Chapter 11

Effect of Parameters on Framework Performance

Recall that the proposed rate-based congestion control framework (see Figure 4) is composed of a number of functions that intercommunicate via several congestion control fields in every network packet, as was shown in Figure 5. Any specific function that meets the framework's requirements can be used as one of the framework's functions. Thus there could be any number of transport protocols, sustainable rate measurement functions, and so on.

In this thesis I have described several specific functions to be used in the congestion control framework. Several of these functions have parameters which affect their operation. In this chapter we will explore how the possible values of these parameters affect the functions and also the overall performance of the framework where these functions are used.

11.1 Available Parameters

The two major function instantiations which have been described are TRUMP, a rate-based Transport protocol, and RBCC, a Sustainable Rate Measurement function. The implementations of RBCC and TRUMP have parameters which can be altered to change their behaviour. RBCC has the following parameters:

- To perform ABT updates based on the two reverse congestion control fields in every packet. This was described in Section 8.4. This parameter can have two values: yes or no.
- To send Rate Quench packets as a form of congestion avoidance. This was described in Section 8.4. This parameter can have two values: yes or no.
- To scale an output interface's available bandwidth when the interface's buffered packet queue length exceeds a threshold of T packets. This was described in Section 4.11 as a form of congestion avoidance. This threshold can have integral values ranging from 1 to MP, the maximum number of packets that the interface can hold.
- The bandwidth scaling value, α, which is used when the old value T is exceeded. Values for this parameter range from 0.0 (exclusive) to 1.0 (inclusive).

The TRUMP transport protocol has one parameter:

- The maximum number of data segments which can be selectively acknowledged in a TRUMP acknowledgment packet. This was described in Section 5.3.1. This parameter can have integral values ranging from 1 to 16.

The set of possible parameter values form a 5-dimensional space with an infinite number of 5-tuples. We will explore some of the 5-space by keeping 4 parameters constant and varying a fifth[1].

11.2 Measurements Made

Most of abstracted measurements presented in this chapter are the same as for the previous chapter where TRUMP/RBCC, TCP Reno and TCP Vegas were compared in the 500 pseudo-random scenarios. The abbreviations given below will be used in the table presented in this chapter.

hiqueue The average buffer queue length (in packets) for the router with the highest average length.

avqueue The average buffer queue length (in packets) for all the routers in the simulated scenario.

hiutil The average utilisation (normalised) for the link with the highest utilisation.

e2edev The end-to-end standard deviation for all sources in the simulated scenario.

hits The total number of cached congestion 'hits' in the simulated scenario, which allowed ABT update decisions to be skipped.

misses The total number of cached congestion 'misses' in the simulated scenario, which forced ABT update decisions to be performed.

abt The total number of ABT table updates for all the routers in the scenario.

lost The total number of packets lost in the simulated scenario.

For each measurement, the median value and the 90% range will be given. The only exception to this is the **lost** result. The median value is not given, as this is nearly always 0. Instead, the total number of packets lost will be given.

11.3 Effect of Rate Quench Packets

Rate Quench packets were described in Section 8.4: If a data packet causes a flow's rate to be lowered in a router's ABT, a Rate Quench packet is returned to the source with the **Return Rate** field set to the flow's new rate in the ABT.

This congestion avoidance mechanism causes traffic flows to throttle back their transmission faster than the usual mechanism of obtaining the new rate via acknowledgment packets, which takes one round trip. The overall effect should be to lower network congestion, as traffic flows are transmitting data at an incorrect rate for less time. However, the

[1] A run of 500 scenarios for a particular 5-tuple typically takes between 7 and 10 hours to complete. It is infeasible to thoroughly explore the 5-space.

Rate Quench packets form a traffic flow themselves, and as a network becomes more congested, more Rate Quench packets are generated, which may lead to even more congestion.

The effect of Rate Quench packets were examined in the 500 pseudo-random scenarios described in the previous chapter. The following table shows the measurements for these scenarios where Rate Quench packets were generated/not generated, and where the remaining four parameters have the values [selacksize=1, revupdates=no, thresh=5, alpha=0.75].

