Recovery of Information Lost by the Least Squares Estimation in Real-Time Network Environment

Hengky Susanto and ByungGuk Kim

Department of Computer Science
University of Massachusetts at Lowell
{hsusanto, kim}@cs.uml.edu

Abstract — Feedback mechanism based algorithms are frequently used to solve network optimization problems. These schemes involve users and network exchanging information to achieve convergence towards an optimal solution. However, in implementation, these algorithms do not guarantee messages will be delivered to the destination, especially when network congestion occurs, which in turn often results in packet drops. This condition may lead to algorithm failing to converge. To prevent this, we propose least square (LS) estimation algorithm to recover the missing information when packets are dropped from the network. Our results from several scenarios demonstrate that LS estimation can provide the convergence for feedback mechanism based algorithm.

Index Terms **- congestion control, resource management, QoS.**

I. INTRODUCTION

Dynamic pricing has been a widely adopted means to overcome network congestion. The relationship between pricing and traffic management is often formulated into the Network Utility Maximization (NUM) framework [1,2,3,4,5,12,15,19]. A common approach to solve NUM problems involves feedback mechanism that converges to an optimal solution. That is, the network adjusts the price to control the level of congestion and users adapt their transmission rate according to the price decided by network [3,4,5]. One major shortcoming of this approach is the reliance on explicit communication of both price and acknowledgement (ACK) notifications. During network congestions, price and ACK notification packets are susceptible to packet loss, or dropped from the network. A proper ACK response is particularly important to determine the appropriate price update intervals. It is because without ACK notification, the exact time interval to broadcast price update is difficult to estimate. In our previous work [13], we show that premature price update leads to algorithm oscillation, and delayed update leads to slower convergence.

To address this problem, we propose a solution based on *Least Square (LS) estimation* [8] algorithm to predict the appropriate time interval for the next update. This technique is based on aggregated historical inputs of network price and update intervals. Thus, network does not need to see the overall picture of what is happening in the network, because it leverages on information accumulated over time to make the estimation. In this paper, we incorporate LS estimation algorithm into feedback mechanism-based solution to resolve situations whenever there is information lost. LS estimation can be resolved in linear time and requires a small memory space, yet it provides effective recovery of information in most situations.

There has been much research done on feedback mechanism, such as: subgradient based algorithms that are often employed to resolve congestion or bandwidth provisioning problem in NUM [2,3,14,15,18], Explicit Congestion Notification (ECN) [6][7], and variations of feedback mechanism schemes on congestion control [10,11]. However, these literatures do not address the time interval of notification update. It is because, in NUM, the information on price and transmission rate is assumed to be available instantaneously. Moreover, the proposed solutions in [6,7,10,11] are TCP/IP based, where the network relies on users to manage their own transmission rate through some control methodology, such as the congestion window. Thus, precise update interval is not necessary in these approaches. Authors of [17] claim that when the estimator of gradient based algorithm is biased, the solution can still converge to a contraction region around the optimal point, even without complete information. In other words, depending on the choice of step size, the algorithm still converges even with missing information. However, determining the appropriate estimator relies on the assumption that the interval time for update is constant, which may not hold when the condition of network traffic fluctuates.

In this paper, we address this gap in interval notification update by providing a technique to estimate appropriate interval, especially when there is a lack of information caused by packet loss. We begin our proposal with problem formulation in section II, where we introduce NUM, subgradient based algorithm, and message exchange mechanism. Then, we present our major contributions: a mathematical model for interval update for price notification and the design for LS estimation algorithm in section III and IV respectively. The simulation results are presented and discussed in section V, followed by concluding remarks in section VI.

II. PROBLEM FORMULATION

We begin with a discussion on NUM to support real-time traffic. Consider a network with a set of links L, and a set of link capacities C over the links. Given a utility function $U_s(x_s)$ of user s with the allocated bandwidth of x_s, the NUM formulation becomes

