CHAPTER 3

Performance Analysis and Experimentation

3.1 Performance Evaluation Model

Following the methodology adopted by the original work on the TCP Friendly marker, we use a performance model as suggested by Figure 3.1.1

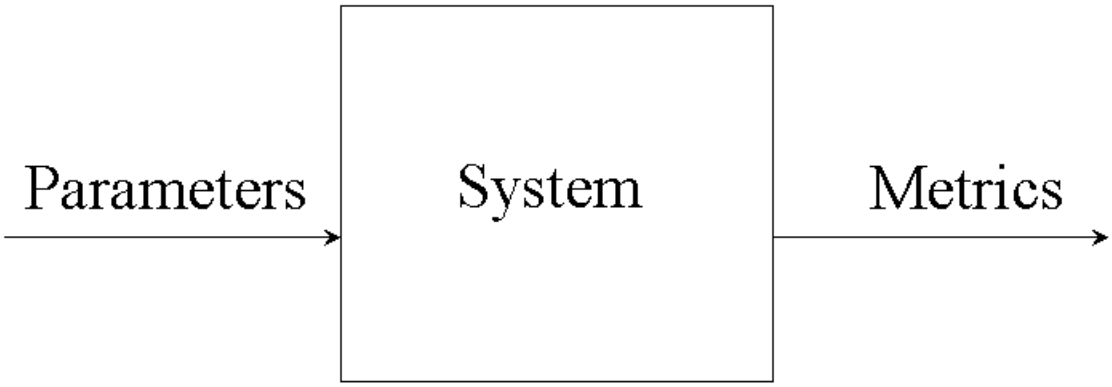

Figure 3.1.1: Performance Evaluation Model

We consider the system as a black box composed of the network and a set of flows that traverse it from source to destination, while considering the inputs to the system as being the specific configurations (defined by the traffic conditioning algorithm and the buffer management scheme, as well as their parameters). In a similar fashion, the outputs to our model correspond to the metrics that we used in order to assess the benefits and superiority of one set of inputs over another. We will now discuss the metrics to consider as well as the permutations on the configuration elements that compose our input to the system.

3.2 Metrics

We aim to define a set of metrics that allow us to asses the performance under the common *best-effort* approach on the Internet and the better than *best-effort* approach that we propose with our implementation. We consider that the results of our scheme should in fact benefit both the consumers of the network services (i.e. the common end user) and the operators of the network resources. We appropriately partition our set of metrics into what we call user metrics and operator metrics.

The operator metrics refer to those of interest to the owners of the network, presumably the ISPs[26]. It is known that an operator is willing to tradeoff buffer space in the network for higher utilization as well as low packet loss probability. It is intuitive that an operator would like its network to be fully utilized for as long as there is traffic waiting to be sent. This will in turn result into a greater capacity to accept traffic since the network will not stay idle if there is traffic to be served. On the other hand, a low probability of loss inherently means less retransmissions flowing through the network, while at the same time provides a much-desired marketing tool used to attract its competitor's business. Therefore, the operator metrics we consider are:

- **Timeouts**: From the previous discussion we have established that instances of timeouts result into increased transfer delays, and in some cases originate short periods of underutilization of the network. We will use the number of timeouts as a proxy to the increase in delays as seen by the end user. In addition, timeouts result into increased data retransmission, which would not be needed otherwise. This metric will also aid us in establishing the degree of retransmission traffic that takes up resources from the network

- **Byte Loss Probability**: This quantity will provide us with a measurement of the packet loss probability in the network.

[26] Internet Service Providers

The user metrics, refer to those of interest to the customer who subscribes for servicing of packets by the operator's network. It is also intuitive that a user would like to experience the largest possible goodput[27] for its packets, as well as a low probability of seeing the service degrading as the operator also services other customers. That is, a customer would like to experience the same service regardless of him/her being the first individual simultaneously trying to access a specific website, or the 100th to do so. That will translate into uncertainties as to how long it will take him/her to access the Internet's desired destination. Therefore, the user metrics we consider are:

- **Average per-flow Goodput**: This metric will assess the improvements in the net data throughput of all the flows contending for the bandwidth resources of the network. The larger this value, the better the network reacts to congestion.

- **Coefficient of Variation of per-flow Goodput**: This metric will evaluate the predictability of network service, and the fairness of this servicing among all the flows accessing its resources

3.3 Experimental Configuration

The information on Figure 3.3.1 refers to the physical topology of the network used for our experimentation. Figure 3.3.2 shows the logical topology of the experimentation setup.

[27] Goodput is defined as the throughput of new data that should not include retransmissions

