This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Implementation and Simulation of DBHPD and CBQ scheduling - A Comparative Study

Johanna Nieminen*, Marko Luoma*, Olli-Pekka Lamminen* and Antti Paju†
*Helsinki University of Technology Otakaari 5A Espoo, FIN 02015, Finland
Email: {Johanna.Nieminen, Marko.Luoma, Olli-Pekka.Lamminen}@netlab.tkk.fi
†Creanord Oy Hämeentie 153 C 00560 Helsinki
Email: Antti.Paju@creanord.com

***Abstract*— In this paper we analyze the operation of two scheduling algorithms, Class Based Queuing (CBQ) and Delay Bounded Hybrid Proportional Delay (DBHPD). We compare results obtained from simulations and measurements in a reasonably equivalent setup and state that considerable differences can be observed in the results produced by these two methods. We also perform kernel profiling to investigate the resource overhead caused by these algorithms. Kernel profiling results support the observation that deviations between simulations and measurements are mainly caused by complex measurement and calculation procedures that generate extra delays in real implementations. As a conclusion we suggest criticism to be used when making judgements about algorithm performance based on simulation results.**

I. Introduction

Packet scheduling is one of the oldest and most important mechanisms for providing Quality of Service (QoS). Numerous scheduling algorithms have been proposed during the last decades from conventional static algorithms such as Generalized Processor Sharing (GPS) and Weighted Fair Queueing (WFQ), Deficit Round Robin (DRR) and Earliest Due Date (EDD) to adaptive approaches ([3], [5], [2]).

Often these algorithms are evaluated only with simulations. Judgements and conclusions are made about algorithm performance based on the simulation results even though simulation is an ideal simplification and abstraction of real world. In discrete event simulations, (like with ns2) router operations such as classification, enqueue, forwarding and dequeue do not consume any time while in real world the effect of packet processing can be considerable. Algorithms which utilize time or resource consumption in scheduling decision require several processor cycles to resolve dispatching order. Some of these cycles are used during the enqueue event of packet but mostly during the dequeue process where several packets are investigated in parallel against the configured resource sharing policy.

The goal of this paper is to evaluate two different types of scheduling algorithms: Class Based Queuing (CBQ) and Delay Bounded Hybrid Proportional Delay (DBHPD). CBQ is a well-known hierarchical bandwidth sharing algorithm proposed by Floyd and Jacobson [4]. DBHPD is our own delay-adaptive algorithm proposed in [1]. It is based on proportional delay constrained scheduling making it suitable for application differentiation: it provides low delay service for real-time traffic and allocates resources between the other classes based on delay ratios.

Evaluations are made by using simulations with ns2 (optimistic) and implementation in PC-based testplatform (pessimistic). Algorithms are compared not only head to head (in simulations and in implementation) but also how their performance changes from the simulation to implementation. This should reveal their relative computational complexity and also how well they cope in real-time operation. We also perform kernel profiling for both DBHPD and CBQ implementation in terms of achievable throughput, time consumed in enqueue and dequeue operations as well as the number of function calls used in these operations. These profiling results provide a view of the router processing overhead and can thus be utilized in explaining the differences between simulations and measurements.

We argue that simulations (with their idealized execution environment) together with implementation measurements (in highly unoptimized environment of PC platform) form a good combination to investigate the overall performance of scheduling algorithms. Actual performance of these algorithms in embedded platforms lies somewhere in between.

In [6] we showed the very first implementation of the DBHPD algorithm in a FreeBSD based ALTQ router. We conducted several measurements with CBQ and DBHPD to investigate their performance. However, these results were obtained indirectly by utilizing traffic probes sending small packets at regular intervals to different traffic classes. Delay distributions were recorded only for the probes and the actual distribution of application packets were argued to resemble the distribution of probes. In this paper all measurement results are based on the data obtained directly from TCPDump as depicted in Figure 1, thus providing more accuracy. In this paper we also compare measurement results to simulation results obtained in an equivalent setup in order to determine the effect of real world phenomena on the performance of scheduling algorithms.

