# On the Level of Guaranteed Services for Signaling Control in Cellular Networks

Saowaphak Sasanus and David Tipper
Telecommunications Program, University of Pittsburgh, Pittsburgh, PA 15260
sasst128,dtipper@pitt.edu

***Abstract*— In this paper, we discuss our preliminary work on signaling overload control for cellular services which requires adaptability and scalability as well as guaranteed level of service. We propose a set of controls that can provide different grades of service to different service classes effectively. The controls are resource sharing schemes proposed consisting of a token rate control combined with priority scheduling and an awareness of the transport network state. Simulation results are given illustrating the behavior of new signaling overload controls and comparative performance with the other existing controls in the literature.**

## I. Introduction

Wireless cellular networks (WCNs) are dependent on signaling networks for secure communications, location tracking of users, and providing intelligent services (e.g., video calling, incoming call restriction, multi-media message service). In overload situations, congestion can easily occur at signaling network database servers such as the visitor location register (VLR) degrading network performance. Numerous examples have been reported in the literature of mass call ins and denial of service attacks (e.g., with short message services) that overload the signaling network resulting in almost zero throughput for a network service area, even though free traffic channels are available in some areas [1] [2]. Clearly, new signaling overload controls are needed as existing techniques have been demonstrated to be lacking.

Unlike basic voice calls in PSTNs where all signaling services, in WCNs the signaling of different mobile applications should not be treated equally as there are many varieties of signaling services and heterogeneity in the traffic. Multi-class signaling overload control for WCNs have recently been proposed in [3] [4] to ensure differentiated QoS among classes of signaling traffic. Whereas, the advantages of adaptive overload control were explored in [5] [6]. By considering both concepts, algorithms on adaptive multi-class token rate control were proposed in [7] [8]. Token rate control is known to provide better bounded maximum departure rate as well as improving throughput by servicing a burst of packets after a low activity period as compared to other rate-based controls (i.e., call gapping, percentage of call blocking). In *Wei Wu, et al.'s algorithm* [7] based on token rate control, load is controlled only when an overload is detected. The algorithm tries to optimally utilize resources by iteratively finding shared ratios among all classes in each control decision. However, the functionality of the algorithm depends solely on overload detection where utilization is chosen as the trigger parameter. When the utilization does not reflect the amount of current arrivals well, the rate distribution scheme will behave incorrectly, as not all unused resources of one class is redistributed to the others. Moreover, since the algorithm completely shares all unused resource, services cannot be guaranteed in the feedback delayed system. In *Karagiannis's algorithm* [8], overload control is always activated, lowering the overshoot of call/service blocking rate and system delay time caused by slow reaction to overload. However, token and job buffer settings of the algorithm sometimes cause too large token accumulation, leading to undesirable performance. Moreover, the algorithm uses static shared ratios where overloaded classes are penalized while underloaded classes are credited with more rate. This absolute resource distribution does not optimally utilize resource or provide scalability in time-varying networks, although it ensures guaranteed services.

The current algorithms do not provide guaranteed services and effective resource sharing simultaneously. Moreover, none of them address issue of ineffective control caused by servicing load that needs new channel allocation when channel is unavailable. We address the requirements of being adaptability and scalability here and considered integration of state of available radio resources in the control [9]. In this paper, we propose two novel multi-class signaling network congestion controls that allow an efficient tradeoff between providing guaranteed and shared resources. Simulation results are given for the proposed congestion control schemes illustrating their performance as well as comparing them with other algorithms. The remainder of the paper is organized as follows. In the next section, we present our proposed adaptive multi-class signaling overload controls. Followed by a comparative simulation based performance evaluation in Section III. Lastly, in Section IV we summarize our results and present our conclusions.

## II. Adaptive multi-class token rate controls

Overload control can be effective by dropping load in the early stage of congestion and at source nodes. We use centralized control approaches where the control decisions are made at the server and the service rejections are done at sources accordingly. Server drops overload that is not already throttled by sources. Control decisions are more precise since the server knows the global view of all sources. We study token rate/bucket control here due to its superior performance over other rate- and window-based controls. We bound the maximum departure rate by limiting the bucket size. Let the token bucket capacity be $B$ bytes. Tokens arrive with the deterministic rate of $r$ bytes/sec where tokens that arrive into a full bucket are dropped and lost. The maximum service rate is $M$ bytes/sec. For a burst length $S$, the arrival input burst should not exceed $MS$ bytes, thus $B + rS \leq MS$. However, token rate control has the disadvantage on large fluctuation of the departure rate, which raises the requirement of large buffer in the downstream node. Especially, the database server which provides service to multiple sources is highly effected by this requirement. To cope with this problem, the token bucket/buffer size at each source is partitioned and used as job buffer. Each class has a separate token buffer and job buffer to easily achieve differentiated QoS among classes.

