x_{1actual}$ this could be used as a second equation. The first equation being that of $\ddot{y}$ evaluated on steady state. This yields two equations with two unknowns which means this is solvable. The idea is to let the non linear controller go to a too high steady state due to a different N_{actual} then N_{obs}. When this state is reached N could be updated using this difference. Implementation however soon showed this would not be feasible. This method would not hold up in NS-2 because of its noisy characteristic on queue length.

More knowledge about the variation of N is one possibility to improve steady state behavior. A more elaborate scheme for an observer (a non-linear observer f.e.) is not needed when viewing Figure 5.3. The 'bottle-neck' in current performance is definitely the varying nature of N. This is confirmed by looking at the following figure where two simulations are compared where the observer does and does not have the real information about the load factor N. As can be seen when actual load is known steady state converges to the desired queue length.

Adding integral action is a common tactic to result in steady state convergence. This will be mentioned in the following section.

5.5 Adding Integral Action

When looking at Equation (5.6) it show the cause of the steady state deviation. Part 1 and part 3 (part 2 is negligible) balance out which means it must contain a steady state deviation for a different load factor since state x_1 is estimated based upon a wrong parameter in its equation. The idea is to add an integral part to the linearizing feedback v is seen in below

$$v = -a_1\dot{x_2} - a_0(x_2 - x_{2desired}) - a_{-1}\int_0^\infty (x_2 - x_{2desired})\,dt.$$

Because the integral term in this equation is a term that 'remembers' the error signal this term can have value even for $(x_2 - x_{2desired})$ is 0. This means theoretically this should provide convergence to a constant queue length and have the steady state error go to zero. However this has its drawbacks. Because for conditions where this extra term is not needed ($N_{act} = N_{controller}$) this term will make this performance worse. The constant a_{-1} can be used to make this choice for trade-off. Choosing this constant high will yield much overshoot and faster convergence and vice versa. The new controller, named $NL3$, is tested and the following combination proved to give an acceptable behavior. Behavior is compared on overshoot and speed of convergence. This extra term still needs to keep total probability or input u, to be between zero and one. adding this term therefore leads to instability very quickly. Figures 5.5, 5.6 and 5.7 show the results for three simulations where the controller constants are: $NL3$: $[N_{con}, N_{obs}, a_1, a_0, a_{-1}] = [100, 60, 6, 9, 1]$). For completion the constant before the integral term must yield a negative contribution. For a higher load then used in the controller the v must have a negative sign to balance the equation for $\ddot{y}$. Now steady state error goes to zero as seen in Figure 5.5. However this will result in a worse transient. Figures 5.5 and 5.6 both show worse results for $N = 20$. These results are can be made better by changing the controller. However the specific results for $N = 20$ are not the issue. The trend towards the steady state reference is. Therefore the results for $N = 20$ is taken

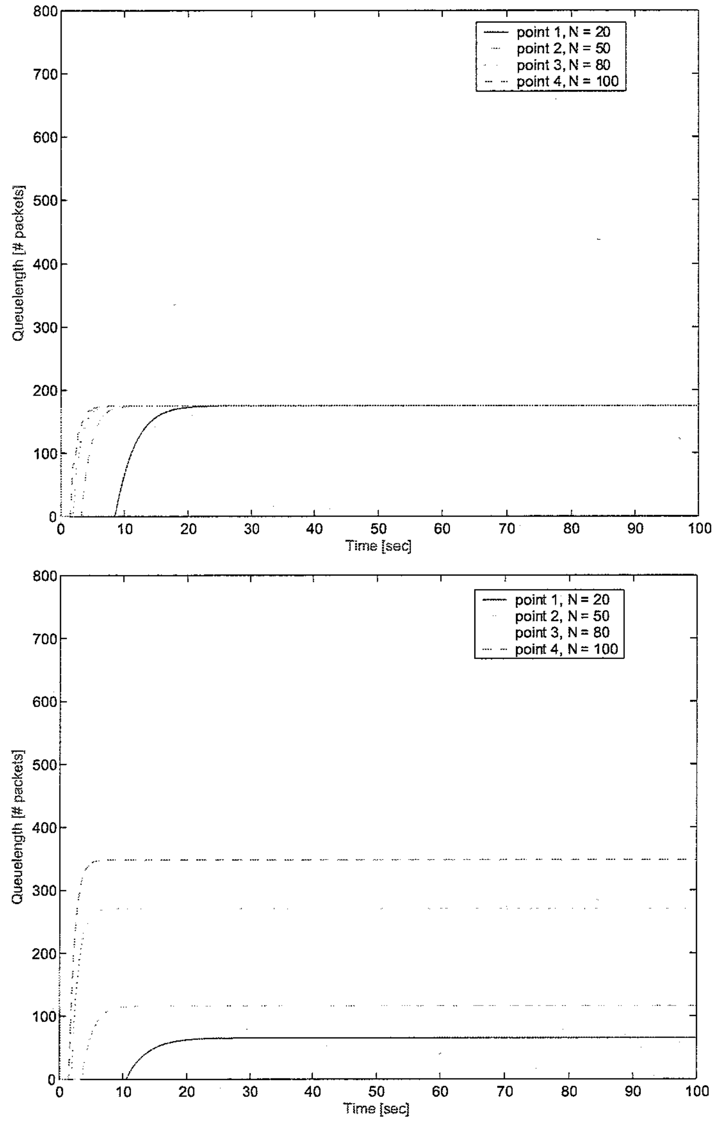


Figure 5.4: Left observer based on actual N (top) and fixed N of 60 (bottom)

out of the figure 5.6 for this will obscure the other results. The results shown in Figure 5.7 are the result of the rapid change in N. This brings a fast and big contribution to the integral term which makes this type of control not very effective for very effective for very large varying N as done in experiment 5. In this Figure also at $t \approx 115$ [sec] some oscillations are seen. This is because the load jumps to 20 [-], which brings along stability issues.

