Copyright © 2014 KSII

A Case Study on Network Status Classification based on Latency Stability

JunSeong Kim
School of Electrical and Electronics Engineering, Chung-Ang University
221 Heukseok-dong, Dongjak-gu, Seoul, 156-756 Korea
[e-mail: junkim@cau.ac.kr]
*Corresponding author: JunSeong Kim

Received July 3, 2014; revised September 3, 2014; accepted October 1, 2014; published November 30, 2014

Abstract

Understanding network latency is important for providing consistent and acceptable levels of services in network-based applications. However, due to the difficulty of estimating applications' network demands and the difficulty of network latency modeling the management of network resources has often been ignored. We expect that, since network latency repeats cycles of congested states, a systematic classification method for network status would be helpful to simplify issues in network resource managements. This paper presents a simple empirical method to classify network status with a real operational network. By observing oscillating behavior of end-to-end latency we determine networks' status in run time. Five typical network statuses are defined based on a long-term stability and a short-term burstiness. By investigating prediction accuracies of several simple numerical models we show the effectiveness of the network status classification. Experimental results show that around 80% reduction in prediction errors depending on network status.

Keywords: latency stability; network status classification; end-to-end network latency; network scheduling; time series

A preliminary version of this paper appeared in ICOIN 2013, Bangkok, Thailand. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(grant number 2013-025387)

http://dx.doi.org/10.3837/tiis.2014.11.020

1. Introduction

Latency and throughput are the essential factors in network performance. Recently latency is getting more attention than peak data rates in many network-based applications. For example, in networked control systems(NCS), which remotely control tasks through networks, variations in network latency result in delayed command order [1][2]. It can lead to instability as well as performance degradations. In online gaming applications, where game consoles and server continuously exchange messages for operations, high latencies freeze the entire game [3][4]. Users cannot react quickly enough or ultimately lose the game. In high performance computing such as cloud or grid computing, where computing platforms are established through networks, ignoring network latencies lead to flawed solutions [5][6]. The results can become useless especially when the application is time critical. Network latency is of great interest in many applications and understanding the behavior of latency is important to provide consistent and acceptable levels of services.

There have been many research efforts to increase our understanding of the behavior of network latency. Queueing theory is used with a small scale network having the static information on individual links involved [7][8]. System identification and time series approaches build mathematical models of network latency oscillations using experimental data [7][9]. Artificial neural networks approximate the dynamics of network latency with its strong adaptive learning ability [9][10]. While much of these works have focused on modeling of network latency, network resources in actual network-based applications have not been actively managed. For example, in NCS areas unrealistic or over-simplified assumptions on network delay such as constant delay, stochastic delay, Markov chain model are frequently used [1][2]. In grid computing environments, the effective scheduling has been focused on controllable shared resources such as processors, memory, bandwidth but managements of network latency have easily been ignored [5][11][12]. This is because networks consist of multiple administrative domains. The complexity and the heterogeneity in numerous factors including congestion, QoS, and routing protocols make it very difficult to derive an accurate modeling of network latency. However, proper managements of network latency are important to optimize applications' performance and resource utilization. A divide-and-conquer approach, in which large complex problems are decomposed into smaller, more tractable subproblems, would be an efficient solution. Network traffic repeat cycles of congested-empty-congested indefinitely and a systematic method to differentiate between a congested state and an empty state would be helpful to simplify issues in many network-based applications.

This paper presents an empirical method to classify network status at the application level. We treat the network, seen by specific source and destination nodes, as a black-box and directly measure end-to-end latencies in real operational networks. Based on the oscillating behavior of network latency both in a long-term and a short-term aspect we define five typical network statuses. Experimental results show that the differences in prediction accuracy across the network statuses are orders of magnitude. The classification of network status provides a simplified intuition of the stability and the burstiness of network latency. We do not expect that networks can be regulated by a single rule for all network status. Instead, a delicate management of network resources can be provided by combining multiple rules based on network status. The proposed classification method is simple to utilize and allows scheduling, modeling, or planning network resources to be pursued more accurately according to network

status.

The rest of the paper is organized as follows. In section 2, a preliminary study on several numerical models with the performance metric of mean square error(MSE) is provided. In section 3, a simple and practical approach to classify network status is presented in detail. In section 4, we evaluate the effectiveness of the proposed method for sets of real network latency data. Finally, in section 5, we summarize our results and conclude.

2. A Preliminary Study

2.1 Network Latency Characterizations

As in many studies on latency dynamics, we use RTTs in this experiment. We collect network latency using the *ping* tool, which measures network round trip time(RTT) by sending ICMP(Internet Control Message Protocol) echo request and receiving echo reply messages. A path between nodes within the Chung-Ang University (Seoul, Korea) campus is continuously observed for a day (Thursday, September 24, 2009). The round trip time of a 64-byte probing packet is measured in every 10 seconds and **Fig. 1** shows the 24-hour RTT sample. Since the source, the destination and the size of packets are all fixed variations in latency results mainly from the queuing delay due to packet spacing at each hop on the transmission.

