Correct Router Interface Modeling

Krzysztof Rusek
krusek@agh.edu.pl

Lucjan Janowski
janowski@kt.agh.edu.pl

Zdzisław Papir
papir@kt.agh.edu.pl

Department of Telecommunications
AGH University of Science and Technology
Krakow, Poland

ABSTRACT

The aim of this paper is to determine how to model router interface in order to accurately predict packet drops. There is an enormous amount of research on traffic models reported, however, a model of router interface has not gained proper consideration yet. Our experiments reveal that an incorrect model of the router interface can result in a significant disparity between drop probabilities measured on a physical interface and derived from a trace-driven simulation study. In this paper an accurate model of the router interface (Cisco IOS-based routers with a non-distributed architecture) was presented.

Categories and Subject Descriptors

B.4.4 [**Input/Output and data communications**]: Performance Analysis and Design Aids— *Formal models, Simulation, Verification*; C.2.6 [**Computer communication networks**]: Internetworking—*Routers*

General Terms

Performance,Verification,Experimentation

Keywords

buffer, router, packet drops, storage policy

1. INTRODUCTION

Efficient dimensioning and effective management of packet network resources require both traffic descriptions and network devices (in particular an interface of a router) modeling. Surprisingly, while traffic models are validated very often, device models receive much less attention. According to the author's knowledge no research addressing accuracy of router queuing modeling has been published.

In order to model a network device one has to consider packet storage and packet service. Currently, there are at least two types of packet storage policies. A buffer with a byte oriented policy can store a fixed amount of data (in bytes) i.e. a large number of short packets or a small number of long packets. On the other hand, in a packet oriented policy, the buffer can store a fixed number of packets regardless of their size.

Note that the second policy is easier to implement since a router operating system reads a packet payload into a special memory and then operates only on payload pointers. Therefore, a router knows how many packets (pointers) it has in a queue, however, in order to know how many bytes are stored additional information (packet size) has to be read and processed.

Both storage policies have corresponding mathematical models (refereed in this paper as byte or packet oriented models). Byte oriented models happen to be more commonly used e.g. [9, 1]. Nevertheless, packet oriented models have been derived [6]. In fact, the default storage policy in the popular network simulator ns-2 is packet oriented [5].

A byte oriented approach is correct for some types of routers where packets are stored in chunks of memory [1], however, it is not adequate for all devices. Sometimes the byte oriented model of the buffer is dictated by the traffic model as it is for fluid approximation models (commonly used for self-similar traffic [7]) where a storage model has to be byte oriented in order to obtain a consistent model of a system. When a buffer is packet oriented, then both an input process and a service rate have to be measured in packets per second. However, in such a case the service rate is variable due to the different sizes of the packets. This problem is discussed in detail in [4].

Different storage policies would result in different Packet Loss Ratios, so when modeling a network device one has to precisely specify which storage model (byte or packet oriented) is used in order to adhere it to a traffic description which comes from network measurements [9, 6].

However, the router interface has a complicated internal structure, and it is not obvious, whether an approximation commonly used in a simulation software is correct, or if parameters used in simulations are equal to parameters of a real device.

The problem of a buffer policy and its parameters in principle can be solved by checking router documentation. The documentation of a Cisco router clearly states that the queue limit in popular Cisco routers is set in packets [2], so a packet policy is implied. Nevertheless, service is counted in bytes per second, therefore we can say that the real router is a mixture of both byte and packet oriented models. Buffer size is counted in packets (as for a packet oriented model)

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

and service is counted in bytes per second (as for a byte oriented model). Moreover, the experiments described in detail further on revealed that a queue size set in a router configuration is too small to reproduce observed packet drops. In order to demonstrate an accurate model of a router interface it is necessary to consider the internal structure of the interface.

In [8] Ivan Pepelnjak described a queuing process in Cisco IOS-based routers with a non-distributed architecture (for example, the ISR series commonly used in small to medium-sized businesses and enterprise branch offices). In Cisco high-end routers (ASR and GSR series used by ISPs) implementations differ significantly from the simple model described in [8]. The buffer described in [8] resembles a network of queues rather than a single queue. Therefore, it is evident that there is a gap between theory and practice. The goal of this paper is to bridge this gap.

