Chapter 6 – Comparing Congestion Control Regimes in a Large, Fast Network

6 Comparing Congestion Control Regimes in a Large, Fast Network

We continue our investigation of congestion control mechanisms by comparing relative behaviors given a large (up to 278 x 10^3 sources), fast (backbone routers operating up to 192 Gbps), simulated network with Web traffic, a few long-lived flows and periods of heavy file transfers among selected sites. We adopt an unrealistic assumption that all sources within our simulated network use the same congestion control regime.[1] We simulate our network under a range of conditions, then change the congestion control regime and repeat the simulation for the same conditions. In this way, we can determine how each congestion control regime responds to various conditions and then identify any differences. Various data analyses given later in this chapter refer to congestion control regimes by the identifiers shown in Table 6-1. The details of each regime were explained previously in Chapter 5.

Table 6-1. Congestion Control Mechanisms Compared

Identifier	Label	Name of Congestion Avoidance Algorithm
1	BIC	Binary Increase Congestion Control
2	CTCP	Compound Transmission Control Protocol
3	FAST	Fast Active-Queue Management Scalable Transmission Control Protocol
4	HSTCP	High-Speed Transmission Control Protocol
5	HTCP	Hamilton Transmission Control Protocol
6	Scalable	Scalable Transmission Control Protocol
7	TCP	Transmission Control Protocol (Reno)

We begin by describing (in Sec. 6.1) our experiment design, including the topology simulated, the input factors varied (and fixed), the conditions adopted, the temporal scenario and measured responses. Subsequently (in Sec. 6.2), we describe how we executed our experiments and collected data. In Sec. 6.3, we discuss our approach to data analysis. We display our most salient results in Sec. 6.4 and then (in Sec. 6.5) report our main findings before concluding in Sec. 6.6.

6.1 Experiment Design

The experiment was conducted within a single topology, illustrated in Fig. 6-1. This four-tier topology was explained and justified in Chapter 3. The top tier is formed by 11 backbone routers and 14 pairs of long-distance links. The second tier consists of 22 POP routers, while the third tier comprises 139 access routers. Access routers come in three varieties: normal (gray), fast (green) and directly connected (red). Fast and directly connected access routers connect sites to the topology at higher speeds than normal access routers. Directly connected access routers bypass POP routers and connect directly to the backbone. The fourth tier, not shown in Fig. 6-1, consists of various sources and receivers distributed throughout the topology and located under access routers.

[1] We change this assumption in Chapters 8 and 9.