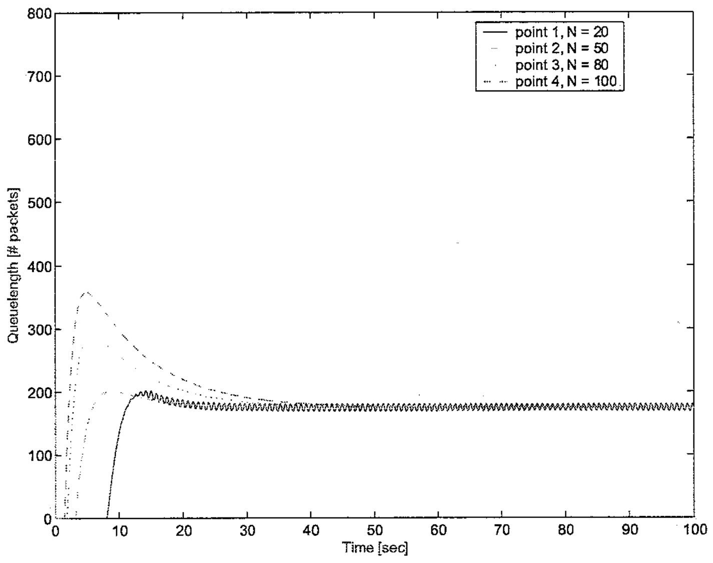


Figure 5.5: NL_3 performance on different load values

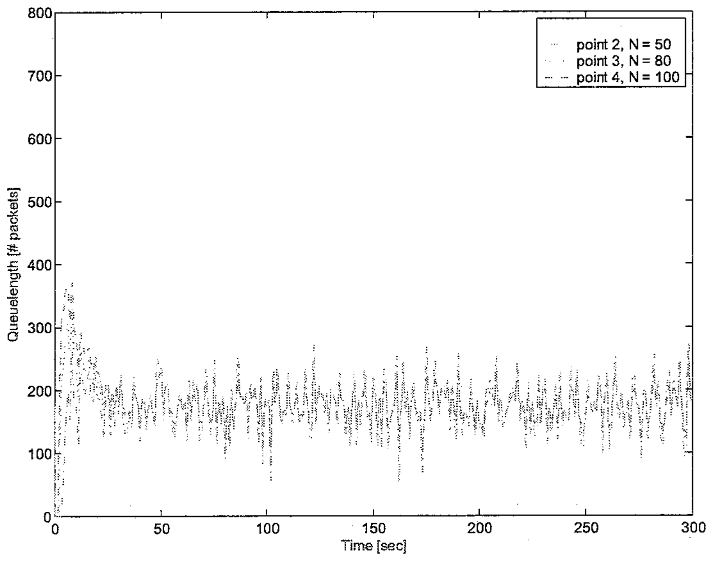


Figure 5.6: NL_3 performance on different load values with noise 1 [Hz], standard deviation 2.5 [-]

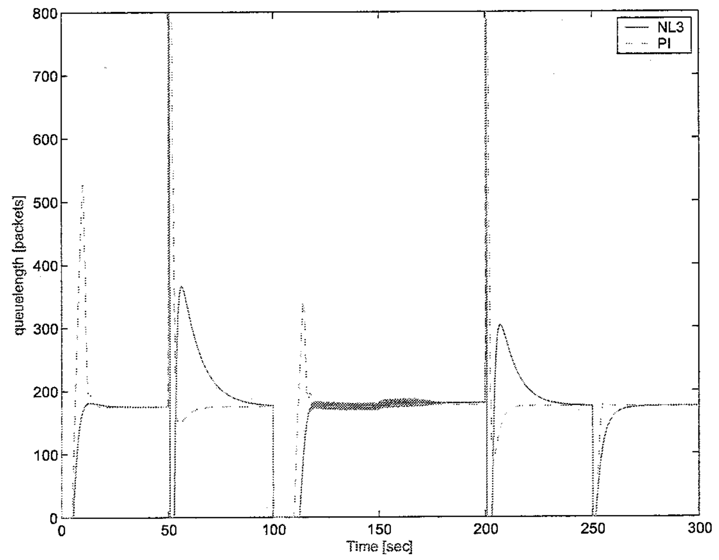


Figure 5.7: Experiment 5: NL_3 versus PI controller

Chapter 6

NS-2 'experiments'

6.1 NS-2 Software

NS-2 is a discrete event software program which enables a simulation which is much better then a fluid-based simulation. These simulations show much more variability then simulations based upon the fluid-model. For instance in [10] results are shown. When drawing a smooth line through queue length results obtained by NS-2, this smooth line looks very much like the fluid model describes.

A real life network experimental facility would be expensive, and very complicated. The advantage of NS-2 is that a network layout can be altered very quickly. Attributes like propagation delay and link capacity are constants in software application. In a real system this will provide more difficulty. Simulations in this program are accepted as genuine in the scientific community.

The main drawback of NS-2 is the programming language for it uses C++. For this discussion it was not possible to therefore apply everything on NS-2. However the PI controller is implemented already and PID has about the same structure which means this can be implemented. It proved not possible to implement the non-linear control scheme. This would be a nice assignment for a computer engineer known with the language C++. This will however be something for future work. To give an insight in NS-2 PI and PID will be implemented.

6.2 NS-2 Experiments

The PI and PID controllers both have to be discretized to be implementable in the discrete event simulator. For this the z-transform will be applied. A definition for the z-transform is taken from [12] and given in the following equations. In analysis of continuous systems the Laplace transform is used which is defined as

$$\mathscr{L}\{f(t)\} = F(s) = \int_0^{\infty} f(t)e^{-st}dt$$

For discrete systems a very similar procedure is available. The z-transform is defined by

$$\mathscr{Z}\{f(k)\} = F(z) = \sum_0^{\infty} f(k)z^{-k}$$

where $f(k)$ is the sampled version of $f(t)$ and k = 0,1,2,... which refers to the discrete sample times $t_0, t_1, t_2, \ldots$. In [18] the discrete form of the PI controller that this discussion was set out to improve is given. A Tustin approximation was used in discretization, which is based upon a trapezoidal integration formule. This is depicted and described very well in [12]. Tustin's Method or the bilinear approximation results in the following substitution in going from the laplace domain to the discrete domain

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right).$$

As seen in [18] this results in the following

$$C_{PI}(s) = \frac{K_{PI}\left(\frac{s}{\omega_g}+1\right)}{s} \qquad (6.1)$$

$$C_{PI}(z) = \frac{az-b}{z-1} = \frac{\partial p(z)}{\partial q(z)}, \qquad (6.2)$$

where $\partial q = q - q_{reference}$ and $\partial p = p$. Equation (6.1) is equal to Equation (3.1) but written in a different form. Here ω_g equals the pole chosen to be 0.53 [rad/s] and K_{PI} equals the constant $9.6426 \cdot 10^{-6}$ [-]. Of the discrete description constants a and b are dependant of the sample time chosen. The chosen sample frequency was 160 [Hz] ([18]) and this resulted in the constant a equal to $1.822 \cdot 10^{-5}$ and b equal to $1.816 \cdot 10^{-5}$ (verifiable with Matlab). When this controller is applied in NS-2 the results are as stated in Figure 6.1.

The non-linear controller described in the previous chapter is not easy to implement. For this also a lot of knowledge about the programming language C++ is needed. Implementation was therefore not achievable in this discussion. This will be mentioned in the conclusion of this discussion under 'future work'. The PID controller with low pass, seen in equation (3.2), is described in the Laplace domein. Therefore the transition into the discrete domein is easily computed and very similar to that of the existing PI implementation. Therefore this controller is implementable and this results into the following transfer function

$$C_{PIDL}(z) = \frac{az^2+bz+c}{z^2+ez+f}.$$

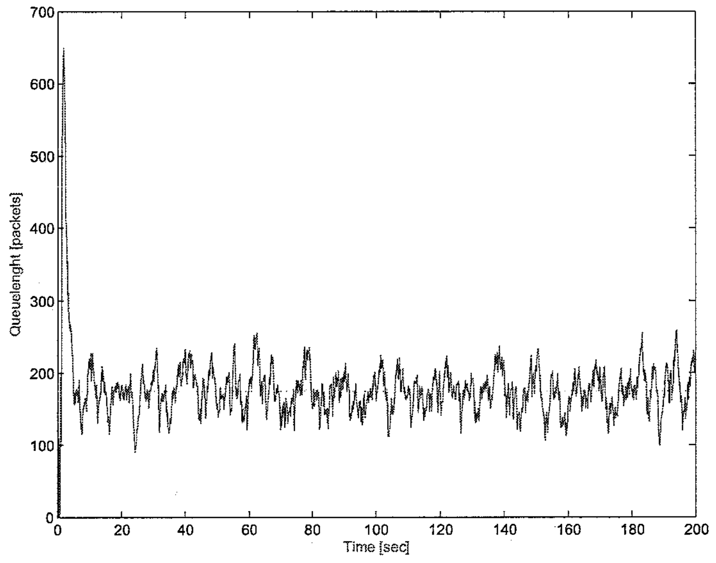


Figure 6.1: PI controller applied in NS-2, Controller constants $a = 1.822 \cdot 10^{-5}$ and $b = 1.816 \cdot 10^{-5}$