This work proposes resource sharing among classes using the concept of *rate sharing* or *buffer sharing*[1], both of which can be explained by the queueing model shown in Fig. 1.

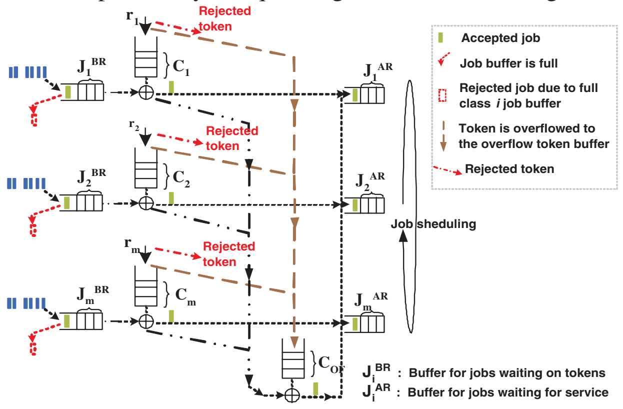


Fig. 1. Queueing model of the proposed resource sharing schemes

Let $C_i$ and $J_i$ be the size of the token buffer and the job buffer of class $i$. The queueing model consists of the token buffers ($C_1$, $C_2$, ..., $C_m$), the job buffers ($J_1$, $J_2$, ..., $J_m$) for class $(1, 2, ..., m)$, and the overflow buffer ($C_{OF}$). Class 1 is the highest priority class with class $m$ the lowest priority class. The job buffer of class $i$ of two logical job buffers. $J_i^{BR}$ stores jobs of class $i$ that are waiting for tokens. The second logical job buffer denoted by $J_i^{AR}$ stores jobs of class $i$ that are waiting for service. The token rate of class $i$ is denoted by $r_i$. Tokens are credited to the token buffer of class $i$ periodically every $1/r_i$ seconds. In the event the token buffer of class $i$ is full when its token arrives, the token is sent to the overflow buffer, $C_{OF}$ if it is available. Otherwise, the token is lost. When a packet which belongs to class $i$ arrives, if there is available token in the token buffer of class $i$ or in the overflow buffer, the packet captures a token and moves to $J_i^{AR}$. Otherwise, the packet is queued in the job buffer of class $i$, $J_i^{BR}$ if space is free. The packet is only rejected if the job buffer is full. A packet in job buffer $J_i^{BR}$ waits until a token becomes available in token buffer of its class or in the overflow buffer. Once a job is in the $J_i^{AR}$ portion of the job buffer, it waits for its turn at the server. This queueing model allows better utilization of the server since some tokens of temporary low-activity classes can be used by packets which belong to the other currently high-activity classes.

In both schemes proposed here, resources (i.e., token rate and buffer) are distributed among classes based on priority weights. Let $\Pi_i$ denote the priority weight for class $i$ where $\sum \Pi_i = 1$. The priority weight of any class is selected based on the significance of that class and the percentage of load which that class contributes to the total load. The token rate of class $i$, $r_i$ is set equal to $\Pi_i r$. The allowed burst size which is the summation of token and job buffer of class $i$ is set to $\Pi_i B$. The burst size of the highest priority class, $B_1$ is first derived according to its maximum system delay time[2] recommended by the ITU [11], and is later used to calculate the burst size of the other classes, $B_i$. However, the value of $B_i$ must not cause the violation to the preferrable maximum system delay time of class $i$. We set the burst size, $B_i$ at each source based on the from the burst size, $B_i$ at the server and the number of participating sources. Since our overload control is only activated when an overload is detected, a large token accumulation is unlikely. In the exchange, overshoot in the system delay time and the probability of service rejection when the overlod is first detected, is expected to be higher than the case that the overload control is always active.

At the server, let $Svc_i$ and $Svc_i^{max}$ denote the service time and the maximum system delay time of class $i$ at the server where $Svc_1^{max} < Svc_2^{max} < ... < Svc_m^{max}$. The burst size is set such that, $B_1^* \times Svc_1 \leq Svc_1^{max}$ and $B_i^* = min(\frac{\Pi_i}{\Pi_1} \times B_1^*, \frac{Svc_i^{max}}{Svc_i})$, where a $^*$ superscript indicates the initial settings. Since the departure rate is limited by the maximum service rate, the burst size is assigned to the token buffer, $C_i^* = B_i^*$. There is no job buffer at the server, $J_i^* = 0$.