This paper is organized as follows. Section 2 explains the operation of selected scheduling algorithms. It gives short introduction to essential aspects of these algorithms. In Section 3 we have a short review of our measurement and simulation setup. Comprehensive representation is given in [6]. Section 4 presents comparative results from the measurements and

1-4244-0353-7/07/$25.00 ©2007 IEEE

simulations and Section 5 presents results from the kernel profiling. Section 6 concludes our work.

II. Description of the Algorithms

This section presents the theoretical models used in simulations for DBHPD and CBQ algorithms. It should be noted that the theoretical models may deviate from the real implementations since the hardware and software set certain limits for what can be implemented in practice. Implementation details of these algorithms are given in [6].

A. Class Based Queueing (CBQ)

Class Based Queuing (CBQ) [4] is the most well known hierarchical bandwidth sharing algorithm. In CBQ the roles of the different hierarchy levels are following:

- **Root class** contains the link resource that is to be divided among the traffic classes.
- **Leaf classes** represent the actual traffic classes that are served by the link.
- **Intermediate classes** are responsible for resource sharing among the leaf classes. The intermediate classes act as parents for the leaf classes, allowing the leaf classes to borrow resources.

The resource allocation in CBQ is enforced with two different schedulers: a general scheduler and a link sharing scheduler.

- **General scheduler** is all that is required when initial provisioning of resources is adequate for each individual leaf class.
- **Link sharing scheduler** is required when some of the leaf classes get congested. In this case, the link sharing scheduler enforces rules by which the congested classes can borrow resources from the parent classes.

The actual implementation for the general scheduler and the link sharing scheduler has not been defined. The idea is that any rate-based scheduler, such as WRR or DRR can be used as the general scheduler. Only the high-level operation of link sharing scheduler is defined. Actual rules and heuristics for borrowing the resources are left for implementations.

B. Delay-bounded HPD (DBHPD)

Scheduling decision in DBHPD is dividen into two phases. In phase one accumulated queueing delay ($t_{curr}(m) - t_{in}(m)$) of head of line (HOL) packet (m) in delay bounded class is checked against delay bound d_{max} and safety margin t_{safe}. Packet is scheduled next to the link if the delay bound of HOL-packet (m) is considered to be violating its bound.

$$t_{curr}(m) - t_{in}(m) + t_{safe} > d_{max} \tag{1}$$

If delay violation is not occurring phase two of algorithm is executed.

In phase two the algorithm calculates the normalized hybrid queueing delays of HOL-packets in each of the classes. The calculation is based on the exponentially weighted moving average of experienced queueing delay of class i packets

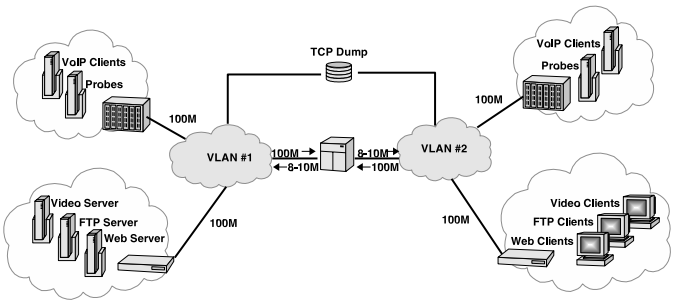


Fig. 1. Measurement network topology.

($\bar{d}_i(m-1)$) and HOL-packet waiting time ($\bar{w}_i(m)$). These delays are normalized with Delay Differentiation Parameter (DDP) of class i.

$$\tilde{d}_i(m-1) = \frac{\bar{d}_i(m-1)}{\delta_i} \quad , \quad \tilde{w}_i(m) = \frac{\bar{w}_i(m)}{\delta_i} \tag{2}$$

The DBHPD algorithm selects for transmission at time t, when the server becomes free, a packet from a backlogged class j with the maximum normalized hybrid delay [3]:

$$j = \arg\max\left(g\tilde{d}_i(m-1) + (1-g)\tilde{w}_i(m)\right), \tag{3}$$

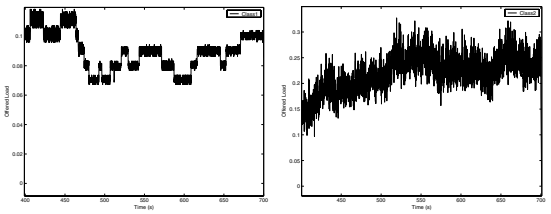
$0 \leq g \leq 1$ is a weighting coefficient between short $\tilde{w}_i(m)$ and long term $\tilde{d}_i(m)$ queuing delays.