This can be written into a difference equation as will be shown here

$$C_{PIDL}(z) = \frac{p(z)}{\partial q(z)} = \frac{az^2 + bz + c}{z^2 + ez + f} \cdot \frac{z^{-2}}{z^{-2}} = \frac{a + bz^{-1} + cz^{-2}}{1 + ez^{-1} + fz^{-2}}. \tag{6.3}$$

NS-2 showed when simulating that when the low pass filter was set to 100 [Hz] the results were incredible noisy. Lowering this low pass pole still ensures the close to 1 [rad/s] bandwidth and admirable phase margin. Through simulations this low pass pole (τ) was set to 6.5 [Hz]. This yields the following for the constants in Equation (6.3): $a = 5.514 \cdot 10^{-5}$; $b = -1.0872 \cdot 10^{-4}$; $c = 5.359 \cdot 10^{-5}$; $e = -1.96$ and $f = 9.60 \cdot 10^{-1}$. The frequency domain representation of the discretized controller is compared that of the original system. This is seen in Figure 6.2, as can be seen the two match closely. If instead of the Tustin method the Zero Order Hold method was used, this does not match a well as seen in Figure 6.3. The vertical line in these figures denote the Nyquist frequency ($\frac{\pi}{T_{sample}}$ [rad/s]). The difference equation at time $t = kT$ where $T = \frac{1}{f_{sample}}$ is given below

$$p(kT) = a\partial q(kT) + b\partial q((k-1)T) + c\partial q((k-2)T) - ep((k-1)T) - fp((k-2)T). \tag{6.4}$$

Applied on NS-2 the simulation results for N is 60 are presented in Figures 6.4 and 6.5. The results show PID has less noise then the PI controller

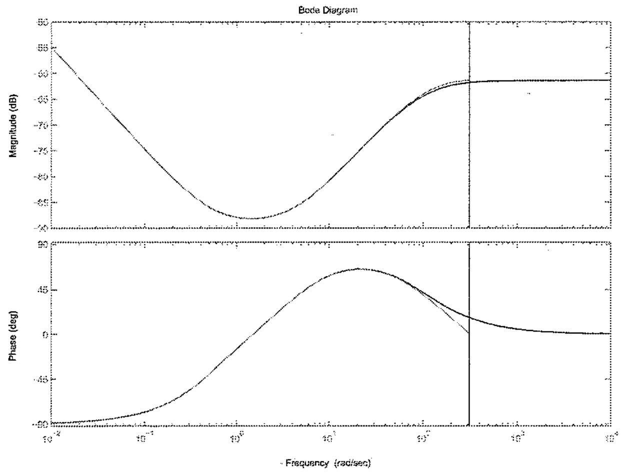


Figure 6.2: Bode diagram of Continuous and Discretized controller using tustin method; dashed line is descritized controller

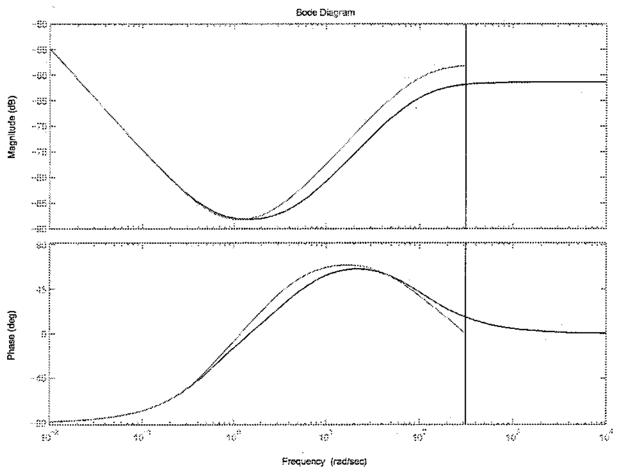


Figure 6.3: Bode diagram of Continuous and Discretized controller using zero order hold method

in 'steady state' behavior. The transients however, PID seems a bit smoother the PI but it results also in an empty queue for a while. Applied on NS-2 the simulation results for N is 100 are presented in Figures 6.6 and 6.7. These do show the smoothness is better meaning PID is better.

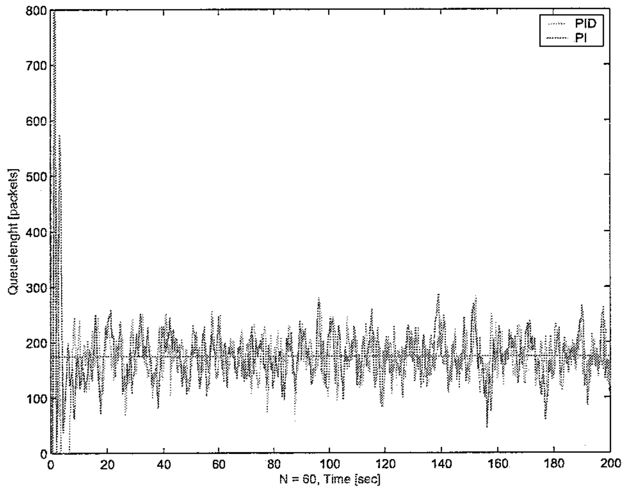


Figure 6.4: NS-2 simulation results of implemented PI and PID controller

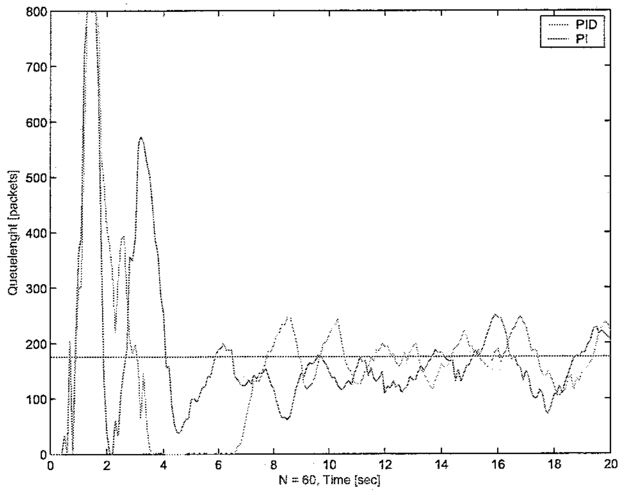


Figure 6.5: NS-2 simulation results of implemented PI and PID controller, emphasis for transient

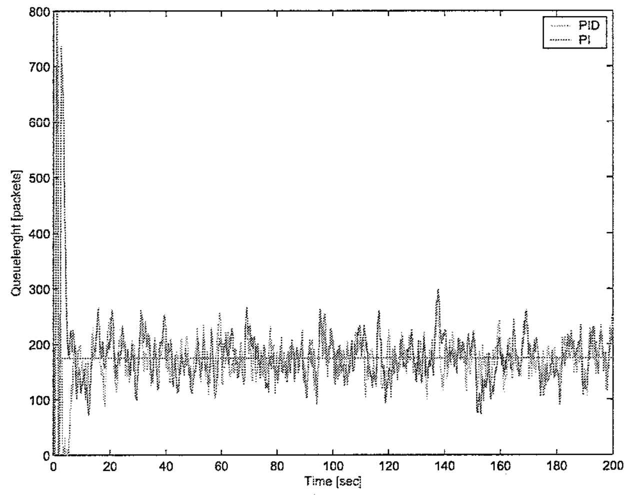


Figure 6.6: NS-2 simulation results of implemented PI and PID controller

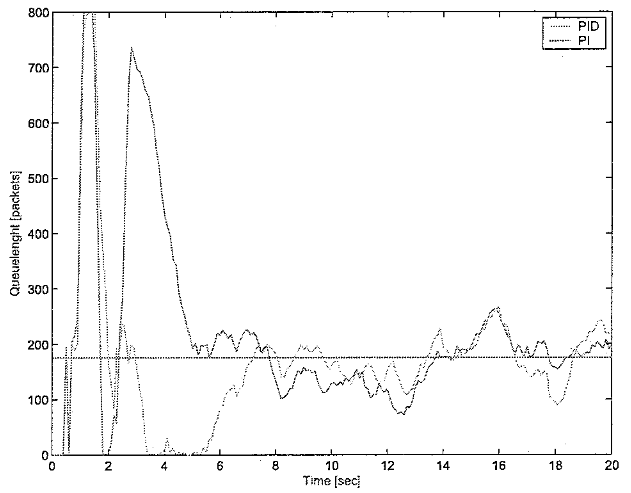


Figure 6.7: NS-2 simulation results of implemented PI and PID controller, emphasis for transient

