A New Estimation Scheme for the Effective Number of Users in Internet Congestion Control

Marios Lestas Member IEEE, Andreas Pitsillides Senior Member IEEE, Petros Ioannou Fellow IEEE, George Hadjipollas Student Member IEEE

Abstract—Many congestion control protocols have been recently proposed in order to alleviate the problems encountered by TCP in high speed networks and wireless links. Protocols utilizing an architecture which is in the same spirit as the ABR service in ATM networks require estimates of the effective number of users utilizing each link in the network to maintain stability in the presence of delays. In this work we propose a novel estimation algorithm which is based on online parameter identification techniques and is shown through analysis and simulations to converge to the effective number of users utilizing each link. The algorithm does not require maintenance of per flow states within the network or additional fields in the packet header, and is shown to outperform previous proposals which were based on pointwise division in time. The estimation scheme is designed independently from the control functions of the protocols and is thus universal in the sense that it operates effectively in a number of congestion control protocols. It can thus be successfully used in the design of new congestion control protocols. In this paper, to illustrate its universality, we use the proposed estimation scheme to design a representative set of Internet congestion control protocols. We demonstrate using simulations that these protocols satisfy key design requirements. They guide the network to a stable equilibrium which is characterized by high network utilization, small queue sizes and max-min fairness. In addition they are scalable with respect to changing bandwidths, delays and number of users, and they generate smooth responses which converge fast to the desired equilibrium.

Index Terms—Congestion Control, Adaptive Control, Internet

I. INTRODUCTION

There are strong indications that TCP will perform poorly in future high speed networks which incorporate links with high bandwidth delay products and wireless links which generate random packet losses ([1], [2], [3]). These problems have triggered intense research activity on Internet congestion control which has led to TCP enhancements and new congestion control protocols. Some of these protocols such as HS-TCP [4] and TCP Westwood [5], in a way similar to TCP, use packet losses to infer congestion, while others such as TCP Vegas [6] and Fast TCP [7] use the queuing delay as the congestion signal. However, as pointed out in [8] such

This work was partly funded by the RPF VIDEO and the UCY ADVIDEO projects.

Marios Lestas is with the Department of Electrical Engineering, Frederick University, Nicosia, Cyprus, eng.lm@frederick.ac.cy
Andreas Pitsillides and George Hadjipollas are with the Computer Science Department, University of Cyprus, Nicosia, Cyprus, andreas.pitsillides@ucy.ac.cy, hpollas@ucy.ac.cy
Petros Ioannou is with the Electrical Engineering Department, University of Southern California, Los Angeles, California, ioannou@usc.edu

binary and implicit feedback schemes impose fundamental performance limitations which can only be resolved by using explicit, multi-bit feedback.

Recently, a number of new Internet congestion control protocols which use explicit-multi bit feedback have been proposed. Some of the most promising proposals utilize a control architecture which is in the same spirit as the one used in the ABR service in ATM networks ([9], [10], [11], [12], [13], [14]). Each link in the network calculates at regular time intervals a signal, which represents the sending rate it desires from the users utilizing the link. A packet, as it traverses from source to destination, accumulates in a designated field in the packet header the minimum of the desired sending rate signals it encounters in its path, and communicates this information back to the source through an acknowledgement mechanism. The user side algorithm then gradually modifies the congestion window to match its sending rate with the value received from the network. This approach is attractive since it leads to congestion control schemes which achieve max-min fairness at equilibrium.

In such a control architecture, the most challenging task is the design of the link algorithm which calculates the desired sending rate. Many algorithms have been proposed. Stability analysis of these algorithms in a single bottleneck link network has shown that in order to maintain stability in the presence of delays, the control parameters of the algorithm must be normalized with the number of users utilizing the bottleneck link ([15], [16], [17], [18], [19]). The extension of this result to networks of arbitrary topology is in most cases intractable and so the most common approach has been to normalize the control parameters at each link with variables which reduce to the total number of users in the single bottleneck link case. Examples of such variables include the number of active users ([20], [21]), the number of bottlenecked users ([22], [23], [24]), or the effective number of users ([25], [26]). These variables are unknown and time varying and thus need to be estimated. A number of estimation algorithms have been proposed in literature, with different degrees of implementation complexity. Some of the algorithms require per virtual connection accounting ([22], [23]). Each link in the network maintains a number of states per connection which it uses, together with some decision rule, to infer the number of connections which are bottlenecked at the link. This approach increases significantly the signaling overhead of the protocol and is not amenable for implementation in Internet type network. To overcome this problem, a number of estimation algorithms ([13], [21], [20], [24]) were developed.
based on the following observation. The number of users utilizing a particular link is equal to the sum of inter-packet departure delays over all packets arriving at the link in a specific period of time $\Delta \tau$ divided by this period of time. Estimation algorithms which were developed based on the above idea do not require maintenance of per flow states within the network but require additional fields in the packet header which store either the sending rate or the inter packet departure delay. In order to remove the overhead associated with these extra fields, Fulton in [25] and much later Aweya in [26] propose an algorithm which estimates an effective number of users simply by dividing the input data rate at the link with the desired sending rate generated at the link. Although not specifically mentioned in the fore-mentioned studies the proposed estimation scheme is derived by considering the single bottleneck link case and the scenario where all $N$ users traversing the bottleneck link inject data into the network with a sending rate which is equal to the desired sending rate $p$. In such a scenario, the input data rate at time $t$, denoted by $y(t)$, satisfies the following relationship:

$$y(t) = Np(t)$$

where $p(t)$ denotes the desired sending rate at time $t$. Estimates of $N$ are then obtained by directly dividing $y(t)$ with $p(t)$. However, such a pointwise division in time may not be desirable, since modeling errors and the effect of noise on $y(t)$ and $p(t)$ may lead to large estimation errors especially when $p(t)$ assumes values arbitrarily close to zero and the signal to noise ratio is small ([27]). A similar approach reported in [12], estimates $N$ by dividing the capacity $C$ of the link with a delayed version of the desired sending rate $p$. This approach suffers from the same problem outlined above and in addition, generates accurate estimates of $N$, only if the input data rate converges to the link capacity.