III. Measurement and Simulation Setup

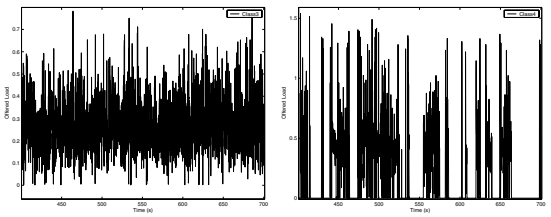
The measurement network topology is shown in Figure 1. The simulation topology was the same except that VLANs and TCPDump were not used and all traffic sources had their own access link to the edge router.

Four different applications were represented in the evaluation: FTP, HTTP, Video Streaming and VoIP. In the measurements, FTP, HTTP and QuickTime Video Streaming traffic was produced with Spirent's test appliances. VoIP traffic was emulated by defining suitable traffic generation pattern, however the actual codec and protocol was missing. In simulations the traffic processes produced by these applications were somewhat different. First, the webcache model used in the simulations for FTP and HTTP traffic deviated from the FTP and HTTP traffic creation process used in appliance. There were also minor differences in the VoIP and Video traffic patterns. The differences in the load patterns were mainly caused by the fact that appliance offers only very basic patterns for starting new sessions. For instance, in the measurements HTTP and FTP session arrival process are flat or sinusoid with a constant phase while in the ns2 webcache model session interarrivals are drawn from an exponential distribution. A flat or sinusoid session arrival process used in the measurements clearly results in more periodic and deterministic behavior. Some deviation in the load processes may also be caused by the differences in TCP implementation details in ns2 and test appliance.

In both simulations and measurements desired load level was generated by changing the bottleneck link capacity. In

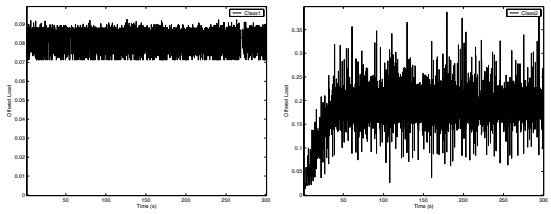


(a) Class 1, VoIP (b) Class 2, Video

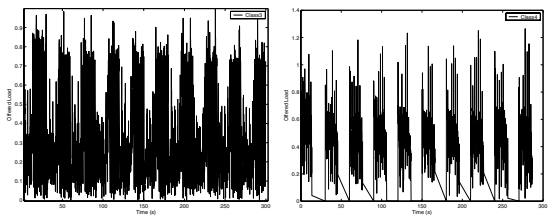


(c) Class 3, WWW (d) Class 4, FTP

Fig. 2. Offered class loads in the simulations (Total offered load 90%)



(a) Class 1, VoIP (b) Class 2, Video



(c) Class 3, WWW (d) Class 4, FTP

Fig. 3. Offered class loads in the measurements (Total offered load 90%)

measurements link capacity can be changed with token bucket rate control at dequeue events. Software based implementation of token bucket rate control is prone to small errors caused by low resolution of kernel timer. This error causes the actual link capacity to vary around the correct link capacity. Figure 2 and Figure 3 show the offered load patterns of different traffic types in the simulations and in the measurements for a desired average offered load of 90%. It can be observed that the average amount of offered traffic is the same in both cases. However, due to these differences in the temporal properties of load patterns within individual traffic classes, the simulation and measurement results can be compared only at a relatively coarse level.

The mapping of traffic into different scheduling classes was performed so that the VoIP uses the first class (delay bounded in DBHPD), Video second class, HTTP third class and FTP fourth class. The relative traffic shares for different applications were following: (FTP: 30%, WWW: 40%, Video: 20%, VoIP: 10%).