Chapter 7

Conclusion & Future Work

We have developed a non-linear AQM that shows in fluid-flow simulation significant performance improvement for single congested link compared with that achieved with a PI AQM. The properties of the system have been described, which can be seen as serious drawbacks when applying control laws. The current controller also encounters these drawbacks. The current derived controller has a profound increase in performance as has been mentioned. However it now lacks the robustness due to the unknown nature of the load factor. It has been shown that if this can be overcome this type of control is significantly better the PI control. It is much less sensitive for time delays and has easily tuneable parameters to try and even squeeze performance ('fine-tuning'). Directional influences on parameters have been made clear so when another feasible domain of parameters will be examined tuning will not be a problem.

For future work this controller could be fitted with either an estimation algorithm for the load or the controller with integrator has to be fine tuned. After this is done it can be applied in NS-2. This will required knowledge of C++ programming and NS-2 in general. This discussion was concluded with only applying PI and PID with low pass to show the improvement it made. This is done and results are as expected. PID control provides better results but not ground breakingly better. This is also done to give a feeling for the properties of NS-2.

Other future work is applying this type of control on a multi congested link network. This is a more complex representation of a computer network. Another step in complexity is adding multiple router layers which will greatly complicate things. If these pieces of work are done and still this provides promising results for this type of control, this type of control should be implemented in actual routers. However that perhaps is a bit far fetched.

Appendix A

Assumed Full State Feedback Non-linear Controller vs PI

Experiment 4 till 12 results depicted below.

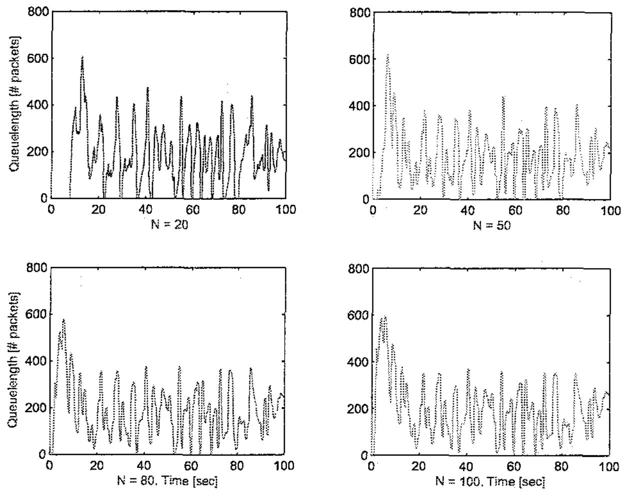


Figure A.1: Experiment 4: queue length on varying T_p, 1 $[Hz]$, PI controller

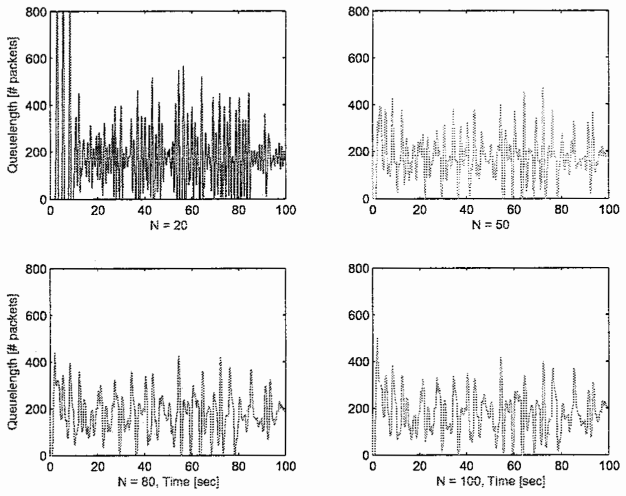


Figure A.2: Experiment 4: queue length on varying T_p, 1 $[Hz]$, PID controller

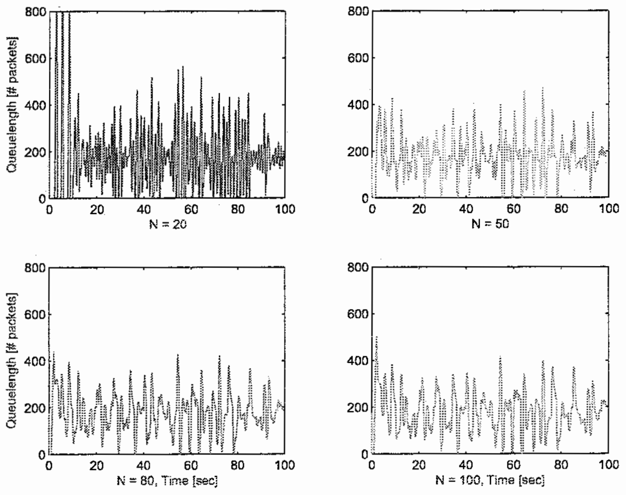


Figure A.3: Experiment 4: queue length on varying T_p, 1 $[Hz]$, Non-linear controller 2

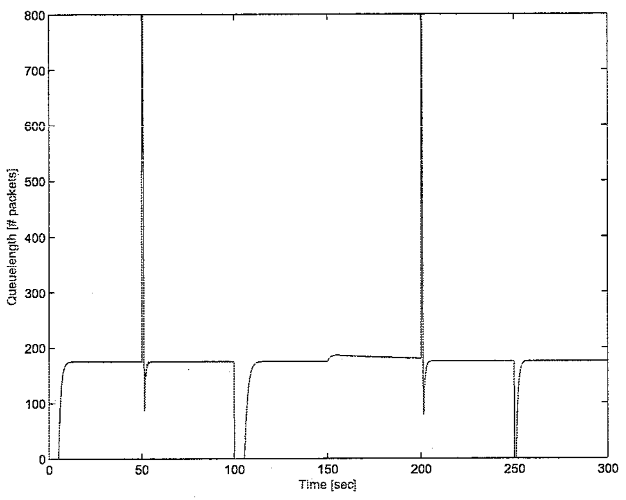


Figure A.4: Experiment 5: controller 2 of table 4.2 applied on Load reference without noise