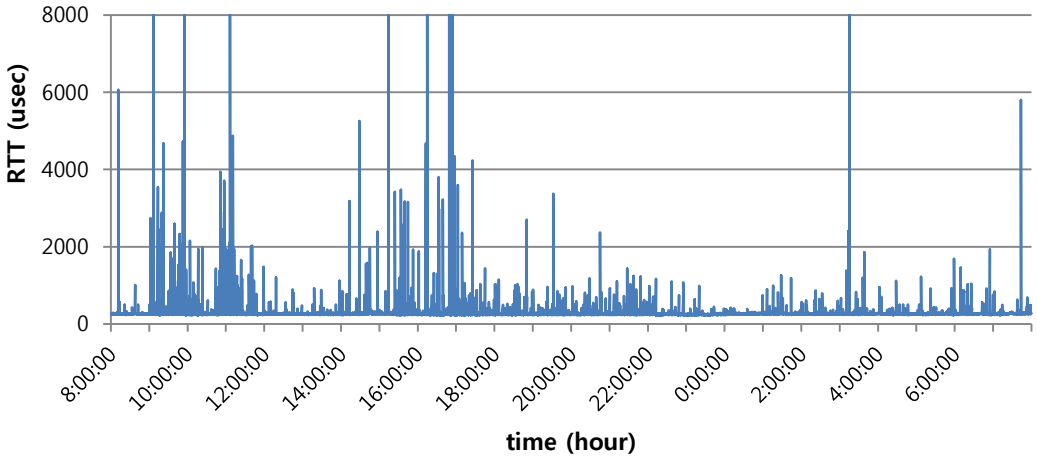


Fig. 1. The round trip time to send a 64-byte probing packet is measured for a day.

To characterize network latency several simple numerical models are considered using a sliding window mechanism. We denote a window record that consists of the most recent K latency history Swin_K. In our previous work, we found that the average-, the median-, and the minimum-based models show good prediction accuracy in general [14].

$$\text{avg}_K(t) = \frac{1}{K}\sum_{i=0}^{K-1} \text{Swin}_K(i,t) \tag{1}$$

$$\text{med}_K(t) = \begin{cases} \text{SortWin}_K\left(\frac{K-1}{2}, t\right) & K = odd \\ \frac{1}{2}\left[\text{SortWin}_K\left(\frac{K}{2}-1, t\right) + \text{SortWin}_K\left(\frac{K}{2}, t\right)\right] & K = even \end{cases} \tag{2}$$

$$\min_K(t) = \text{SortWin}_K(0,t) \tag{3}$$

$$\text{t\_avg}_K(t) = \frac{1}{K-2T}\sum_{i=T}^{K-T-1}\text{SortWin}_K(i,t) \tag{4}$$

$$\text{t\_min}_K(t) = \text{SortWin}_K(T,t) \tag{5}$$

where $\text{Swin}_K(i,t)$ is the $(i+1)^{th}$ element, which is measured at time (t-K+i), of Swin_K; SortWin_K is the sorted sequence of Swin_K in increasing order such that $\text{SortWin}_K(0,t)$ is the smallest element and $\text{SortWin}_K(K-1,t)$ is the largest element in Swin_K; T is the fraction of elements in a window that are ignored for a trimmed list. We consider a trimmed version of Swin_K since network latency is naturally bursty.

2.2 Prediction Accuracy and Window sizes

We use a trace-driven simulation approach. The record of the 24-hour RTT data is fed into a prediction model. By comparing the actual latency value at time *t* with the simple numerical models of a window Swin_K, we evaluate the prediction accuracy. It is effective because the RTT values are the only concern in this study and because the statistical independence of network latencies cannot be guaranteed. We use the performance metric of the *mean square error*(MSE) [9]. For each simple numerical model *f*, we have

$$\text{MSE}_f(t) = \frac{1}{t}\sum_{i=0}^{t-1}(\text{rtt}(i) - \text{prediction}_f(i))^2 \tag{6}$$

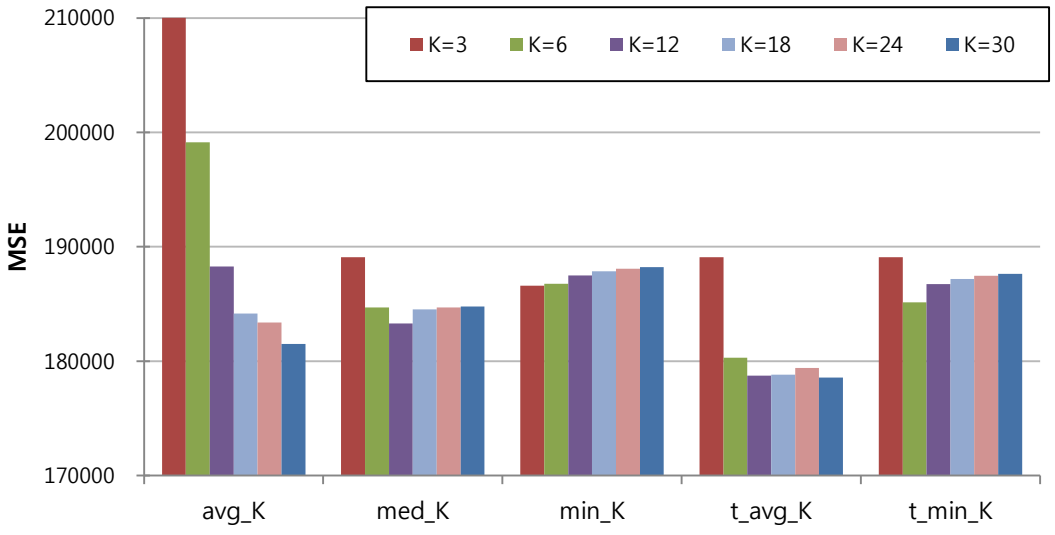


Fig. 2. The prediction accuracy of the simple numerical models for various window sizes, K.