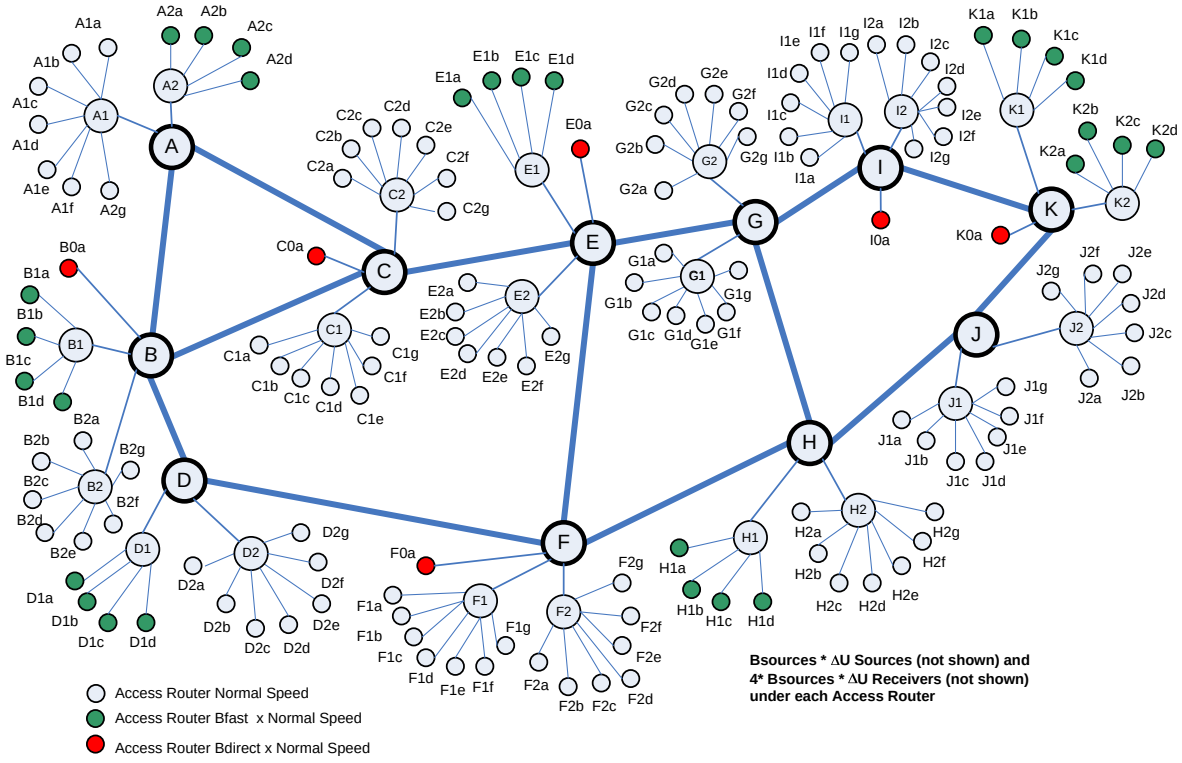

Figure 6-1. Topology Adopted for Experiments

One of the reasons for adopting such a topology is to permit flows to transit paths with a variety of characteristics, so congestion control mechanisms can be compared with respect to path class as defined in Table 6-2, where each path class consists of one or more flow type. A flow type is defined by the type of the access routers under which the source and receiver are located. The flow types in Table 6-2 are color coded to match the access routers depicted in Fig. 6-1. Since a flow cannot expect better performance than access routers provide, a flow is placed into the class dictated by its slowest access router. Thus, the "Very Fast" path class includes only DD flows, while DF flows are allocated to the "Fast" path class and DN flows are allocated to the "Typical" path class and so on.

Table 6-2. Definition of Three Path Classes (note that the correspondent of a source is a receiver and the correspondent of a receiver is a source)

Path Class	Flow Type	Definition
Very Fast	DD	Source & receiver under directly connected access router
Fast	DF	Source or receiver under directly connected access router and correspondent under fast access router
	FF	Source & receiver under fast access router
Typical	DN	Source or receiver under directly connected access router and correspondent under normal access router
	FN	Source or receiver under fast access router and correspondent under normal access router
	NN	Source & receiver under normal access router

6.1.1 Simulation Parameters

Within the framework provided by the topology given in Fig. 6-1, a wide range of network conditions may be simulated by specifying values for various parameters, or input factors, as discussed previously in Chapters 3 and 4. Guided by our sensitivity analysis, we selected six parameters, shown in Table 6-3, which we vary to establish the conditions under which we compare the congestion control mechanisms listed in Table 6-1. These six parameters are called *robustness factors* because any conclusions we draw hold (i.e., are robust) only over our simulated combinations of these parameters. Other simulation parameters are fixed across all experiments, as we document below.