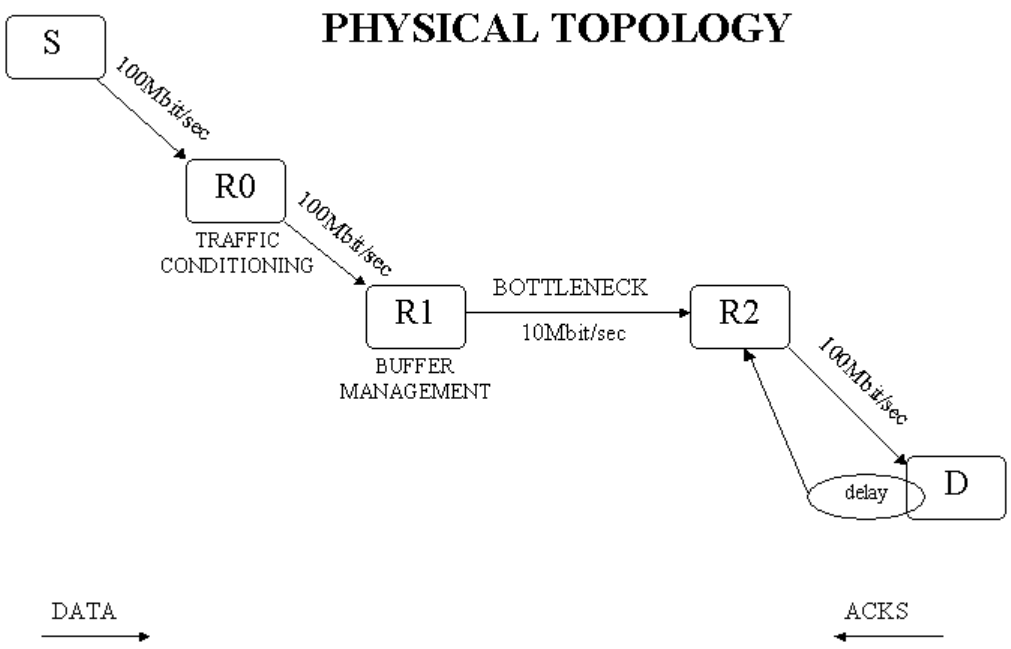

Figure 3.3.1: Experimental Physical Network Topology

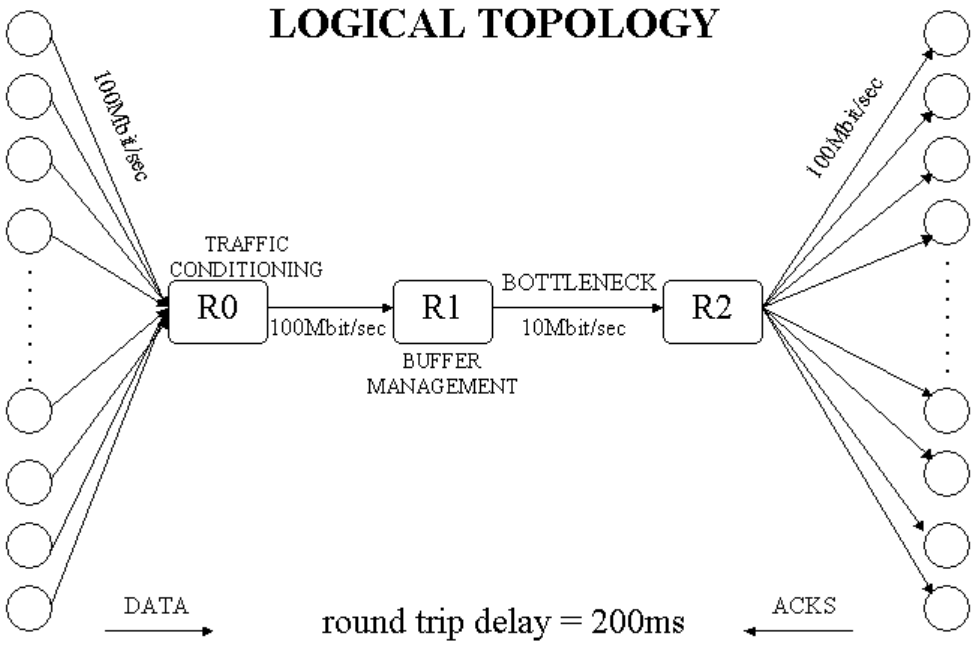

Figure 3.3.2: Experimental Logical Network Topology

We have a single source machine (S) that generates **multiple** TCP flows destined to a single machine (D) at the other end of the network. The source and destination machines have large (1Mbyte) buffers that prevent them from loosing outgoing data packets and outgoing acknowledgements, respectively. The source is connected to the marking router (R0) through a 100Mbit/sec link and the marking router feeds into the bottleneck gateway (R1) through a 100Mbit/sec link. It is clear that the traffic should not experience any bottleneck congestion up until its arrival to R1. The bottleneck is achieved by connecting R1 to R2 through a 10Mbit/sec link. We implement the correspondent buffer management algorithms at the interface of R1 connecting to R2. The link from R2 to the destination, supports a 100Mbit/sec speed, which clearly does not generate congestion on the traffic from R2. Finally we implement a delay component on the outgoing interface of the destination with the intention of emulating a round trip delay of 200ms[28] as seen by the source machine. The implementation details for this delay component, as well as the motivations for its inclusion in our experimentation are discussed in the Appendix.

Our set of experiments was composed of four major configuration settings with a focus on analyzing the performance of our overall scheme using the two most commonly deployed TCP implementations (Reno and SACK), as well as the setting of appropriate and inappropriate RED-like parameters.

We should discuss the meaning of “appropriate” and “inappropriate” in terms of RED-like parameters. We consider “appropriate” parameters, those in which the max and min threshold are relatively far apart, in order to allow the probabilistic dropping phase of RED to be in effect during most of the time. That is, the closer together these two parameters are set, the more RED behaves like tail drop. We assume that these parameters should be sufficiently far apart in order to give RED a fair chance to do its intended job. In addition the “appropriateness” of the parameters is greatly dependent on the characteristics of the traffic across the gateway. Thus, parameters that were

[28] Such delay corresponds to the typical round trip time of TCP transfer across the backbone of the Internet, from the Eastern Time Zone to the Pacific Time Zone.

appropriate in the past might not be so appropriate given the current traffic characteristics. This last circumstance could be also thought of as parameter misconfiguration. Using this premise, our RED-like parameters are:

Appropriate RED-like parameters:

- minimum threshold for OUT packets = 10Kbytes
- maximum threshold for OUT packets = 200Kbytes
- maximum probability of packet loss for IN packets = 0.25
- minimum threshold for IN packets = 50Kbytes
- maximum threshold for IN packets = 490Kbytes
- maximum probability of packet loss for IN packets = 0.005

Inappropriate RED-like parameters:

- minimum threshold for OUT packets = 10Kbytes
- maximum threshold for OUT packets = 30Kbytes
- maximum probability of packet loss for IN packets = 0.25
- minimum threshold for IN packets = 50Kbytes
- maximum threshold for IN packets = 490Kbytes
- maximum probability of packet loss for IN packets = 0.005

The four configurations settings are:

- **Configuration A**: TCP SACK with appropriate RED-like parameters
- **Configuration B**: TCP Reno with appropriate RED-like parameters
- **Configuration C**: TCP SACK with inappropriate RED-like parameters
- **Configuration D**: TCP Reno with inappropriate RED-like parameters

Finally, each one of the configurations included the following cases reflecting the use of traffic conditioning markers and buffer management schemes. In addition, each case was further examined using a total of 50 and 100 simultaneous flows.

- **Case 1**: TCP Friendly marker and FRIO
- **Case 2**: TCP Friendly marker and RIO
- **Case 3**: Token Bucket marker and RIO
- **Case 4**: No marker and RIO (defaults to RED using OUT parameters)

The parameters used for both markers, for all the experiments, are as follows:

- Number of byte tokens per interval = 296K
- Update interval = 400ms

CHAPTER 4

Experiments Results

4.1 Configuration A: SACK & appropriate RED-like parameters

Table 4.1 shows the values for the obtained metrics under this configuration. The graphs presented in Figure 4.1.1 through Figure 4.1.4 represent a comparison between all of the experimental cases in terms of the resulting values for the operator and user metrics.

	Configuration A							
	100 flows				**50 flows**			
	Operator Metrics		*User Metrics*		*Operator Metrics*		*User Metrics*	
	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation
TCP Friendly & FRIO	**41**	**4**	**11421**	**0.06**	**2**	**2.14**	**22236**	**0.03**
TCP Friendly & RIO	**120**	**5.46**	**10575**	**0.12**	**9**	**2.67**	**20742**	**0.04**
Token Bucket & RIO	911	7.75	9705	0.33	253	4.13	18892	0.12
No Marker & RED	1322	9.01	9015	0.38	491	5.25	17865	0.17