Fig. 2 shows the prediction accuracy of the simple numerical models when applied over the 24-hour RTT sample with various window sizes. In this experiment, we set T = 1 such that only the largest and the smallest elements of Swin_K can be ignored. From the graph we can say that the window size of K = 3, which corresponds to 30 seconds measurement period in the

RTT sample, is not enough to reflect the history of network status since the prediction accuracy of $K = 3$ is, in general, worse than that of larger K. The average-based models prefer long latency history since they show that the larger K the better prediction accuracy in the experimental range. The minimum-based models, however, show that the more latency history the worse prediction accuracy. In contrast, the median-based model seems to have an optimal window size of K=12.

3. A Classification of Network Status

In this section, we present an empirical method to classify network status. First, we divide RTT values into two groups: the *majority* and the *outlier*. Next, we count the number of outliers among the K latency history using a window, Swin_K. Based on the number of outliers both in long-term and short-term aspects we determine the status of a network on the fly. This is a simple and easy-to-utilize approach.

3.1 Distribution of Network Latency and Outliers

Fig. 3 shows the latency distribution of the 24-hour RTT sample. The X-axis represents the RTT range in usec. The primary and the secondary Y-axis represent their occurrences in count and their accumulated occurrences in percentage, respectively. Note that the scale of the X-axis is neither linear nor uniform.

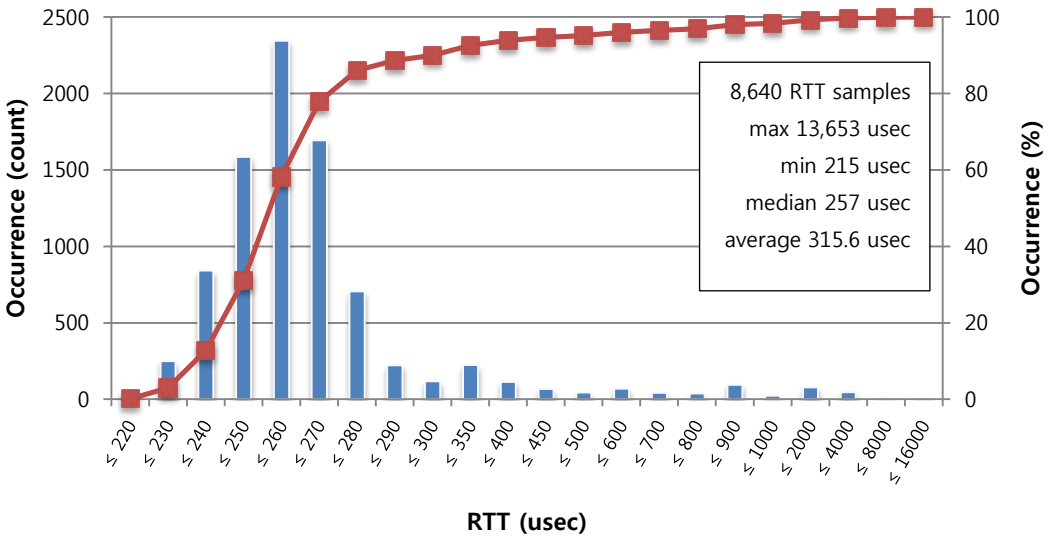


Fig. 3. The distribution of latency values for the 24-hour RTT sample.

From the graph we can see that the RTT values are widely distributed between 215 usec and 13,653 usec. However, most of the RTT values are clustered over a single narrow range. In this case, the RTT values between 215 usec and 300 usec account for about 90% of all. They seem to represent normal operations. That is, their network latencies include the most deterministic behavior of the queuing delay at each hop on the transmission. We classify these RTT values as the *majority* group. The density function of the majority group shows a bell-shaped curve that is symmetric about the median. The rest of the RTT values are

distributed over a very wide range but are few in number. They seem to represent bursty traffic in abnormal operations. These include congestions, retransmissions, ARP table aging, etc. We may further divide them into groups but their portion is so small. We classify these RTT values as the *outlier* group. The network latency of the outlier group seems to be non-deterministic.

3.2 Five Typical Network Statuses

It has been known that networks repeat the cycle of being congested-empty-congested indefinitely [8][13]. In fact, when we carefully look into the preliminary study results we can find that there are certain patterns in network latency variations. For example, at one time, network remains stable and becomes very predictable. At another time, network tends to be very bursty and makes good prediction difficult. We define five typical network statuses based on a long-term stability and a short-term burstiness of latency history. By counting the number of outliers among the K latency history of a window $Swin_K$ we determine the status of a network.