Table 6-3. Robustness Factors Selected for Comparing Congestion Control Mechanisms

Identifier	Definition	PLUS (+1) Value	Minus (-1) Value
x1	Network Speed	8000 packets/ms	4000 packets/ms
x2	Think Time	5000 ms	2500 ms
x3	Source Distribution	Uniform (.33/.33/.33)	Skewed (.1/.6/.3)
x4	Propagation Delay	2	1
x5	File Size	100 packets	50 packets
x6	Buffer Sizing Algorithm	RTTxCapacity	RTTxCapacity/SQRT(*N*)

In Chapter 4, seven of the 11 parameters considered exhibited most significant influence. We adopted six of those seven as robustness factors for our current experiment. We omitted the multiplier on number of sources and receivers because we will consider a smaller network separately in Chapter 7. For each factor, we selected two settings, so we use a two-level experiment design. Network speed (x1) defines the fundamental capacity of backbone routers in packets per ms (p/ms). Recall, however, that this fundamental capacity is multiplied by BBspeedup to determine the full capacity of each backbone router. The speeds of other routers within the topology are derived from the value of x1 using various transformations, as shown in Table 6-4, which lists fixed parameters associated with the network model.

Table 6-4. Fixed Network Parameters

Parameter	Definition	Value
BBspeedup	Backbone router speed = x1xBBspeedup	2
R2	POP routers speed = x1/R2	4
R3	Access routers speed = x1/R2/R3	10
Bdirect	Directly connected access router speed = x1/R2/R3xBdirect	10
Bfast	Fast access router speed = x1/R2/R3xBfast	2
Hbase	Speed of basic sources (96 Mbps)	8
Hfast	Speed of fast sources (960 Mbps)	80
P(FastHost)	Probability that a source is fast	0.4
Qfactor	Factor by which buffer size will be multiplied	1

Sources and receivers may operate at one of two speeds: Hbase (8 p/ms) or Hfast (80 p/ms). This simulates the situation in real networks, where some computers connect at 100 Mbps, while others connect at 1 Gbps. For this experiment, we permit 40 % of

sources (and receivers) to connect at the fast speed, while remaining sources (and receivers) connect at the slower speed.

Factors x4 (propagation delay) and x6 (buffer-sizing algorithm) also alter characteristics of the network. When x4 = 2, the fundamental propagation delays encoded in the topology are doubled. Factor x6 selects the algorithm used to size router buffers. Setting the Qfactor = 1 ensures that the results of the chosen algorithm are used without further scaling of buffer sizes.

The factors controlling network characteristics may be translated into domain-specific values to give a sense of the nature of the network being simulated. For example, the speed of a backbone router when x1 = 8000 p/ms may be translated as 8000 p/ms x 2 x 1000 sec/ms x 12000 bits/packet = 192 Gbps. Table 6-5 shows the simulated speeds for all types of routers given the two values for factor x1. Similar reasoning indicates that fast sources operate at 960 Mbps and basic sources operate at 96 Mbps.

Table 6-5. Domain View of Router Speeds

Router	PLUS (+1)	Minus (-1)
Backbone	192 Gbps	96 Gbps
POP	24 Gbps	12 Gbps
Normal Access	2.4 Gbps	1.2 Gbps
Fast Access	4.8 Gbps	2.4 Gbps
Directly Connected Access	24 Gbps	12 Gbps

Table 6-6 illustrates the range of propagation delays being used within the experiment. Setting x6 = 1 (Minus) simulates a topology with an average one-way path propagation delay comparable to a network in the continental United States that has some links to Europe. Setting x6 = 2 (PLUS) simulates a topology that could span from East Asia to Western Europe, while transiting across North America. Since buffer sizes are computed based on router speed and propagation delay, Table 6-7 gives the range of buffer sizes that are simulated in our experiments.

Table 6-6. Path Propagation Delays Simulated

	Min	Avg	Max
PLUS (+1)	12	81	200
Minus (-1)	6	41	100

Table 6-7. Buffer Sizes Simulated

	PLUS (+1)			Minus (-1)		
Router	Min	Avg	Max	Min	Avg	Max
Backbone	325.528×10^3	732.437×10^3	130.211×10^4	1.153×10^3	2.606×10^3	4.654×10^3
POP	40.691×10^3	91.555×10^3	162.764×10^3	221	505	908
Access	6.47×10^3	14.557×10^3	25.879×10^3	91	207	369