In this paper we build on our work in [14], to develop a universal estimator for Internet congestion control which overcomes the aforementioned problems and addresses the estimation of the effective number of users in single and multiple bottlenecked links. The estimation of an effective number of users is attractive since it leads to schemes which do not require maintenance of per flow states or additional fields in the packet header. For a given $p$, the effective number of users at a particular link of a network of arbitrary topology can be realized as the equivalent number of greedy users in a single bottleneck link network, which would generate at equilibrium the same input data rate as the one reported at the link under consideration. This implies that the effective number of users in networks of arbitrary topology must reduce to the actual number of users in the single bottleneck link case when the latter is used by greedy users only. We thus start by considering a discrete time fluid flow model of a single bottleneck link network which we use to derive a linear static parametric model of the unknown parameter. We then use online parameter identification techniques to derive a novel estimation algorithm whose properties are established analytically. Parameter identification techniques ([27], [28]) are known to overcome the accuracy and robustness problems encountered by estimation methods which are based on pointwise division in time. Despite the fact that parameter identification techniques have been successfully used in other engineering fields, their use in Internet congestion control has so far been limited. So, one of the main contributions of this work is to demonstrate the effectiveness of these methods in such a complex system as the Internet.

As pointed out earlier, the main reason that we need estimates of the effective number of users utilizing each link, is to use them to design effective congestion control protocols. Our objective in this work has been to design the estimation algorithm independently from the control algorithm which updates the desired sending rate at each link so that the same estimation algorithm can be used in the design of various congestion control protocols. To verify that our estimation algorithm has the above universality property, we use it to develop a representative set of Internet congestion control protocols which we evaluate using simulations. These protocols differ in the control algorithm which calculates the desired sending rate at each link. To our knowledge three types of control algorithms appear in the literature: algorithms whose objective is to track a reference queue size and algorithms which attempt direct calculation of the max-min fair allocation by dividing the available capacity with the effective number of flows. In this paper we consider one algorithm from each category; the link side algorithm proposed in [14], a modified version of the queue length approach proposed in [20], and an algorithm which is similar to the one proposed in [25]. Our simulations indicate that unlike previous proposals which were based on pointwise division in time, the proposed estimation scheme works effectively in all three cases. The congestion control protocols are found to guide the system to a stable equilibrium which is characterized by high network utilization, small queue sizes and max-min fairness. In addition they are scalable with respect to changing bandwidths, delays and number of users, and they generate smooth responses which converge fast to the desired equilibrium.

In [14] we report our initial findings related to the development of an effective estimation algorithm which is based on online parameter identification techniques. The estimation scheme proposed therein is designed using a continuous time fluid flow model of a single bottleneck link network and its performance is evaluated in the context of the proposed ACP congestion control protocol. In this paper we utilize a discrete time fluid flow model, to develop a new scheme which requires modifications in both the source and link side algorithms. In addition, we extend our discussion to the multiple link case, to show that the proposed scheme converges to the effective number of users utilizing each link in the network. However, a major contribution in this work is to validate through simulations, that the proposed estimation scheme is
universal in the sense that it is successfully used to design a number of congestion control protocols which satisfy their design requirements. This opens up new directions for protocol design to meet the challenging characteristics of currently deployed networks and the emerging network technologies which result in drawbacks of present protocols.

The paper is organized as follows. In section II we formulate the problem mathematically and in section III we present the proposed estimation algorithm and its convergence properties which are established analytically in Appendix I. In section IV we describe the implementation details of the three representative congestion control protocols and in section V we evaluate using simulations the performance of both the estimation scheme and the congestion control protocols. Finally in section VI we offer our conclusions and future research directions.

II. Problem Formulation

In this section, we use a discrete time fluid flow model to provide a definition for the effective number of users and formulate the problem mathematically. The starting point in this formulation is the special case of a synchronized single bottleneck link network utilized by greedy users only, where the effective number of users reduces to the actual number of users. In such a simple network, the problem formulation and analysis is fairly tractable. We then extend our model and analysis to networks of arbitrary topology. We map the local dynamics at each link of the arbitrary network to the dynamics of a single bottleneck link network and we use this mapping to define the effective number of users and formulate the estimation problem.

A. The Single Bottleneck Link Case

We consider a packet switched store and forward network comprising of a single bottleneck link. The network is utilized by a finite set of $N$ greedy users $U = \{s_1, s_2, ..., s_N\}$, where $s_i$ denotes user $i$. Let $I$ denote the index set of the users. The path of each user consists of the single bottleneck link only. We assume that all users share a common round trip delay denoted by $\tau$. Let $\tau_f$ denote the forward delay, which is the delay that a packet incurs from its point of entry in the network until it reaches the buffer at the bottleneck link. Correspondingly let $\tau_b$ denote the backward delay, which is the time required for any information to be transmitted from the bottleneck buffer to one of the sources. It follows that:

$$\tau = \tau_f + \tau_b$$  \hspace{1cm} (2)

The single bottleneck link has a buffer which accommodates packets waiting to be transmitted. The queue size at the link is denoted by $b$. The capacity of the link is denoted by $C$. At the link we also associate a signal processor which generates a signal $p$ which denotes the desired sending rate of the flows traversing the link. The signal processor updates the desired sending rate once every round-trip time. The control time is thus slotted with the slot duration $[k, k+1)$, $k = 0, 1, ...$ equal to one round trip time. Let $d(k)$ denote the number of packets entering the link in time slot $k$ and let $y(k)$ denote the input data rate at time $k$ calculated by dividing $d(k)$ with the slot duration $\tau$. Let also $b(k), p(k)$ denote the queue size and the desired sending rate respectively at time $k$. The desired sending rate is updated according to a control law of the form:

$$x(k+1) = v[y(k), b(k), x(k)] , \quad x(0) = x_0$$  \hspace{1cm} (3)

$$p(k) = g[y(k), b(k), x(k)]$$  \hspace{1cm} (4)

where $x$ denotes a vector of states which are maintained at the bottleneck link. It must be noted that since $x$ and $p$ are updated once every round trip time at times $k$, $k = 0, 1, ...$, it follows that $x(t) = x(k)$ and $p(t) = p(k)$ for $t \in [k, k+1)$. The functions $v(\cdot)$ and $g(\cdot)$ are to be generated by the congestion control strategy and $g(\cdot)$ is chosen so that:

$$\lambda_1 \leq p(k) \leq \lambda_2$$  \hspace{1cm} (5)

where $\lambda_1, \lambda_2 > 0$. The signals generated at the links are communicated back to the sources in the reverse direction of the traffic, using acknowledgement packets. This mechanism generates at each source $s_i$ a feedback signal $q_i$ whose value at any time $k$, denoted by $q_i(k)$, is given be the following equation:

$$q_i(k) = p(k) - \frac{\tau_b}{\tau}$$  \hspace{1cm} (6)

where $\frac{\tau_b}{\tau}$ denotes the backward propagation delay measured in time slots $\tau$.