Table 4.1

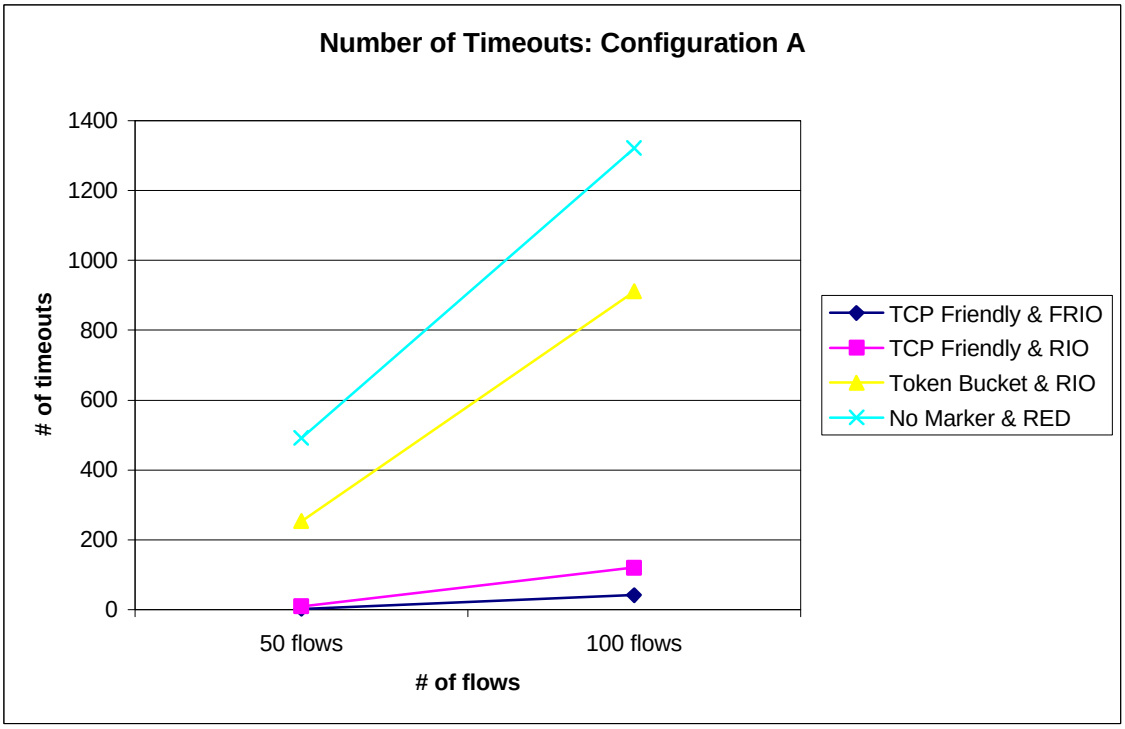

Figure 4.1.1

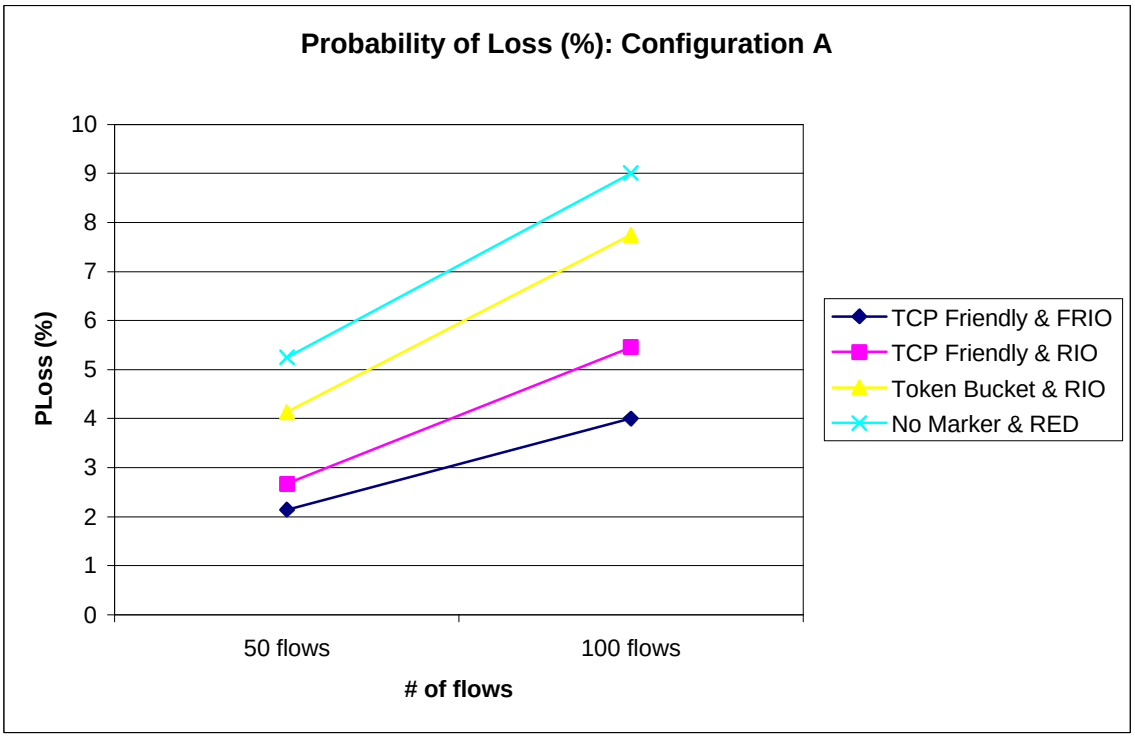

Figure 4.1.2

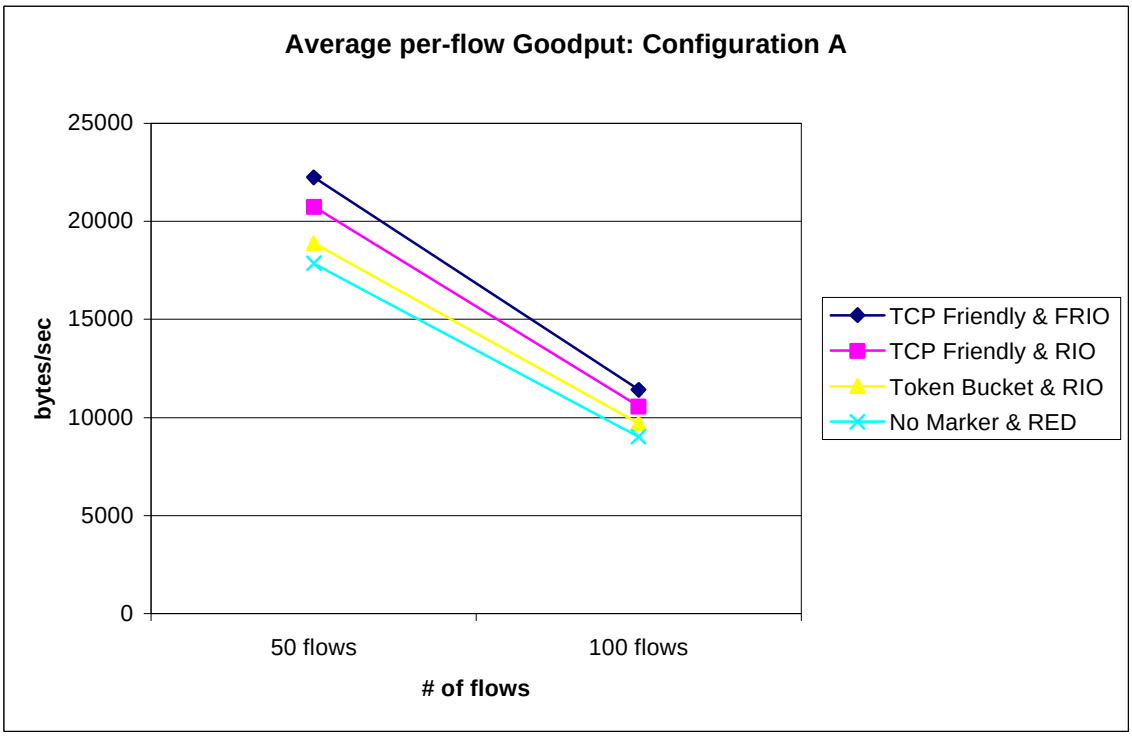

Figure 4.1.3

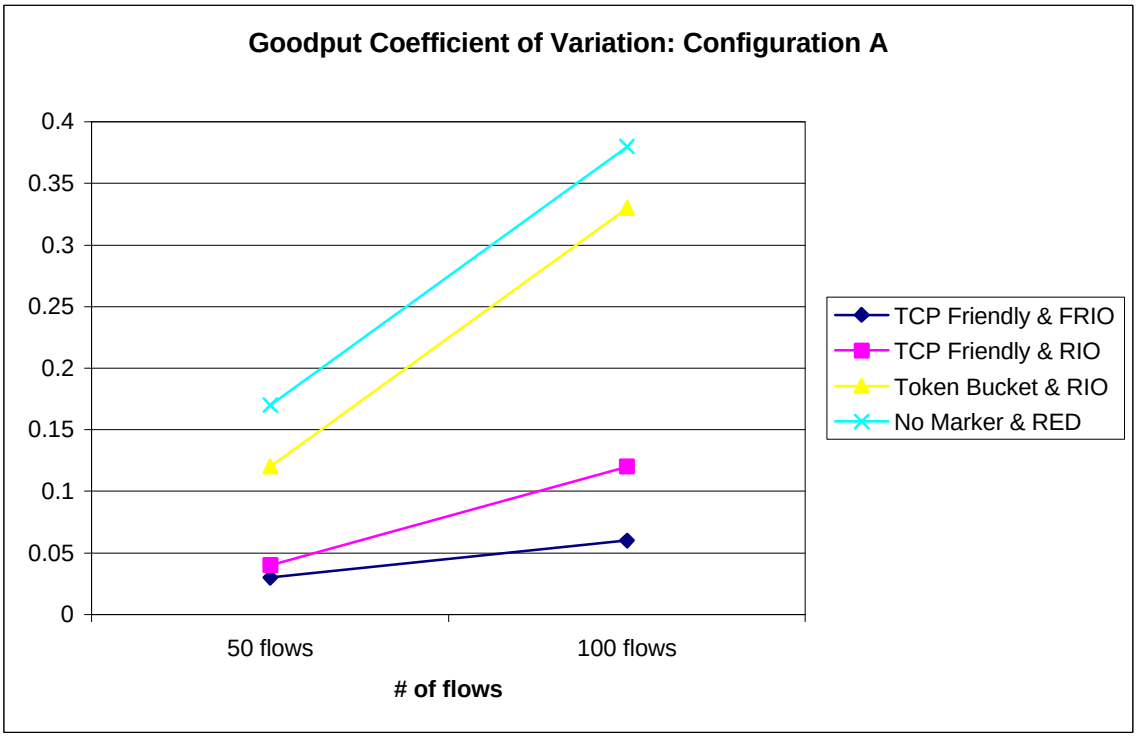

Figure 4.1.4

We clearly observe significant improvements achieved by the TCP Friendly marker across all metrics. On the case of 100 flows we see a reduction of **timeouts from 911 to only 41** when using the TCP Friendly marker over FRIO as opposed to the token bucket marker over RIO. This corresponds to an **improvement of almost two orders of magnitude**. We note, however, that there are timeouts when using SACK, which is indeed designed to avoid instances of timeout. We encountered than under high degrees of multiplexing (when the number of flows is large) SACK is still vulnerable to timeouts. We now have seen dramatic improvements to this vulnerability when using the TCP Friendly marker.