$$maximize \sum_{s\in S} U_s(x_s) \qquad (P)$$

$$s.t.\ \ Ax \le C, over\ \ x \ge \bar{0},$$

where S and A denote sets of users and routing paths, respectively, and $\bar{0}$ is a vector of zeros. A route r consists of a series of links l such that $A_{lr} = 1$ if $l \in r$ and $A_{lr} = 0$, otherwise. The user utility function is defined as $U_s(x_s) = \frac{1}{(1+e^{-x_s})}$. The NUM formulation is solved by the Lagrangian method. Typically, a dual problem to the primal problem of (P) is constructed as follows. $L(x,\lambda) = \sum_{s\in S} U_s(x_s) - \lambda^T(C - Ax)$. The dual problem D of (P) is then defined as $\min D(\lambda), s.t\ \ \lambda \ge \bar{0}$, where the dual function $D(\lambda) = \max_{\bar{0}\le x\le x^{max}} L(x,\lambda)$. Subgradient method algorithm [3] is described as follows. Given, $\lambda_s = \sum_{l\in r} \lambda_l$,

$$x_s(\lambda_s) = \underset{0 \le x \le x^{max}}{\arg\max} \big(U_s(x_s)\big). \qquad (1)$$

$$\lambda^{t+1} = \big[\lambda^t - \sigma^t\big(C - A\,x(\lambda^t)\big)\big]^+, \qquad (2)$$

where $x(\lambda^t)$ denotes the rate allocation at λ^t at time t, for $\lambda^t \ge \lambda^{min}$ and λ_l denotes the price at link l. price λ^{min} can be interpreted as the network's operation cost [16]. Also, σ^t denotes the step size to control the tradeoff between a convergence guarantee and the convergence speed [3,5]. The feedback loop in the pair of equations (1) and (2) allows users to adjust their transmission rate according to the price λ_s by solving (1), and for the network to control the amount of traffic flow by adjusting the price until λ_l^t converges to a solution by solving (2).

In practice, message exchanges between users and network in subgradient based algorithm is initiated when congestion is detected at link l. The network (router at link l) broadcasts the price notification to users whose flow traverse through it. Then, users respond with an ACK message upon reception of the price notification. The mechanism is repeated until congestion is resolved. However, there is no guarantee packets will reach intended destinations. As a result, network may not receive ACK packet from users and not able to determine the appropriate time to broadcast the next price notification.

III. INTERVAL UPDATE

Information delay is unavoidable in any system that employs feedback mechanism. Thus, we employ Little theorem [9] to model the delay for the price and ACK notifications to reach their destination. First of all, the average delay d_s between the origin of the notification and the destination can be estimated as follows.

$$d_s = \sum_{l\in r(s)} \left(\frac{x_l^{min}}{x_l\big(x_l - x_l^{min}\big)} + \frac{1}{x_s^{min}}\right), \qquad (3)$$

where x_l^{min} is the minimum time required to process notification packet, x_l is the amount of bandwidth allocated on link l along path $r(s)$ associated to user s. Here, bandwidth x_l^{min} can be interpreted as user minimum requirement for bandwidth allocation. So, the time required d_s^{info} for the price notification packet to reach destination, user s, can be estimated by $d_s^{info} = d_s + \sum_{l\in r(s)} d_l^{propa}$, where d_l^{propa} denotes the negligible propagation delay at link l. Thus, with this estimation, the longest delay d_l^{max} from l to every user that uses link l is

$$d_l^{max} = max\left(d_s^{info} | \ s \in S(l)\right), \qquad (4)$$

where $S(l)$ is a set of users who use link l. The interval length between price updates can be bounded with Round Trip Time RTT_l by utilizing (3) and (4), such that $RTT_l \ge 2\,d_l^{max}$. So the maximum interval length RTT_{max} can be estimated as

$$RTT_{max} = \max(\{RTT_l^t\}) + \epsilon, \qquad \forall s \in l, \ (5)$$

where ϵ is a positive constant error term and RTT_l^t denotes RTT_l over a period time of $t \to \infty$. Hence, the time for network to send price notification and receive ACK notification can be estimated by RTT_{max}, which also implies the price update notification interval can be estimated by RTT_{max}. From this point onward, RTT_{max} is referred as RTT. Next, we investigate the relationship between network price and RTT.

Proposition 1. RTT fluctuation corresponds to price change.

Proof: *Case 1*: $x_s > x_s^{min}$. By using Little theorem [9], we have a delay function $(x_s) = \frac{\rho_s}{x_s(1-\rho_s)} = \frac{1}{x_s - x_s^{min}}$, where α_s is the arrival rate on l. Since function $d(x_s)$ is influenced by ρ_s, where $\rho_s = \frac{x_s^{min}}{x_s}$ and $\rho_s < 1$, and the rate allocation x_s is adjusted according to the network price λ_s by solving equation (1) and (2). The changes in price λ_s influences ρ_s, which affects $d(x_s)$. Thus, in this case, the fluctuation of RTT of (5) corresponds to price change.