At each source, the token buffer size of each class is set to the same token buffer size at the server, or $C_i^*$ at source is equal to $B_{i_{Server}}^*$. The job buffer size of the highest priority class is set, so that the system delay time of the last job in the queue will not violate the maximum system delay time of its class. Let $S\acute{v}c_i$ be the token waiting time of class $i$ job, and $S\acute{v}c_i^{max}$ be the preferred maximum waiting time of last job in the class $i$ job queue where $S\acute{v}c_1^{max} < S\acute{v}c_2^{max} < ... < S\acute{v}c_m^{max}$. Let $\rho_{targ}$ be the target average utilization of the server. $S\acute{v}c_i$ is equal to $\frac{1}{r \times \rho_{targ} \times \Pi_i}$. Since the system delay time here is a job's waiting time for a token, the class $i$ job buffer size is set such that $J_1^* \times S\acute{v}c_1 \leq S\acute{v}c_1^{max}$. The burst size of the highest priority class is the summation of the job buffer and the token buffer, $B_1^* = J_1^* + C_1^*$. After the burst size of the highest priority class is derived, the burst size of each lower priority class can be calculated with the constraint of its own budget of maximum system delay time, $B_i^* = min(\frac{\Pi_i B_1^*}{\Pi_1}, \frac{S\acute{v}c_i^{max}}{S\acute{v}c_i} + C_i)$. Then, the job buffer size of each lower priority class can be derived from $J_i^* = B_i^* - C_i^*$.

[1]A. Berger studies static control for the buffer sharing scheme [10]. Token buffer of each class and the overflow buffer is set to 10.

[2]Assuming that delay time due to accessing radio channel and relaying packet at a BS is very small, the maximum delay time budget is equally distributed to a source (BSC) and a server (MSC/VLR).

The algorithm monitors arrivals at the server and checks whether the system is overloaded at every end of the control interval. By following Kasera et al's study [12], overload is detected using both the processor utilization and the acceptance rate. The utilization is dimensionless which makes it relatively system-independence but with slow reaction to overload. The acceptance rate reacts fast to overload, but it does not represent the inner situation of the processor as well as the utilization. To prevent a ping-pong effect, we consider change in overload status only when both indicators changed from detection or abatement thresholds to the other.

When an overload is detected, a token rate of each class is reassigned. We adopt Kasera et al.'s single-class control algorithm [12] to the multi-class case. Let $r_{n_i}$ be the class $i$ token rate in the $n^{th}$ control time interval $(n = 0, 1, 2, ...)$. The token rate in the next control interval $(r_{n+1_i})$ is reduced when the utilization of class $i$ denoted by $\rho_i$ is greater than the target utilization of class $i$ denoted by $\rho_{targ_i}$, but at least $r_{min_i}$ to allow some transmission. If $\rho_i$ is less than $\rho_{targ_i}$, the token rate $r_{n+1_i}$ is increased but limited by the server's service rate $r$. The specific formula adopted is given by.

$$r_{n+1_i} = \begin{cases} min\left(\frac{\rho_{targ_i}}{\hat{\rho}_i} \times r_{n_i}, \Pi_i r\right) & : \hat{\rho}_i < \rho_{targ_i} - \frac{\epsilon}{2} \\ max\left(\frac{\rho_{targ_i}}{\hat{\rho}_i} \times r_{n_i}, r_{min_i}\right) & : \hat{\rho}_i > \rho_{targ_i} + \frac{\epsilon}{2} \\ r_{n_i} & : otherwise \end{cases} \quad (1)$$

where $\epsilon$ is the percentage fluctuation allowed in the utilization

In *rate sharing*, the priority weights used in rate and buffer distribution are adaptively adjusted while the overflow buffer $C_{OF}$ is set to 0. Let $\widehat{\pi}_i$ denote the adaptive priority weight. The token rate of class $i$ that theoretically allows satisfaction to all classes denoted by $r_i^A$. The weights $\widehat{\pi}_i$ is equal to $\frac{r_i^A/r}{\rho_{targ}}$ where $r_i^A$ is calculated using the similar concept to min-max sharing[3]. In min-max sharing, the lowest token rate depends only on the amount of the current load of each class, and all classes have the same priority to access unused resource. Whereas, in our sharing, the lowest token rate of any class is set at a certain threshold indicating the maximum resource that one class is allowed to share with the others, and the higher priority classes have an easier chance to grasp unused resources than the other lower priority classes.

