Switches under Real Internet Traffic

Paolo Giaccone, Marco Mellia, Luca Muscariello, Dario Rossi
Dipartimento di Elettronica – Politecnico di Torino
Corso Duca degli Abruzzi, 24 – I-10129 Torino, Italy
{giaccone,mellia,muscariello,rossi}@mail.tlc.polito.it

Abstract—**In this paper we propose a novel methodology to generate realistic traffic traces to be used for performance evaluation of switches. Indeed, real Internet traffic shows long and short range dependency characteristics, difficult to be captured by flexible, yet simple, synthetic models. One option is to use real traffic traces, which however are difficult to obtain, as requires to capture traffic in different places with synchronization and management problems.**

We therefore present a methodology to generate several synthetic traffic traces from a *single real trace* of packets, by carefully grouping packets belonging to the *same flow* to guarantee to keep the same statistical properties of the original trace. After formalizing the problem, we solve it and apply the results to assess the performance of scheduling algorithms in high performance switches, comparing the results to other simpler traffic models traditionally adopted in the switching community. Our results show that realistic traffic degrades the performance of the switch by more than one order of magnitude with respect to the traditional traffic models.

I. Introduction

In the last years, many different studies have pointed out how the Internet traffic behaves, focusing their analysis on the statistical properties of IP packets and traffic flows. The whole network community is now more conscious that traffic arriving at an IP router is considerably different from the traditional models (Bernoulli, on/off and many others). The seminal paper of Leland [1] gave new impulses in traffic modeling, leading to a huge number of papers deeply investigating such problem from different point of view. A group of studies focused on *statistical analysis* and data fit, e.g. [2]. These works highlighted traffic properties such as Long Range Dependence (LRD) at large time scales and also multi-fractal properties. LRD is probably the most relevant cause of degradation in system performance because it heavily influences the buffer performance whose behavior, being characterized by a Weibull tail [3], is considerably different from the exponential tail of conventional Markovian models. However, no commonly accepted model of Internet traffic has yet emerged, because either the proposed models are too simple (e.g., Markovian models), or very complex and difficult to understand and tune (e.g., multi-fractal models).

Therefore, *Internet traffic* is intrinsically different from the random processes commonly used in performance evaluation of networking systems, where trace-driven simulations are the most commonly used approach. This applies in particular to performance evaluation of high speed switches/routers, since the overall complexity of switching systems cannot be fully captured by analytical models. Hence, all the switch designers use simulation to validate their architectures by stressing its performance under critical situations.

How to generate the traffic to feed the simulation model is still an open question, because: i) traffic models (like Bernoulli, on/off, etc.) are indeed flexible and easy to tune, but they are not good model of real Internet traffic; ii) real traffic traces are more difficult to tune, and are not flexible since they refer to a particular network topology/configuration. In this paper, we propose a novel approach to generate synthetic traffic traces to be used for performance evaluation. It steams from the generation of synthetic traffic from a real trace, and adds the capability of building different scenarios (e.g., number of input/output ports and traffic pattern), keeping real traffic characteristics and providing synthetic traffic flexibility. The main idea is to generate the traffic which follows the time correlation of packets at IP flow level and, at the same time, satisfies some given traffic pattern relations. As interesting example of application, the performance of basic switching architectures are discussed and compared to the tradition benchmarking models.

II. Internet Traffic Synthesis

We consider a $N \times N$ switch, where each switch port is associated to an input/output link toward external networks, as shown in top plot of Fig. 1. One possible approach to feed this switch with real packet traces requires to sample the traffic at each of the N links; unfortunately, this approach is not easily viable, since it requires either to have a real $N \times N$ switching architecture or to manage and synchronize several distributed packet sniffers.

In a more realistic situation, only one trace referring to one link is available: for example in this work, traffic traces have been sniffed at Politecnico's egress router, as shown in bottom part of Fig. 1. Once the trace is available, a methodology to create different traffic scenarios is required, for example by imposing specific traffic relations among the input/output ports. The output of the methodology will be a set of N traces, satisfying the constraints imposed by the selected scenario. Our approach tries to establish the best mapping among

This work was supported by the Italian Ministry for Education, University and Research, within the EURO project.