Each user injects data packets in the network. We assume persistent [29] sources which always have data to send. We also assume a window based scheme where each user $s_i$ maintains a congestion window $w_i$ of unacknowledged packets. Let $w_i(k)$ denote the congestion window of user $s_i$ at time $k$. In a way similar to TCP Westwood [5], the latter is calculated based on the feedback signal $q_i$ as follows:

$$w_i(k) = q_i(k) \ast \tau$$  \hspace{1cm} (7)

From equation (6) it follows that $w_i(k) = p(k) - \frac{\tau_b}{\tau} \ast \tau$. Since the signal processor at the bottleneck link updates the signal $p$ once every round trip time at times $k$, $k = 0, 1, ...$, it follows that for each $k = 1, 2, ...$, the congestion window $w_i(k)$ is constant and equal to $p(k-1) \ast \tau$ in the time interval $L_k = [k - \frac{\tau}{\tau}, k + \frac{\tau}{\tau})$. Let $r_i(k - \frac{\tau_f}{\tau})$ denote the sending rate of source $s_i$ at time $k - \frac{\tau_f}{\tau}$ which is defined as the number of new packets sent by source $s_i$ in the interval $L_k$, divided by the round trip time $\tau$. The number of new packets sent in the interval $L_k$ is equal to the congestion window $w_i(k - \frac{\tau_f}{\tau})$. It follows that:

$$r_i(k - \frac{\tau_f}{\tau}) = \frac{w_i(k - \frac{\tau_f}{\tau})}{\tau} = p(k - \frac{\tau_f}{\tau} - \frac{\tau_b}{\tau}) = p(k-1)$$  \hspace{1cm} (8)

At the bottleneck link the input data rate at time $k$ is equal to the summation of the sending rates of the users traversing the link and is thus given by:

$$y(k) = \sum_{i \in I} r_i(k - \frac{\tau_f}{\tau}) = Np(k-1)$$  \hspace{1cm} (9)
The model of equation (9) is based on a number of simplifying assumptions, such as the network being synchronized, the users being greedy and the users immediately adopting the signal advertised by the network as their sending rate. However, some of these simplifying assumptions do not hold in practice and thus lead to model uncertainties. These model uncertainties can be accounted for in the underlying model by introducing a noise term \(d(k)\) which satisfies \(|d(k)| \leq \nu_0\), \(\nu_0 > 0\), so that equation (9) becomes:

\[
y(k) = Np(k - 1) + d(k)
\]

(10)

\(N\) denotes the number of effective users which, in the special case that we consider reduces to the actual number of users utilizing the network. In an actual network this number is unknown and changing over time. The objective of this paper is to estimate the number \(N\) of effective users at each instant of time by processing the measurements \(y(k)\) and \(p(k)\).

\[ \lambda_1 \leq p_j(k) \leq \lambda_2 \]  

(14)

where \(\lambda_1, \lambda_2 > 0\). The signals generated at the links are communicated back to the sources resulting in the generation of the feedback signal \(q_i\) at each source \(s_i\). The mechanism with which \(q_i\) is generated is as follows. A packet as it traverses from source to destination it accumulates the minimum of the congestion signals it encounters in its path. The value that reaches the destination is communicated back to the sources in an acknowledgement packet resulting in the generation of the feedback signal \(q_i\).

The system is assumed to have a unique equilibrium point. Let \(y_j^*\) denote the input data rate value at equilibrium at link \(l_j\), let \(p_j^*\) denote the desired sending rate at equilibrium at link \(l_j\). We change the indexing of the links so that the set \(\{p_j^*, j \in J\}\) is rearranged in ascending order of its elements: \(p_1^* \leq p_2^* \leq p_3^* \ldots \leq p_n^*\). Let \(n_{jk}\) denote the number of users utilizing links \(j\) and \(k\) but are not utilizing any of the links \(m < \min(j, k)\). At link 1, all users utilizing the link will adopt a sending rate equal to \(p_1^*\) and so \(y_1^* = n_{1j}p_1^*\). At link 2, all users utilizing the link but are not utilizing link 1 will have a sending rate equal to \(p_2^*\). The sending rate of all other sources utilizing both links 1 and 2, will be equal to \(p_2^*\). It is thus true that \(y_j^* = n_{jj}p_j^* + \sum_{k=1}^{j-1} n_{jk}p_k^*\). The procedure is applied up to link \(L\) to obtain the following:

\[
y_j^* = n_{jj}p_j^* + \sum_{k=1}^{j-1} n_{jk}p_k^*, \quad j \in J
\]

(15)

In our problem formulation and analysis, we define the effective number of users based on the following interpretation:

**Interpretation:** For a given \(p\), the effective number of users at a particular link in a network of arbitrary topology is the number of greedy users in a single bottleneck link network that would generate at equilibrium the same input data rate as the one observed at the link under consideration.

Whether, such a notion of the effective number of users leads to effective congestion control protocols, cannot be verified analytically in networks of arbitrary topology but can only be evaluated using simulations and actual implementation. Based on the above interpretation and equation (9) we can define the effective number of users at each link \(j\) as:

\[
y_j^* = N_j^{\text{eff}}p_j^*
\]

(16)

Combining equation (15) with (16) we obtain:

\[
N_j^{\text{eff}} = n_{jj} + \sum_{k=1}^{j-1} n_{jk} \frac{p_k^*}{p_j^*}
\]

(17)

Equation (17) defines mathematically the effective number of users. It indicates that the effective number of users at a particular link can be realized as the number of users bottlenecks at the link plus the fractional effect of the other users which are bottlenecked elsewhere. A similar notion
appears in [26]. It follows that the effective number of users at a particular link is lower bounded by the number of users bottlenecked at the link and upper bounded by the total number of users utilizing the link.

The model of equation (16) and the definition of equation (17) are expressed in terms of equilibrium values of the related quantities. During the transient period, until the system converges to its equilibrium state, the number of effective users remains constant, however, the signals \( y_j(t) \) and \( p_j(t) \) may deviate significantly from the linear space defined by equation (16). These deviations are incorporated in the model by introducing an error term \( d_j(k) \). This error term can also be used to account for model uncertainties which result from relaxations of the simplifying assumptions on which our analysis relies. The equation thus becomes:

\[
y_j(k) = N_j^{e,f} p_j(k - 1) + d_j(k) \tag{18}
\]

where \( d_j(k) \) is a bounded noise signal which satisfies \(|d_j(k)| \leq \nu_0, \nu_0 > 0\). We choose to express our model in terms of \( p_j(k - 1) \) instead of \( p_j(k) \) so that in the single bottleneck link case it reduces to (10).