Measurement	**No Quenches**		**Quench Packets**	
	Median	**90% Range**	**Median**	**90% Range**
hiqueue	1.686	1.122:2.130	1.677	1.121:2.104
avqueue	0.784	0.434:1.122	0.789	0.437:1.127
hiutil	0.924	0.837:0.987	0.924	0.845:0.995
avutil	0.097	0.061:0.132	0.098	0.060:0.131
e2edev	0.001	0.000:0.006	0.001	0.000:0.006
hits	109052	93060:128708	109052	93060:128685
misses	394	58:817	394	79:826
abt	214	44:402	214	44:401
lost	551	0:0	411	0:0

It is very difficult to distinguish between the two parameter values. Highest queue lengths are smaller, but average queue lengths are slightly higher. Utilisation is negligibly higher, and all other results are equal. We must look at the total number of packets lost to see any significant difference. With no quenches, 551 packets are lost over the 500 scenarios. With Rate Quench packets used, only 411 packets are lost.

The use of Rate Quench appears to lower overall packet loss without affecting other network characteristics such as utilisation or end-to-end variance. Given the results above, I would recommend that Rate Quench packets be used in any implementation of the rate-based congestion control framework.

11.4 Effect of Reverse ABT Updates

A reverse ABT update occurs when the router updates its ABT table when a packet's **Return_Rate** is less than the flow's rate in the table, or when the flow's bottleneck has altered the **Return_Rate**.

Allowing a router to perform reverse ABT updates provides it with 'downstream' information from packets flowing upstream. Without reverse ABT updates, the router only receives downstream congestion information after:

- the packet reaches the flow's source,
- the source alters its transmission rate, and
- a packet with the new transmission rate information from the source reaches the router.

It would appear that reverse ABT updates should help the network infrastructure react more quickly to overall network congestion. The results below indicate, however, that in fact reverse ABT updates increase network congestion.

The following table shows the measurements for the 500 pseudo-random scenarios where reverse ABT updates were allowed/disallowed, and where the remaining four parameters have the values [selacksize=1, quench=no, thresh=5, alpha=0.75].

Measurement	No Reverse		Reverse Updates	
	Median	**90% Range**	**Median**	**90% Range**
hiqueue	1.686	1.122:2.130	1.737	1.037:2.237
avqueue	0.784	0.434:1.122	0.793	0.394:1.098
hiutil	0.924	0.837:0.987	0.917	0.821:0.984
avutil	0.097	0.061:0.132	0.097	0.056:0.128
e2edev	0.001	0.000:0.006	0.001	0.000:0.008
hits	109052	93060:128708	108893	92861:128302
misses	394	58:817	459	55:1095
abt	214	44:402	290	47:655
lost	551	0:0	550	0:0

Reverse updates have no significant effect end-to-end standard deviation. However, reverse updates cause queue sizes for the most congested router, and for all routers, to be larger. Utilisation across the most-used links decreases, and there are more router ABT updates as expected. Overall packet loss is the same.

At first, the result appears to be counter-intuitive: the reverse congestion information should help routers to determine better rates for traffic flows. This should lead to better network utilisation. Instead, there is higher queue lengths and less utilisation. A hypothesis for this follows. Reverse ABT updates cause more router table changes, which cause more rate changes in the sources. Source rates fluctuate more, and so the network is slightly more unstable and higher congestion results.

Given the result, I would recommend that reverse ABT updates not be used in RBCC. This reduces the overhead of the RBCC on network routers, and keeps router queue lengths slightly lower, with no detrimental effect on network performance.

11.5 Effect of the Threshold, T

One of the framework's design goals, given in Section 4.2, is that 'A congestion control scheme should strive to keep queue lengths at length 1'. The threshold T causes each router to scale the bandwidth of an output interface by α if there are more than T packets queued for transmission on that interface. This is a form of congestion avoidance, as the scaled bandwidth lowers traffic flows' rates through the interface, which will help bring the interface's queue length back to length 1.