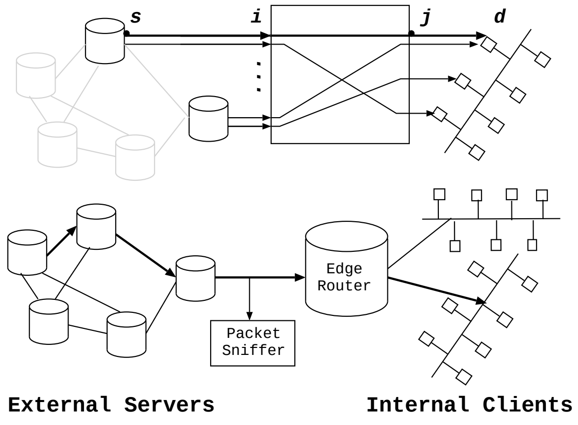

Fig. 1. Internet traffic abstraction model (on the top) and measure setup (on the bottom).

source-destination IP addresses (s, d) and input-output ports (i, j) of the switch (i.e., $s \leftrightarrow i$ and $d \leftrightarrow j$), in order to generate a total of N synthetic traces that can be "replayed", i.e. fed to the N input links of the switch under analysis.

The traffic relations are described by a traffic matrix T, of size $N \times N$, expressing the normalized average offered load between any input and output ports.

Additional constraints must be met in order to keep the statistical behavior of the original traffic trace, e.g., to keep the packet time correlation among the IP packets having the *same* source and destination addresses.

This problem is intuitively not trivial, given the number of constraints that must be satisfied. To better discuss the synthesis problem, we introduce the notation used throughout the rest of the paper, then formalize the problem, and solve it using a greedy heuristic.

A. Preliminary Definitions

When traffic is routed on a network, it is possible to focus the attention on different level of aggregations – namely IP packet, IP Flow, and Flow Aggregate levels. In particular, we define:

• **IP Flow:** An IP flow aggregates all IP packets having the same IP source address $s \in \mathcal{S}$ and IP destination address $d \in \mathcal{D}$. We state its size expressed in bytes with f_{sd}, and its normalized load $\rho_{sd} = f_{sd} / \sum_{h \in \mathcal{S}} \sum_{k \in \mathcal{D}} f_{hk}$. This is a natural aggregation level which entails that IP packets routed from s to d will follow the same route, closely mimicking Internet behavior.

• **Flow Aggregate:** We define a flow aggregate $\mathcal{F}_{\mathcal{S}_i \mathcal{D}_j}$ as the aggregation of all packets having source address $s \in \mathcal{S}_i \subseteq \mathcal{S}$ and destination address $d \in \mathcal{D}_j \subseteq \mathcal{D}$. We will choose address sets such that $\{\mathcal{S}_i\}$ and $\{\mathcal{D}_j\}$ are *partitions* of $\mathcal{S}$ and $\mathcal{D}$ respectively.

The flow aggregate $\mathcal{F}_{\mathcal{S}_i \mathcal{D}_j}$ represents the traffic crossing the switch from input i to output j. Let us denote the *address-2-port* mapping function with A2P(); then, for any source address tied to a specific input i we have A2P$(s) = i, \forall s \in \mathcal{S}_i$, as well as for any destination address it holds A2P$(d) = j, \forall d \in \mathcal{D}_j$ for a specific output port j.

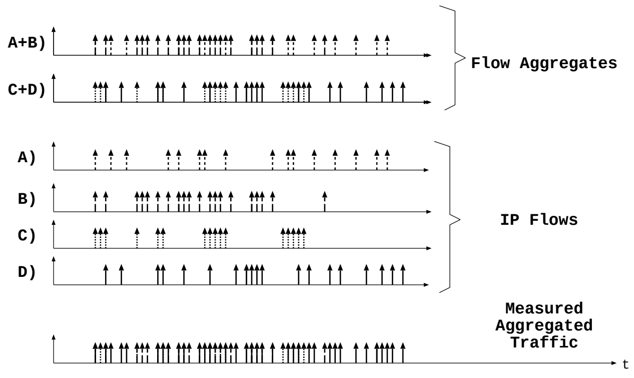

Fig. 2. Internet traffic at different levels of aggregation

Fig. 2 reports an example of the previous classification: at the bottom there is the original traffic trace, which is composed by four IP flows (labeled A,B,C,D). Then two flow aggregates are generated, considering the union of $\{$A,B$\}$, and $\{$C,D$\}$ respectively.