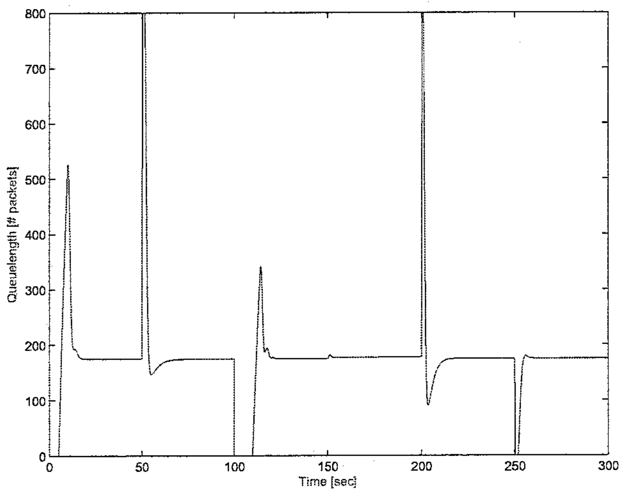


Figure A.5: Experiment 5: PI controller applied on Load reference without noise

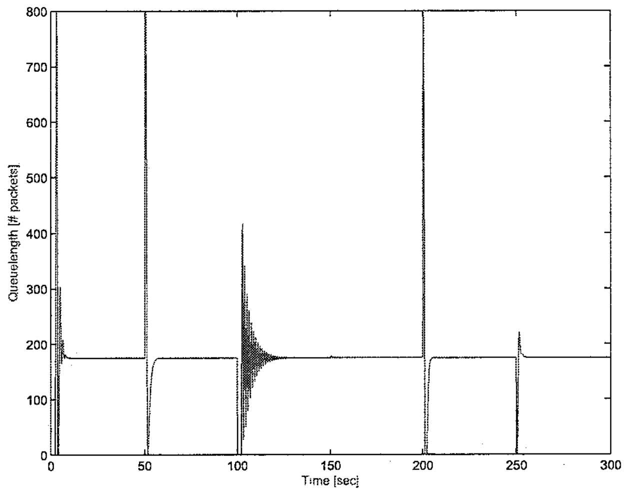


Figure A.6: Experiment 5: PID controller applied on Load reference without noise

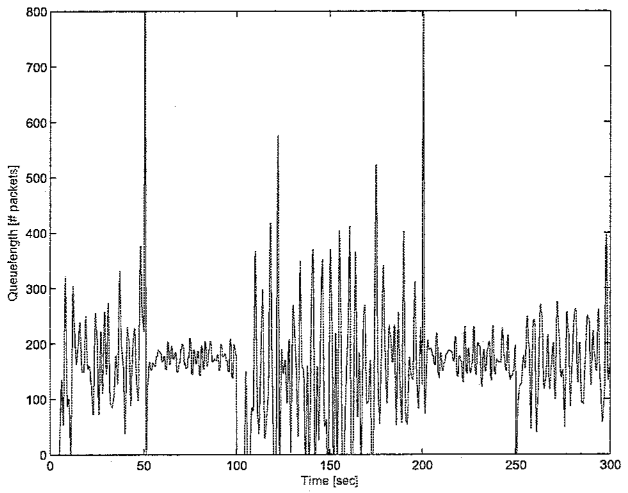


Figure A.7: Experiment 6: Non-linear controller 2 of Table 4.2 applied on load reference with noise

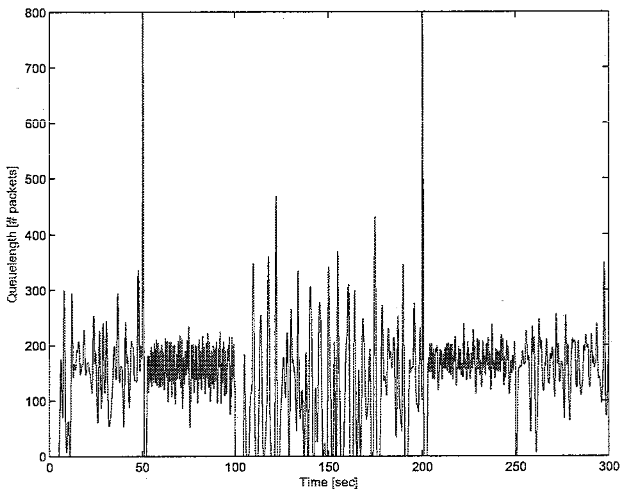


Figure A.8: Experiment 6: Non-linear controller 4 of Table 4.2 applied on load reference with noise

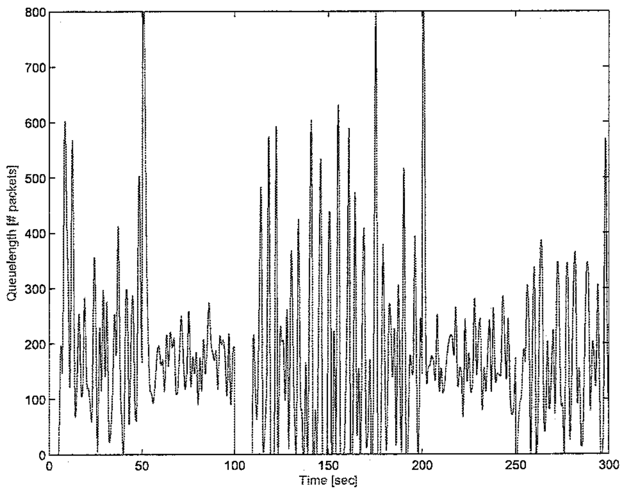


Figure A.9: Experiment 6: PI controller on load reference with noise

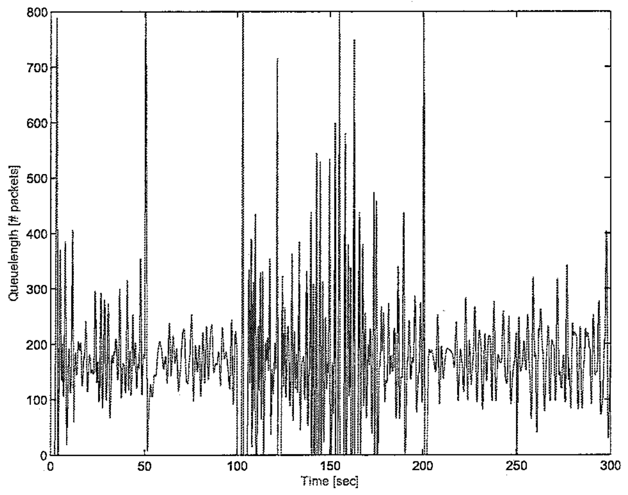


Figure A.10: Experiment 6: PID controller on load reference with noise

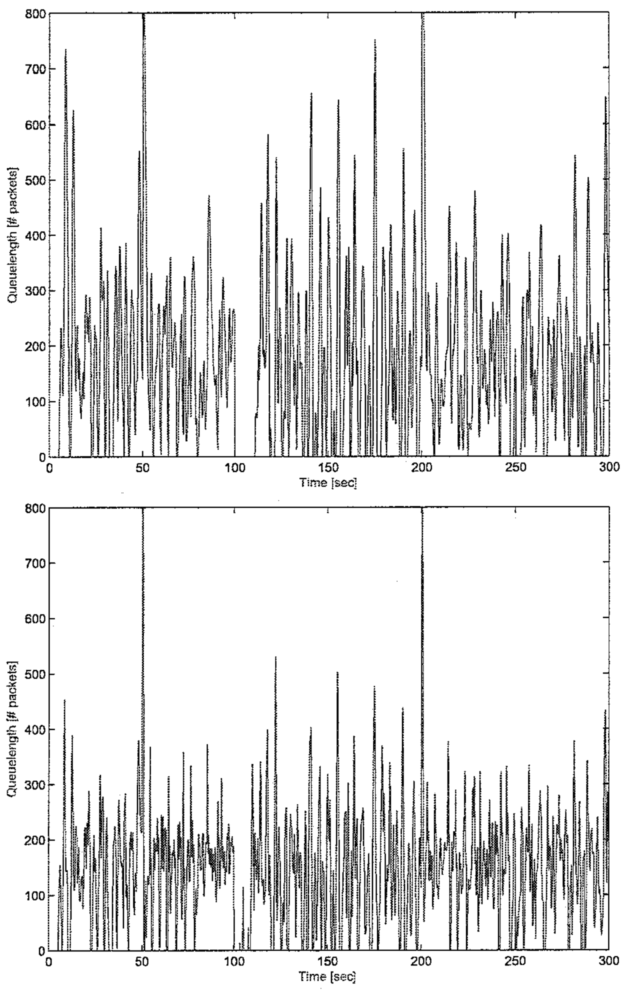


Figure A.11: Experiment 7: PI (top) vs. Non linear controller 2 (bottom); load reference with noise as Figure 4.8 with varying T_p