If T is small, than the scaling occurs before the queue size is large. This will help to maintain the queue length near 1, but will cause more router ABT updates, and will also cause more traffic flow rate fluctuations, as the queue length will cross the threshold very often.

If T is large (close to the maximum number of packets which can be queued on the interface), then the threshold is rarely crossed, no congestion avoidance is performed, and high queue lengths are permitted.

The effect of threshold values of 2, 3, 4, 5, 7 and 10 packets were examined in the 500 pseudo-random scenarios, where an interface is able to queue up to 15,000 octets (10 data packets). The remaining four parameters have the values [selacksize=1, quench=no, revupdates=no, alpha=0.75]. The following tables give the median and 90% range results for each threshold value.

Median	**T=1**	**T=2**	**T=3**	**T=5**	**T=7**	**T=10**
hiqueue	1.575	1.635	1.660	1.686	1.714	1.712
avqueue	0.779	0.778	0.791	0.784	0.791	0.791
hiutil	0.906	0.920	0.924	0.924	0.924	0.924
avutil	0.096	0.097	0.097	0.097	0.097	0.097
e2edev	0.001	0.001	0.001	0.001	0.001	0.001
hits	108399	108855	108920	109052	109052	108920
misses	677	427	396	394	392	392
abt	512	249	221	214	212	211
lost	522	535	543	551	560	761

90% Range	**T=1**	**T=2**	**T=3**
hiqueue	1.069:1.884	1.152:2.024	1.094:2.013
avqueue	0.429:1.090	0.438:1.129	0.434:1.118
hiutil	0.815:0.983	0.837:0.990	0.835:0.987
avutil	0.054:0.126	0.060:0.131	0.060:0.131
e2edev	0.000:0.008	0.000:0.007	0.000:0.007
hits	92186:126924	93060:128252	92908:128475
misses	52:2586	52:1192	52:844
abt	33:1695	33:688	33:424
lost	0:0	0:0	0:0

90% Range	**T=5**	**T=7**	**T=10**
hiqueue	1.122:2.130	1.017:2.078	1.000:2.140
avqueue	0.434:1.122	0.437:1.131	0.432:1.140
hiutil	0.837:0.987	0.837:0.987	0.837:0.987
avutil	0.061:0.132	0.060:0.131	0.060:0.131
e2edev	0.000:0.006	0.000:0.007	0.000:0.009
hits	93060:128708	92918:128574	92899:128566
misses	58:817	79:815	83:815
abt	44:402	39:393	36:393
lost	0:0	0:0	0:2

From the tables, it is apparent that all of the measurements, except the number of congestion cache misses and ABT updates, increases as T increases. Router queue lengths increase as the threshold is raised. As a byproduct of this, utilisation in the congested router increases as there are more packets ready for transmission. Because of the higher router queue lengths, end-to-end delays increase, and the higher occupancy in queue lengths causes higher end-to-end variance. More packets are lost as higher queue lengths are tolerated. However, because the threshold is crossed less at higher values of T, the number of ABT updates decreases.

The tables show that a lower threshold value provides better congestion control in the rate-based framework. The only drawbacks are a higher number of ABT updates, and a slightly lower network utilisation. If this form of congestion avoidance was to be implemented, I would recommend a value of T between 3 and 6 packets, inclusive.

11.6 Effect of the Scaling Factor, α

Hand-in-hand with the threshold value T is the scaling factor α. When the threshold is crossed, an interface's total bandwidth is scaled by α and its ABT is recalculated. α can take values between 0.0 (exclusive) and 1.0 (inclusive). Values of α near 1.0 have little scaling effect, and should provide little congestion avoidance. Values near 0.0 clamp traffic flows much below their ideal rate (until short-term congestion is reduced), and should have a negative effect on the overall network utilisation.