Let $x_i^l$ be the minimum service rate that class $i$ will receive and $x_i^h$ be the baseline rate for class $i$. $x_i^h$ is set to $\Pi_i r_n$, and $x_i^l$ is set to $x_i^h(1 - H)$, where $H$ denotes the maximum percentage that one class is willing to share with the others. We calculate $r_i^A$ as follows. In the first round, $r_i^A$ will be set to at least $x_i^l$ and at most $x_i^h$ if arrival load of class $i$ denoted by $\lambda_i$ is less than or equal to $x_i^l$, and greater than or equal to $x_i^h$, respectively. If some classes require service rate less than $x_i^l$, classes that still unsatisfy with their assigned $r_i^A$ can acquire more rate in the following rounds. That is only if unused resource from the first round are not already claimed by some higher priority classes. When all classes satisfy with their assigned $r_i^A$, whatever left is distributed to all classes using the priority weights. Assuming $m$ classes of services, $r_i^A$ ranges from $x_i^l$ to $x_i^h + \sum(x_a^h - max(\lambda_a, x_i^l)) \mid \forall a \neq i, \lambda_a < x_a^h$.

[3]According to [13], "a min-max sharing allocates a user with a small demand what it wants, and evenly distributes unused resources to the big users". First, demand of resource is allocated in increasing order. Second, no source gets a resource larger than its demand. Third, source with unsatisfied demands get an equal share of resource, which is continued until resource is depleted.

In *buffer sharing*, we use a constant priority weight $\Pi_i$ in rate and buffer distribution. When the arrival load changes, the percentage of class $i$ token buffer that is contributed to the overflow token buffer denoted by $C_{OF_i}^p$ is changed according to Eq. 3 with the constraint that it must not exceed the preferrable maximum percentage of resource sharing, $H$. The value of $C_{OF_i}^p$ is zero when the arrival load of class $i$, $\lambda_i$ is greater than or equal to the token rate of class $i$, $r_{n_i}$. The value is increased linearly when the arrival load is less than the token rate. When the arrival load of class $i$, $\lambda_i$ is greater than or equal to $(1-H)r_{n_i}$, $C_{OF_i}^p$ is equal to $H$. The portion of the overflow token buffer contributed by class $i$ denoted by $C_{OF_i}$ is calculated by Eq. 2. The overflow token buffer $C_{OF}$ is the summation of $C_{OF_i}$ from all classes. Eq. 4 shows the buffer settings as the percentage of sharing is changed.

$$C_{OF_i} = C_{OF_i}^p \times B_i^* \quad (2)$$

$$\text{where } C_{OF_i}^p = \begin{cases} min(\frac{r_{n_i} - \lambda_i}{r_{n_i}}, H) & : \forall i | \lambda_i \leq r_{n_i} \\ 0 & : otherwise \end{cases} \quad (3)$$

$$C_i = (1 - C_{OF_i}^p)\, C_i^*, \text{ and } J_i = (1 - C_{OF_i}^p)\, J_i^* \quad (4)$$

In both sharing schemes, the token rate of class $i$ is distributed to each source node similarly to rate distribution used in *rate sharing scheme* except that priority is not exercised in acquiring unused resources here. Let $(r_{n+1_{i,j}})$ denote class $i$ token rate that node $j$ receives. Let $H_j$ denote the maximum percentage that one node is willing to share with the others. Let $x_{i,j}^l$ be the minimum service rate that node $j$ will receive and $x_{i,j}^h$ be the baseline rate for node $j$. Assuming the total of $n$ nodes, $x_{i,j}^h$ is set to $\frac{r_{n+1_{i,j}}}{n}$, and $x_{i,j}^l$ is set to $x_{i,j}^h(1 - H_j)$. In the first round, $r_{n+1_{i,j}}$ is set to at least $x_{i,j}^l$ and at most $x_{i,j}^h$ if arrival load of node $j$ denoted by $\lambda_{i,j}$ is less than or equal to $x_{i,j}^l$, and greater than or equal to $x_{i,j}^h$, respectively. If some nodes require service rate less than $x_{i,j}^l$, nodes that still unsatisfy with their assigned rate will receive more rate but will not exceed $\lambda_{i,j}$. When all nodes satisfy with their assigned rate, whatever left is distributed to all nodes equally.

## III. Performance Evaluation

We studied a simple network model as shown in Fig. 2. There were three sources of aggregate signaling service requests at base station controllers (BSCs) each of which supports seven base stations (BSs). Each BSC was directly connected to a common shared visitor location register (VLR), a database server co-located at the MSC.

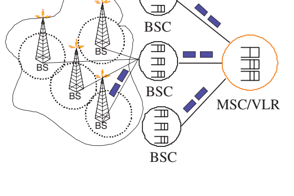


Fig. 2. Node model in this study

A BSC requested services from the VLR according to the request originated from BSs. OPNET v.10 was the software tool used to develop the simulation.