- *Strongly Stable*(SS) status: Network tends to be very stable with respect to a long-term aspect. This is the case when there is no outlier in a long-term window $Swin_{LT}$. Network latency is highly predictable in this status.
- *Relatively Stable*(RS) status: Network shows only occasional bursts of traffic such that it is stable when we ignore the sporadic bursts with respect to a long-term aspect. This is the case when there are one or two outliers in a long-term window $Swin_{LT}$. It can be considered as either a prelude or a postlude of the SS status.
- *Highly Bursty*(HB) status: Network tends to be very bursty with respect to a short-term aspect. This is the case when there are more than one half outliers in a short-term window $Swin_{ST}$. It is pointless to predict network latency in this status.
- *Relatively Bursty*(RB) status: Network shows stable and bursty patterns repeatedly. Network seems to be stable with respect to a short-term aspect but not with respect to a long-term aspect.
- *Transient*(T) status: All other cases. It relays between the stable statuses of the SS, the RS and the busrty statuses of the HB, the RB.

In this experimental study, we consider the window sizes used in the preliminary study of section 2: $Swin_{30}$, $Swin_{24}$, $Swin_{18}$, $Swin_{12}$, $Swin_6$ and $Swin_3$. We exclude the window size of K = 3 since we aware that it is too small to reflect the history of network status. The largest one, the window size of K = 30, which corresponds to 5 minutes measurement period in the RTT sample, is used to determine network's long-term stability. We expect that larger windows can give only a limited benefit since the preliminary study results seem to be getting saturated. The smallest one other than K = 3, the window size of K = 6, which corresponds to 1 minute measurement period in the RTT sample, is used to determine network's short-term burstiness. The window sizes of K = 12, 18, 24, which corresponds to 2, 3, 4 minutes measurement period, respectively, are used in between. In order to consider both long-term and short-term aspects, we use those multiple window sizes at the same time. The latency history of the various K can be given by a single window $Swin_{30}$, which is a superset of $Swin_{24}$, $Swin_{18}$, $Swin_{12}$, and $Swin_6$ as well.

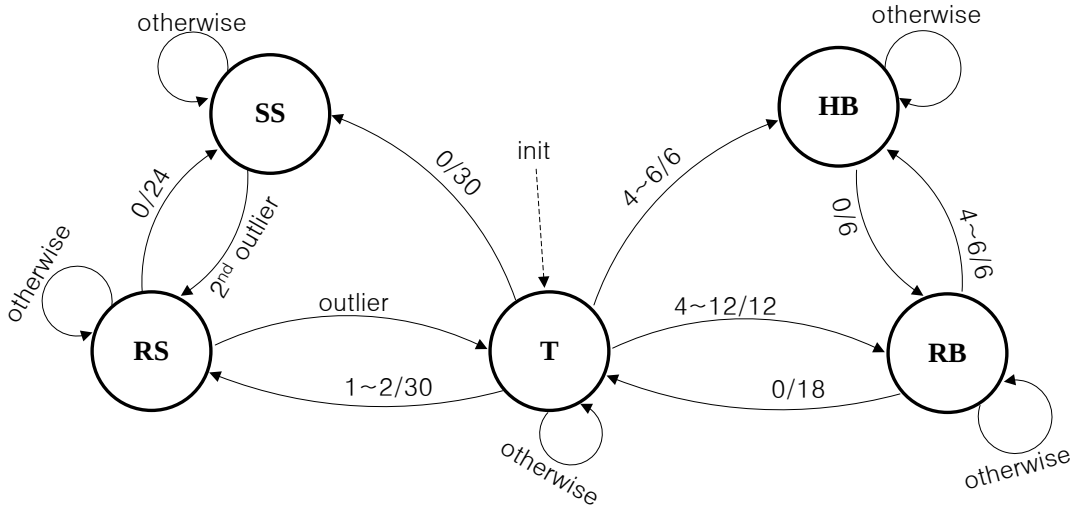


Fig. 4. The state transition diagram of the five typical network statuses. State transitions are induced by counting the number of outliers within a window $Swin_K$.

Fig. 4 shows the state transition diagram of the five typical network statuses. The labels on the transition arc of *n/K* represent the number of outliers, *n*, among the *K* latency history of $Swin_K$. From the definition of the five typical network statuses with the windows in various sizes of $Swin_{30}$, $Swin_{24}$, $Swin_{18}$, $Swin_{12}$, $Swin_6$ we complete the state transition diagram. Note that the condition to enter a status differs from the condition to exit the status reflecting the inertial nature of network statuses.