Factor x2, think time, represents the average (exponentially distributed) interval (in ms) before a source initiates a new flow after completing a previous flow. A longer think time leads to lower demand on the network. Factor x3 controls the distribution of sources throughout the topology. The uniform distribution tends to spread congestion more evenly across the topology, while the skewed distribution tends to concentrate congestion more toward fast access routers. The number of sources in the topology is determined by a combination of factor x3 and two fixed factors, Bsources and ΔU, shown in Table 6-8. The net effect on the maximum number of simulated sources is given in Table 6-9. Table 6-8 also gives the fixed distribution of receivers, which creates a bias toward placing receivers under typical access routers. Further, Table 6-8 records the initial slow-start threshold – fixed to an arbitrarily large number of packets for the current simulation experiment.

Table 6-8. Fixed Parameters Related to Sources and Receivers

Parameter	Definition	Value
Bsources	Basic number of sources per access router	1000
ΔU	Avg. sources per access router = Bsources x ΔU	2
P(Nr)	Probability receiver under normal access router	0.6
P(Nrf)	Probability receiver under fast access router	0.2
P(Nrd)	Probability receiver under directly connected access router	0.2
sst_{INT}	Initial slow-start threshold in packets	$2^{31}/2$

Table 6-9. Number of Simulated Sources

PLUS (+1)	Minus (-1)
278×10^3	174.6×10^3

Several fixed parameters, shown in Table 6-10, control the operation of the simulation. The basic simulation time step is set to 1 ms and measurements are taken 5 times/sec, i.e., measurement interval (mi) duration is 200 ms. Total simulated time is (7500 mi/5 mi/s) = 1500 s, which amounts to (1500 s/60 s/m =) 25 minutes simulated for each condition. In order to reduce memory consumption, measures are buffered for only (1500 mi/5 mi/s/60 s/m =) 5 minutes before being written to disk. Table 6-10 also shows the fixed random number seed used for each run.

Table 6-10. Fixed Simulation Control Parameters

Parameter	Definition	Value
M	Number of Time Steps per Measurement Interval	200
MI	Number of Measurement Intervals Simulated	7500
MB	Number of Measurement Intervals Buffered	1500
Rnseed	Random Number Seed	200000
TSD	Duration of Each Time Step in seconds	0.001

For each condition, the 25 simulated minutes are orchestrated into the same scenario, shown in Fig. 6-2. Each time period consists of simulated traffic with specific properties, as defined below. The first 10 minutes, used primarily as a warm-up period,

consists of simulated Web traffic. The subsequent 15 minutes are divided into three, five-minute periods. At the beginning of the first time period (TP1), ongoing Web traffic is augmented by three long-lived flows, which continue for the duration of the simulation. All flows initiated on very fast paths (i.e., DD flows) during TP2 carry jumbo file transfers. At the onset of TP3, all newly initiated flows return to a pattern of simulated Web traffic; any residual backlog of ongoing, jumbo file transfers started during TP2 will continue into TP3 until they complete or the simulation ends. As explained below in Sec. 6.1.3, separate measurements are made in each time period, and selected measurements are totaled over the entire 25 minutes of the simulated scenario.