### III. Estimation Algorithm

The model of equation (18) is widely known in the literature of adaptive control as a linear static parametric model of the unknown parameter. Having obtained such a linear static parametric model at each link, the next step is to use this model to derive an adaptive law which generates at each link estimates of the unknown parameter. The adaptive law takes the form of a one dimensional difference equation whose solution converges to the unknown parameter. The procedure for developing such a difference equation can be found in a number of manuscripts e.g. [27], however, for clarity of presentation we give an overview of the procedure here with reference to the linear static parametric model (18).

Let \( \hat{N}_j \) denote an estimate of the effective number of users \( N_j^{e,f} \), generated by the adaptive law. We use this estimate to calculate the predicted or estimated value \( \hat{y}_j \) of the signal \( y_j(k) \) as follows:

\[
\hat{y}_j = \hat{N}_j p_j \tag{19}
\]

We then calculate the estimation error \( \epsilon_j \) as the difference between \( \hat{y}_j \) and \( y_j \):

\[
\epsilon_j = \hat{y}_j - y_j \tag{20}
\]

The estimation error reflects the deviation of the parameter estimate \( \hat{N}_j \) from the actual parameter \( N_j^{e,f} \). Our objective is then to develop, using the steepest descent method, a recursive algorithm which at equilibrium minimizes a cost function of \( \epsilon_j \) with respect to \( \hat{N}_j \). Choosing a quadratic cost function of the form:

\[
J(\hat{N}_j) = \frac{\epsilon_j^2}{2} = \frac{(y_j - \hat{N}_j p_j)^2}{2} \tag{21}
\]

we obtain the following recursive scheme:

\[
\hat{N}_j(k + 1) = \hat{N}_j(k) + \gamma \mu_j(k), \quad \hat{N}_j(0) = \hat{N}_0 \tag{22}
\]

\[
\mu_j(k) = (y_j(k) - \hat{N}_j(k) p_j(k - 1)) p_j(k - 1) \tag{23}
\]

where \( \gamma > 0 \) is the adaptive gain. Equation (22) forms the basis of the proposed adaptive law. In order to avoid problems associated with the signals \( y_j(k) \) and \( p_j(k - 1) \) becoming unbounded, we divide both with a signal \( 1 + p_j^2(k - 1) \), often referred to as the normalizing signal. The proposed estimation scheme at each link \( l_j \) then becomes:

\[
\hat{N}_j(k + 1) = \hat{N}_j(k) + \gamma \delta_j(k), \quad \hat{N}_j(0) = \hat{N}_0 \tag{24}
\]

\[
\delta_j(k) = (y_j(k) - \hat{N}_j(k) p_j(k - 1)) \frac{p_j(k - 1)}{1 + p_j^2(k - 1)} \tag{25}
\]

We summarize the properties of the proposed algorithm in the following theorem which is proved in Appendix I.

**Theorem 1:** Let \( \hat{N}_j(k) = \hat{N}_j(k) - N_j^{e,f} \). At each link \( j \in \{1, 2, \ldots, L\} \), \( |\hat{N}_j(k)| \leq c_0 \nu_0 + \epsilon_k \), where \( c_0 > 0 \) is a constant that depends on the rate of convergence, \( \nu_0 > 0 \) is an upper bound on the noise signal \( d_j(k) \) and \( \epsilon_k \) is a term exponentially decaying to zero.

It is clear that the size of the parameter error is of the order of the noise term \( \nu_0 \). It is also clear that in the absence of the error term \( d_j(k) \), the parameter estimate \( \hat{N}_j(k) \) converges exponentially fast to the desired parameter \( N_j^{e,f} \), i.e., \( \lim_{k \to \infty} \hat{N}_j(k) = N_j^{e,f} \).

The effective number of users at the bottleneck links, which are the ones which control the sending rates, are known a priori to be greater than one. At the non-bottleneck links, the effective number of users can be less than 1. However, at those links, even if the estimate converges to a value of one instead of the actual value, the equilibrium sending rates will not be affected. So, we introduce a projection operator in the estimation algorithm which ensures that the estimates do not attain values less than 1. This lower bound does not affect the equilibrium properties of the system and confines the attainable values of the parameter estimates, thus improving the speed of convergence properties of the algorithm. Equation (24) then becomes:

\[
\hat{N}_j(k + 1) = Pr_1[\hat{N}_j(k) + \gamma \delta_j(k)], \quad \hat{N}_j(0) = \hat{N}_0 \tag{26}
\]

where \( \delta_j(k) \) is given by (25) and the projection operator \( Pr_1[\cdot] \) is defined as:

\[
Pr_1[x] = \begin{cases} 
  x & \text{if } x > 1 \\
  1 & \text{otherwise} 
\end{cases} \tag{27}
\]

The initial condition \( \hat{N}_0 \) is chosen such that \( \hat{N}_0 > 1 \) and \( \gamma \) is chosen such that \( \gamma < 1 \).

### IV. Implementation

Our main objective in this work has been to design an estimation algorithm which has the ability to operate effectively with a number of algorithms which calculate the desired sending rate. To verify that the proposed estimation
scheme has the above property, we evaluate its performance in a representative set of Internet congestion control protocols that we develop. These protocols only differ in the algorithm which calculates the desired sending rate. All the other control functions are similar to the ones in [14]. In this section, we describe the main features of the protocols:

A. The packet header

In a way similar to XCP [30], the packet carries a congestion header which consists of 3 fields: the $H_{rtt}$ field, the $H_{feedback}$ field and the $H_{congestion}$ bit. The $H_{rtt}$ field carries the current round trip time estimate of the source which has generated the packet. The field is set by the user and is never modified in transit. It is read by each router and is used to calculate the control period. The $H_{feedback}$ field carries the sending rate which the network requests from the user which has generated the packet. This field is initiated with the user’s desired rate and is then updated by each link the packet encounters in its path. At each link, the value in the field is compared with the desired sending rate value and the smallest value is stored in the $H_{feedback}$ field. In this way, a packet as it traverses from source to destination it accumulates the minimum sending rate it encounters in its path. The $H_{congestion}$ bit is a single bit which is initialized by the user with a zero value and is set by a link if the input data rate at that link is more that 95% of the link capacity. In this way, the link informs its users that it is on the verge of becoming congested so that they can apply a delayed increase policy and avoid excessive instantaneous queue sizes and packet losses.

B. The sender

As in TCP, each sender maintains a congestion window $cwnd$ which represents the number of outstanding packets and a smoothed estimate of the current round trip time $srtt$ which is calculated using an exponentially weighted moving average filter.

The initial congestion window value is set to 1 and is never allowed to become less than this value because this would cause the source to stop sending data. On packet departure, the $H_{feedback}$ field in the packet header is initialized with the desired sending rate of the application and the $H_{rtt}$ field stores the current estimate of the round trip time. If the source does not have a valid estimate of the round trip time the $H_{rtt}$ field is set to zero.