- T → SS (*n/K* = 0/30): from the definition of the SS with the long-term window $Swin_{30}$
- T → RS (*n/K* = 1~2/30): from the definition of the RS with the long-term window $Swin_{30}$
- T, RB → HB (*n/K* = 4~6/6): from the definition of the HB with the short-term window $Swin_6$
- T → RB (*n/K* = 4~12/12): from the definition of the RB with the window $Swin_{12}$ (heuristically larger than the short-term window size – in this experiment, we use $Swin_{12}$ among $Swin_{30}$, $Swin_{24}$, $Swin_{18}$, $Swin_{12}$, $Swin_6$)
- RS → SS (*n/K* = 0/24): from the definition of the SS and the RS with the window $Swin_{24}$ (heuristically smaller than the long-term window size – in this experiment, we use $Swin_{24}$ among $Swin_{30}$, $Swin_{24}$, $Swin_{18}$, $Swin_{12}$, $Swin_6$) considering the inertial nature of network statuses
- SS → RS (*n/K* = 2nd outlier), RS → T (*n/K* = outlier): from the definition of the SS and the RS considering the inertial nature of network statuses
- HB → RB (*n/K* = 0/6): from the definition of the HB and the RB with the short-term window $Swin_6$ considering the inertial nature of network statuses
- RB → T (*n/K* = 0/18): from the definition of the RB with the window $Swin_{18}$ (heuristically between the long-term window size and the short-term window size – in this experiment, we use $Swin_{18}$ among $Swin_{30}$, $Swin_{24}$, $Swin_{18}$, $Swin_{12}$, $Swin_6$) considering the inertial nature of network statuses

When there are conflicts in the state transition, if any, we give high priority to SS, RS, HB,

RB, T in the order. For example, assume that network is in the T status and that the 30 latency history composed of 26 consecutive majorities followed by 4 consecutive outliers. This is rare but fulfills the conditions for both the HB and the RB transitions at the same time. In this case, according to the priority policy, the network status changes to the HB.

4. Experiments and Results

4.1 A Boundary Value between the Majority and the Outlier

In section 3.1 we divide network latencies into two groups: the *majority* and the *outlier*. When we know the distribution of network latency in advance, we can use a static boundary value between the two groups directly. However, it might not be the case in most situations. Instead, a dynamic boundary value can be estimated by using a median and a minimum value of latency history. We aware that network latencies are random processes and that the majority follows the normal distribution. With a sliding window Swin_K the boundary value can be

$$\text{boundary(t)} = \text{med}_K(t) + [\text{med}_K(t) - \min_K(t)] \tag{7}$$

We can confirm that the estimation works by applying the statistics of the 24-hour RTT sample. That is, the median of 257 usec and the minimum of 215 usec result in the boundary value of 299 usec, which is close to the static boundary value of 300 usec we used in section 3.1.

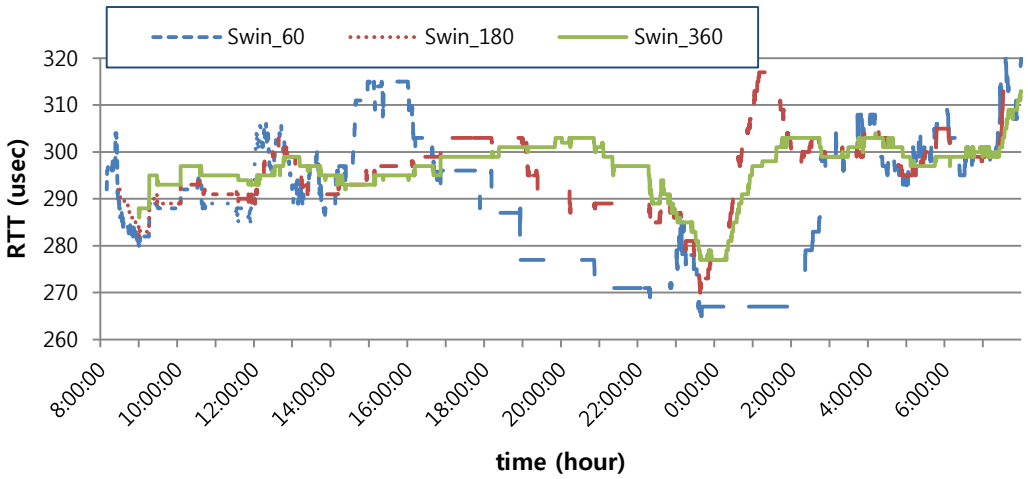


Fig. 5. The boundary values of the 24-hour RTT sample for various window sizes, K.

While we can contrive a float to sample a minimum and a median value of latency history, in this experiment we use an extra sliding window for simplicity. **Fig. 5** shows the dynamic boundary values between the majority and the outlier groups for various window sizes when applied over the 24-hour RTT sample. We consider the window sizes of K = 60, 180, 360, which correspond to 10, 30, 60 minutes measurement period, respectively. From the graph we can see that the window size of K = 60 is not enough to reflect the latency history since the boundary value remains around the median in many cases. In this experiment, we arbitrarily

choose K = 180 for our convenience. With a window $Swin_{180}$ the 8,640 RTT samples are divided into the five typical network statuses such that 26.5% of SS, 17.1% of RS, 14.5% of T, 33.1% of RB, and 8.8% of HB.