The effect of α values of 0.125, 0.25, 0.375, 0.5, 0.625, 0.75 and 0.875 were examined in the 500 pseudo-random scenarios. The remaining four parameters have the values [selack-size=1, quench=no, revupdates=no, thresh=5]. The following two tables give the median and 90% range results for each α value.

Median	α**=0.125**	α**=0.25**	α**=0.375**	α**=0.5**	α**=0.625**	α**=0.75**	α**=0.875**
hiqueue	1.692	1.686	1.675	1.682	1.689	1.686	1.691
avqueue	0.792	0.785	0.778	0.787	0.791	0.784	0.779
hiutil	0.914	0.919	0.923	0.923	0.924	0.924	0.925
avutil	0.097	0.097	0.097	0.097	0.097	0.097	0.097
e2edev	0.001	0.001	0.001	0.001	0.001	0.001	0.001
hits	107293	108518	108920	109052	109052	109052	109040
misses	394	394	393	393	394	394	394
abt	219	214	214	214	214	214	214
lost	27389	4130	357	408	473	551	639

90% Range	α**=0.125**	α**=0.25**	α**=0.375**	α**=0.5**
hiqueue	1.000:2.460	1.017:2.119	1.017:2.012	1.020:2.026
avqueue	0.432:1.216	0.432:1.153	0.437:1.141	0.434:1.131
hiutil	0.784:0.995	0.824:0.987	0.835:0.987	0.842:0.991
avutil	0.055:0.132	0.060:0.132	0.060:0.131	0.056:0.127
e2edev	0.000:0.013	0.000:0.006	0.000:0.006	0.000:0.006
hits	81070:133242	92699:129335	93060:128841	92896:128517
misses	79:928	55:833	79:837	83:823
abt	33:468	33:409	44:409	39:400
lost	0:4	0:0	0:0	0:0

90% Range	α**=0.625**	α**=0.75**	α**=0.875**
hiqueue	1.020:2.071	1.122:2.130	1.120:2.161
avqueue	0.434:1.138	0.434:1.122	0.432:1.121
hiutil	0.842:0.991	0.837:0.987	0.845:0.993
avutil	0.056:0.127	0.061:0.132	0.056:0.127
e2edev	0.000:0.006	0.000:0.006	0.000:0.007
hits	92893:128498	93060:128708	92901:128520
misses	83:823	58:817	79:831
abt	39:398	44:402	33:393
lost	0:0	0:0	0:0

Low values of α lower the utilisation of the highest utilised link as predicted. Router queue lengths increase and packet loss also increases as α approaches 1.0. The number of ABT updates is not substantially affected by any value of α.

However, as α approaches 0.0, router queue lengths increase and packet loss increases dramatically. From the results given in the tables above, I would recommend α values in the range 0.375 to 0.5, where the T/α congestion avoidance scheme is implemented.

11.7 Effect of the Selective Acknowledgment Size

Another design goal for the rate-based congestion control framework, given in Section 4.2, is that retransmission schemes such as Selective Acknowledgment should be used, as they retransmit packets only when necessary. This gives a parameter in the framework: the number of data packets that a Selective Acknowledgment can acknowledge.

A higher number of data packets acknowledged per ack packet lowers the required rate for the acknowledgment traffic flow, and thus lowers the load on the network. However, the drawback of a higher number of data packets acknowledged per ack packet is that the framework's congestion information takes longer to reach the source, as the information is delayed until a 'full' selective acknowledgment packet is transmitted. Another drawback to Selective Acknowledgments is the complexity that they add to a transport protocol.

The TRUMP protocol, implemented in REAL, allows between 1 and 16 data packets to be acknowledged in each acknowledgment packet. The effect of selective acknowledgment sizes 1, 2, 4, 8 and 16 data packets were examined in the 500 pseudo-random scenarios. The remaining four parameters have the values [quench=no, revupdates=no, thresh=5, alpha=0.75]. The following two tables give the median and 90% range results for each selective acknowledgment size.