In this article a realistic yet mathematically tractable model of a single interface in a router is presented. Model predictions match packet drops obtained in a real network with acceptable accuracy. Two main steps of router interface modeling were addressed. Firstly, it has been empirically proven, that the Cisco ISR router can be accurately described by queuing theory. Secondly, the parametrization of the proposed model was discussed.

This work is divided into five sections. In Section 2 a testbed and details of performed experiments are described. In Section 3 a model of a single router interface is discussed. Obtained results are presented and assessed in Section 4. Finally, Section 5 concludes the paper. Finally in the Appendix network measurement accuracy is discussed.

2. EXPERIMENT

The purpose of this article was to investigate what the storage policy in a real router is and how accurate in Packet Loss Ratio prediction its mathematical model is. Toward this end a series of two step experiments were performed. In the first step artificially generated traffic was transmitted through a lossy interface in the router with simultaneous sniffing of incoming and outgoing traffic. In the second step a trace-driven simulation was used to verify if the dropped packets are the same. All simulations were performed in MATLAB, using custom routines, though identical results can be obtained from another simulator e.g ns-2. The concept of the experiment is presented in Fig. 1.

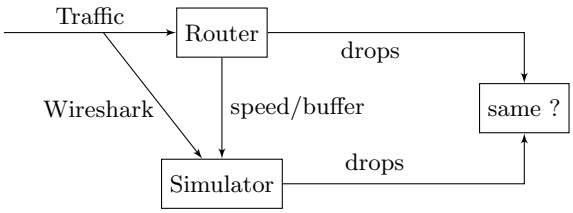

Figure 1: Concept of the experiment. We compared drops occurring in router and in trace-driven simulation

Each transmitted packet carried a unique ID in its payload, so it was possible to identify dropped packets by checking missing IDs in the output trace file. The experiments took place conducted in a controlled and isolated environment,so there were almost no packets outside the experiment.

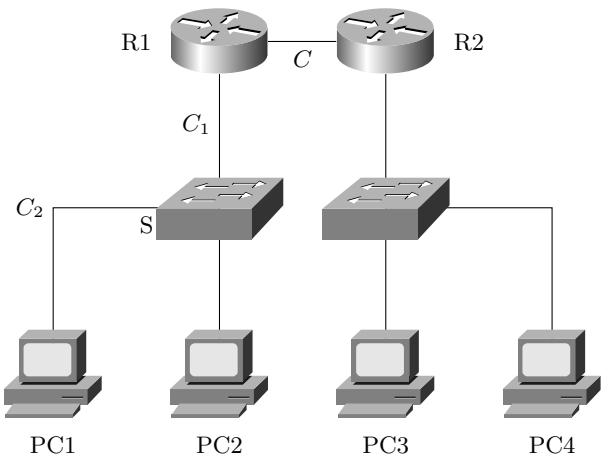

Figure 2: Network used for tests, R1 – tested router, PC2 – traffic generator, PC3 – traffic destination, PC1,PC4 – sniffers

A simple network depicted in Fig. 2 was set up for testing purposes. In the tests Cisco 2800 Series routers being typical of the ISR family were used. For all of the connections 100 Mbit Etherernet ($C_1 = C_2 = 100$) was used and the bottleneck was $C = 10$ Mbit link. Such a setup guarantees that drops occur only on a tested interface. The PC2 computer was a traffic generator and the PC3 was a traffic destination. For simplicity UDP was used as a transport protocol. Computers PC1 and PC4 served as sniffers, dumping respectively, incoming and outgoing traffic. The Wireshark application was used for traffic capturing. It is crucial for an experiment accuracy that links C_1 and C_2 have the same speed which is explained in the appendix. The tested router was R1 and the bottleneck was link $C = 10$Mbit between R1 and R2. One may consider the router R2 as redundant but such architecture allows replacing the Ethernet link R1R2 by a serial link.

In all the experiments surprising results were revealed. In the simulations the number of dropped packets was at least two times greater than the drop number observed in reality. In the next section we present a model that is able to accurately reproduce observed packet drops.