4.2 Prediction Accuracy across the Five Typical Network Statuses

To evaluate the effectiveness of the network status classification we examine the prediction accuracy of the simple numerical models over the five typical network statuses. **Fig. 6** shows the prediction accuracy of the t_min_6, the med_{12} and the t_avg_{30} models over the 24-hour RTT sample dynamically applying the classification of network status on the fly. Also, shown for comparison are the overall prediction accuracy results of section 2.2. We can see that the overall prediction accuracy is close to that of RB, in which network latency oscillates up and down very badly. This means that without the status classification there would be no chance of accurate prediction and that we cannot expect any delicate network resource managements. It is desirable to detect network status and choose appropriate rules according to the status. We want to separate sand and gravel from a bottom of river.

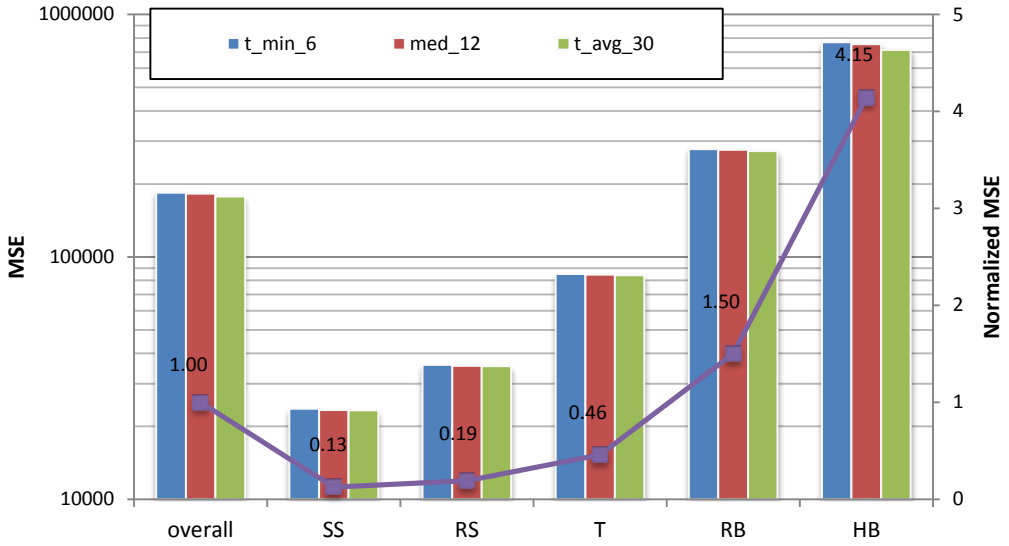


Fig. 6. The comparison of prediction accuracy for the five typical network statuses.

While the absolute values of MSE depend on the geographic distance of a transmission path we'd rather focus on the relative values of them across different network statuses. The line graph with the secondary Y-axis of **Fig. 6** presents a normalized MSE of the t_min_6 model with respect to the overall case. We can see from the graph that the prediction accuracy falls as the network status changes from SS to RS, T, RB, HB in the order. Though the RTT values come from the same resources the differences in prediction accuracy are orders of magnitude between network statuses. It is true regardless of prediction model. We can say that the network statuses are correlated with oscillating behaviors of latency. More than 80% reductions in MSE has been found in stable statuses of the SS and the RS over the overall case. We can expect realistic planning and effective scheduling of network resources when networks in stable statuses. Burstiness is a nature of network latency but now we can filter out the bursty statuses.

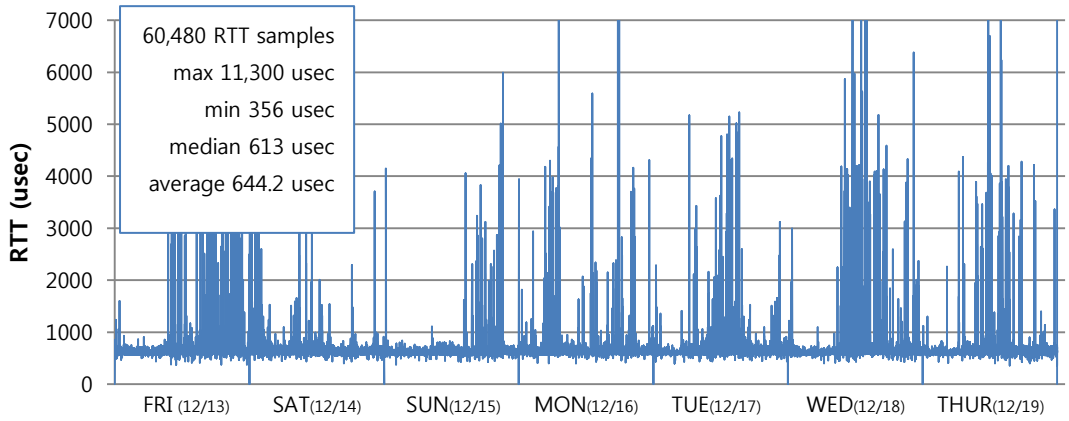


Fig. 7. The round trip time to send a 64-byte probing packet was measured for a week.

Table 1. A summary of the node used in the experiment.