The assumption in this preliminary study was that all signaling services had the same service delay time of $2.5ms$[4]. The delay time in dropping a packet due to unavailable job buffer was set to $1ms$. The control interval was set to $1.0s$ to follow the setting in [12] and to suit database networks where query and storage take time in the order of a second. The simulation run time was set to $720s$. The total target utilization was set to $0.8$. The detection and the abatement thresholds were set to $0.8$ and $0.7$ for the utilization, and $0.7$ and $0.6$ for acceptance rate since it is less stable than the utilization. The percentage of fluctuation allowed from the target utilization was set to $0.01\%$. Three classes of service requests were considered: high, medium, and low denoted by class 1, 2, and 3, respectively. The arrival load was independently originated among classes and followed a Poisson distribution. The assumption was that all signaling services consists of a packet with the same size. The priority weight of high, medium, and low priority class denoted by $\Pi_1$, $\Pi_2$, and $\Pi_3$ were initially set to $0.5$, $0.35$, and $0.15$, respectively. These numbers were only set based on priority in these experiments, not the contributed load.

For the proposed controls, table I below shows the initial size of each class' job and token buffers. The maximum percentage of sharing was set to $40\%$. For other algorithms, we follow rules exercised in the original work. In Karagiannis's algorithm, the minimum token buffer of each class was 90, and a shared job buffer for all classes was 20. The settings purposely reduced the packet loss due to unavailable job buffer. In Wei Wu et al.'s algorithm, the token buffer of each class was 10, and a shared job buffer for all classes is 200.

Table I. Initial Settings of Token and Job Buffers

| Class | Server | | | Source | | |
|---|---|---|---|---|---|---|
| | $B_i$ | $J_i$ | $C_i$ | $B_i$ | $J_i$ | $C_i$ |
| HI | 120 | 0 | 120 | 136 | 16 | 120 |
| MED | 84 | 0 | 84 | 95 | 11 | 84 |
| LOW | 36 | 0 | 36 | 40 | 4 | 36 |
| **Total** | **240** | **0** | **240** | **271** | **31** | **240** |

The performance parameters under the inspection were the system delay time and the utilization of each class. The system delay time is the time between a packet arriving and resided in the job buffer until the packet receives service and departs. The utilization of class $i$ was a ratio of time that the server/VLR serviced class $i$ signaling packets to the summation of time that the server/VLR serviced signaling packets from all classes and the control packets that were successfully served within the control interval.

For reliability of the results, data was collected from 57 runs using different seed numbers. Each point is the average value of the measurements from 57 run sets where each point in a run set is the sample mean over $3s$. However, we represent the results in term of the average only as they are conclusive.

[4]The response time was selected such that it simplified the analysis and it was proper with $2s$ post selection of authentication service and $4s$ post-selection delay of paging/alerting services according to [11] by considering the propagation delay time and queueing delay time at both source and the server.

In the first experiment, the reaction of overload control algorithm was inspected to a sudden overload by setting high arrivals beginning at time $180s$ and ending at time $540s$. In the first experiment, the functionality of the proposed controls was studied where all classes overloaded their resource shares. The arrival load of high, medium, and low priority classes from each source/BSC were set to 60, 60, and 70 packets/sec in the period of $180s - 300s$ and $420s - 540s$, and 70, 60, and 70 packets/sec in the period of $300s - 420s$.

The simulation results show that the utilization of rate sharing and buffer sharing could be maintained approximately at the target utilization, 0.8. As shown in Fig. 3, each class be able to utilize resource roughly at its share, which was the product of the target utilization and the priority weight. Fig. 4 shows that the proposed controls provided differentiated QoS among classes in term of the system delay time. Load from the lower priority classes faced longer delay than load from the other higher priority classes.

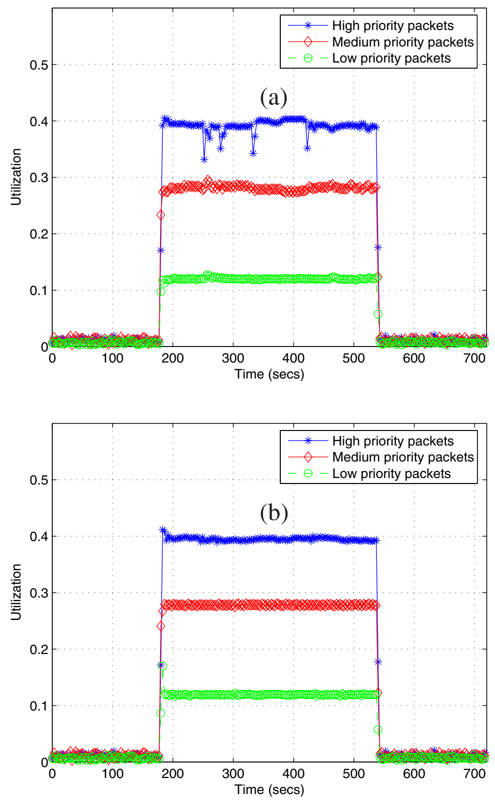
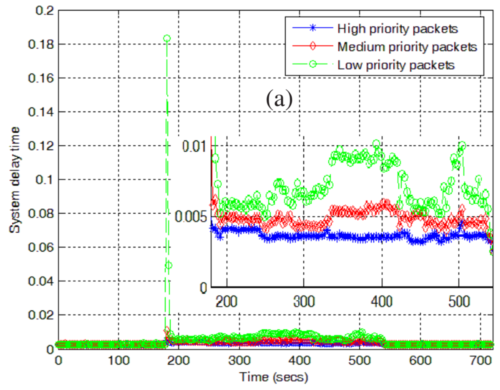