B. Traffic Matrix Generation

The mapping of the IP flows to a given traffic matrix T establishes a binding among IP addresses and switch ports. More formally, this binding can be thought as a generalization of the $P \| C_{max}$ optimization problem of scheduling jobs over identical parallel machines[4], and its formalization is provided in Fig. 3, in which element (i, j) of matrix X is denoted by X_{ij}, the i-th row by X_i and the j-th column by X_j^T, being X^T the transpose matrix of X.

The normalized IP flows load matrix $A \in [0, 1]^{|\mathcal{S}| \times |\mathcal{D}|}$ is the input of the optimization problem, in which each IP flow represents a *job* of size ρ_{sd} which have to be scheduled without deadline. A fixed number N^2 of machines are available, each corresponding to an input output couple of the switch; each machine is assigned a target *completion time*, i.e., the target (normalized) traffic matrix $T \in [0, 1]^{N \times N}$, which can be exceeded without penalty. Matrices $S \in \{0, 1\}^{N \times |\mathcal{S}|}$ and $D \in \{0, 1\}^{|\mathcal{D}| \times N}$ are the output of the problem, i.e., the mapping of jobs to machines, or, in our case, the mapping of source IP addresses to switch input ports and destination IP addresses to switch output ports respectively.

The objective is to *minimize* the *maximum* error committed in the approximation, i.e., the maximum deviance from the target traffic matrix. The first two system constraints lower bound the error z_{ij} with the absolute difference among the approximated $(S_i\, A) D_j^T$ mapping and

$$
\begin{array}{l}
\min p \\
p \geq z_{ij}, \quad \forall\, i,j \in \{1,2,\dots,N\} \\
\text{s.t.} \\
\begin{cases}
z_{ij} \geq (S_i A) D_j^T - T_{ij}, & \forall i,j \\
z_{ij} \geq T_{ij} - (S_i A) D_j^T, & \forall i,j \\
\sum_j S_{sj} = 1, & \forall s \\
\sum_i D_{id} = 1, & \forall d
\end{cases}
\end{array}
$$

Fig. 3. The optimization problem

the target T_{ij} completion time.

With respect to the classic $P \| C_{max}$ formulation, two additional constraints are present: once an IP destination address has been mapped to a particular switch output port, then all IP flows with the same destination IP address must be mapped to the same output port, as we assume that only one path is used to route packets. The same applies for IP source addresses and switch input ports. Therefore, each flow source will use just one row of the machine grid and each flow destination will use just one column, as enforced by the last two constraints.

Since the well known $P \| C_{max}$ optimization problem [4] is known to be strongly NP-hard, its bi-dimensional extension cannot have a polynomial time solution. Moreover, due to the size of our problem, we look for a simple and fast approximation of the optimal solution: among all the possible strategies, a greedy approach has been selected due to its extreme simplicity.

C. Greedy Partitioning Algorithm

In this section, we will briefly highlight some of the main features of the greedy adopted strategy, whose pseudo-code is shown in Fig. 4. The intuition at the base of the algorithm is to try to map the heaviest (un-mapped) IP flow (s, d) to the freest port pair (i, j), and then force all flows having the same IP source address to enter from the same switch input port, while flows having the same destination address will be force to exits from the same switch output port. This is done by updating the approximated traffic matrix Γ, whose elements account for the size of IP flows assigned to that particular port pair. This is repeated until all the IP flows have been mapped.

In the context of a greedy solution, the choice to accommodate at each step the heaviest remaining IP flow –which simply yields to process IP flows in a reversed sorted order– is quite intuitive. However, this not the case for the port pair selection; indeed, we tried several policies, and tested their performance under different traffic scenarios. For example:

- the port pair corresponding to the *globally largest* element in $T - \Gamma$, i.e. $(i,j) \leftarrow \arg\max_{m,n}(T_{m,n} - \Gamma_{m,n})$;
- the *row-wise largest* element, i.e. the largest element of the largest row $i \leftarrow \arg\max_m(\sum_k T_{m,k} - \Gamma_{m,k})$, $j \leftarrow \arg\max_n(T_{i,n} - \Gamma_{i,n})$ — or, symmetrically, the column-wise largest;
- the *coupled (row,column)-wise largest* element, that is the element that lies at the intersection of the largest row and column, i.e. $i \leftarrow \arg\max_m(\sum_k T_{m,k} - \Gamma_{m,k})$, $j \leftarrow \arg\max_n(\sum_k T_{k,n} - \Gamma_{k,n})$.

```
$\Gamma_{mn} = 0, \ \forall m,n$
while( $\exists \rho_{sd} > 0$ ) {
// select freer ports and heaviest IP flow
  $(i,j) \leftarrow \arg\max_{mn}(T_{mn} - \Gamma_{mn})$
  $(s,d) \leftarrow \arg\max_{mn} \rho_{mn}$

// set address-to-port mapping if unset
  if( $\nexists a2p\{s\}$ ), $a2p\{s\} \leftarrow i$
  if( $\nexists a2p\{d\}$ ), $a2p\{d\} \leftarrow j$

// update $\Gamma$ involving already mapped dests
  foreach( already mapped $\hat{d}$ ) {
     if( $\rho_{s\hat{d}} > 0$ ) && ( $\exists a2p\{\hat{d}\}$ ) {
       $\Gamma_{i,\text{a2p}\{\hat{d}\}} = \Gamma_{i,\text{a2p}\{\hat{d}\}} + \rho_{s\hat{d}}$
       $\rho_{s\hat{d}} = 0$
     }
  }

// update $\Gamma$ involving already mapped sources
  foreach( already mapped $\hat{s}$ ) {
     if( $\rho_{\hat{s}d} > 0$ ) && ( $\exists a2p\{\hat{s}\}$ ) {
       $\Gamma_{\text{a2p}\{\hat{s}\},j} = \Gamma_{\text{a2p}\{\hat{s}\},j} + \rho_{\hat{s}d}$
       $\rho_{\hat{s}d} = 0$
     }
  }
}
```

Fig. 4. The greedy algorithm

All the former approaches gave similar results only with uniform target matrix T; in the other cases the global approach gave the best results, even when compared to the coupled (row,column)-wise. Indeed, the global approach tries to minimize the *maximum error* among the target T and approximated traffic matrix Γ at a *local* level, i.e. for a specific input/output pair. Other strategies try to minimize respectively the error on either the input ports (row-wise strategy) or the *average error* (coupled strategy), explaining thus the relatively worse performances.

III. Performance Study

A. Measurement setup

In order to collect traffic traces, we observed the data flow on the Internet access link of our institution, i.e., we focus on the data flow between the edge router of our campus LAN and the access router of GARR/B-TEN, the Italian and European Research network. Since our university hosts mainly act as clients, we recorded only the traffic flows originated by external servers reaching internal clients (i.e., the direction highlighted in Fig. 1).

The trace has been sampled during a busy period of six hours, by collecting data on 28 million of packets and 42400 IP flows. The time window has been chosen such that the overall traffic is tested to be stationary both for the first and second order statistics. The property of real traffic that we mainly take into account is the long range dependency which is well known to be responsible of buffer performance degradation. It is not the topic of this paper to provide a statistical analysis of the traffic measured at our institution router but it is relevant to highlight that the measured traffic exhibit LRD [5] properties from the scales of hundreds of milliseconds to the entire length of the data trace with the Hurst parameter in the range of 0.7-0.8 [6].

B. *The switching architectures under study*

An IP switch/router is a very complex system [7], composed by several functionalities: here we focus our attention only on the performance of switching systems. We consider a simple model of the switching architecture, based on the one described in [8]. The incoming, variable size, IP packets are chopped into fixed size cells which are sent to the internal switch, where they are transferred to output port, and then reassembled into the original packets before being sent across the output link. The internal switch, which operates in a time slotted fashion, can be input queued (IQ), output queued (OQ) or a combined solution, depending of the available bandwidth inside the switching fabric.

IQ switches are usually considered scaling better than OQ switches with the line speed, and for this reason they are considered in practical implementations of high speed switches. Input queues are organized into the well known virtual output queue (VOQ) structure, necessary to maximize the throughput. One disadvantage of IQ switches is that they require a scheduling algorithm to coordinate the transfer of the packets across the switching fabric; the performance of an IQ switch, in terms of delays and throughput, is very sensible to the adopted scheduling algorithm and depends also on the traffic matrix considered. Scheduling algorithms can work either in *cell mode* or in *packet mode* [8]. In cell mode, cells are transferred individually. In packet mode, cells belonging to the same IP packet are transferred as a train of cells, in subsequent time slots; hence, the scheduling decision is correlated to the packet size.

In the past, the performance of several scheduling algorithms have been compared [8], [9] under Bernoulli or correlated on/off traffic. Here we compare the performance of maximum weight matching (MWM) [10] and iSLIP [11] scheduling algorithms under different traffic models. We selected MWM as example of theoretical optimal algorithm which is too complex to be implemented, whereas iSLIP was chosen as example of practical implementation with suboptimal performance.