For all of the cases the packet loss probability when using our marker over FRIO is half of that when using the token bucket marker and RIO. In addition the **goodput is always higher** when using our scheme and the coefficient of variation is also 1.5 orders of magnitude lower when using our own marker and FRIO.

The graphs of the timeouts and the coefficient of variance show the **enormous gaps between the performance using the TCP Friendly marker and the performance without it**. The gaps are smaller for the probability of loss and the average goodput, but still considerable in terms of performance improvement. These improvements will undoubtedly translate into **better utilization of the network and fairer treatment of flows**.

In addition we note the **scalability** of our approach as the **performance enhancement increases with the number of flows introduced to the system**. We expect that a larger number of flows, as it is the case on the backbone routers on the Internet, will result into a significant smaller degradation of network performance as the one seen in the absence of marking or even in the presence of a token bucket marker. Furthermore, we note that, even though the use of FRIO does enhance our results, the TCP Friendly marker greatly improves the performance with RIO itself, **without the need for flow state information at the core of the network**. This last

observation certainly aids the feasibility of **deployment of our approach**, since we support **complex flow classification solely at the edges of the network**.

4.2 Configuration B: Reno & appropriate RED-like parameters

Table 4.2 summarizes our results obtained by replicating the previous configuration with the use of TCP Reno. The graphs shown in Figure 4.2.1 through Figure 4.2.4 represent a comparison between all of the experimental cases under this second configuration.

	Configuration B							
	100 flows				**50 flows**			
	Operator Metrics		*User Metrics*		*Operator Metrics*		*User Metrics*	
	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation
TCP Friendly & FRIO	**62**	**4.33**	**11391**	**0.07**	**8**	**2.49**	**22079**	**0.03**
TCP Friendly & RIO	**176**	**5.47**	**10472**	**0.14**	**16**	**2.83**	**20635**	**0.04**
Token Bucket & RIO	1127	12.49	9660	0.31	350	6.66	18486	0.19
No Marker & RED	1405	16.47	8807	0.37	610	9.63	17319	0.25