The congestion window is updated every time the sender receives an acknowledgement. When a new acknowledgement is received, the value in the $H_{feedback}$ field, which represents the sending rate requested by the network in bytes per second, is read and is used to calculate the desired congestion window as follows:

$$desired\_window = \frac{H_{feedback} \times srtt}{size}$$

where size is the packet size in bytes. We multiply with the $srtt$ to transform the rate information into window information and we divide by the packet size to change the units from bytes to packets. A similar approach was adopted in [5] to calculate the slow start threshold in the case of a lost packet. The $desired\_window$ is the new congestion window requested by the network. We do not immediately set the $cwnd$ equal to the desired congestion window because this abrupt change may lead to bursty traffic. Instead we choose to gradually make this change by means of a first order filter. The smoothing gain of this filter depends on the state of the $H_{congestion}$ bit in the acknowledgement received. If this is equal to 1, which indicates congestion in the source destination path, we apply a less aggressive increase policy by decreasing the smoothing rate of incoming bytes entering the queue in one control period divided by the control period. To calculate the number of received bytes,
the router maintains a variable which is incremented with the packet size in bytes, every time the queue receives a packet. When the control timer expires, the link calculates \( y \) by adding the received number of bytes and then dividing with the control period. It then resets the received number of bytes.

The above variables are used to calculate the desired rate \( p \) every control period. In this paper we consider three representative algorithms which calculate the desired sending rate. It is beyond the scope of this paper to describe the rationale behind these algorithms since this is described in detail in the references provided.

1) _Adaptive Congestion Protocol (ACP)_: At each link, the objective is to match the input data rate \( y \) to the link capacity \( C \) and at the same time maintain small queue sizes. To achieve this, we use the algorithm proposed in [14] which updates the desired sending rate \( p \) as follows:

\[
p(k + 1) = Pr_1^C[p(k) + \frac{1}{N(k)}q(k)], \quad p(0) = 0
\]

where

\[
g(k) = k_i(0.99 * C - y(k)) - \frac{1}{d(k)}k_q q(k)
\]

\( k_i \) and \( k_q \) are design parameters, \( d(k) \) represents the average round trip time at time \( k \), \( N(k) \) represents an estimate of the effective number of users utilizing the link at time \( k \) and the projection operator is defined as follows:

\[
Pr_1^C[x] = \begin{cases} 
1 & \text{if } x < 1 \\
C & \text{if } x > C \\
x & \text{otherwise}
\end{cases}
\]

Following the design guidelines given in [14] \( k_i \) and \( k_q \) are chosen to be 0.1587 and 0.3175 respectively.

2) _Queue Length Approach (QLA)_: At each link, the objective is to regulate the queue size \( q \) so that it tracks a reference queue size \( q_{ref} \) chosen by the designer. To achieve the latter, we use a modified version of the algorithm proposed in ([20]).

The algorithm is as follows:

\[
p(k) = Pr_1^C[\frac{C}{N(k)}(q(k) - q_{ref})]
\]

The algorithm applies proportional action.

3) _Direct Max-Min Approach (DMM)_: At each link, the objective is to directly calculate the fair share of each user by dividing the available capacity with the effective number of users utilizing the network ([25]). To achieve the latter we use the following algorithm:

\[
p(k) = Pr_1^C[\frac{0.98 * C}{N(k)} - \frac{1}{d(k)N(k)}q(k)]
\]

Note, that we have introduced a queue size term to ensure that at equilibrium the queue size is equal to zero. In addition we have chosen to allocate 98% of the link capacity. We do this to reserve bandwidth resources which can be used to accommodate statistical fluctuations of the bursty network traffic. This prevents excessive instantaneous queue sizes.

In all three algorithms presented above, the control parameters are updated by estimates of the effective number of users utilizing the link. These estimates are generated by the adaptive law (26). The control period \( T \) which appears in (26) is set equal to the average round trip time and the initial value of the estimates \( \hat{N}_j(0) \) is set equal to 100. We choose this value to ensure a relatively conservative policy when initially updating the desired sending rate. The parameter \( \gamma \) affects the convergence and stability properties of the algorithm and is chosen to be equal to 0.1.

V. PERFORMANCE EVALUATION

We conduct our simulations on the ns-2 simulator. We consider persistent FTP sources, the packet size is equal to 1000 bytes and the buffer size of all links is set equal to the bandwidth delay product. The simulation time is not constant. It varies depending on the round trip propagation delay. We simulate for a sufficiently long time to ensure that the system has reached an equilibrium state. It is highly unlikely that in an actual network the network users will enter the network simultaneously. So, in all scenarios, the users enter the network with an average rate of one user per round trip time.

A. Accuracy of Estimation-Single Bottleneck Link Network

![Fig. 1. Single bottleneck link network used to investigate the ability of the estimation algorithm to track the correct number of users. The number of users changes from 50 to 10, at 30 seconds and from 10 to 30 at 45 seconds.](image-url)
the network is reduced to 10. At 45 seconds, however, 20 additional users enter the network thus causing the number of users to increase to 30.

We simulate the above scenario using the three representative congestion control protocols that we have developed. In Fig. 2 we show the output of the estimation algorithm in each of these protocols. We observe that the time responses are almost identical. In all cases, the estimation algorithm tracks the correct number of users at equilibrium: 50 users at the beginning, 10 after 30 seconds and 30 after 45 seconds. In addition, the algorithm exhibits nice dynamical properties as it generates smooth responses which converge fast to the desired equilibrium with no oscillations or overshoots.

![Fig. 2](image)

**Fig. 2.** The output generated by the estimation algorithm in the three congestion control protocols. The estimation algorithm generates smooth responses and calculates the correct number of users utilizing the link as this changes from 50 to 10, at 30 seconds and from 10 to 30 at 45 seconds.

The above scenario only considers persistent FTP flows. However, real Internet traffic is characterized by relatively few long flows (elephants) and a very large number of short flows (mice). In addition, Internet wires, apart from forward traffic, accommodate reverse traffic which is known to have negative effects on TCP [31]. In order to investigate the performance of the proposed estimation scheme in such realistic scenarios we conduct the following experiment. We consider the network topology of Fig. 1 utilized by 40 persistent sources at the beginning, 20 after 30 seconds and 40 after 50 seconds. In addition we consider short web like flows both in the forward and reverse direction which are of the same type as the persistent flows. The transfer size of these flows is derived from a Pareto distribution with an average of 60 packets. The shape of the distribution is set to 1.35. The inter-arrival times of these flows are derived from a Poisson distribution and on the average 10 new users enter the network every second. This generates a highly dynamic scenario of 1600 on-off sources in both the forward and reverse direction. The estimates generated by the estimation algorithm are shown in Fig. 3 (a) and are compared with the actual number of users which are also shown in the figure.