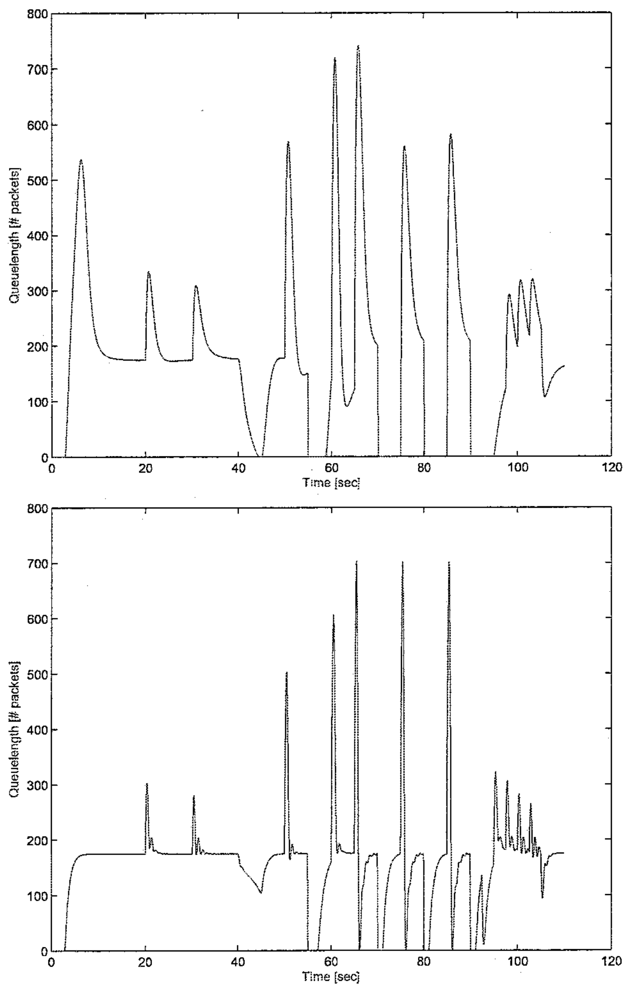


Figure A.12: Experiment 8: PI (top) vs. Non linear controller 2 (bottom); load reference without noise as depicted in Figure 4.9

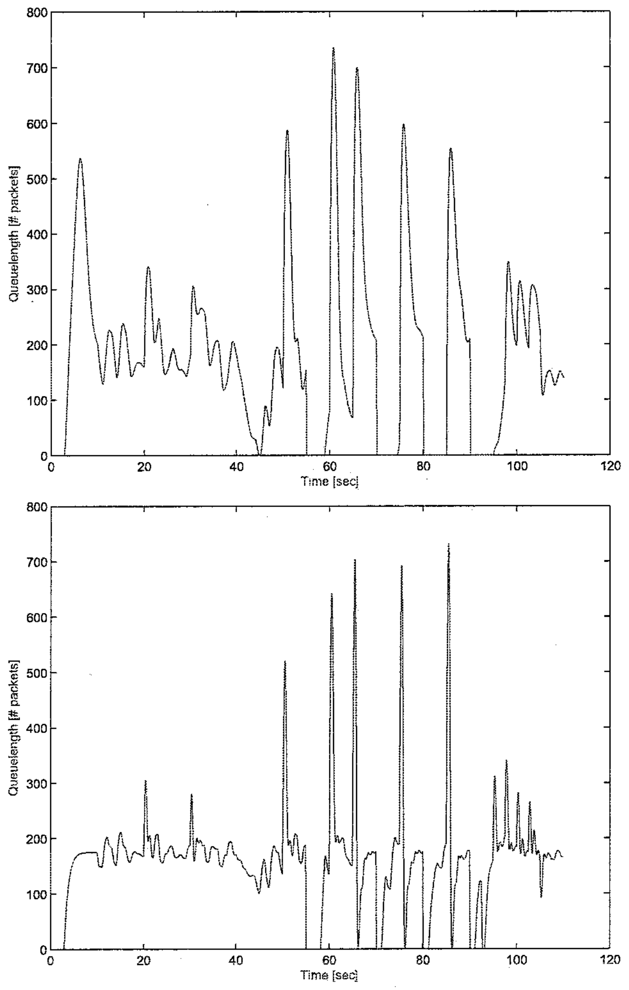


Figure A.13: Experiment 9: PI (top) vs. Non linear controller 2 (bottom); load reference with noise as depicted in Figure 4.9

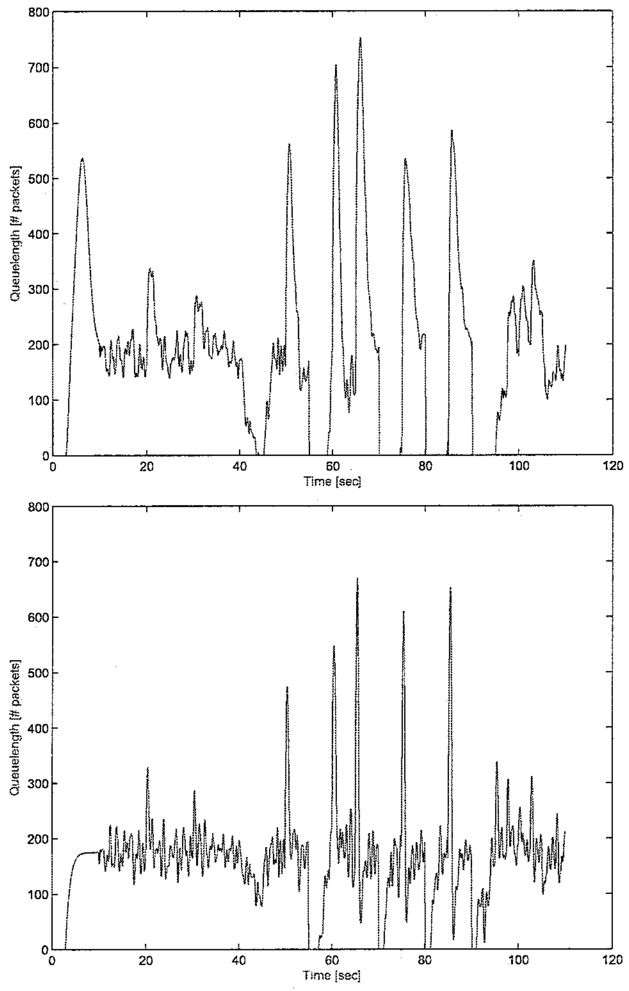


Figure A.14: Experiment 10: PI (top) vs. Non-linear controller 2 (bottom); load reference with noise