Table 4.2

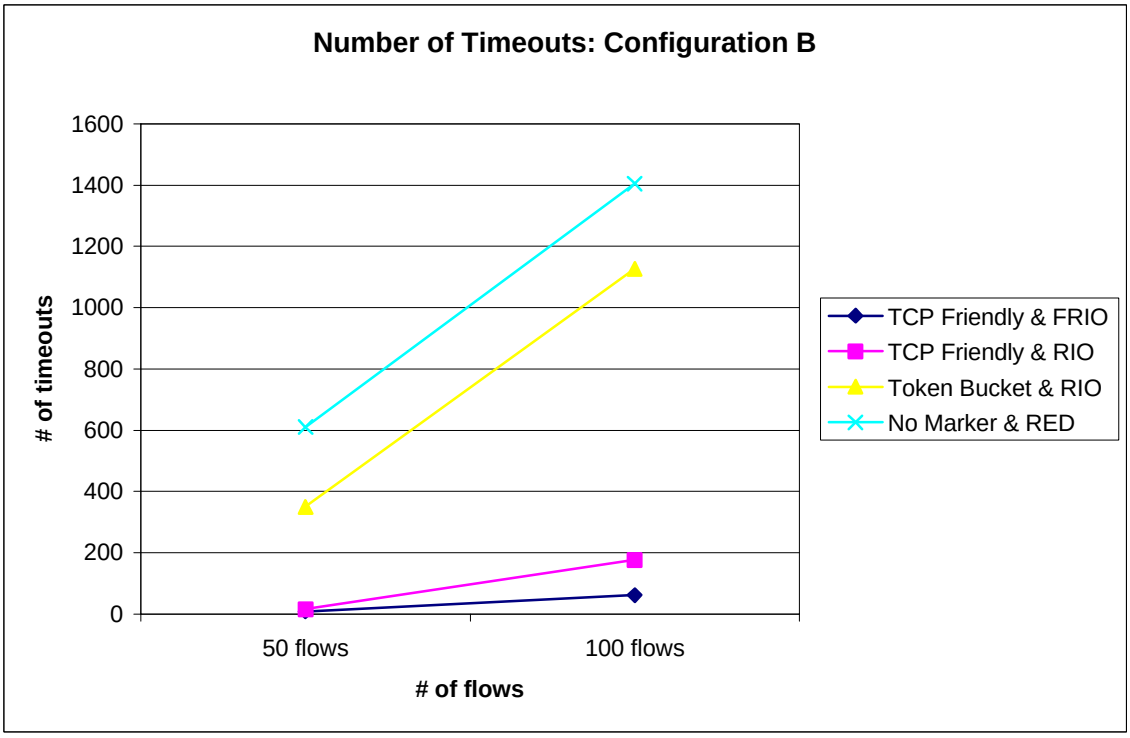

Figure 4.2.1

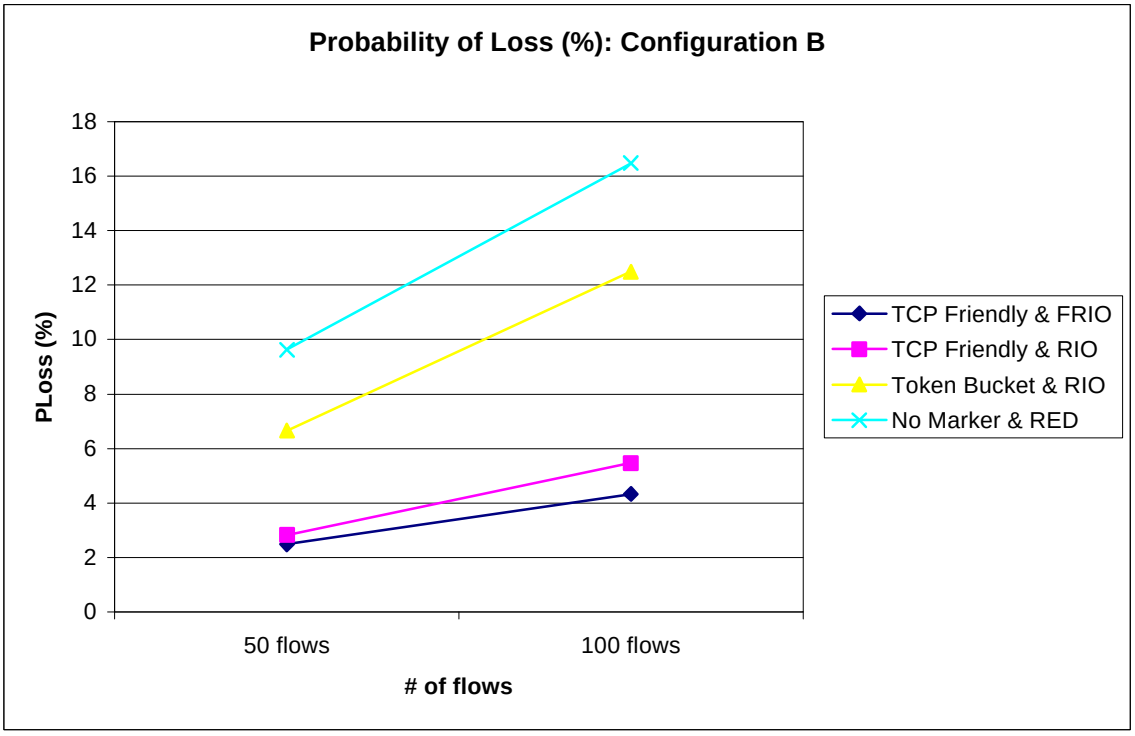

Figure 4.2.2

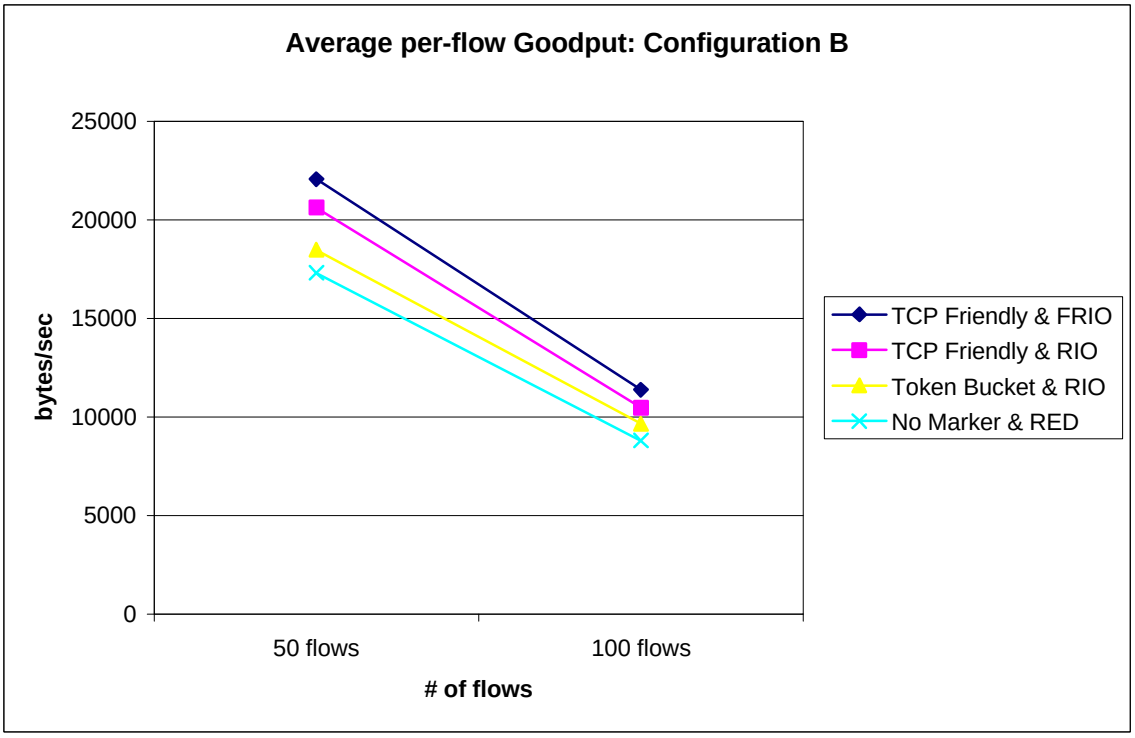

Figure 4.2.3

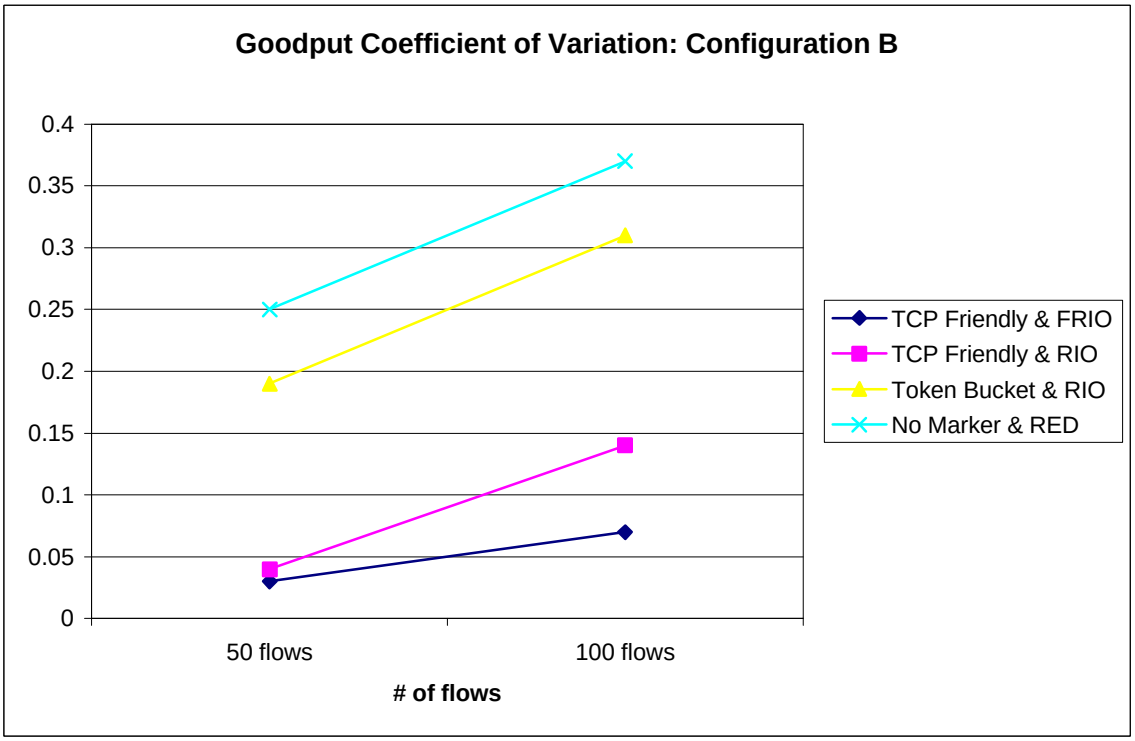

Figure 4.2.4

We note that the results obtained using Reno are consistent with the improvements seen with SACK. We see the number of **timeouts go from 1127 down to only 62**, when using the TCP Friendly marker and FRIO instead of the token bucket marker and RIO. We again see **improvements of close to two orders of magnitude** for the timeouts (610 to 8) when using **our scheme instead of best-effort approach** (i.e. no marking).

All of the graphs better show the improvements by evidencing the large gaps between the metrics obtained with the TCP Friendly marker and any other approach.

We observe that the absolute magnitudes of our metrics degrade in performance with respect to those of SACK. Nonetheless, the **performance improvements** of the TCP Friendly marker **using Reno are even larger** than those seen in SACK, clearly decoupling the benefits of our marker to the use of any particular TCP implementation. This once again suggests that our approach would in fact incorporate significant benefits to the performance of currently installed TCP Reno systems, and thereby the incentives for its **immediate deployment**.

4.3 Configuration C: SACK & inappropriate RED-like parameters

Table 4.3 presents the metrics under this configuration. The graphs seen in Figure 4.3.1 through Figure 4.3.4 represent a comparison between the defined experimental cases under this third configuration.

	Configuration C							
	100 flows				50 flows			
	Operator Metrics		*User Metrics*		*Operator Metrics*		*User Metrics*	
	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation
TCP Friendly & FRIO	**92**	**5.02**	**10854**	**0.07**	**7**	**2.79**	**21687**	**0.05**
TCP Friendly & RIO	**152**	**6.17**	**10168**	**0.13**	**10**	**3.63**	**20104**	**0.06**
Token Bucket & RIO	1200	9.16	9092	0.4	320	5.71	17277	0.22
No Marker & RED	1864	10.75	8316	0.45	689	7.03	16825	0.25