3. MODEL OF A ROUTER INTERFACE

A FIFO queue system is characterized by three components: input process, service process and buffer capacity. Each of these three components have to be fitted in order to create an accurate router interface model. In the case of modeling the router, the storage policy is also important. The input process comes from traffic measurements while queue size is obtained from router configuration. Combing the information from a router configuration (a link speed) and traffic measurements (packet lengths) one can derive the service process.

3.1 Buffer Capacity

Each interface in the router has an input and an output queue. When a packet enters the router, it is queued in

the input queue of the incoming interface and waits to be processed. If the input queue is full the packet is dropped. Input queue drops generally occur when processing is too slow, so it is a hardware related issue [2]. Output drops on the other hand are caused by a congested interface (the outgoing interface cannot send all packets) [2].

The output queue is more complicated than a simple FIFO [1] buffer [8] as it is split into at least two queues: one hardware queue (tx ring) and one or more software queues. The size of the software queues is set by a `hold-queue` command whereas hardware queues are managed by the Cisco IOS. The size of the additional hardware queue can be verified by a `show controller` command in a tx_limit section. The hardware queue is shared between the CPU and the interface chipset. Its size is computed automatically by Cisco IOS (the router tries to minimize the hardware queue size) and depends on the interface speed. For interfaces with FIFO discipline, the hardware queue can store up to 128 packets.

All queuing mechanisms offered by the Cisco IOS performe in the software queue. After packet processing the CPU inserts a packet into the shared hardware queue where it waits for transmission. When the hardware queue is full a packet is queued in the software queue. The interface picks up packets from the hardware queue when it is ready to transmit another one. If the hardware queue is empty, the interface enters an idle state. In an idle state the interface hardware periodically polls the hardware queue and starts the transmission as soon as it finds a new packet. During normal operations the interface interrupts the main CPU when it needs more packets to transmit. On slow interfaces interruption may occur on every successfully transmitted packet. Because packets are transferred from the software queue to the shared hardware queue very often it was assumed that *hardware and software queues may be modeled as single queue where size is equal to sum of the sizes of both of the components*. Such an assumption happens to be very accurate.

3.2 Service Process

The last but not least part of a router model is the service process. Simulation software does not typically require explicitly service time but rather frame lengths and an interface speed are required. Service time τ is then computed according to following formula

$$\tau = \frac{l}{C}, \tag{1}$$

where l is a layer 2 frame length in bits and C is the interface speed in bits per second. Precise calculation of service times requires precise information about a protocol stack (overhead) and the interface speed.

The overhead added by protocols can be easily obtained from a documentation. Unfortunately, some protocols (e.g. MAC in Ethernet [10]) have interframe gaps (IFGs) between frames. In order the equation (1) to be still valid for those protocols, the gap was treated as part of the frame thus increasing its length.

Two popular protocols MAC in Ethernet (10M) and Cisco HDLC (version of HDLC modified by Cisco) were investigated. The Ethernet frame without VLAN adds a 38B overhead to an IP packet(including IFG [10]). The Cisco HDLC frame has a smaller overhead (only 8B per IP packet), but the protocol is more complicated than the Ethernet because each frame starts and ends with one octet frame flag and two frames may share the same flag. When sharing the flag between successive frames, the ending flag in the first frame is also treated as the starting flag of the next frame [3]. Sharing flags has a small impact on a model accuracy in highly congested interfaces when almost all flags are shared.

The link speed can be obtained from a router configuration, however, it was decided to measure the service time directly which resulted in slightly better accuracy. The service time was measured as interarrival time of the output from the highly congested interface flooded by packets with known length. The link speed for the simulation software was computed according to equation (1).

[1]We assume, that the interface is configured for FIFO discipline.

4. RESULTS

Applying a queue parametrization described in Section 3 results in an accurate Packet Loss Ration prediction on a real device. Further on the experimental results are presented. We conducted several experiments with different types of generated traffic. Each experiment was conducted for both fixed and variable packet lengths in order to be certain about the storage policy.

In the variable packet length scenarios sizes ware drown from discreet random variable taking values 48B and 1408B witch equal probability. Size distribution was chosen to mimic size distribution observed in real traffic. Exact packet size distributions can be found in e.g. [9]. The experiment was performed for both Ethernet and serial links at different level of utilization (including the overloaded state) and different buffer capacities. Because real packets have variable lengths only such results are presented. We mostly concentrated on Ethernet interfaces but the model is also valid for serial interfaces.