IP	Location	Distance	# Hops
165.194.95.137	Chung-Ang Univ., Seoul, Korea	within ~1.0Km diameter	5

To confirm the generality of the classification method for network status we repeat the same experiment on another set of network latency. **Fig. 7** shows another RTT sample, which are collected within the Chung-Ang University campus for a week (between December 13 and December 19, 2013) and **Table 1** summarizes the node used in this experiment. Packets can be corrupted or lost during the transmission through internet. There are 60,480 RTT samples and 13 among them show time-out. The missing data is replaced with a global maximum latency value(11,300 usec) for the experiment. From the graph we can see relatively low fluctuations in latency during the night and the weekend as we expected. Other than that, it shows a form similar to that of **Fig. 1.** However, the measurements come from different resources and their statistics are different from those of **Fig. 1**.

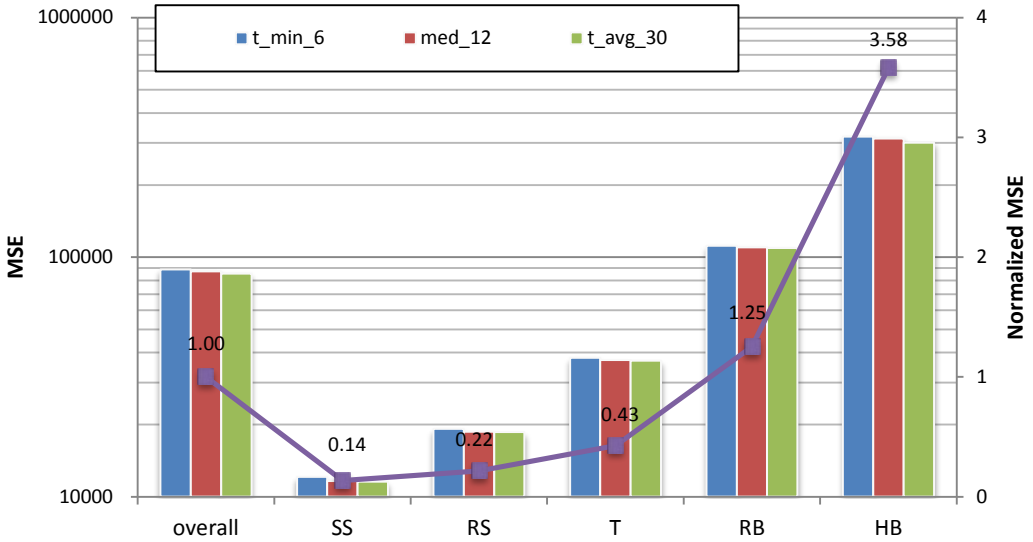


Fig. 8. Another comparison of prediction accuracy for the five typical network statuses.

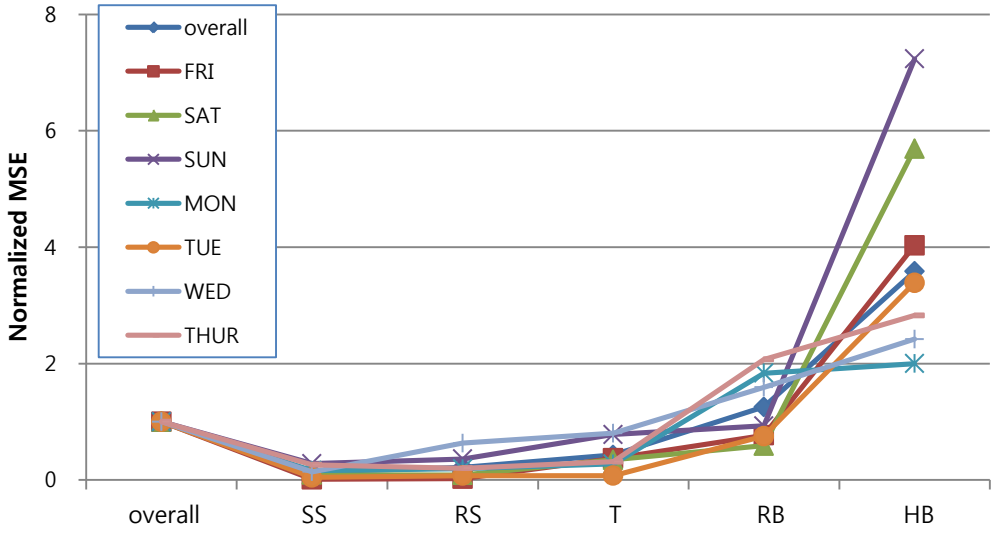


Fig. 9. Normalized MSE of the t_min_6 model for the five typical network statuses by date.

Fig. 8 shows the prediction accuracy over the new set of latency data. Compared with the graph of **Fig. 6**, we can see that the absolute values of MSE are scaled down. It is expected since the difference between the maximum and the minimum values of the new set of latency data is less than that of the 24-hour RTT sample. As with the graph of **Fig. 6**, however, we can easily see the same patterns of prediction accuracy across the five typical network statuses. That is, the prediction accuracy falls appreciably as the network status changes from SS to RS, T, RB, HB in the order. The stable statuses show almost 80% reductions in MSE over the overall case. Also, **Fig. 9** presents a normalized MSE of the t_min_6 model for the five typical network statuses by date. Since network is more stable during weekend(Saturday and Sunday) we can see that the normalized MSE in the HB status is relatively high on those days. Depending on date the stable statuses show more than 90% reductions in MSE over the overall case. From these observations we can conclude that the classification of network status is quite effective to provide a simplified intuition of the stability and the burstiness of network latency.