Provisioning for scheduling algorithms was performed based on best knowledge i.e. traffic was known in advance and resource provisioning was set to meet the service requirements with our traffic mix. Provisioning of CBQ was performed by simply allocating each class a bandwidth share that corresponds to the load fraction of that class. CBQ hierarchy and provisioned bandwidth fractions are shown in Figure 4. In delay-bounded HPD the delay-bound (d_{max}) was set to 5ms, safety margin (t_{safe}) to 0, long-range delay EWMA coefficient (γ) to $1/2^{10}$, long-term short-term weighting coefficient (g) to 0.875 and the delay ratios to 4. Queue lengths in both algorithms are set to 15, 15, 80 and 200 packets.

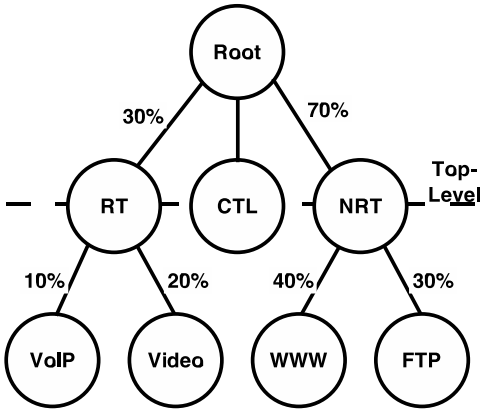


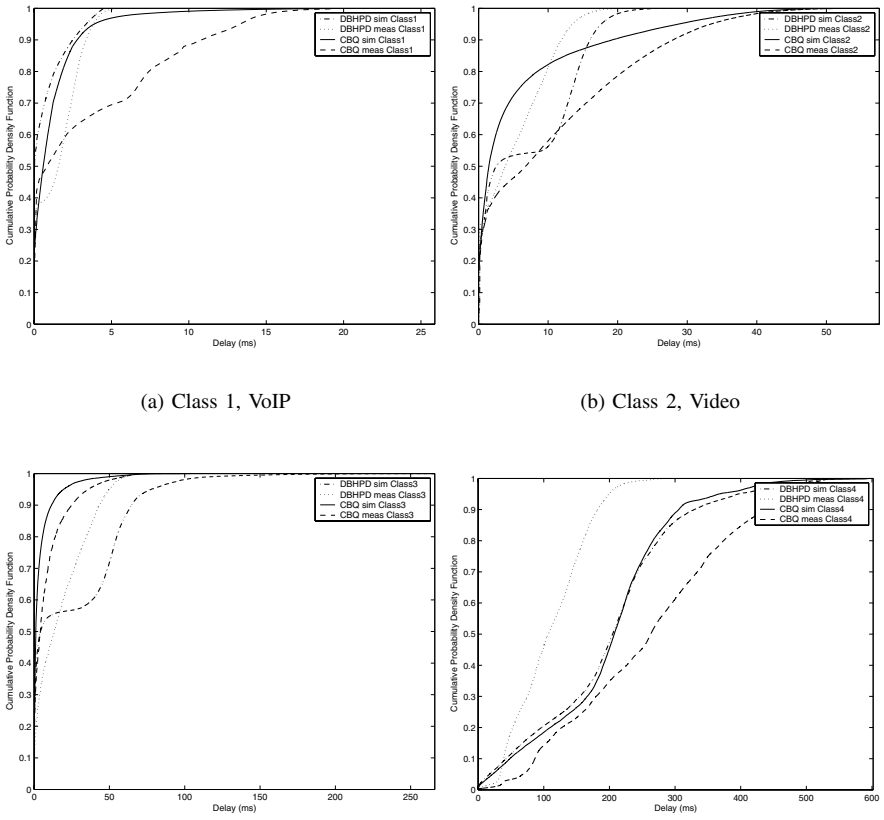
Fig. 4. CBQ link-sharing hierarchy

IV. Results of the Simulation and Measurement Comparison

In this section we present performance statistics in terms of queuing delay distributions and bandwidth allocations for both simulations and measurements of DBHPD and CBQ algorithms.