Case 2: $x_s \le x_s^{min}$. Whenever rate allocation is below the required minimum bandwidth, x_s^{min} is adjusted to x_s, such that $x_s^{min} = x_s$, because it is physically impossible to transmit more data than the available bandwidth. It is because as $\rho_s \to 1$, the number of packets in the system approaches to infinity [9]. Thus, in this case, the delay function is formulated as $d(x_s) = \frac{B_s}{x_s}$, where B_s denotes the buffer size allocated for user s or the maximum number of packets that the buffer can hold. Observe in $d(x_s)$ that the changes in λ_s also influences x_s, which in turn affects delay function $d(x_s)$ because rate allocation x_s is adjusted according to the network price λ_s by solving equation (1) and (2). ■

Proposition 1 describes the relationship between network price and delay such that any changes in price will impact the length of RTT. It also shows that the change in RTT is proportional to the change in pricing. For this reason, pricing is employed as one of the factors to estimate RTT in LS estimation algorithm.

IV. LEAST SQUARE ESTIMATION

In this section, we introduce LS estimation algorithm to predict RTT on the basis of aggregated historical

information of network price and RTT. The section on RTT includes a discussion on predicting user demand for bandwidth using the same approach. Let h be the observed time interval between two iterations, so

$$h = RTT + \epsilon. \quad (6)$$

Furthermore, the price λ^t is merely a weighted price of past *observed* interval h^t over t iteration, which is adjusted over time by appropriate updates between source and network.

$$\lambda^t = h^t \; w\,. \quad (7)$$

Let $\hat{h}^{(t+1)}$ be the estimate time interval between two iterations. Given the past history of t, observed intervals h and network price λ, as $t \to \infty$, the network estimates the weight distribution w, which is used to determine $\hat{h}^{(t+1)}$, especially when the price notification packet that corresponds to the longest path is lost. The estimated time interval is defined as follows.

$$\hat{h}^{t+1} = \frac{\lambda^{t+1}}{w}. \quad (8)$$

The model above is formulated into an LS problem by introducing an error term ε^t so that $\lambda^t = h^t.w + \varepsilon^t$. In a matrix form $\vec{\lambda}^t = \vec{h}^t.w + \vec{\varepsilon}^t$, where

$$\vec{\lambda}^t = [\lambda^t, \lambda^{t-1}, \lambda^{t-1}, \dots, \lambda^1]^T,$$
$$\vec{h}^t = [h^t, h^{t-1}, h^{t-2}, \dots, h^1]^T,$$
$$\vec{\varepsilon}^t = [\varepsilon^t, \varepsilon^{t-1}, \varepsilon^{t-2}, \dots, \varepsilon^1]^T.$$

The LS estimator w is based on the minimization of the scalar cost function $J(w)$, given by $J(w) = \frac{1}{2}(\vec{\varepsilon})^T \vec{\varepsilon}$, we have $J(w) = \frac{1}{2}\left[\vec{\lambda}^t - w.\vec{h}^t\right]^T\left[\vec{\lambda}^t - w.\vec{h}^t\right]$. The LS estimator, w, satisfies the condition of first order derivative of $J(w)$, $\frac{\partial J(w)}{\partial w} = \left[(\vec{h}^t)^T \; \vec{h}^t\right] w - \vec{\lambda}^t \left(\vec{h}^t\right)^T = 0$. Or $\left[(\vec{h}^t)^T \; \vec{h}^t\right] w = \vec{\lambda}^t \left(\vec{h}^t\right)^T$. Since $\vec{\lambda}^t$ and $\vec{h}^t$ are $m \times 1$ matrix, where

$$\vec{\lambda}^t \approx \vec{h}^t.w$$
$$\begin{bmatrix}\lambda^1\\ \lambda^2\\ \vdots\\ \lambda^t\end{bmatrix} \approx \begin{bmatrix}h^1\\ h^2\\ \vdots\\ h^t\end{bmatrix}[w]\,,$$

the solution for w is obtained as follows.

$$w \approx \left[(\vec{h}^t)^T \; \vec{h}^t\right]^{-1} \vec{\lambda}^t \; (\vec{h}^t)^T$$
$$= \frac{\sum_{t=0}^{m} h^t \; \lambda^t}{\sum_{t=0}^{m} (h^t)^2}\,. \quad (9)$$

Eq. (9) indicates that network does not require a large memory space to store information. Instead of allocating space for each h^t and λ^t, as $t \to \infty$, network only needs a space to store each value of $\sum_{t=0}^{m} h^{(t)} \lambda^t$, $\sum_{t=0}^{m} (h^t)^2$, and w_s , which is $O(1)$ of memory space for each value. When new information becomes available, network simply aggregates the new information to (9). Next, we evaluate the effectiveness of LS estimation. From this point onward, s refers to user with the farthest distance from the origin of price notification.