We consider a 8×8 switch, with internal cell format of 64 bytes. In the IQ switch, buffers are set equal to

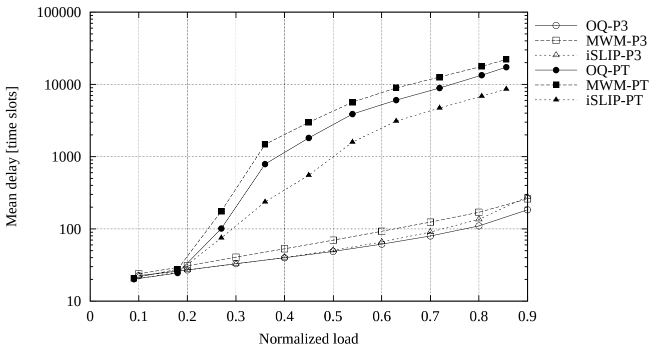

Fig. 5. Mean packet delay under PT and P3 scenarios for cell mode policies.

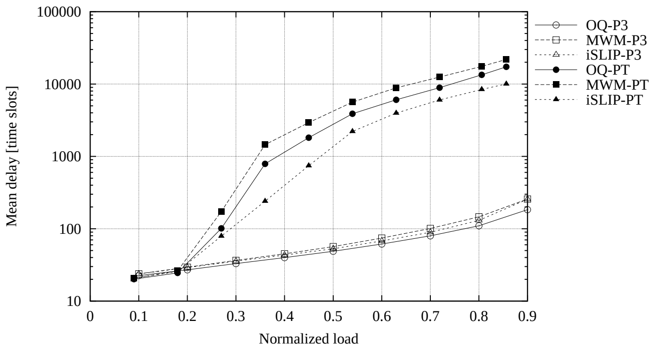

Fig. 6. Mean packet delay under PT and P3 scenarios for packet mode policies.

5000 cells per VOQ, i.e. about 320 KBytes per VOQ and about 2.5 MBytes per input port – which is a reasonable amount of high-speed memory available today in an input card. In the OQ switch, buffers are set equal to 40000 cells ($= 5000 \times 8$) to compare fairly with the IQ switch.

C. *Traffic scenarios*

For the sake of space, we present our results only for uniform traffic, i.e. $T_{ij} = \alpha/N$, where α (between 0.1 and 0.9) is the average input load, normalized to the link speed. We consider two scenarios, depending on the process of packet generation.

• *Packet trace (PT).* Packets are generated according to the trace, following the methodology of traffic synthesis we presented in Sec. II-B.

• *Packet trimodal (P3).* Packet generation is modulated by an on/off process, satisfying the traffic matrix T. Packet lengths are generated according to a trimodal distribution, which approximates the distribution observed in our trace. This can be considered a traditional good synthetic model, tuned according to the features of the real trace.

D. *Simulation results*

Figs. 5 and 6 plot the average delay as a function of the normalized load for tree configurations: OQ switch, IQ switch with MWM scheduler and IQ switch with iSLIP

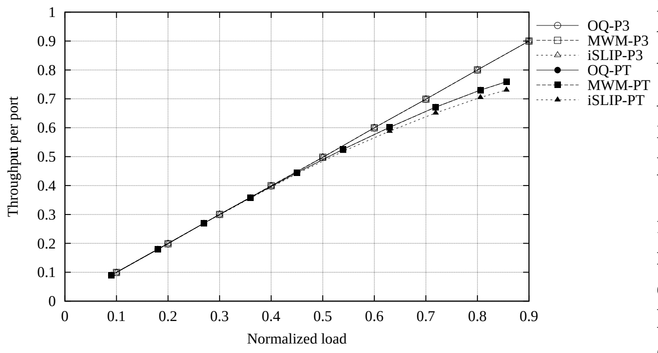

Fig. 7. Throughput under PT and P3 scenarios for cell mode policies.

scheduler. The first graph refers to cell mode (CM) schedulers for IQ, the second to packet mode (PM) schedulers for IQ. In all cases, the delays experienced under Packet Trace model are much larger than in the case of trimodal traffic. This holds true not only for high load, but also at low load the effect of the LRD traffic causes much higher delay. Note also that the performance of CM and PM are almost the same in both scenarios; this is reasonable, since the estimated coefficient of variation of the packet length distribution is 1.05 and, according to the approximate model in [8], CM and PM should behave the same.