Table 4.3

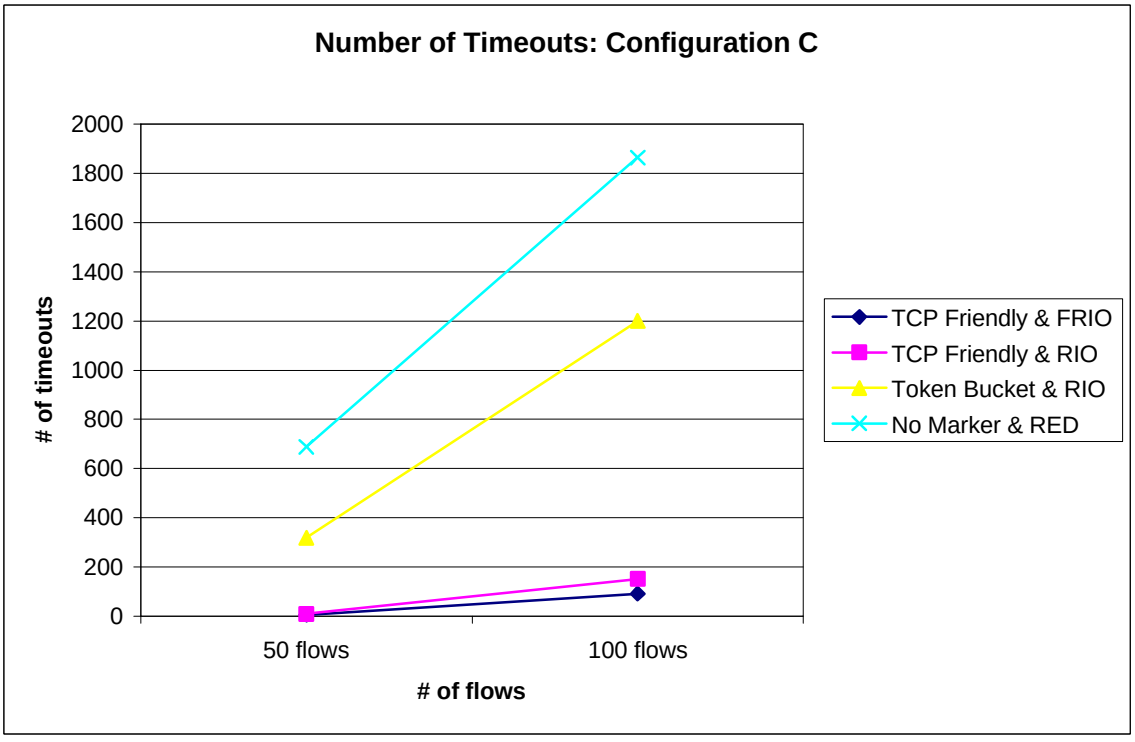

Figure 4.3.1

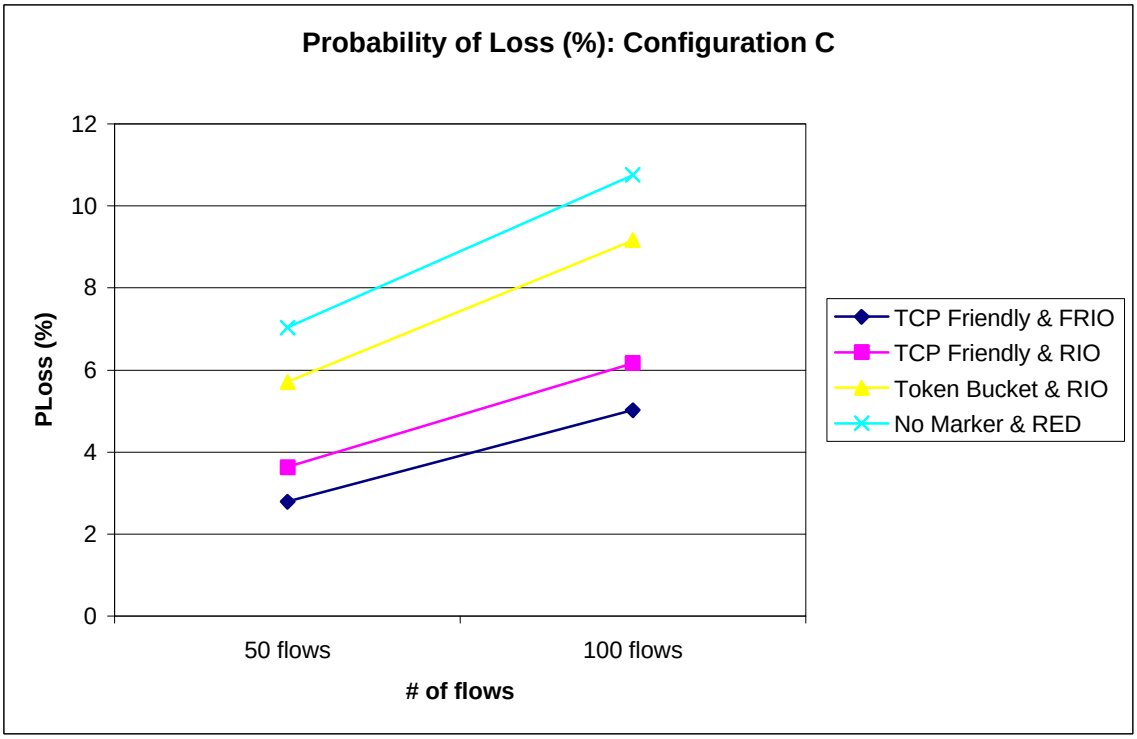

Figure 4.3.2

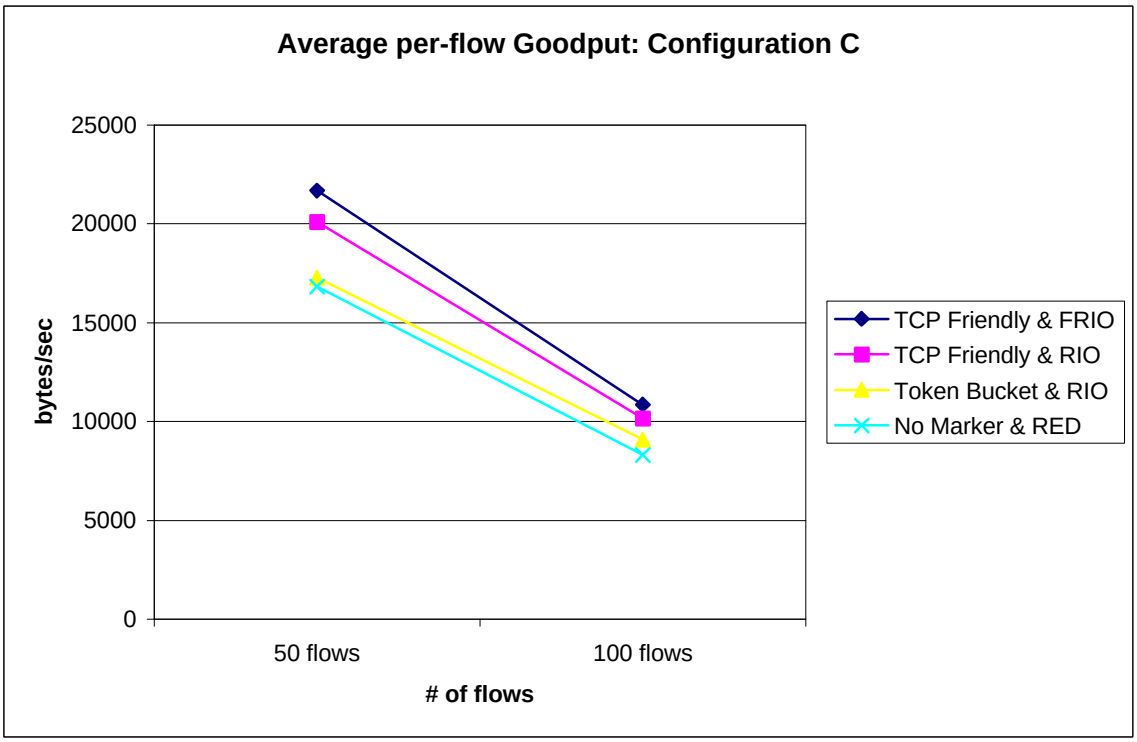

Figure 4.3.3

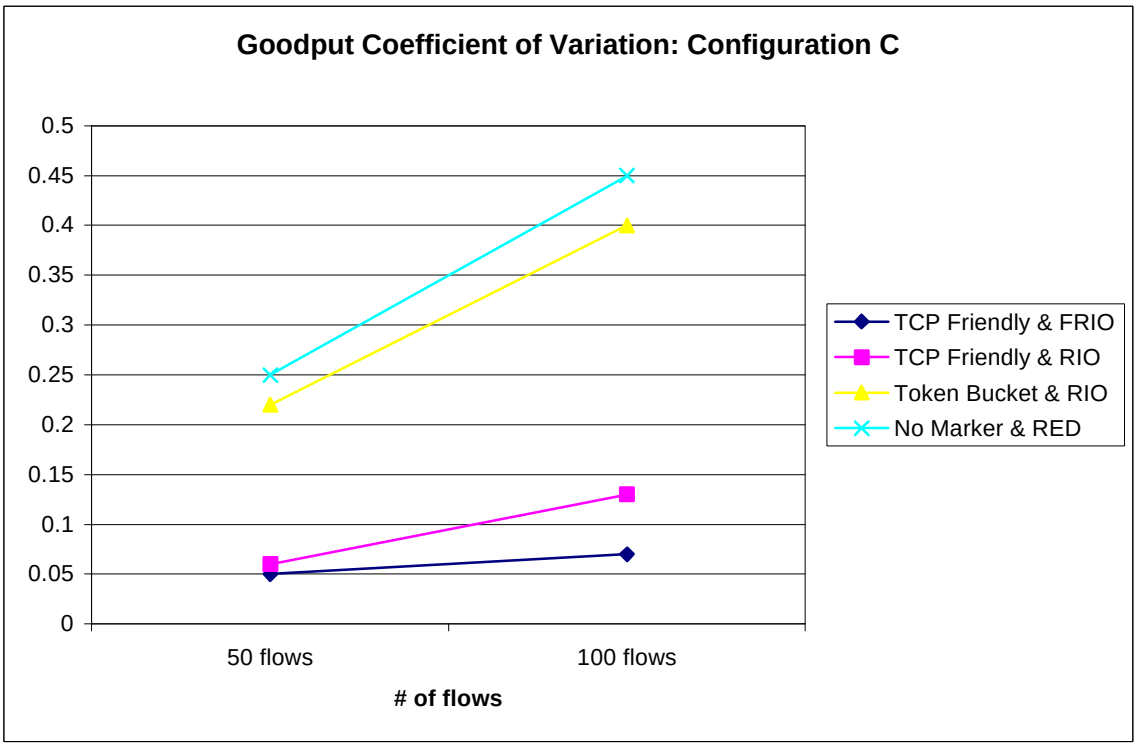

Figure 4.3.4

The graphs for this configuration illustrate performance enhancements that are consistent with the previously shown results. The graphs show that gaps between the metrics with different schemes remain very large. The number of timeouts is **2000%** larger without marking and **1300%** larger with a token bucket marker. The net throughput is consistently larger when using the TCP Friendly marker and the coefficient of variation is **600%** larger when using the best-effort approach instead of the scheme we propose.

On close observation of the graphs, we notice that the **usage of inappropriate RED-like parameters does not significantly affect the improvements** achieved by our marking scheme. Even though there is performance degradation with respect to the usage of appropriate RED-like parameters, the **improvements accomplished are sustainable** even disregarding the complex analysis that results into setting appropriate parameters. This is an important idea, since we now **have a tool that**

does not depend on the accuracy of the RED-like parameters to support improvements across all of the fundamental metrics.

Furthermore, we see that adopting inappropriate parameters does in fact close the gap between the performance of the TCP Friendly marker over FRIO and that of the marker over RIO. Once again, we have demonstrated the **powerful contributions** of our traffic-conditioning tool **without depending on robustness of parameters or complex flow state classification at the core of the network**.

4.4 Configuration D: Reno & inappropriate RED-like parameters

Table 4.4 shows the metrics obtained by replicating the previous configuration with the use of TCP Reno. The graphs presented in Figure 4.4.1 through Figure 4.4.4 represent a comparison between all the experimental cases for this configuration.

	Configuration D							
	100 flows				**50 flows**			
	Operator Metrics		*User Metrics*		*Operator Metrics*		*User Metrics*	
	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation	Timeouts	Ploss (%)	Average Goodput (bytes/sec)	Coefficient of Variation
TCP Friendly & FRIO	**130**	**6.33**	**10668**	**0.08**	**13**	**3.43**	**21315**	**0.05**
TCP Friendly & RIO	**239**	**6.97**	**10214**	**0.15**	**19**	**4.01**	**20168**	**0.06**
Token Bucket & RIO	1410	13.98	8954	0.43	413	7.8	17112	0.33
No Marker & RED	2163	20.58	8046	0.5	844	11.2	16152	0.35