Proposition 2. w_s converges as $t \to \infty$.

Proof: Assuming there exists an optimal solution for the dual problem D, such that price λ_s^t and rate allocation x_s^t converges at time t, as $t \to \infty$. With this assumption, $\rho_s^{(t)}$ also converges, where $\rho_s^t = \frac{x_l^{min}}{x_s^t}$, for $l \in r(s)$. Since delay function d_l^{info} in eq. (4) is influenced by x_s^t through ρ_s^t, then d_s^{info} converges, which implies d_l^{max} in eq. (4) also converges. Since time interval h^t is obtained by solving (5) and (6), then h^t also converges if and only if d_l^{max} converges. Thus, given the relationship between h_s^t and λ_s^t in (8), w_s also converges as $t \to \infty$. ■

Proposition 2 implies that as $t \to \infty$, if there exists a solution to the dual problem D, then the predicted outcomes should also converge to a value, which is a similar behavior to subgradient algorithm. Additionally, in order to save memory space, instead of estimating the RTT of every user in l, network only needs to keep track the longest RTT of every link, which results in $O(|L|)$ memory space.

The gap between the estimate time interval $\hat{h}_s^t$ and the actual time interval h_s^t can be summarized as $\Delta h_s^t = \left|h_s^t - \hat{h}_s^t\right|$. Subsequently, given h_s^i and $\hat{h}_s^i$, for $0 \le i \le t$, Δh_s^{t+1} can be minimized, such that $h_s^{t+1} \approx \left|\Delta h_s^{t+1} - \hat{h}_s^{t+1}\right|$.

$$\Delta \bar{h}_s^t = \frac{1}{t}\sum_{t=0}^{\infty} \Delta h_s^t = \frac{1}{t}\sum_{t=0}^{\infty} \left|h_s^t - \hat{h}_s^t\right|.$$

Corollary 1. $\hat{h}_s^{t+1}$ converges as $t \to \infty$.

Proof: As shown in Proposition 2 that w_s converges if there exists an optimal solution for the dual problem, such that λ_s^t converges, as $t \to \infty$. Since $\hat{h}_s^{t+1}$ is obtained by solving (8), so $\hat{h}_s^{(t+1)}$ also converges as $t \to \infty$.■

Proposition 3. $\lim_{t\to\infty} \frac{\left|\Delta \bar{h}_s^t - \hat{h}_s^{t+1}\right|}{h_s^{t+1}} = 1$

Proof. First of all, we show that $\Delta \bar{h}_s^t$ converges, as $t \to \infty$. By proposition 2 and corollary 1, if there exists an optimal solution to the dual problem D, then h_s^t and $\hat{h}_s^t$ also converge, which also implies that the mean value $\Delta \bar{h}_s^t$ also converges. For this reason, $\left|\Delta \bar{h}_s^t - \hat{h}_s^t\right| \to h_s^t$, as $t \to \infty$. Hence, we conclude that

$$\lim_{t\to\infty} \frac{\left|\Delta \bar{h}_s^t - h_s^t\right|}{h_s^t} = \lim_{t\to\infty} \frac{\left|\Delta \bar{h}_s^t - \hat{h}_s^{t+1}\right|}{h_s^{t+1}} = 1\,. \quad ■$$

Proposition 3 shows that when there is sufficient information, the gap between the estimation and the actual RTT can be predicted, which also means $\Delta \bar{h}_s^t$ can be minimized. To minimize the gap, we consider two cases: first, when the gap $\Delta \bar{h}_s^t$ between the estimate and the actual value at time t is too large and error correction to minimize $\Delta \bar{h}_s^t$ may be necessary. Second, when the gap $\Delta \bar{h}_s^t$ is too small, then it may not be necessary to perform an error correction. Observe that $\lim_{t\to\infty} \Delta \bar{h}_s^t = \lim_{t\to\infty} \frac{\sum_{t=0}^{\infty} \Delta h_s^t}{t} = 0$. Let ε_s denote a positive constant decided by the network. The

error correction is performed only when $\Delta\bar{h}_s^t > \varepsilon_s$. The objective of error correction is to

$$minimize\ |\hat{h} - \Delta h|, \text{over } \hat{h}, \Delta h > 0.$$

Any methodology for *estimation error minimization* in real-time environment has to be executed quickly without requiring heavy computing resources and a large memory space. It is because higher computation and memory space also mean higher overhead cost.