Fig. 7 shows the throughput achieved in both scenarios considering CM policies (PM behave the same). Because of the LRD property of the input traffic, the queue occupation under PT is much larger than P3 causing higher loss probability and reduced throughput.

The most surprising result is that the relative behavior of the three schedulers changes from P3 to PT. Indeed, OQ, IQ-MWM an IQ-iSLIP give almost the same performance with the traditional traffic model, while a degradation in the throughput curves is present considering the Packet Trace model (up to 10% reduction) and an increase in delays. Moreover, IQ-MWM behaves the worse considering the delay metric, which depends mainly on metric based on queue length [8]; iSLIP on the contrary shows shorter delays than OQ: one partial explanation of this is that, for high loads, iSLIP is experiencing larger losses than OQ under PT (larger than IQ-MWM for instance), as Fig. 7 shows.

Finally, the most important fact is that OQ is penalized in terms of average delay: the shared buffer at output queue allow much longer queues to build up, therefore degrading the delay performance because of the Weibull tail [12]. These results underline that traffic models traditionally adopted to assess switching performance are not capable of showing real world figures.

IV. Conclusion

This work proposed a novel and flexible methodology to synthesize realistic traffic traces to evaluate the performance of switches and, in general, of controlled queuing networks. Packets are generated from a single packet trace from which different synthetic traffic traces are obtained fulfilling a desired scenario, e.g., a traffic matrix. Additional constraints are imposed to maintain the original traffic characteristics, mimicking the behavior imposed by Internet routing.

We compared the performance of a switch adopting different queuing and scheduling strategies, under two scenarios: the synthetic traffic of our methodology and traditional traffic models. We observed that not only absolute values of throughput and delays can change considerably from one scenario to the other, but also their relative behaviors. This fact highlights the importance of some design aspects (e.g., the buffer management) which are traditionally treated separately. These results show new behavioral aspects of the queuing and scheduling in switches, which requires more insight in the future.

References

[1] W.E. Leland, M.S. Taqqu, W. Willinger and V. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Transaction on Networking*, Vol.2, No. 1, pp. 1–15, Jan. 1994.

[2] A. Feldmann, A.C. Gilbert and W. Willinger, "Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic", *ACM SIGCOMM'98*, Boston, Ma, pp. 42–55, Sep. 1998.

[3] I. Norros, " A storage model with self-similar input", *Queueing Systems*, Vol. 16, pp. 387–396, 1994.

[4] S. Graves et al. (eds.), "Handbook in Operations Research and Management Science: Logistics of Production and Inventory", *North-Holland*, 1993.

[5] M.S. Taqqu, "Fractional Brownian Motion and Long Range Dependence", *Theory and Application of Long-Range Dependence* P.Doukhan, G. Oppenheim, M.S. Taqqu Editors, 2002.

[6] L.Muscariello, M.Mellia, M.Meo, R.Lo Cigno, M.Ajmone Marsan, "A Simple Markovian Approach to Model Internet Traffic at Edge Routers", *COST279 , Technical Document*, TD(03)032, May 2003

[7] H. J. Chao, "Next generation routers", *Proceedings of the IEEE*, Vol. 90, No. 9, pp. 1518–1558, Sep. 2002,

[8] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi and F. Neri, "Packet-mode scheduling in input-queued cell-based switches", *IEEE/ACM Transactions on Networking*, Vol. 10, No. 5, pp. 666–678, Oct. 2002

[9] N. McKeown and T. E. Anderson, "A Quantitative Comparison of Scheduling Algorithms for Input-Queued Switches", *Computer Networks and ISDN Systems*, Vol. 30, No. 24, pp. 2309–2326, Dec. 1998.

[10] N. McKeown, A. Mekkittikul, V. Anantharam and J.Walrand, "Achieving 100% Throughput in an Input-Queued Switch", *IEEE Transactions on Communications*, Vol. 47, No. 8, pp. 1260–1272, Aug. 1999,

[11] N.McKeown,"The iSLIP scheduling algorithm for input-queued switches", *IEEE/ACM Transactions on Networking*, vol.7, n.2, Aug.1999, pp.188-201

[12] B. V. Rao, K. R. Krishnan, and D. P. Heyman, "Performance of Finite Buffer Queues under Traffic with Long-Range Dependence," *Proc. IEEE Globecom*, Vol. 1, pp. 607–611, Nov. 1996.