We observe that despite the dynamic environment, the estimation scheme manages to estimate to a very good extent the actual number of users utilizing the network. Slight underestimates are recorded which are due to some sources completing their transfers in less than one round trip time. These sources do not have sufficient data to comply with the desired sending rate advertised by the bottleneck link and so on the average they send data at a smaller rate. This causes the output of the estimation algorithm to converge to values which lie in between the number of persistent sources and the actual number of sources. As one might expect, the deviation from the actual number of sources becomes larger as the number of small flows increases.

Non responsive traffic such as UDP, is a significant part of Internet traffic and it is thus important to investigate its effect on the performance of the proposed estimation scheme. We consider the previous scenario which incorporates responsive long and short flows in both the forward and reverse direction, and we introduce two additional UDP sources which inject CBR traffic in the forward direction. The sending rate of each source is set to 20Mbits/sec. Following the concepts introduced in section II one can easily infer that the addition of non responsive flows causes the effective number of users to become equal to $\frac{C}{\tau} N$, where $C$ is the bandwidth of the bottleneck link, $\tau$ is the rate of UDP traffic and $N$ is the actual number of users utilizing the network. In Fig. 3 (b) we compare the output of the estimation algorithm in the three congestion control protocols that we have developed with the effective number of users. Similar to the behavior in the previous experiment, the output of the estimation algorithm manages to converge to a value close to the effective number of users, as required. The slight underestimates are again due to some sources completing their transfers in less than one round trip time.

![Fig. 3](image)

**Fig. 3.** The output generated by the estimation algorithm in the presence of short web like flows in both the forward and reverse direction. Note that the estimates generated by the ACP scheme are almost identical to the ones generated by the DMM scheme and are thus not clearly identified in the figure.

The model of equation (9) in the single bottleneck link case was based on a mathematical model of a synchronized network. Synchronization is a rather strong assumption and our objective in this section is to relax this assumption and evaluate through simulations the performance of the estimation algorithm in a non-synchronized network. We consider the single bottleneck link network of Fig. 1 utilized by 100 users. In order to implement a non-synchronized network we consider access links with different propagation delays. The propagation delay of the access links of the first 10 users is equal to 20msec and every 10 users the propagation delay of the access links increases by 25msec. In this way we establish users with round trip propagation delays in the range 80msec to 490msec. We then consider the following dynamic scenario of these 100 users entering and leaving the network at different times: 80 users originally utilize the single bottleneck link
network and at 60 seconds, 30 of these users stop sending data simultaneously. So the number of users utilizing the network is reduced to 50. At 105 seconds, however, 20 additional users enter the network thus causing the number of users to increase to 70. We simulate the above scenario using the three congestion control protocols that we have developed. In Fig. 4 we show the output of the estimation algorithm in each of these protocols. We observe that in all cases the algorithm converges to the correct number of users utilizing the network thus demonstrating that our analytical findings which were based on the assumption of synchronized transmission are also true in asynchronous networks.

B. A Multi-Link Network

As in the single bottleneck link case, in the multiple link case, Theorem 1 predicts that the estimation error is of the order of the noise error. The size of this estimation error and thus the accuracy of estimation in the case of multiple congested links is investigated in this section. We consider the parking lot topology shown in Fig. 5.

The network consists of 8 links which are connected in series. All links have a bandwidth of 155Mbits/sec except link 4 which has a bandwidth of 80Mbits/sec. The propagation delay of all links is set equal to 15msec. 20 users utilize the network by traversing all 8 links. Moreover, each link in the network is utilized by an additional 20 users which have single hop paths as shown in Fig. 5. This way, all links in the network are bottleneck links and link 4 is the single bottleneck link for the 20 users which traverse the whole network. Note, however, that link 4 is the bottleneck link for all 40 users traversing the link, whereas the rest of the links only bottleneck 20 out of the 40 users that utilize them. In the above scenario, if the users are allocated their max-min fair share at equilibrium, equation (17) predicts that the effective number of users at all links except link 4 is equal to 27, whereas the effective number of users at link 4 is equal to the total number of users which is equal to 40.

We simulate the above scenario using the three congestion control protocols that we have developed. The output of the estimation scheme in all three protocols eventually converges to an effective number of users whose value for each link is shown in Fig. 6. We observe that at all links except link 4, the output converges to the predicted value which is equal to 27. At link 4 we observe that ACP and DMM estimate the correct value which is equal to 40 whereas QLA estimates a slightly smaller value which is close to 39.

C. A more Complex Scenario

In this section we evaluate the performance of the proposed estimation algorithm in a more complex scenario shown in Fig. 7. The network represents a section of the Abilene network which is now a part of the Internet2 network. The propagation delays of the links are shown in Fig. 7 and their bandwidth is set to 155Mb/s. The network is utilized by 100 users through access links whose bandwidth is set equal to 155Mb/s and their propagation delay is equal to 20msec. 20 users access the network through router R4 sending data to hosts attached on R2, 20 users access the network through router R4 sending...
data to hosts attached on R2 and 40 users access the network through R0 with half of them sending data to hosts attached on R3 and the rest to hosts attached to R5. In such a scenario, links R1-R2 and links R2-R3 constitute bottleneck links for some flows and their effective number of users are 30 and 60 respectively.

We simulate the above scenario using the three congestion control protocols and in Fig. 8 we show the estimates generated by the three protocols in the two bottleneck links. We observe almost perfect matching between the estimated values and the effective number of users. In order to evaluate the performance of the protocol in the presence of persistent reverse traffic we introduce 20 additional users which access the network through R5 and send data to hosts attached on R2. In Fig. 9 (a) we show the estimates generated by the three protocols on link R2-R3 through which the reverse traffic traverses. We observe that slight overestimates are generated relative to the true number of users traversing the link. The reason is that the reverse traffic generates acknowledgement traffic in the forward path. The acknowledgement traffic is unresponsive and is thus similar in nature to UDP traffic. It was shown in one of the previous subsections that the effect of UDP traffic is to increase the value of the effective number of users. This happens in this case as well and is reflected in the estimates generated by the proposed algorithm.

In order to render our experiment even more realistic we introduce short web-like flows in the forward and reverse direction of the path R1 to R5 via R2. In Fig. 9 (b) we show the actual number of flows traversing link R2-R3 and the estimates generated by the three protocols. We observe very good agreement between the two. As in our previous experiments involving short flows, slight underestimates of the actual number of users are observed which are due to some sources completing their transfers in less than one round trip time.