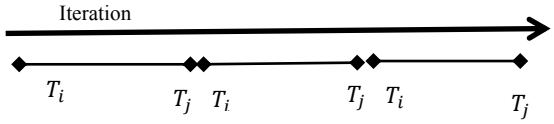

Fig. 1. Iteration Interval.

Let T_i be the initial iteration when Δh is observed and T_j is the last iteration before the network stops observing Δh, for $i < j$, as depicted in figure 1. The mean value of Δh between iteration $[T_i, T_j]$ for s with the longest RTT is defined as follows.

$$\Delta h_s^{|T_j - T_i|} = \frac{\sum_{t=T_i}^{T_j} \Delta\bar{h}_s^t}{|T_j(s) - T_i(s)|}. \quad (10)$$

The reason for computing a new mean value of Δh in every interval of $|T_j - T_i|$ iterations is because $\Delta h_s^{|T_j - T_i|}$ is sensitive to the gradient of RTT. On the other hand, $\Delta h_s^{|T_n - T_0|}$, for $n \to \infty$, is less sensitive to the gradient. After network computes estimated $\hat{h}_s^{T_{j+1}}$ at T_{j+1}, $\hat{h}_s^{T_{j+1}}$ is adjusted according to this procedure.

$$\hat{h}_s^{T_{j+1}} = \begin{cases} \hat{h}_s^{T_{j+1}} - \Delta h_s^{|T_j - T_i|}, & h_s^{|T_j - T_i|} < \hat{h}_s^{|T_j - T_i|} \\ \hat{h}_s^{T_{j+1}} + \Delta h_s^{|T_j - T_i|}, & h_s^{|T_j - T_i|} > \hat{h}_s^{|T_j - T_i|}, \\ \hat{h}_s^{T_{j+1}} \quad , & h_s^{|T_j - T_i|} \approx \hat{h}_s^{|T_j - T_i|} \end{cases} \quad (11)$$

where $h_s^{|T_j - T_i|}$ and $\hat{h}_s^{|T_j - T_i|}$ are the mean values of h and $\hat{h}$ respectively between iteration $[T_i, T_j]$. Furthermore, $T_j = T_i + \Gamma$, where Γ denotes a positive variable decided by the network and $\Gamma \leq$ RTT. In the subsequent interval, the next $T_i = T_{j+1}$. However, if the network does not receive the ACK notification packet by time $T_{j'}$, where $T_{j'} = T_i +$ RTT, then $T_j = T_{j'}$. Then $T_j = T_{j'}$ when $T_i + \Gamma > T_{j'}$. Otherwise, $T_j = T_i + \Gamma$. In other words, the ACK notification packet can be considered lost when it does not arrive after the last recorded RTT.

In addition to estimating RTT, we extend our study to estimate user demand for bandwidth by using LS estimation. This allows us to further evaluate the performance of the estimation algorithm. The problem of bandwidth estimation can be reformulated by modifying equation (8) to $\lambda^{(t)} = x^{(t)}\, w$. The solution for w_x is obtained with $w_x \approx \frac{\sum_{t=0}^{m} x^t \lambda^t}{\sum_{t=0}^{m} (x^t)^2}$, where x^t is the observed rate allocation at time t. Next, the estimated rate allocation $\hat{x}^{t+1}$ is processed according to $\hat{x}^{t+1} = \frac{\lambda^{t+1}}{w_x}$. Thus, the aggregated estimated flow on link l at $(t+1)$ is $\sum_{s \in l} \hat{x}_s^{t+1}$. Thus, this allows network to have a better picture of user demands.

V. DISCUSSION AND SIMULATION

In this section, we present the performance of estimated RTT in various failure rates in the network below (figure 2) that is shared by three users (user 0, 1, and 2). Each link has a capacity of 10 and each user initially transmits data at 10 units per second. Thus, link BC and CD become congested. The network congestion is resolved with subgradient based algorithm. Since the behaviors of three users are identical, simulation results only focus on user 0.