4.1 Drop Function

Suppose that from M input packets N was dropped for both the model and the real queue. Note that it does not indicate that both queues behave similarly. Knowing a number of dropped packets in one experiment is not enough, because those drops can be achieved in many different ways. We needed more information about the drop process, therefore we introduced drop function $d(i), i \in \mathbb{N}$, a more fundamental characteristic describing drop process at a deeper level. Drop function $d(i)$ has two values:

$$d(i) = \begin{cases} 0 & \text{if the } i\text{-th packet was not dropped} \\ 1 & \text{if the } i\text{-th packet was dropped} \end{cases} \tag{2}$$

and fully describes the drop process, but it is hard to visualize on a plot. For cleaner drop presentation we use the cumulative drop function defined as:

$$D(i) = \sum_{k=1}^{i} d(k). \tag{3}$$

The cumulative drop function is simply the number of dropped packets from the beginning of the observation up to the i-th packet arrival. Further,if there is no explicit name, the term drop function means cumulative drop function.

Drop function defined in a such way is an easy tool to compare two drop systems. When the difference of two drop

functions for the compared systems is equal to zero for all possible inputs then those systems are indistinguishable in the sense of the drop process.

The drop function is a primary tool used to validate the model used in this work. In an ideal case the drop function obtained from router (D_r) and the one from simulation (D_s) should be the same. In reality there is always some level of uncertainty. Also a model may simplify reality too much, therefore we expect the drop function to be close but not identical. To measure the accuracy of the model we used the difference between experimental and simulation drop functions D_Δ named the error function.

$$D_\Delta(i) = D_r(i) - D_s(i), \quad i \in \mathbb{N} \tag{4}$$

By dividing equation (4) by i, one can find that quantity

$$D_*(i) = \frac{D_\Delta(i)}{i}, \quad i \in \mathbb{N} \tag{5}$$

is the difference between real and simulated Loss Ratios named relative error function.

4.2 Model verification

Our main proposition was to bind software and hardware queues into one queue. We tested this proposition for three different software queues for the Ethernet[2] interface at the utilization level $\rho = 0.8$. The error functions are presented in Fig. 3,4 and 5. Each of the presented error charts is

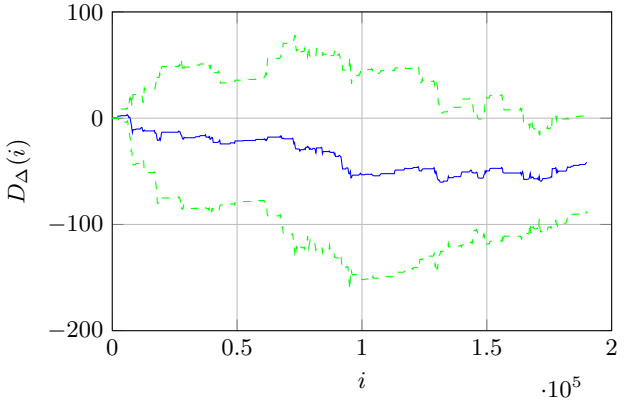

Figure 3: Error function for 20 packet queue, $\rho = 0.8$. Solid line – error function, dashed line – 95% confidence interval

a mean value of error functions for five experiments with the same input traffic. Observed error functions are very chaotic and quite large even for very similar input traffic. Confidence intervals for the observed error functions are also depicted In Fig. 3,4 and 5 to show the dispersion of results. Although the error is random, 0 value is almost always in 95% confidence interval, so statistically our model is accurate. Obtained results indicate that the chosen model and its parametrization are correct.