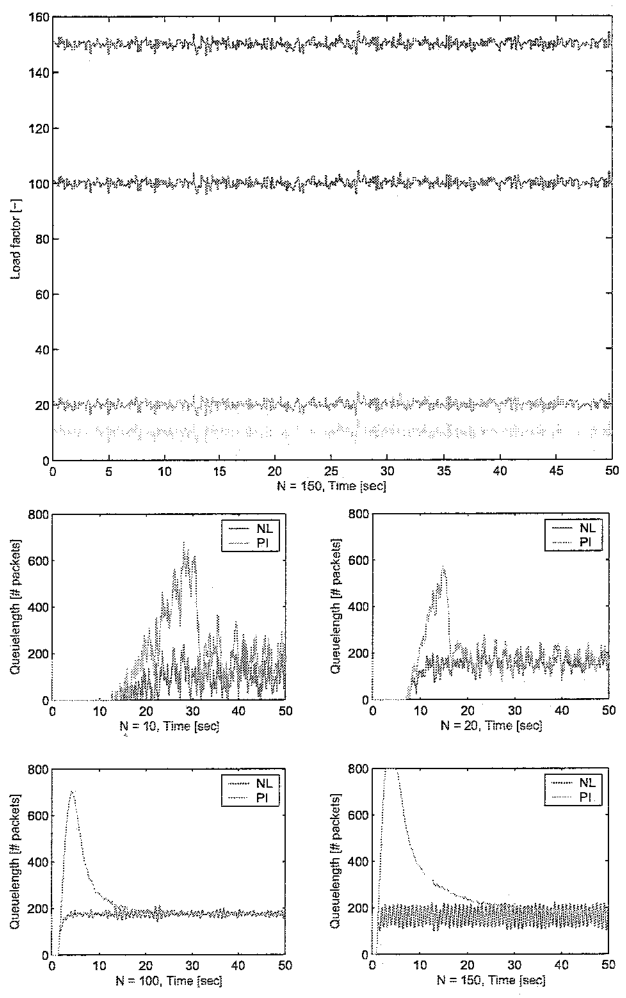


Figure A.15: Experiment 11: Varying load references with noise 5 [Hz] (top); PI and Non-linear controller 2 (bottom)

Appendix B

Lower Bound Non-linear Controller

The non-linear controller performs well for low N, just slow because of reasons stated, the left figure in Figure B.1 presents this. The setpoint is reached very smooth. Because of the low load the window size is high about 926 packets. Because real TCP has a slow start mechanism and not just additive increase multiplicative decrease as the mathematical model represents, the raise of the window size would go faster (if a high enough threshold is set).

T_p has no influence on high frequent gain ($\frac{2N}{C}$). For high propagation delay the time delay depending on the round trip time obviously result the PI controller to be useless. However the non-linear controller performs well. This value for propagation delay is unrealistically high, this again results in the actual dynamics of the system starting very late (t = 1550 s).

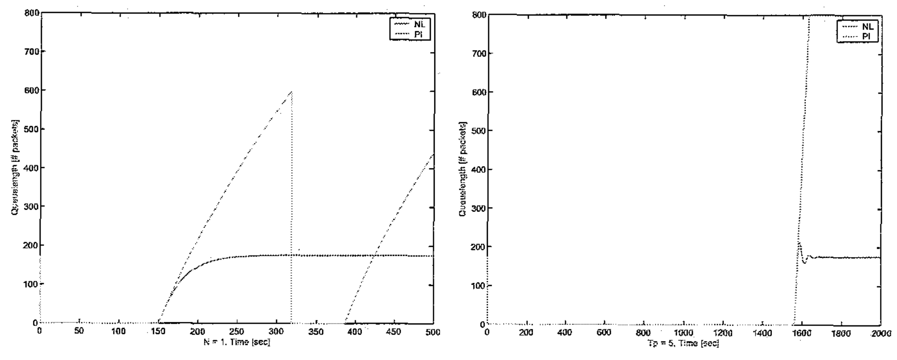


Figure B.1: (left) N = 1; (right) $T_p = 5$

Appendix C

Observer Assisted Full State Feedback Non-linear Controller vs PI

Experiment 4 till 12 results depicted below.

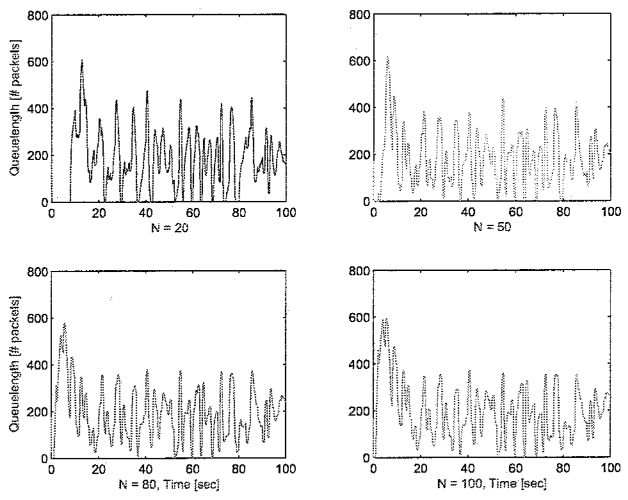


Figure C.1: Experiment 4: queue length on varying T_p, 1 $[Hz]$, PI controller

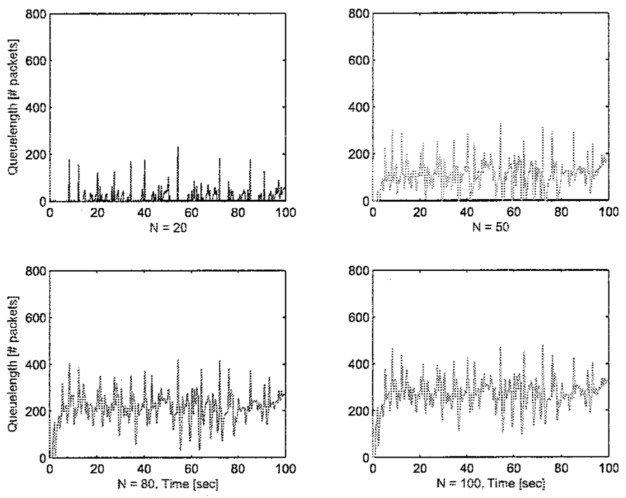


Figure C.2: Experiment 4: queue length on varying T_p, 1 $[Hz]$, $NL2$ controller with N_o of observer set to 60

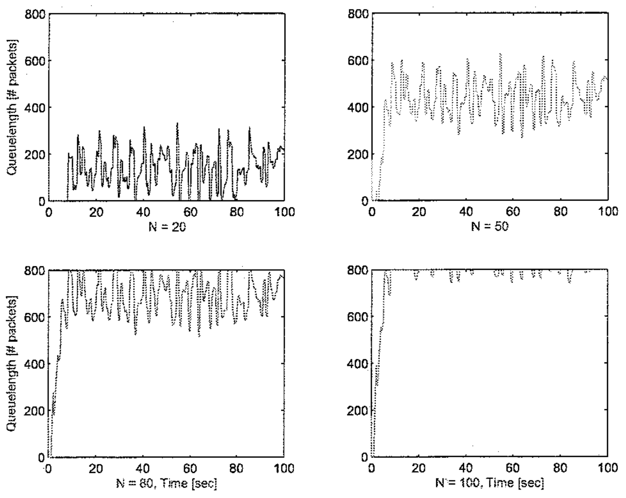


Figure C.3: Experiment 4: queue length on varying T_p, 1 $[Hz]$, $NL2$ controller with N_o of observer set to 20