### D. Comparison with other Estimation Schemes

In this paper we propose an estimation scheme which is based on online parameter identification techniques and does not require maintenance of per flow states within the network. Estimation schemes which appear in the literature and also do not require maintenance of per flow states within the network, are based on point wise division in time. The question that arises is the following: What are the disadvantages of these estimation schemes which necessitate the development of a new one. In this section we demonstrate that estimation schemes which are based on pointwise division in time are not always effective. In two of the congestion control protocols that we develop in this work, they generate oscillatory responses, they lead to erroneous estimates and they may even lead to instability.

We compare the proposed estimation scheme with two other proposals: the method proposed in [25] for ATM networks, where the computed rate is directly divided with the desired sending rate $\hat{N} = \frac{\hat{R}}{C}$ and the method proposed in [12] for the RCP protocol, where the link capacity is divided with a delayed version of the desired sending rate, $\hat{N} = \frac{C}{p(t-\tau)}$.

We refer to the proposed estimation scheme as the online parameter identification (OPI) estimation scheme, we refer to the scheme used in ATM networks as the pointwise division in time (PDT) estimation scheme and we refer to the estimation scheme used in the RCP protocol as the RCP estimation scheme. We use these algorithms to estimate the number of flows in the three protocols that we have developed and we compare the estimation output with the output of the scheme that we propose in this paper. Our simulations indicate that although in ACP all methods exhibit similar performance, in the other two protocols the proposed estimation scheme exhibits superior performance as it avoids oscillations and achieves good tracking of the unknown parameter.

We conduct our simulations using the network topology shown in Fig. 1 and we consider 50 users gradually entering the network. We first conduct simulations with ACP users. In each simulation, we estimate the number of users utilizing the bottleneck link using a different estimation scheme. The results are shown in Fig. 10 (a). It is obvious that within the ACP protocol, the three estimation schemes generate smooth responses which converge to the correct number of users. The OPI scheme, however, combines a number of fine attributes relative to the other approaches such as faster convergence compared to the PDT scheme and significantly less overshoots compared to the RCP scheme.

The next set of simulations involves users which use the QLA protocol to control the flow of data. In each simulation we implement a different estimation scheme. The results are shown in Fig. 10 (b). We observe that the estimation scheme proposed in this paper, again generates smooth responses.
which converge to the correct number of users. The other two schemes generate oscillatory responses with the effect being most vivid and erratic in the case of the RCP estimation scheme. In addition, the response of the PDT scheme is not only oscillatory but its mean value is larger than the true number of users utilizing the network. We have also tested the three estimation schemes in the DMM protocol. We have observed that the only estimation scheme that manages to stabilize the protocol is the one proposed in this paper. The above results demonstrate that although estimation schemes which rely on pointwise division in time are successful in some protocols, they lack robustness and they fail to fulfill the design objectives in two of the protocols that we develop. The estimation scheme that we propose, which is based on online parameter identification techniques, overcomes this problem and manages to perform well in these protocols.

E. Protocol Evaluation

The main reason that we need to design an estimation scheme, is to use it to develop effective congestion control protocols. In this section we demonstrate through simulations that the three representative protocols that we have developed using the proposed estimation scheme satisfy key design requirements of congestion control protocols.

We conduct our study by considering the single bottleneck link network of Fig. 1 and the parking lot topology of Fig. 5. We first consider the single bottleneck link network of Fig 1 and the simulation scenario presented in section V-A. We evaluate the performance of these protocols in terms of the network utilization achieved, the queue size observed at the bottleneck link and the congestion window of the network users. In Fig. 11 we show the time response of the utilization achieved by the three protocols at the bottleneck link. We observe that in all cases the utilization converges fast to a value close to 1 (QLA achieves 100% utilization whereas DMM and ACP achieve 99% utilization). When the 40 users leave the network, the flow of data suddenly decreases thus causing an instantaneous decrease in utilization. However, the system reacts quickly by increasing the sending rate of the remaining users, thus achieving almost full utilization in a short period of time. In Fig. 12 we show the time evolution of the queue size at the bottleneck link for each of the protocols. We observe that ACP and DMM achieve almost zero queue size at equilibrium whereas QLA guarantees that the queue size converges to 100 packets. These observations are consistent with the design objectives of these protocols. The objective of ACP and DMM is to achieve zero queue size at equilibrium, whereas the objective of QLA is to track a reference queue size which is chosen to be equal to a 100. In the transient periods during which new users enter the network, the three protocols exhibit almost identical behavior experiencing instantaneous increases in the queue size. It must be noted that the maximum queue size recorded in the transient periods, increases as the bandwidth delay product increases. However, careful choice of the control parameters at the links and the delayed increase policy that we apply at the sources ensure that these overshoots do not exceed the buffer size and thus do not lead to packet drops. When the 40 users stop sending data at 30 seconds, the queue size experiences undershoots mostly evident in the case of the QLA protocol. This is due to the time it takes for the sending rate of the remaining users to take up the slack created by the users leaving the network.

The last thing we examine is the time responses of the congestion window of a representative number of users. We consider user 1, which utilizes the network throughout the simulation, user 11 which stops sending data at 30 seconds and user 51 which enters the network at 45 seconds. The transient behavior of the other users is very similar to the behavior of
the users that we consider. Due to lack of space we only show the time responses generated by the QLA scheme in Fig. 13. The other schemes generate similar responses. We observe that the protocols achieve smooth responses which converge fast to the desired equilibrium. At equilibrium the network users are assigned the same congestion window value. Since all users experience the same round trip propagation delay, the latter implies that the users are assigned the same sending rate, thus achieving max-min fairness.

Having established that the three protocols work effectively in a single bottleneck link network we now investigate their performance in more complex topologies. We consider the parking lot topology of Fig. 5 and we simulate the scenario described in section B. We use two performance metrics: the utilization achieved at equilibrium and the queue size achieved at equilibrium. The values recorded for each of the congestion control protocols at each link are shown in Fig. 14. Since all links in the network are bottleneck links for some flows, we expect them to be fully utilized. Indeed, we observe that all three protocols achieve almost full utilization at all links. QLA achieves 100% utilization, whereas ACP and DMM achieve 99% utilization. One of the design objectives of the ACP and DMM schemes is to achieve zero queue sizes at equilibrium. We observe that both protocols satisfy this objective to a very good extent as they achieve small queue sizes. The main objective of the QLA scheme, on the other hand, is to track a reference queue size. In our simulations we choose the latter to be equal to 100. We observe, that at all links the protocol achieves excellent tracking of the reference queue size.