Besides the random nature of the errors, they have a clear trend. That kinds of trends may be caused by biases in link speed or queue size estimation. Changing the queue size by ± 1 leads to a worst model so we believe that the error is caused by bias in the link speed estimator and other

[2]Similar results were observed for serial interface

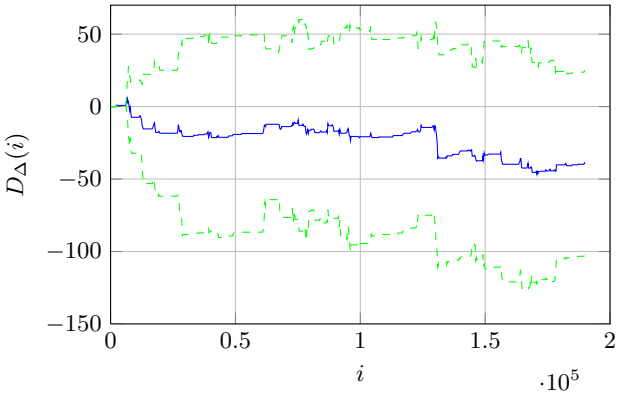

Figure 4: Error function for 40 packet queue, $\rho = 0.8$. Solid line – error function, dashed line – 95% confidence interval

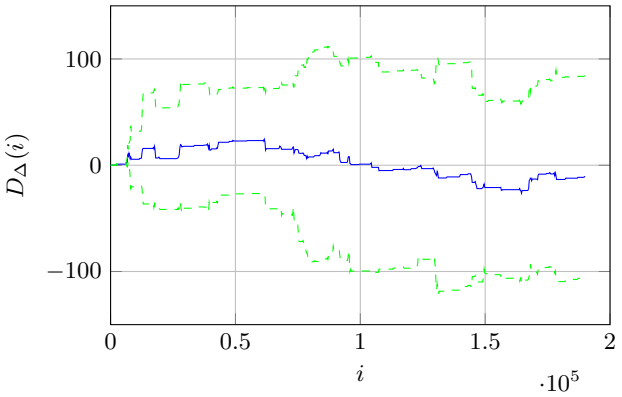

Figure 5: Error function for 60 packet queue, $\rho = 0.8$. Solid line – error function, dashed line – 95% confidence interval

random effects. This also means that we have found the most accurate approximation of buffer capacity.

The model accuracy is not clearly dependent on software queue size, but we observed such a dependency on link utilization. Figures 3,4 and 5 present the results of experiments conducted at utilization level $\rho = 0.8$. Higher utilization level (0.9 and 1.1) results are presented in Fig. 6 and 7. Each of the presented results contains some error, but the question is if this error is acceptable. To answer this question we have to use the relative error function to estimate the accuracy of the Packet Loss Ratio. Let us consider the worst scenario Fig. 7. The maximum of an error function can be approximated (with trend) by a linear function. Such an approximation is also depicted in Fig. 7. The relative error function for a large number of packets is a tangent of the fitted line and is equal to 0.0028. It is quite a small error considering that the observed packet loss ratio was equal to 0.189, the model error is less then 1.5 % of the observed drop probability.

4.3 Errors

Each measurement we did has an uncertainty error. In the experiments errors appeared in two places: traffic sniffing and interface speed calculation. We did not use the DAG card but instead simple Wireshark application to dump the

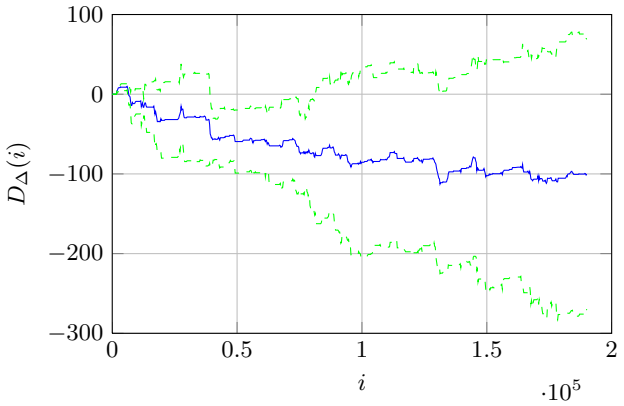

Figure 6: Error function for $\rho = 0.9$, 20 packet queue. Solid line – error function, dashed line – 95% confidence interval