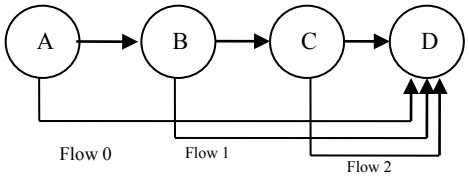

Fig. 2. Parking Lot Topology.

In this scenario, notification packet loss is introduced at different rates: Every 50^{th},30^{th}, and 5^{th} iteration. During the occurrence of packet loss, network predicts the next RTT value according to the previous actual RTT. Furthermore, the simulator randomly decides whether ACK or price notification packets are dropped. For the purpose of analysis, the entire packets of the selected notification are dropped to make the inaccuracy more visible. Whichever notification packets are dropped, the network will not receive updates from users.

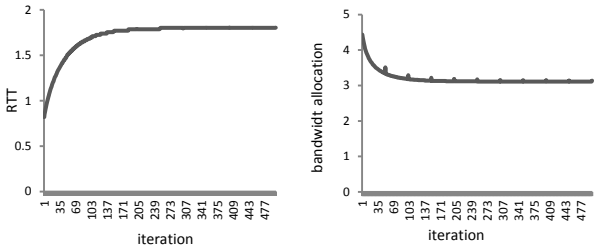

Fig. 3.a and 3.b: Failure rate of every 50^{th} iteration.

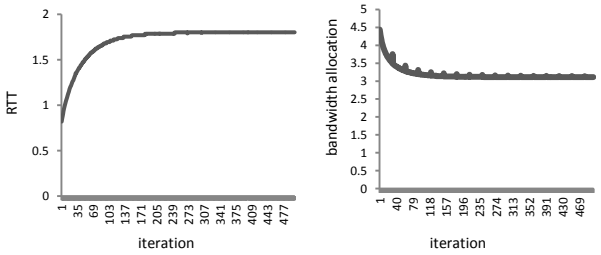

Fig. 4.a and 4.b: Failure rate of every 30^{th} iteration.

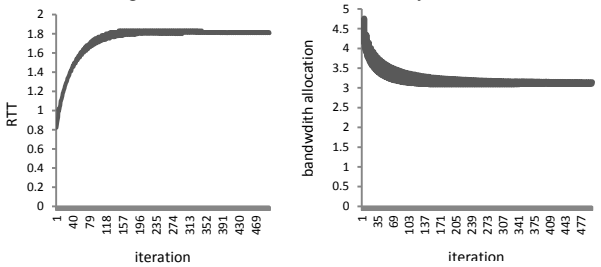

Fig. 5.a and 5.b: Failure rate of every 5^{th} iteration.

The simulation begins with a failure rate at every 50^{th} iteration, where the network must predict the RTT at every 50^{th} iteration as depicted in figure 3.a. Notice that the RTT converges smoothly even with notification packet loss in every 50^{th} iteration. This means the estimation value falls on the line of the actual value. In other words, the network makes a close estimation to actual RTT. Furthermore, figure 4.a and 5.a show that as the number of notification packets loss increases, the RTT convergence becomes less smooth and the line in the graph grows thicker. The line spikes whenever there is a gap between the actual and the estimated RTT: a larger gap leads to a sharper spike, which causes the line to be thicker. Thus, the thickness of the line in these graphs indicates the estimation accuracy in predicting RTT when there is packet loss. This phenomenon becomes more noticeable when there is packet loss at every 5^{th} iteration, as shown in figure 5.a. These results demonstrate that estimation degrades as more information is missing and there is less actual data to compare the estimation with. However, despite the estimation degradation, the algorithm is still able to achieve convergence, especially when the estimation is computed in real time setting.

To further the analysis of the performance of LS estimation, we extend our experiment to estimate user request for bandwidth allocation. In this simulation, user demand for bandwidth is included in the ACK packet. This information is used by network to determine how much bandwidth should be allocated to each user. As shown in figure 3.b, the line spikes at every interval of 50^{th} iteration. This is caused by a discrepancy between the corrected estimated value and the actual value. Notice that the spike subsides as the algorithm converges, as the demand for bandwidth stabilizes and the estimation becomes more accurate. Furthermore, as the rate of packet loss increases, the bandwidth demand line grows thicker and the spikes taller (figures 4.b and 5.b). In other words, as there is lesser information about users, estimation of demand for bandwidth becomes less accurate. For example, in this simulation to mimic a real condition of missing information, we assume that user utility of eq. (1) is not known to network.