Median	**Size 1**	**Size 2**	**Size 4**	**Size 8**	**Size 16**
hiqueue	1.686	1.432	1.285	1.343	1.649
avqueue	0.784	0.743	0.715	0.716	0.726
hiutil	0.924	0.925	0.923	0.924	0.925
avutil	0.097	0.097	0.097	0.098	0.100
e2edev	0.001	0.001	0.001	0.002	0.004
hits	109052	81726	67469	60379	56829
misses	394	427	504	518	471
abt	214	222	252	268	258
lost	551	582	617	710	835

90% Range	**Size 1**	**Size 2**	**Size 4**	**Size 8**	**Size 16**
hiqueue	1.122:2.130	1.000:1.827	1.000:2.133	1.000:2.365	1.000:2.331
avqueue	0.434:1.122	0.421:1.060	0.405:1.007	0.404:1.052	0.411:1.069
hiutil	0.837:0.987	0.843:0.990	0.843:0.993	0.842:0.990	0.841:0.990
avutil	0.061:0.132	0.056:0.125	0.057:0.124	0.063:0.127	0.067:0.131
e2edev	0:0.006	0:0.007	0:0.012	0:0.031	0:0.031
hits	93060:128708	70437:97142	57886:80942	49765:69982	48604:66874
misses	58:817	106:866	142:988	145:955	157:831
abt	44:402	37:407	36:471	35:483	46:475
lost	0:0	0:0	0:0	0:1	0:3

The tables show that higher selective acknowledgment size increase packet loss, queue lengths and ABT updates. End-to-end variance also increases, and this is due to the extra delays in acknowledging several data packets. Surprisingly, a selective acknowledgment size of one appears slightly worse than a size of two: router queue sizes are increased, and network utilisation is lowered across the most congested link. However, the best packet loss results are obtained for a selective acknowledgment size of one.

The effects on other measurements are not significant. Given that the results indicate some problems with high selective acknowledgment sizes on queue lengths, packet loss, end-to-end variance and most congested link utilisation, I would suggest that small selective acknowledgment sizes be used. Taking into account the packet loss results and the complexity of implementing a Selective Acknowledgment scheme, I would recommend that selective acknowledgments *not* be used in the rate-based congestion control scheme, and that transport protocols implement a 'One data packet, one acknowledgment packet' Acknowledgment scheme.

Rate Quenches also help to mitigate the effect of network delays, as was shown in Scenario 9 in Chapter 9. As large selective acknowledgments add delays to the network, then I would also highly recommend the use of Rate Quenches where large selective acknowledgment sizes are employed.

11.8 Effect of the Packet Dropping Function

If there is no room to queue a newly-arrived packet in a router, one or more packets must be dropped by the router. The function which determines which packet(s) are dropped is the Packet Dropping Function. The following dropping functions are available in the REAL network simulator:

Drop Tail: The newly-arrived packet is dropped.

Drop Head: The oldest queued packet in the router is dropped.

Drop Random: An already-queued packet in the router is randomly chosen and dropped.

Decongest First: The traffic flow with the most bytes queued in the router is found, and the most recently-arrived packet for the flow is dropped.

Decongest Last: The traffic flow with the most bytes queued in the router is found, and the oldest queued packet for the flow is dropped.

In fact, more than one packet may be dropped in the last four schemes so that enough room is made available to queue the newly-arrived packet. For example, several small acknowledgment packets may need to be dropped to queue a newly-arrived data packet.

In order to distinguish between the results of the five schemes, each was run on those randomly-generated scenarios where packets were lost with TRUMP/RBCC. Other parameter values were set at [selacksize=1, revupdates=no, quench=no, thresh=5, alpha=0.75].