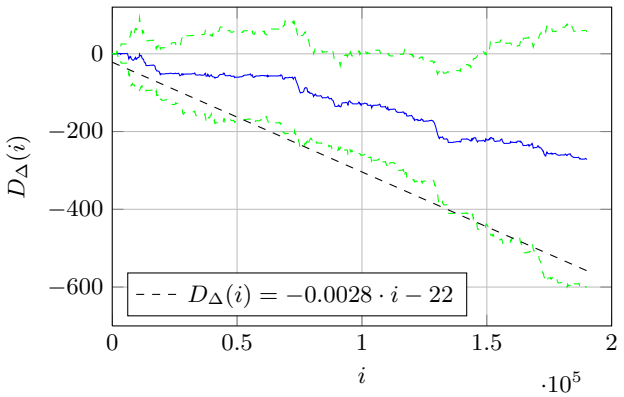

Figure 7: Error function for $\rho = 1.1$, 20 packet queue. Solid line – error function, dashed line– 95% confidence interval. Straight line – linear approximation of maximal model error

traffic, which may create an error related to packet processing in sniffer. Wireshark trace files were also used to estimate link speed, so speed measurement is also not extremely accurate. The question is how those errors effect a proposed model? In order to answer this question, we did some tests similar to the original experiments.

In our main experiment we compared drop functions measured in a router with ones obtained from the trace-driven simulation. To estimate an error, we compared drop functions obtained from two simulations run with different parameters. The changed parameters were the arrival time of each packet (introducing the jitter), link speed and buffer length.

The observation is that the error in the arrival time measurement has the lowest impact on drop function. This type of error is mostly important in highly congested links. Simulation results were confirmed by real experiments where an error increases when link utilization increases (compare Fig. 6 and 7).

The accuracies of link speed and buffer length estimations are far more significant then arrival time accuracy. Perfect fit of link speed is crucial. The simulations showed, that in a real world example, the relative error in link speed at level 10^{-5} causes an error function reaching up to 60 packets in a 200000 packet long simulation. What is also important is that observed error patterns are similar to those observed in the experiments. Based on simulation results we assumed, that link speed error is the main source of the total error in the model.

4.4 Traditional Router Models

Accurate router interface modeling is not possible without taking into account packet length. Traditional router models involving packet lengths are mostly developed for byte oriented storage [9, 1]. Byte oriented storage policy contradicts the model recommended in this paper, nevertheless the traditional approach to router modeling was also tested. Observed error functions for byte storage policy are depicted in Fig. 8.

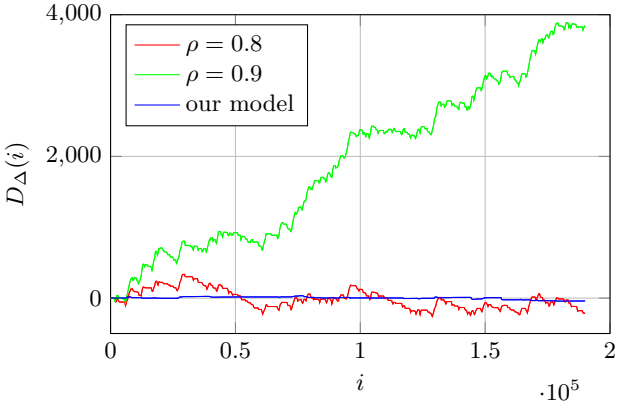

Figure 8: Model error for $\rho = 0.8$ and $\rho = 0.9$

The failure of byte oriented models is easily seen in Fig. 8. The plot marked as $\rho = 0.8$ is the error function for a byte oriented model of a router interface at utilization level 0.8. In such conditions the buffer size was chosen to minimize the error function across the entire trace. Next, the estimated byte buffer size was used in a model of a system at utilization level 0.9. The resulting error function is marked as $\rho = 0.8$. The line marked as "our model" is the error function for a model proposed in this paper. Clearly the byte oriented models can not be used for the real devices discussed in this paper because the required buffer size depends on its utilization, and it is not related to queue sizes for witch an interface was configured.

5. CONCLUSION

The results indicate that a single finite FIFO queue with buffer counted in packets (packet policy) is a good model for a single interface in a typical router used in LAN networks. Despite its complicated internal structure consisting of several buffers, we can tune the queue size and the link capacity to mimic drops occurring on a real device by model prediction with acceptable errors. The accuracy of a model largely depends on the accuracy of the link speed estimation. The protocol stack and the overhead resulting from lower level protocols is also crucial for accuracy. The proposed parametrization of a standard model leads to more realistic simulations and may be used for network dimensioning or router performance evaluation.