We have demonstrated both the strengths and limitations of LS estimation algorithm in addressing missing information caused by network congestion. The simulation results also confirm our theoretical assertion discussed earlier: that algorithm which relies on feedback mechanism can achieve convergence with LS estimation. These simulation results also show that, in most cases, LS algorithm still delivers positive outcomes, specifically, in providing the environment for subgradient algorithm to achieve convergence.

VI. CONCLUSION

We propose to use LS Estimation to resolve the challenge of information loss due to notification packet drops during excessive congestion that leads to algorithm convergence. The estimation carried out by LS estimation techniques minimizes the squared errors between the measured and predicted RTT and bandwidth demand. The LS estimation algorithm requires a solution to a linear equation. The advantages of this approach are: LS estimation algorithm can be computed linearly, does not require high computation complexity, and only requires $O(1)$ memory space for information stored. These characteristics are particularly favorable for system that operates in real-time environment. Our simulations demonstrate that LS estimation algorithm still achieves desirable results in spite of its simplicity.

Additionally, this paper primarily focuses on network actively addressing information loss. It is because we assume users do not have sufficient information on network traffic intensity. Thus, user relies upon network's input to adjust his/her transmission rate. For this reason, in our future work, we will explore methodologies to incorporate users into the estimation algorithm to achieve higher of accuracy in prediction when there is information lost.

REFERENCE

[1] F. P. Kelly, "Charging and rate control for elastic traffic", European Transaction on Telecommunication, 1997.

[2] F. P. Kelly, A. Maullo, D. Tan "Rate control in communication networks: shadow prices, proportional fairness and stability", J. of the Op. Research Soc. 1998.

[3] W. Lee, R.Mazumdar, and N. B. Shroff, "Non-convex optimization and rate control for multi-class services in the Internet," *ACM Trans. on Net.*, vol. 13, no. 4 August 2005.

[4] H. Susanto and B. G. Kim, "Congestion Control with QoS and Delay Utility Function", in *IEEE ICCCN*, Sep. 2013.

[5] H. Susanto and B. G. Kim, Congestion Control and User Utility Function for Real-Time Traffic", *IEEE GlobeCom-MEMS*, 2013..

[6] M. Westerlund et al., "Explicit Congestion Notification (ECN) for RTP over UDP," RFC 6679 , 2012.

[7] Sally Floid, "TCP and Explicit Congestion Notification," in *ACM SIGGCOM*, 1994, pp. 10-23.

[8] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, "Numerical recepes in C++", second edition, Cambridge University Press.

[9] D. Bertsekas and R. Gallagher, *Data Networks*. Englewod Cliffs, NJ: Prentice-Hall, 1992.

[10] Giam Percelli and ByungGuk Kim, "Dynamic behavior of feedback congestion control schemes ," in *INFOCOM*, 1995.

[11] J. CS Lui, V. Misra, and D. Rubenstein, "On the Robustness of Soft State protocols," in *ICNP*, 2004.

[12] Pu Wan and Michael Lemmon, "Distributed Network Utility Maximization using Event-triggered," in *American Control Conference*, 2009.

[13] H. Susanto and B. G. Kim," Message Passing Delay in Network Congestion Management ", IEEE WOCC, 2014.

[14] J. Lee, M. Chiang, and R. Calderbank, "Price-based distributed algorithm for optimal rate-reliability tradeoff in network optimal rate-reliability tradeoff in network," *IEEE Journal on selected areas in communication*, vol. 24, no. 5.

[15] S. Low and D. Lapsely, "Optimization Flow Control, I:Basic Algorithm and Convergence", *ACM Trans. On Net.*, 1999.

[16] A. Couch, N. Wu, and H. Susanto, "Toward a cost model for system administration," in *USENIX LISA*, 2005.

[17] M. Mehyar, D.Spanos, and S. Low, "Optimization Flow Control with Estimation Error" IEEE INFOCOM, 2004.

[18] J. Lee, M. Chiang, and R. Calderbank, "Price-based distributed algorithm for optimal rate-reliability tradeoff in network optimal rate-reliability tradeoff in network," *IEEE Journal on selected areas in communication*, vol. 24, no. 5.

[19] H. Susanto," From Self-Regulate to Admission control in Real-Time Traffic Environment", *IEEE AINA 2014.*