1) Scalability Analysis: In this section we investigate the scalability of the three protocols with respect to changing link propagation delays and number of users utilizing the network. In our basic setup we consider 50 users utilizing the single bottleneck link network of Fig. 1. When investigating the scalability of the protocol with respect to a particular parameter, we fix the other parameters to the values of the basic setup and we evaluate the performance of the protocol as we change the parameter under investigation. The performance metrics that we use are the equilibrium utilization at the bottleneck link and the equilibrium queue size of the buffer at the bottleneck link. We do not report packet drops, as in all simulations we do not observe any. In addition, we do not show fairness plots, as in all simulations the network users are assigned the same sending rate at equilibrium, which implies that max-min fairness is achieved in all cases. Effect of Delays: We first evaluate the performance of the three protocols as we change the propagation delays of the links. Any change in the link propagation delay causes a corresponding change in the round trip propagation delay of all source destination paths. We fix the link bandwidths to 155Mbits/s, we fix the number of users to 50 and we consider round-trip propagation delays in the range 10ms-1sec. Plots of the bottleneck utilization and the equilibrium queue size versus the round trip propagation delays are shown in Fig 15. We observe that the three protocols scale well with increasing round trip times. They achieve high network utilization at all bandwidths, with the QLA scheme pushing the system to its limits as it achieves full network utilization (100%). In addition, the queue sizes are found to converge to the desired values at all bandwidths; the QLA achieves equilibrium queue sizes close to 100 and the other two protocols achieve equilibrium queue sizes close to 0.

Effect of the Number of Users We then investigate the performance of the three protocols as we increase the number of users utilizing the single bottleneck link network in Fig. 1. We consider different number of users in the range 1-1000. Plots of the bottleneck utilization and the average queue size versus the number of users are shown in Fig. 16. We observe that up to approximately 800 users the protocols satisfy their design objectives as they achieve high network utilization and
the queue sizes converge to the desired equilibrium values. When however, the network is utilized by more than 800 users the utilization drops to about 80% and the queue sizes are observed to increase significantly. This effect is not due to the inability of the estimation scheme to estimate the correct number of users but rather due to the number of users becoming so large relative to the link bandwidth that the system reaches its limits. As the number of users increases, the fair congestion window assigned to them decreases, thus amplifying the effect of rounding errors caused by the fact that the congestion window can only take integer values. These rounding errors induce oscillations which degrade the system performance. The system reaches its limits when the number of users is so large that the fair congestion window is less than 1. The users cannot assign congestion window values smaller than 1 since this would cause them to stop receiving acknowledgements and thus stop sending new data. Since the users are forced to send data at a rate higher than their max-min share, a significant degradation in performance may be observed. This behavior is common to all window based protocols.

VI. CONCLUSIONS

Our main contribution in this paper is the design of a novel estimation scheme of the effective number of users utilizing a link or multiple bottleneck links, which is based on online parameter identification techniques and is shown to work effectively, outperforming previous proposals. The estimation scheme is designed independently from the control functions of the protocol and is thus universal in the sense that it operates effectively in a number of congestion control protocols. It can thus be successfully used to improve recently proposed congestion control protocols and also to design new ones. In this paper we use the proposed estimation scheme to design three representative Internet congestion control protocols. We demonstrate through simulations that all three representative protocols satisfy key design requirements.

APPENDIX A

PROOF OF STABILITY

Proof:

For clarity of presentation we omit the subscript \( j \). Substituting (18) in (24), we obtain:

\[
\tilde{N}(k+1) = \tilde{N}(k) + \gamma (N^{\text{eff}}(k) p(k-1) + d(k) - N(k)p(k-1)) + \gamma p(k-1)d(k)
\]

\[
= \tilde{N}(k) - \gamma \tilde{N}(k) + \frac{p(k-1)}{1 + p(k-1)} + \gamma p(k-1)d(k)
\]

\[
+ \gamma N^{\text{eff}}(k) p(k-1) \frac{1}{1 + p^2(k-1)}, \quad \tilde{N}(0) = \tilde{N}_0
\]

Let \( \tilde{N}(k) = \tilde{N}(k) - N^{\text{eff}}(k) \). Substituting the latter in equation (36) yields:

\[
\tilde{N}(k) - \gamma \tilde{N}(k) + \frac{p(k-1)d(k)}{1 + p^2(k-1)} = \tilde{N}(k) - \gamma \tilde{N}(k) + \frac{p(k-1)d(k)}{1 + p^2(k-1)}
\]

\[
= \tilde{N}(k) - \gamma \tilde{N}(k) + \frac{p(k-1)d(k)}{1 + p^2(k-1)}
\]

Let \( \lambda(k) = 1 - \gamma \frac{p(k-1)}{1 + p^2(k-1)}, \quad \sigma(k) = \gamma \frac{p(k-1)}{1 + p^2(k-1)} \). The above equation then becomes:

\[
\tilde{N}(k+1) = \lambda(k) \tilde{N}(k) + \sigma(k)d(k), \quad \tilde{N}(0) = \tilde{N}_0
\]

Due to equation (14), \( p(k-1) > 0 \). Since \( \gamma < 1 \) and \( p(k-1) > 0 < \gamma \frac{p(k-1)}{1 + p^2(k-1)} < 1, \forall k \) and so \( \alpha > 0 \) such that \( \forall k, 0 < \lambda(k) < \alpha < 1 \). In addition, \( \beta \) such that \( \forall k, 0 < \sigma(k) < \beta < 1 \). Let \( h = \max(\alpha, \beta) \).

From (38) it can be deduced that:

\[
\tilde{N}(1) = \lambda(0) \tilde{N}_0 + \sigma(0)d(0)
\]

\[
\tilde{N}(2) = \lambda(0)\lambda(1) \tilde{N}_0 + \lambda(1)\sigma(0)d(0) + \sigma(1)d(1)
\]

\[
\tilde{N}(3) = \lambda(0)\lambda(1)\lambda(3) \tilde{N}_0 + \lambda(2)\lambda(1)\sigma(0)d(0) + \lambda(2)\sigma(1)d(1) + \sigma(2)d(2)
\]

Since \( \lambda(k) < h, \sigma(k) < h, \) and \( |d(k)| < v_0 \) it follows that:

\[
|\tilde{N}(3)| \leq \tilde{N}_0 h^3 + v_0(h^3 + h^2 + h)
\]

and so

\[
|\tilde{N}(k)| \leq \tilde{N}_0 h^k + v_0 \sum_{i=1}^{k} h^i
\]

\[= \tilde{N}_0 h^k + v_0 \frac{h(1-h^k)}{1-h} = \epsilon_k + v_0 c_0
\]

where \( \epsilon_k = \tilde{N}_0 h^k \) and \( c_0 = \frac{h(1-h^k)}{1-h} \).

REFERENCES