5. Conclusions

Understanding the dynamic behaviors of network latency is of great interest in network-based applications. In this paper, we present a simple empirical method to classify network status and define five typical network statuses with respect to the stability of latency history - *strongly stable*(SS), *relatively stable*(RS), *transient*(T), *relatively bursty*(RB), and *highly bursty*(HB). The basic idea of the approach is that network traffic repeat cycles of congested states and that a variation of network latency is strongly correlated with the past history of the latency. We first differentiate between two types of network latency; the *majority* and the *outlier*. Then, by simply counting the number of outliers among the most recent latency history both in a long-term and a short-term aspect we determine networks' status on the fly. Experiments with several simple numerical models show notable differences in prediction accuracy among the different network statuses. The classification of network status provides a simplified intuition of the stability and the burstiness of network latency. We expect that the proposed method can be used for scheduling, monitoring and planning network resources in many network-based applications. As future works, it would be interesting to

investigate the network status classification method with latency data sets of varying sizes and granularities.

References

[1] Y.-B. Zhao, J. Kim, G.-P. Liu and D. Rees, "Compensation and Stochastic Modeling of Discrete-Time Networked Control Systems with Data Packet Disorder," *International Journal of Control, Automation, and Systems*, vol. 10(5), pp. 1055-1063, 2012. Article (CrossRef Link)

[2] Y. Ge, Q. Chen, M. Jiang and Y. Huang, "Modeling of Random Delays in Networked Control Systems," *Journal of Control Science and Engineering*, 2013. Article (CrossRef Link)

[3] M. Claypool and K. Claypool, "Latency Can Kill: Precision and Deadline in Online Games," in *Proc. of ACM SIGMM conference on Multimedia systems*, pp. 215-222, 2010. Article (GoogleScholar Link)

[4] R. Amin, F. Jackson, J. E. Gilbert, J. Martin and T. Shaw, "Assessing the Impact of Latency and Jitter on the Perceived Quality of Call of Duty Modern Warfare 2," *Lecture Notes in Computer Science*, vol. 8006, pp. 97-106, 2013. Article (CrossRef Link)

[5] S. Kumar and N. Kumar, "Network and Data Location Aware Job Scheduling in Grid: Improvement to GridWay Metascheduler," *International Journal of Grid and Distributed Computing*, pp.87-100, 2012. Article (GoogleScholar Link)

[6] M. Rahman, R. Hassan, R. Ranjan and R. Buyya, "Adaptive workflow scheduling for dynamic grid and cloud computing environment," *Concurrency and Computation: Practice and Experience*, vol. 25(13), pp. 1816-1842, 2013. Article (CrossRef Link)

[7] M. Yang, X. R. Li, H. Chen and N. SV Rao, "Predicting internet end-to-end delay: An overview," in *Proc. of IEEE Southeastern Symposium on Systems Theory*, pp. 210-214, 2004. Article (GoogleScholar Link)

[8] Y. Xie, J. Hu, Y. Xiang, S. Yu, S. Tang and Y. Wang, "Modeling Oscillation Behavior of Network Traffic by Nested Hidden Markov Model with Variable State-Duration," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24(9), pp. 1807-1817, 2013. Article (CrossRef Link)

[9] R. Adhikari and R. K. Agrawal, *An Introductory Study on Time Series Modeling and Forecasting*, LAP Lambert Academic Publishing, Germany, 2013. Book (GoogleScholar Link)

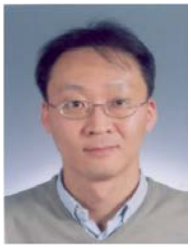
[10] S. Belhaj and M. Tagina, "Modeling and prediction of the internet end-to-end delay using recurrent neural networks," *Journal of Networks*, vol. 4, no. 6, pp. 528-535, 2009. Article (CrossRef Link)

[11] H. Casanova, "Network Modeling Issues for Grid Application Scheduling," *International Journal of Foundations of Computer Science*, vol. 16(2), 2005. Article (CrossRef Link)

[12] M. M. Yousaf and M. Welzl, "Network-Aware HEFT Scheduling for Grid," *The Scientific World Journal*, 2014. Article (CrossRef Link)

[13] L. Zhang and D. D. Clark, "Oscillating behavior of network traffic: A case study simulation," *Internetworking: Research and Experience*, pp. 101-112, 1990. Article (GoogleScholar Link)

[14] J. Kim and J. Yi, "A Pattern-based Prediction: An Empirical Approach to Predict End-to-End Network Latency," *Journal of Systems and Software*, vol. 83, pp. 2317-2321, 2010. Article (CrossRef Link)



JunSeong Kim received a Ph.D. in Electrical Engineering from the University of Minnesota at Minneapolis, an M.S. and a B.S. both in Electronics Engineering from Chung-Ang University in Seoul, Korea. He is currently a Professor of School of Electrical and Electronics Engineering at the Chung-Ang University in Seoul. His main research interests include computer architecture, high-performance computing, parallel processing, and embedded system design.