6. ACKNOWLEDGMENT

The work presented in this paper was supported by the national project 647/N-COST/2010/0

7. REFERENCES

[1] A. Chydzinski and R. Winiarczyk. Blocking probability in a bmap queue. In *ISCC '06: Proceedings of the 11th IEEE Symposium on Computers and Communications*, pages 547–553, Washington, DC, USA, 2006. IEEE Computer Society.

[2] Cisco. *Troubleshooting Input Queue Drops and Output Queue Drops*, document id: 6343 edition, Sep 20 2004.

[3] M. Gast. *T1: a survival guide*. O'Reilly, Sebastopol, Calif., 1st ed edition, 2001.

[4] L. Janowski and P. Owezarski. Assessing the accuracy of using aggregated traffic traces in network engineering. *Telecommunication Systems*, 43(3-4):223–236, 2010.

[5] K. V. K. Fall. *The ns Manual*. The VINT Project, January 2009.

[6] L. Muscariello, M. Mellia, M. Meo, M. A. Marsan, and R. L. Cigno. Markov models of internet traffic and a new hierarchical mmpp model. *Computer Communications*, 28(16):1835 – 1851, 2005.

[7] K. Park and W. Willinger. *Self-similar network traffic and performance evaluation*. Wiley, New York, 2000.

[8] I. Pepelnjak. Queuing principles in cisco ios. `http://wiki.nil.com/Queuing_Principles_in_Cisco_IOS`.

[9] P. Salvador, A. Pacheco, and R. Valadas. Modeling ip traffic: joint characterization of packet arrivals and packet sizes using bmaps. *Computer Networks*, 44(3):335 – 352, 2004.

[10] B. P. Sosinsky. *Networking bible*. WIley Pub., inc., Indianapolis, IN, 1st ed edition, 2009.

APPENDIX

A. TESTBED ACCURACY

Here we explain why the link speed on which the sniffer listen is so important for accurate measurements. Let us go back to Fig. 2. In switch S a traffic directed to router R1 is duplicated. The original traffic is sent to router R1 trough link witch capacity C_1, ad a copy is sent to the sniffer trough link witch capacity C_2. Further we will prove that if C_1 is not equal to C_2 and packets have different sizes, that the sniffed traffic is not the same as incoming to router R1.

Suppose that at time $t = 0$, a l_1 long packet arrives to switch S. The packet is duplicated and transmitted to R1 and PC1. Both packets reach their destination after time $\frac{l_1}{C_1}$ for packet to R1, and $\frac{l_1}{C_2}$ for the sniffed packet. We do not consider the propagation time but only the time required to pump the packet into the wire according to equation (1). Suppose that at time $t = t_1$ the second packet arrives to switch S. Let the second packet has length l_2. The second packet reaches router R1 at time $t_1 + \frac{l_2}{C_1}$ and the sniffer at time $t_1 + \frac{l_2}{C_2}$.

Let us now consider the inter-arrival time of those two packets observed by router R and sniffer PC1. The inter-arrival time observed by router R1 is equal:

$$\Delta_1 = t_1 + \frac{l_2}{C_1} - \frac{l_1}{C_1}. \tag{6}$$

On the other hand, a sniffer PC1 observes packets separated by time:

$$\Delta_2 = t_1 + \frac{l_2}{C_2} - \frac{l_1}{C_2}. \tag{7}$$

The difference of the observed inter-arrival times is equal

$$\Delta_\Delta = \Delta_1 - \Delta_2 = (l_2 - l_1)\left(\frac{1}{C_1} - \frac{1}{C_2}\right) \tag{8}$$

In equation (8) we can clearly see that the inter-arrival times are different when packets have different lengths and link speeds are not the same. Let us put some numbers from one of our experiments into formula (8) to see how big the described phenomena is. Real world values are: $l_1 = 1446B$ and $l_2 = 96B$ and $C_1 = 100Mbps$ and $C_2 = 1Gbps$. The relative error

$$\frac{\Delta_\Delta}{\Delta_1} = -14\%,$$

for inter-arrival time $t_1 = 800\mu s$. The described phenomena is a real problem for accurate measurements and have to be taken into account when planing experiments.