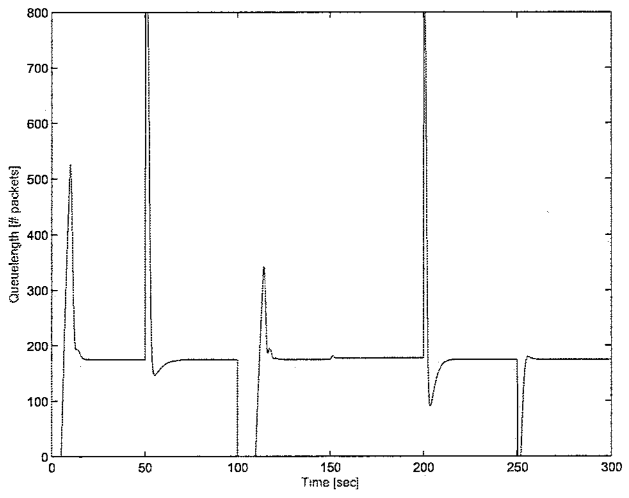


Figure C.4: Experiment 5: PI controller applied on Load reference without noise

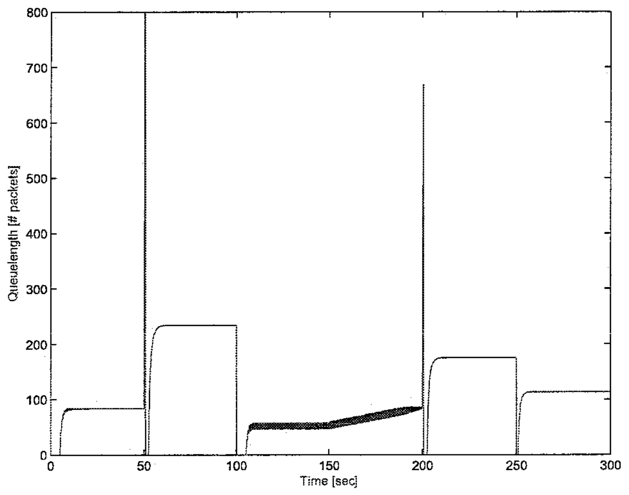


Figure C.5: Experiment 5: NL_2 controller applied on Load reference without noise

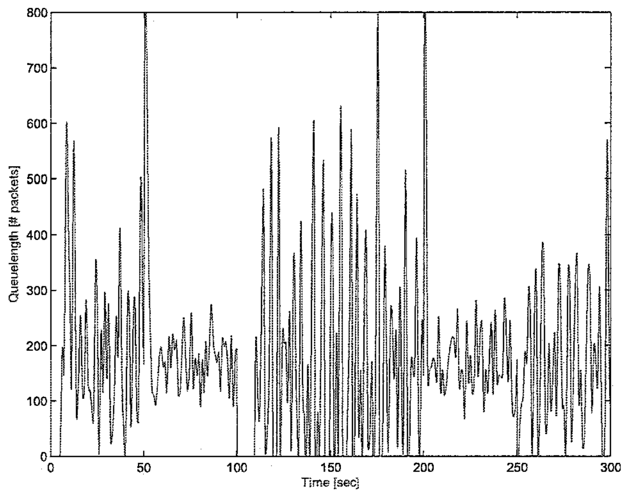


Figure C.6: Experiment 6: PI controller on Load reference with noise

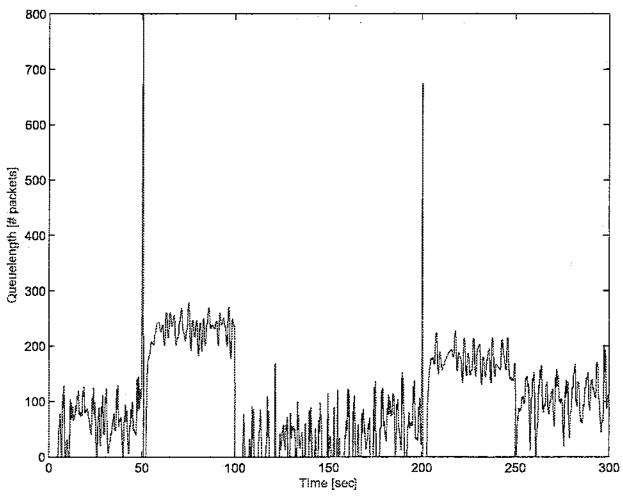


Figure C.7: Experiment 6: NL_2 controller applied on Load reference with noise

Bibliography

[1] *http : //www.cisco.com/en/US/products/sw/iosswrel/ps1828/products _configuration_guide_chapter09186a00800ca59c.html*, 1992 – 2003 Cisco Systems, Inc.

[2] J.F. Kurose & K.W. Ross, *Computer Networking*, Addison-Wesley, 2001

[3] K. Claffy, G. Miller, and K. Thompson *The nature of the beast: recent traffic measurements from an Internet backbone*, in Proceedings of INET'98, ISOC, Washington, DC, 1998

[4] "Skype", obtain via *http://www.skype.com/.*

[5] J. Martin & A. A. Nilsson, *The Evolution of Congestion Control in TCP/IP: from Reactive Windows to Preventive Rate Control*, CACC Technical Report TR-97/11, 1997

[6] http://www.internettrafficreport.com/main.htm, dated 1-12-2003

[7] "ns-2 simulator", obtain via *http://www.isi.edu/nsnam/ns/.*

[8] V.Misra, D.Towsley & W. Gong, *Fluid-based Analysis of a network of AQM Routers Supporting TCP Flows with an Application to RED*, in proceedings of ACM/SIGCOMM, 2000

[9] Y.Chait, C.V.Hollot, V.Misra, D.Towsley, H. Zhang & J.C.S.Lui, *Providing Throughput Differentiation for TCP Flows Using Adaptive Two-Color Marking and Two-Level AQM*, IEEE/INFOCOM, 2002

[10] C.V. Hollot, V. Misra, D. Towsley & W. Gong, *Analysis and Design of Controllers for AQM Routers Supporting TCP Flows*, IEEE transactions on automatic control, vol. 47, no. 6, june 2002

[11] C.V.Hollot, V.Misra, D.Towsley & W. Gong, *A control theoretic analysis of RED*, Proceedings of IEEE/INFOCOM, April 2001

[12] G.F. Franklin, J.D. Powell & A. Emami-Naeini, *Feedback Control of Dynamic Systems* , Prentice Hall fourth edition, 2002

[13] Y. Chait, S. Oldaky, C.V. Hollot & V Misra *An Adaptive Control Strategy for AQM Routers Supporting TCP Flows,* ASME International Mechanical Engineering Congress and Exposition, 2001

[14] W. Wu, Y. Ren & X. Shan, *A Self-configuring Proportional-Integral Controller for AQM Routers Supporting TCP-like Flows,* Proceedings of the 7th Asia-Pacific Communication Conference (APCC'2001), Tokyo Japan, 2001

[15] prof.dr.ir. H. Nijmeijer, *Coarse notes: Controlling of Non-linear systems, TUE,* 2002

[16] S.Sastry *Non-linear Dynamics, Analysis, Stability and Control,* Springer Verlag, Berlin, 1999, ISBN 0387985131

[17] H.A.B.te Braake, H.L.J. van Can, J.M.A. Scherpen & H.B. Verbruggen, *Control of Non-linear Chemical Processes Using Neural Models and Feedback Linearization,* Computers & Chemical Engineering, 1997

[18] C.V.Hollot, V.Misra, D.Towsley & W. Gong, *On designing Improved Controllers for AQM Routers Supporting TCP Flows,* in proceedings IEEE INFOCOM, 2001