Table 4.4

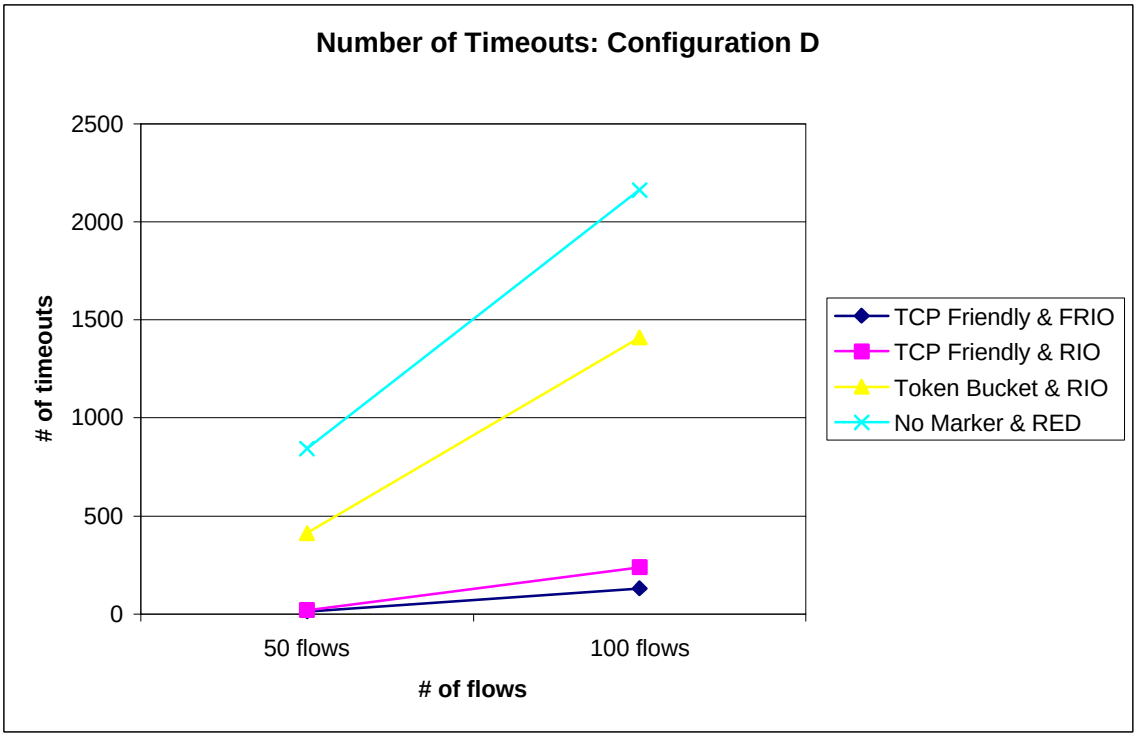

Figure 4.4.1

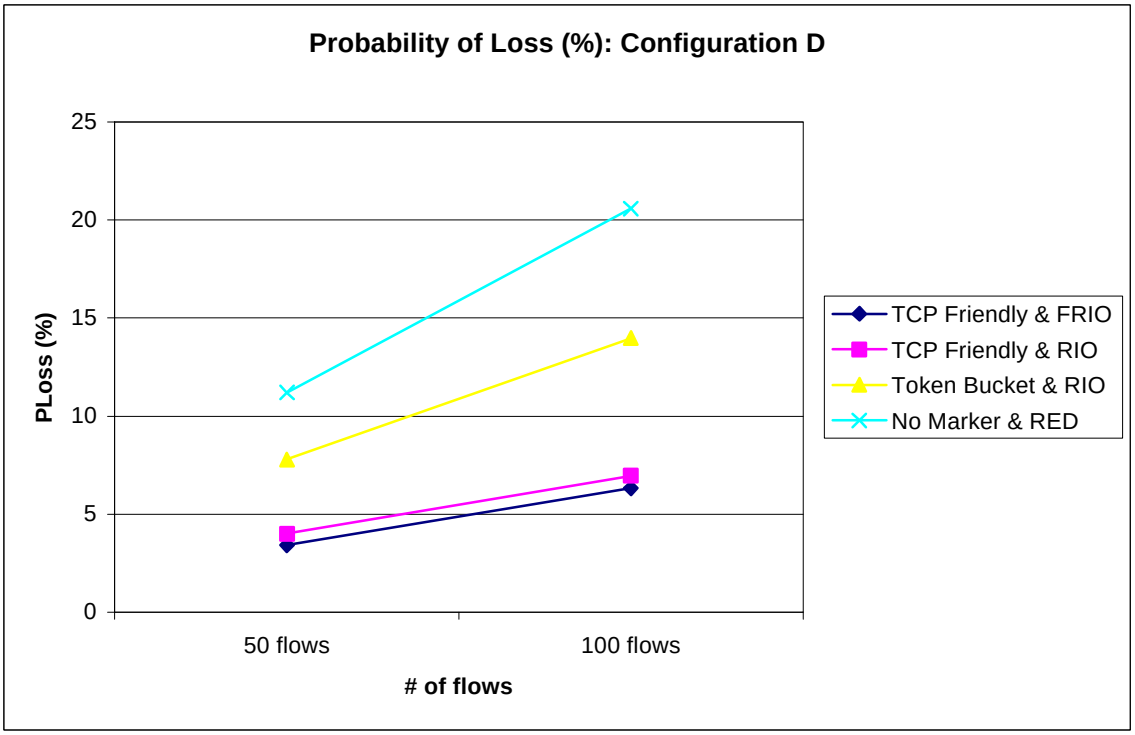

Figure 4.4.2

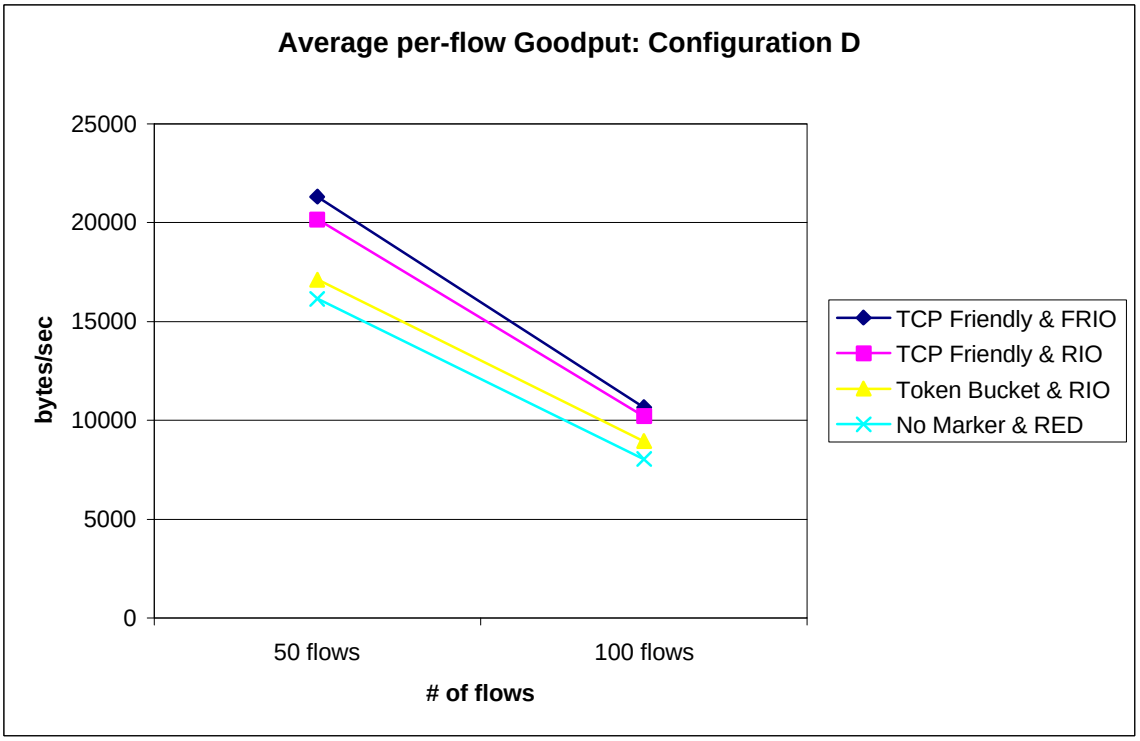

Figure 4.4.3

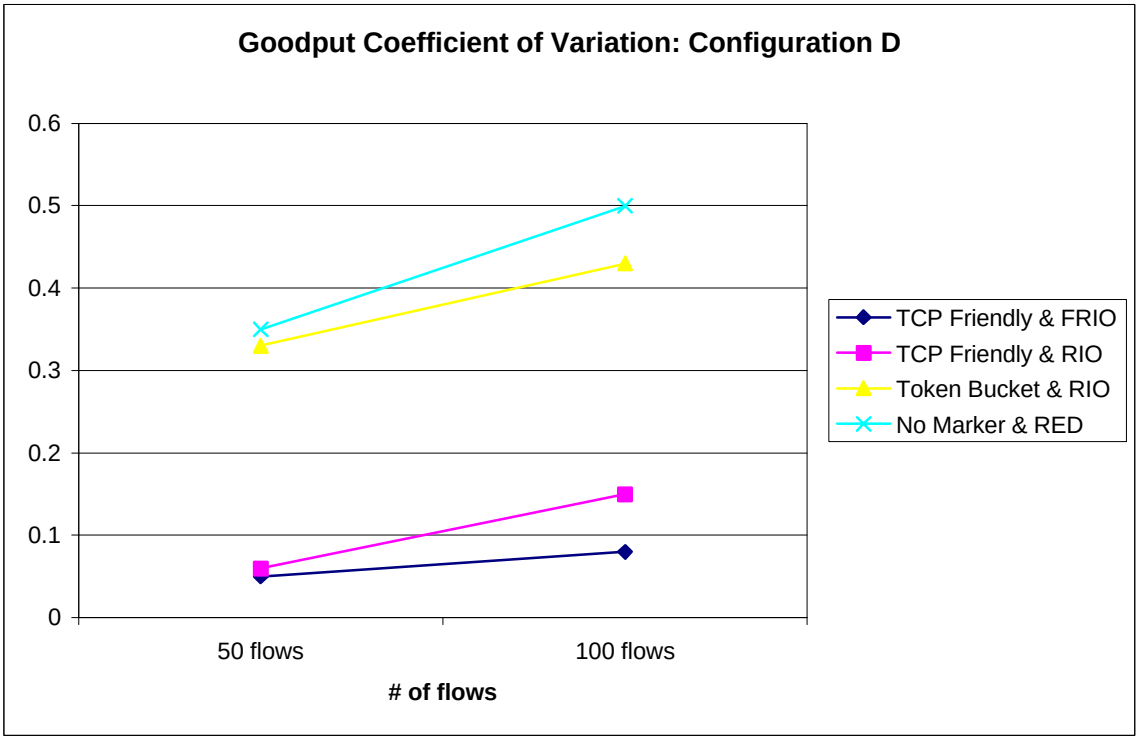

Figure 4.4.4

The results on this final configuration support all of the ideas previously discussed. We observe **consistent improvements across all metrics independent of the RED-like parameters and the TCP implementation used**. We see some minor degradation on the absolute performance with respect to Configuration C, which is expected simply because of the superior nature of SACK. Nonetheless, we continue to demonstrate the powerful capabilities of our marking scheme in terms of **reducing timeouts and probability of packets loss**, as well as **increasing per-flow goodput and favoring fairness among all flows**. All this without a basic need for flow state classification in the core of the network. We stress the **feasibility of our approach** since it is **beneficial for all TCP implementations and counteracts the inherent sensibility of RED to its parameters**.

4.5 Summary of the results

In this chapter we have shown the detailed results obtained for all of the cases we considered. We now present a synopsis of the benefits achieved by the architecture we have proposed:

- **Reductions of two orders of magnitude on the number of timeout instances.**
- **Consistently larger net data throughput for all TCP flows.**
- **Halving of the probability of packet loss across all instances of implementation.**
- **Increments of 200% on the predictability of service by the network.**
- **Improvements immunity to the inappropriate configuration of RED-like parameters.**
- **Improvements independence from version of TCP implementation.**
- **Scalability of the results to any number of TCP flows through a gateway.**

- **Improvements independence from flow classification at the core of the network.**

We finish the presentation of our experimental results, by arguing the validity of our findings. These results are not the product of software-based simulations that ultimately depend on the correctness of the model assumed, and disregard any of the details and limitations that are to be considered when implementing a scheme on the operating system of a router. These results are the outcome of real traffic monitoring as it occurs daily in the Internet. We have used our testbed lab to emulate long delay transfers. Nonetheless, the results using longer links and more hops in the path ought to translate into improvements of the same level.

We do not base our scheme on anything not already deployed on the Internet. The routers for the corporate world and the world of the ISP's do already support TOS byte marking and TOS byte retrieval. They also support standard RED and other proprietary buffer management techniques. The porting of our code towards the operating system of those routers is a relatively simple step, given the already existent functionality.

Furthermore, we have completed a running implementation of these algorithms that could be instantly deployed in any portion of the network that uses Linux-based routers. In the recent years, there has been a tendency[29] to accept the widespread growth of Linux routers as a reliable and economical alternative to the use of the now common proprietary-system routers. It is expected that such a growth will aggressively compete with the use of the now more common routers and will aid to the potential of instant deployment of the implementation of our work.

[29] [Man98]