A. Delay distributions

Figure 5 presents the delay distributions for each traffic class when the target load level is 90%. The ranges of delay values seem to be relatively close to each other with both algorithms but there are clear deviations in the medians and exact delay distributions for both algorithms. The deviations can partly be explained by the differences in the offered load processes. However, most of the difference is due to the simplifications used in real implementation as well as overhead caused by the estimation procedures. For CBQ simulations



(a) Class 1, VoIP

(b) Class 2, Video

(c) Class 3, WWW

(d) Class 4, FTP

Fig. 5. Delay distributions for DBHPD and CBQ simulations and measurements (Load 90%)

tend to provide smaller delays for each class compared with measurements. This is natural since demanding borrowing operations occur frequently with 90% load that cause extra delays for the packets in a real router. It should also be noted that in measurements operations performed before dequeue, such as packet header analysis, classification and enqueue add some extra delay that is not modelled in simulations.

As we already concluded from the measurement results in [6], also in simulations CBQ provides smaller delays for the HTTP class at the expense of the real time classes. Even with a relatively small load the maximum delay for VoIP is 5 ms for DBHPD and 20 ms for CBQ and the maximum delay for Video is 20 ms for DBHPD and nearly 50 ms for CBQ.

Since this observation can be made from both simulation and measurement results, it can be deduced that high delays for real-time traffic are caused by the CBQ's bandwidth allocation mechanism itself, not by router processing overhead.

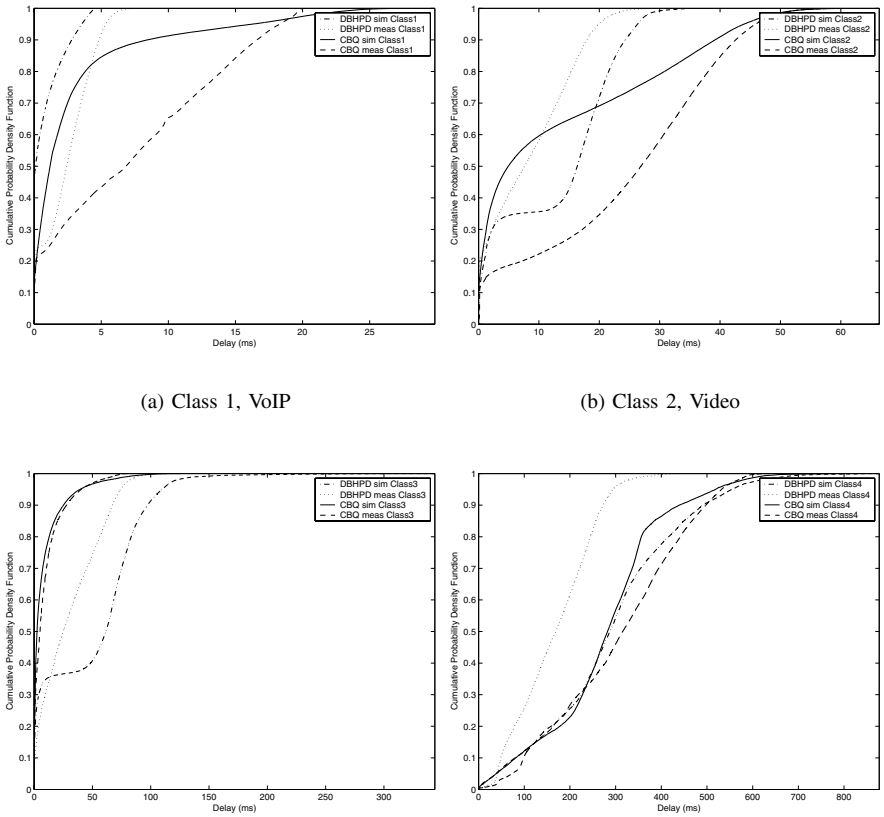
Figure 6 show the corresponding delay distributions with offered load of 100%. From the distributions CBQ's problem of reversed service order between class 2 and 3 can be observed for both simulations and measurements. Thus, even though CBQ might provide smaller delays than DBHPD for some classes, the overall service differentiation is not as predictable when offered load increases.