Fig. 3. The utilization of each class in a) rate sharing and b) buffer sharing

In the second experiment, the proposed controls were compared with Wei Wu et al.'s algorithm [7] and Karagiannis's algorithm [8], which are based on token rate control in [14]. To prove the concept of resource sharing, the arrival load from high priority class was set to require resource less than its share, while others overloaded their shares. High, medium, and low priority class from each source/BSC were set equal to 60, 40, and 30 packets/sec in the period of $180s - 300s$ and $420s - 540s$, and 30, 40, and 60 packets/sec in the period of $300s - 420s$.



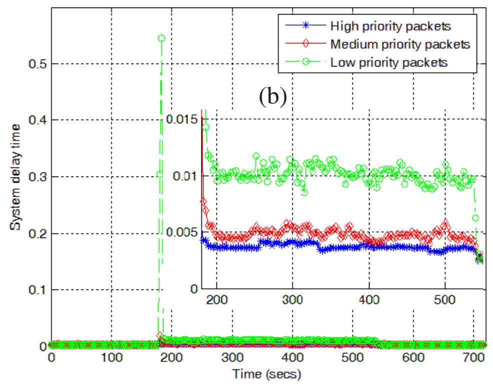


Fig. 4. The system delay time of each class in a) rate sharing and b) buffer sharing

As shown in Fig. 5-6, the utilization of each class and the system delay time of the compared algorithms. In Karagiannis's algorithm, the lower priority classes lost the resource that high priority class did not acquire. In the proposed controls, the medium priority class could achieve higher utilization than the guaranteed threshold and higher than the utilization of low priority class. Wei Wu et al.'s algorithm serviced more load from the low priority class than load from the medium priority class, because classes that violated their guaranteed resource were intended to relatively share unused resources based on the priority weights. However, because of the implementation of rate distribution that only function well when the server's status is overload, some resource may be left unused when the overload trigger parameter does not reflect status of the current arrivals well. Although our proposed controls and Wei Wu, et al.'s algorithm all provided the differentiated services among classes in term of the system delay time, our controls provide less fluctuation. In Karagiannis's algorithm, the system delay time of medium priority class was higher than that of low priority class because of large token accumulations caused by the selection of the token buffer size.

## IV. Conclusion

In this paper, we discuss the first part of a study on adaptive signaling overload control for cellular networks. The second part which integrates overload control decisions with the state of transport network is in [9]. The proposed overload controls allows tradeoff between providing guaranteed service and resource sharing among multiple classes, allowing adaptive and scalable control. Guaranteed level of service can be accomplished as well as better utilization of the processor capacity while differentiated services in utilization and the system delay time can be maintained.

## Acknowledgments

This work was supported by TOT Corporation Public Co. ltd. Saowaphak would like to express her gratitude to OPNET Technologies in providing the simulation tool for this study.

## References

[1] W. Enck, P. Traynor, P. McDaniel, and T. LaPorta, "Exploiting open functionality in SMS-capable cellular networks," in *Proceedings 12th ACM Conference on Computer and Communications Security (CCS'05)*, Alexandria, VA, Nov. 2005.

[2] T. Lewis, *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation, forthcoming*. Wiley-Interscience, Apr. 2006.

[3] S. Kasera, J. Pinheiro, C. Loader, T. LaPorta, M. Karaul, and A. Hari, "Robust multiclass signaling overload control," in *Proceedings of 3th IEEE International Conference on Network Protocols (ICNP'05)*, Nov. 2005, pp. 246–258.

[4] B. D. Choi, S. H. Choi, B. Kim, and D. K. Sung, "Analysis of priority queuing system based on thresholds and its application to signaling no. 7 with congestion control," *Computer Networks*, vol. 32, no. 2, pp. 149–170, Feb. 2000.

[5] R. A. Farel and M. Gawande, "Design and analysis of overload control strategies for transaction network databases," in *Proceedings of the 13th International Teletraffic Congress (ITC 13)*, Copenhagen, Denmark, June 1991, pp. 115–120.

[6] D. E. Smith, "Ensuring robust call throughput and fairness for SCP overload controls," *IEEE/ACM Transactions on Networking*, vol. 3, no. 5, pp. 538–548, Oct. 1995.