Median	Drop Tail	Drop Head	Drop Random	Decongest First	Decongest Last
hiqueue	1.815	1.844	1.947	1.762	1.827
avqueue	0.929	0.891	0.925	0.907	0.909
hiutil	0.921	0.923	0.923	0.923	0.923
avutil	0.098	0.098	0.098	0.098	0.098
e2edev	0.007	0.006	0.007	0.007	0.008
hits	113270	113268	113261	113526	113515
misses	684	688	692	692	696
abt	308	311	308	310	320
lost	551	640	614	551	549

90% Range	Drop Tail	Drop Head	Drop Random	Decongest First	Decongest Last
hiqueue	1.451:2.071	1.438:2.177	1.451:2.207	1.496:2.136	1.370:2.158
avqueue	0.702:1.089	0.711:1.207	0.721:1.230	0.703:1.184	0.689:1.167
hiutil	0.823:0.971	0.820:0.964	0.819:0.964	0.847:0.989	0.820:0.964
avutil	0.053:0.114	0.054:0.115	0.054:0.115	0.054:0.114	0.054:0.114
e2edev	0.001:0.017	0.001:0.014	0.001:0.015	0.001:0.015	0.001:0.015
hits	99503:129323	99503:129320	99497:129320	99503:129311	99503:129320
misses	297:1081	297:1091	303:1118	279:1086	385:1208
abt	136:497	136:495	136:497	136:495	136:495
lost	1:32	1:40	1:32	1:32	1:32

The two schemes with the best combined packet loss/queue length results are Decongest First and Drop Tail. Decongest First has better queue lengths, network utilisation and end-to-end variance than Drop Tail. Despite this, I would argue for the use of Drop Tail as the preferred packet loss mechanism, as it is simple and imposes less overhead on routers than Decongest First.

11.9 Conclusion

The parameters available within the implemented functions of the rate-based congestion control framework do influence the characteristics of the framework, congestion and otherwise. At least two of the five parameters described above, the threshold T and the scaling factor α, have a significant negative impact on the framework's congestion control operation for certain values. Other parameters have some impact, but in general the framework works quite well for all their values.

If the congestion control framework was deployed with the function instantiations examined in this chapter, I would recommend the following parameter values:

- Rate Quench packets should be used,
- No reverse ABT updates should be performed,
- A threshold value of between 3 and 6 packets inclusive should be used,
- An α value of between 0.375 and 0.5 inclusive should be used, and
- No selective acknowledgments should be used. Instead, 'One data packet, one acknowledgment packet' should be used.
- Drop Tail should be used as the packet dropping mechanism.

11.10 Results with Recommended Parameters

A 5-tuple of parameters was chosen from the range of recommended parameters above: Rate Quench packets, no reverse updates, Drop Tail as the packet dropping mechanism, a threshold value of 4 packets, and an α value of 0.4. The following table gives the results in the 500 random scenarios, for these parameters against the 5-tuple used in Chapter 10.

Measurement	Ch. 10 Parameters		Recommended Parameters	
	Median	90% Range	Median	90% Range
hiqueue	1.686	1.122:2.130	1.647	1.122:2.058
avqueue	0.784	0.434:1.122	0.784	0.437:1.130
hiutil	0.924	0.837:0.987	0.922	0.837:0.990
avutil	0.097	0.061:0.132	0.097	0.060:0.131
e2edev	0.001	0.000:0.006	0.001	0.000:0.005
hits	109052	93060:128708	108920	92699:128565
misses	394	58:817	403	83:928
abt	214	44:402	221	35:441
lost	551	0:0	259	0:0

The recommended parameter values give a 47% drop in packet loss, with a lowering of the queue average in the most congested routers. Link utilisation for the most congested link is slightly lower, as is the end-to-end variance. With the threshold T lower, congestion avoidance occurs more frequently, and so the number of ABT updates has increased. The result is a small increase in the amount of RBCC work performed which gives a marked improvement in packet loss, offset by a slight lowering in link utilisation.

By tuning the parameters available with TRUMP and RBCC, the rate-based congestion control framework achieves 600 times fewer packets losses than TCP Reno, and 270 times fewer packets losses than TCP Vegas, in the 500 randomly-generated scenarios. Overall, throughput and network utilisation is improved, end-to-end standard deviation is lower, and the extra workload on the intermediate routers appears small. The framework provides excellent congestion control for connectionless packet-switched networks.