The results also show that as the offered load closes to 100% the deviation between CBQ measurement and simulation results decreases. This is due to the fact that with high load each traffic class is using its fixed share of resources and thus expensive borrowing operations do not occur. The algorithm mainly operates as the DRR general scheduler, which is a basic bandwidth sharing algorithm.

B. Bandwidth allocation

Figure 7 and Figure 8 present bottleneck router bandwidth allocations for the traffic classes with offered load level of 100% in both simulations and measurements. From these figures it is evident that DBHPD allocates more bandwidth for the real-time classes while CBQ provides more capacity for the non-real-time classes. This corresponds to the delay results presented earlier where DBHPD guaranteed small delays for the real-time classes and CBQ provided clearly smaller delays for the HTTP traffic at the expense of VoIP and Video delays.

It should be noted that especially in the CBQ measurements there are clear darker bands near the guaranteed minimum capacity for each class, suggesting that at these points CBQ operates as a basic bandwidth sharing algorithm. DBHPD does not provide a guaranteed bandwidth but the capacity allocation fluctuates according to the incoming load and queuing delays.



(a) Class 1, VoIP

(b) Class 2, Video

(c) Class 3, WWW

(d) Class 4, FTP

Fig. 6. Delay distributions for DBHPD and CBQ simulations and measurements (Load 100%)

TABLE I
ACHIEVABLE THROUGHPUT

Scheduler	Throughput (Mbps)	Frames/s	Relative
PRIQ	84.38	65 322	0.987
CBQ	65.31	50 528	0.764
DBHPD	70.51	54 579	0.824

TABLE II
RESOURCE CONSUMPTION IN DEQUEUE AND ENQUEUE OPERATIONS

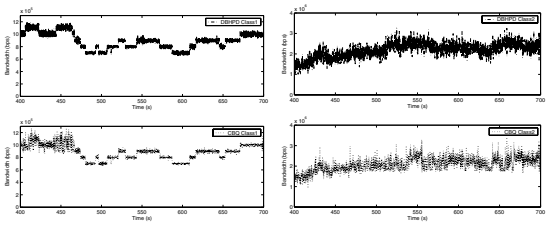
	Cumulative time (s)		Function calls	
Scheduler	Enqueue	Dequeue	Enqueue	Dequeue
PRIQ	0.25	0.07	3998480	2998860
CBQ	0.35	0.87	7996792	13994386
DBHPD	0.18	0.24	6997254	5997648

In simulations the CBQ bands are not that clear. This is most likely due to the fact that the implemented algorithm does not operate exactly according to the theoretical model used in the simulations and thus is not able to perform so sophisticated borrowing.

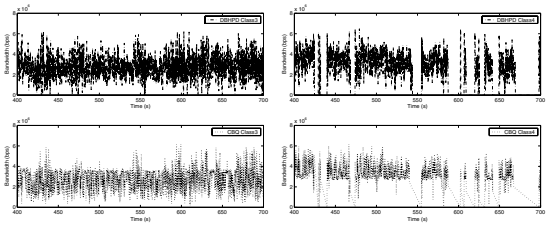
V. KERNEL PROFILING RESULTS

We also performed initial kernel profiling for simple PRIQ (Priority Queueing), CBQ and DBHPD implementation for investigating the achievable throughput and resource overhead of these algorithms. Resulting throughputs for PRIQ, CBQ and DBHPD are shown in Table I. It can be observed that throughput is highest for PRIQ algorithm since they do not use any estimation or time measurement procedures for scheduling. Correspondingly, DBHPD is able to achieve better throughput than CBQ due to its more simple implementation.

In order to examine how much more resources CBQ consumes compared to DBHPD one million TCP/IP packets of size 110 bytes were sent and processed by both of the algorithms with a sending rate of 15000 packets/s. Table II shows the cumulative time consumed in enqueue and dequeue operations as well as the number of function calls used by these algorithms. Results for PRIQ are also shown for comparison. The cumulative time used in enqueue and dequeue operations as well as the number of function calls is in line with the throughput results and confirm the observation that CBQ dequeue function results in significantly larger overhead than DBHPD's dequeue. PRIQ has smallest overhead but it lacks control from the resource sharing, thus making it unsuitable for providing reasonable differentiation.