[7] W. Wei, Y. Fangchun, and Z. Hua, "The study on overload control of application server in next-generation networks," in *Proceedings International Conference on Communication Technology (ICCT'03)*, vol. 2, Apr. 2003, pp. 1429–1432.

[8] G. Karagiannis, "Scalability and congestion control in broadband intelligent and mobile networks," Ph.D. dissertation, Twente University, P.O. Box 217, 7500 AE Enschede, the Netherlands, June 2002. [Online]. Available: http://doc.utwente.nl/fid/1363

[9] S. Sasanus and D. Tipper, "Adaptive multi-class signaling control for cellular networks," in *IEEE Symposium on Computers and Communications (ISCC'07)*, 2007.

[10] A. W. Berger and W. Whitt, "A multiclass input-regulation throttle," in *Proceedings of the 29th IEEE Conference on Decision and Control (CDC'90)*, vol. 4, Honolulu, Hawaii, Dec. 1990, pp. 2106 – 2111.

[11] D. Grillo, "Personal communications and traffic engineering in itu-t: the developing e.750-series of recommendations," *IEEE Personal Communications*, vol. 3, pp. 16 – 28, Dec. 2006.

[12] S. Kasera, J. Pinheiro, C. Loader, M. Karaul, A. Hari, and T. LaPorta, "Fast and robust signaling overload control," in *Ninth International Conference on Network Protocols*, Nov. 2001, pp. 323 – 331.

[13] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*. Addison-Wesley, 1997, Dec. 1998.

[14] A. W. Berger, "Overload control using rate control throttle: Selecting token bank capacity for robustness to arrival rates," *IEEE Transactions on Automatic Control (AC'91)*, vol. 36, no. 2, pp. 216–219, Feb. 1991.