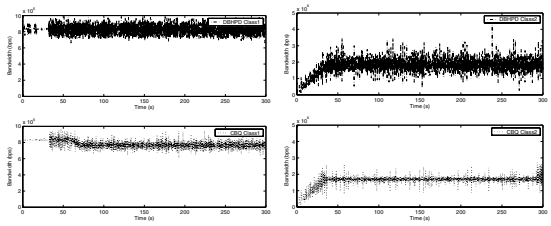


(a) Class 1 (b) Class 2

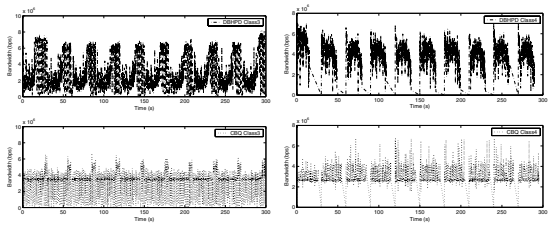


(c) Class 3 (d) Class 4

Fig. 7. Bandwidth allocations for DBHPD (top) and CBQ (bottom) in simulations (Load 100%)



(a) Class 1 (b) Class 2



(c) Class 3 (d) Class 4

Fig. 8. Bandwidth allocations for DBHPD (top) and CBQ (bottom) in measurements (Load 100%)

VI. Conclusions

In this paper two scheduling algorithms, Class Based Queuing (CBQ) and Delay Bounded Hybrid Proportional Delay (DBHPD) were evaluated. Evaluations were made with discrete event simulator (ns2) and with prototype implementation in FreeBSD operating system. According to the results prototype implementations of the algorithms produced results that were in the same range of magnitude with the simulation models. However, the medians, exact shapes of the delay distributions and bandwidth allocations exhibited clear differences. The deviations between simulations and measurements can be explained by the differences in the offered load processes and the simplifications used in the real implementations as well as overhead caused by the estimation procedures. For example, in CBQ, demanding borrowing operation forces to use approximations in implementation. These approximations cause deviation from the ideal scheduling order. However, the major difference comes still from the fact that discrete event simulation lacks the effect which comes from the processing of the packets within a router. These effects are additional delays and performance bottlenecks that cannot be modelled with analytical formulas nor with easy approximations. The more complex the scheduling algorithm is the more time it takes to process the packet and less time is devoted to other actions within a router. The performance of the processor in contrast to link speed ultimately dictates the operational range of popular open source software routers. High volume of interrupts from the network interfaces and computational complexity of scheduling algorithms usually limit the performance of these devices to some tens of megabits per second or some tens of thousands packets per second.

These results were further confirmed with kernel profiling that indicated high number of function calls and high cumulative time used in enqueue and dequeue operations with the advanced scheduling algorithms. The results presented in this paper suggest that any algorithm on the datapath should be evaluated carefully with both simulations and measurements on realistic scenario before making any judgements about their performance. This paper presented the behavior of scheduling algorithms in one router case where traffic is destined into single output link. Also all of the servers reside on the same side of the network causing highly asymmetric network usage. However, results show comparative differences between algorithms and evaluation methods. As a future work, we are going to evaluate the operation in real network scenario where we have multiple routers, cross traffic and distributed servers and clients througout the network.

References

[1] J. Antila and M. Luoma, "Scheduling and quality differentiation in differentiated services," in *Proceedings of Multimedia Interactive Protocols and Systems (MIPS)*, November 2003.

[2] N. Christin, J. Liebeherr, and T. Abdelzaher, "A quantitative assured forwarding service," in *Proceedings of IEEE Infocom*, 2002.

[3] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 12–26, February 2002.

[4] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 365–386, 1995.

[5] A. Moore, "Measurement-based management of network resources, ph.d thesis, university of cambridge, computer laboratory," April 2002.

[6] J. Nieminen, M. Luoma, and A. Paju, "Measurements and implementation of an adaptive scheduling algorithm," in *Proceedings of CoNEXT*, October 2005.