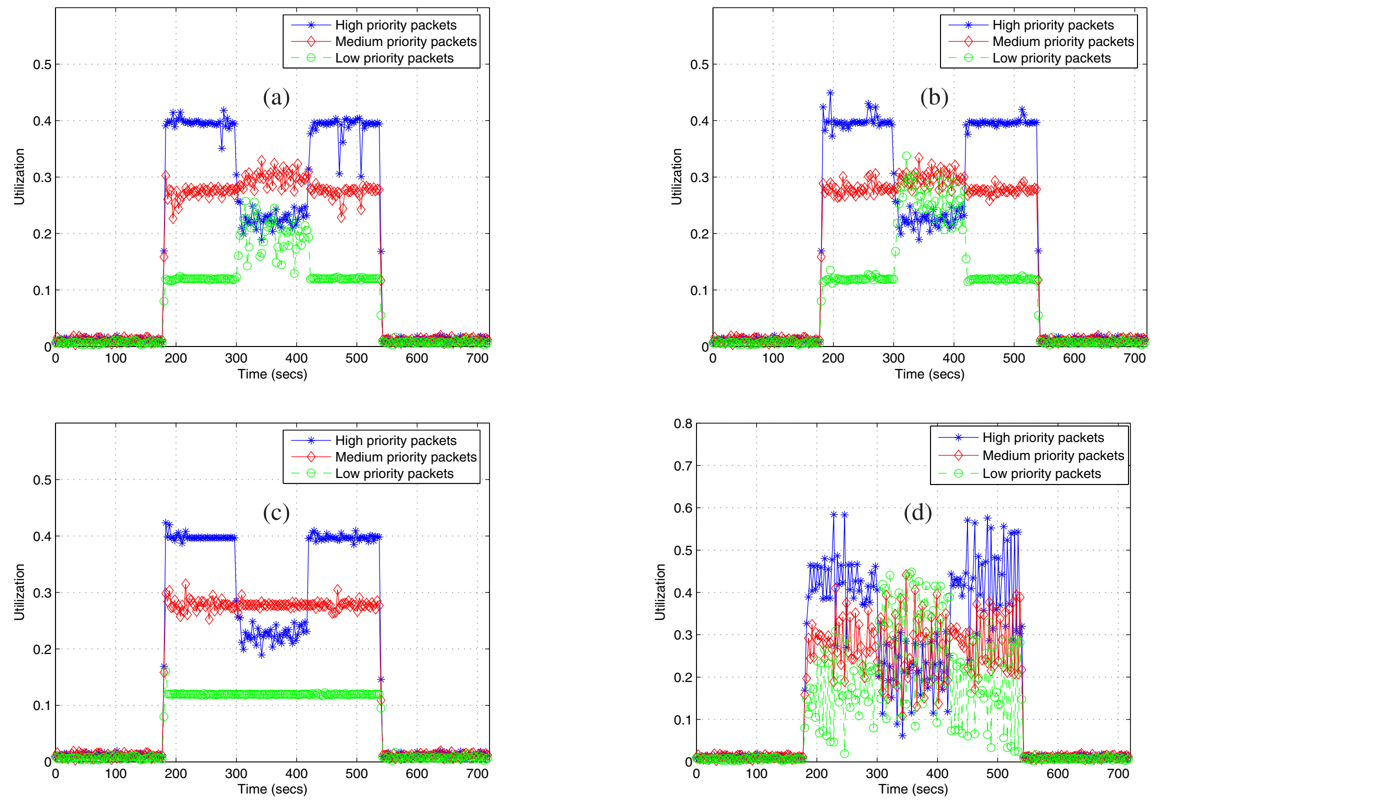


Fig. 5. The utilization of each class in a) rate sharing b) buffer sharing, c) the Karagiannis' algorithm, and d) Wei Wu, et al.'s algorithm

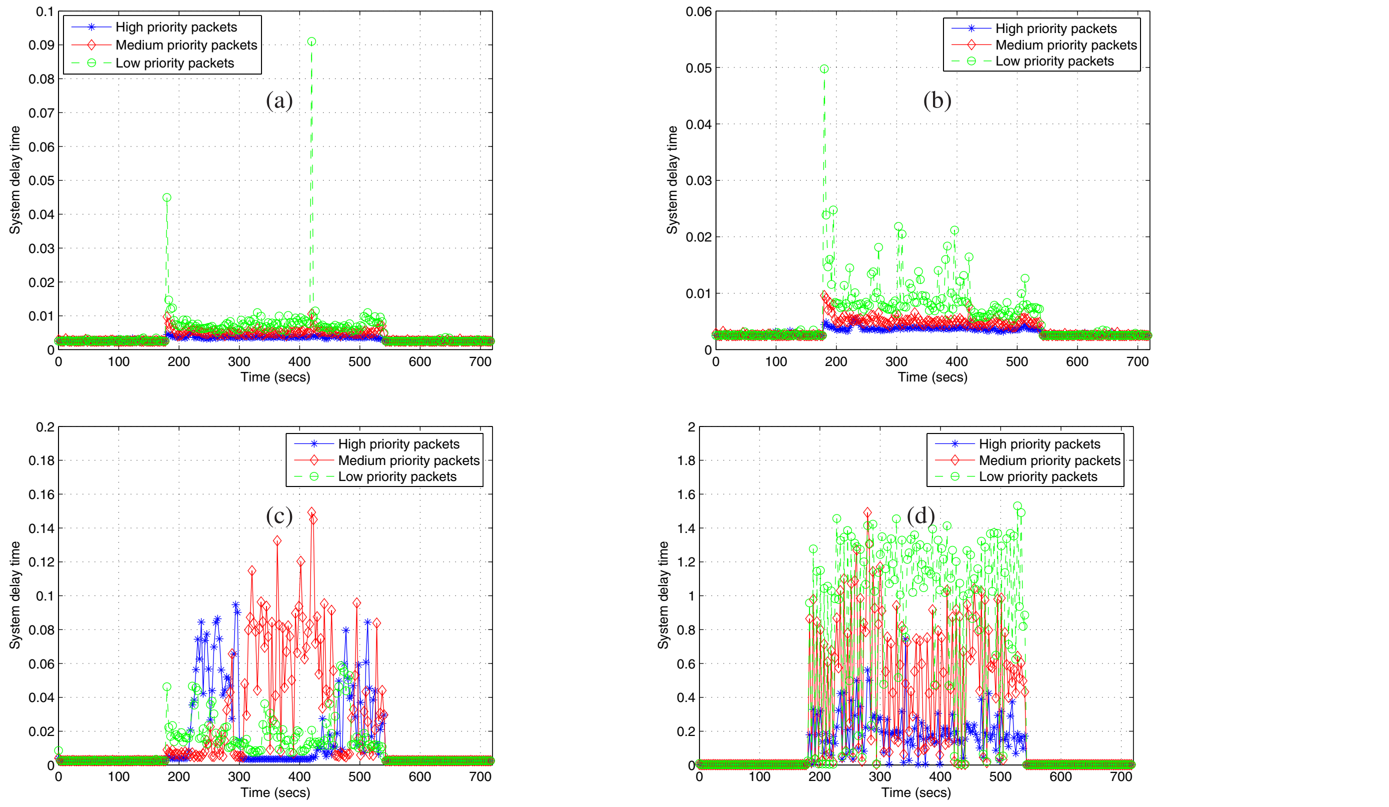


Fig. 6. The system delay time of each class in a) rate sharing b) buffer sharing, c) the Karagiannis' algorithm, and d) Wei Wu, et al.'s algorithm