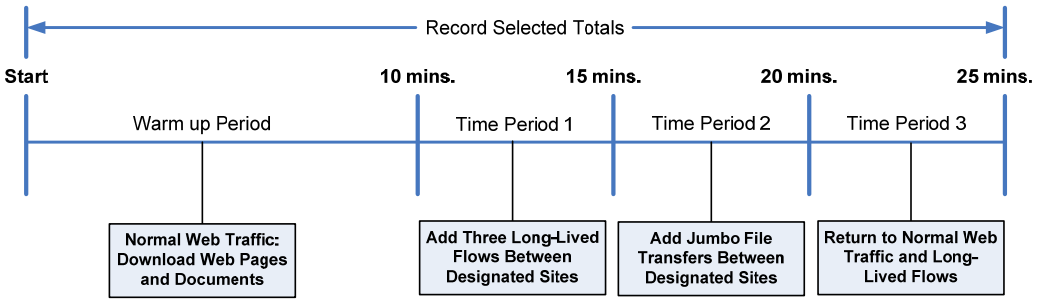

Figure 6-2. Scenario Adopted for Each Simulated Condition

Table 6-11. Fixed Parameters Specifying Simulated User Traffic

Parameter	Definition	Value
α	Shape parameter for Pareto distribution of file sizes	1.5
Fx	Document size = x5 x Fx	10
P(F)	Probability a file is a document	0.01
Jon	Jumbo file transfers begin after Jon x 25 minutes	0.6
Joff	Jumbo file transfers cease after Joff x 25 minutes	0.8
Jx	Jumbo file size = file (or document) size x Jx	100

Table 6-11 specifies the primary fixed parameters controlling generation of user traffic over the 25-minute scenario. Table 6-12 gives fixed parameters for the three long-lived flows. Fundamental file sizes within the simulation are chosen from a Pareto distribution with a mean given by factor x5, which equals either 50 or 100 packets depending on the level of the factor. The shape parameter for the Pareto distribution is fixed at 1.5. Factor x5 represents Web pages with an average size of (50 packets x 1500 bytes/packet =) 75 Kbytes or (100 packets x 1500 bytes/packet =) 150 Kbytes. Recall, however, that MesoNet packets have no size, so file sizes are specified in packets. With a fixed probability of 0.01, i.e., P(F), a document will be downloaded from a Web site. Document sizes are determined by multiplying a file size selected for a Web page by a fixed factor of 10, i.e., Fx, so downloaded documents average either 500 packets (750 Kbytes) or 1000 packets (1.5 Mbytes), depending on the value of x5. This combination of Web pages and documents makes up the pattern of user traffic labeled as normal Web traffic.

Jumbo file transfers, initiated on all DD flows started during TP2, are controlled by three parameters. Jon determines the proportion of elapsed simulation time before jumbo transfers begin and Joff defines the proportion after which initiation of jumbo transfers cease. The size of a jumbo transfer is determined by multiplying the file size chosen for normal Web traffic by a factor of 100 (Jx). This means that jumbo file transfers will average ((50 x .99 + 500 x .01) x 100 =) 251 packets (376.5 Kbytes) or ((100 x .99 + 1000 x .01) x 100 =) 1089 packets (1.63 Mbytes), depending upon the setting of factor x5. Note that all transfers – whether Web pages, documents or jumbo files – are subject to the heavy-tailed property of the Pareto distribution, so transfers may be much larger than the average size.

Table 6-12. Fixed Parameters Specifying Long-Lived Flows

Identifier	Definition	Source Router	Receiver Router	Start Time
L1	Long-distance flow	B0a	K0a	0.4 x 25 mins.
L2	Medium-distance flow	C0a	I0a	0.4 x 25 mins.
L3	Short-distance flow	E0a	F0a	0.4 x 25 mins.

Table 6-12 gives the details for the three long-lived flows that commence in TP1 and continue throughout the remainder of the simulated scenario. Each long-lived flow transmits continuously at whatever rate can be achieved over a very fast (DD) path. The maximum transmission rate for long-lived flows is 80 x 10^3 pps (i.e., long-lived sources and receivers operate at the rate defined by Hfast). Flow L1 traverses the length of the topology. Flow L3 traverses the width of the topology. Flow L2 traverses the middle of the topology. These flows serve several purposes. First, the flows can be individually tracked and measured in detail. This reveals the temporal evolution of the flows, as well as how the flows are influenced by other flows. Second, since the flows transit different distances across the network, measurements can be taken to determine the lag time before each flow reaches its maximum transmission rate. Third, the flows transit directly-connected access routers, so in TP2 the influence of jumbo file transfers may be observed.

6.1.2 Conditions Simulated

For the six factors enumerated in Table 6-2, a two-level experiment design would require simulating (2^6 =) 64 conditions. Given the size and speed of the network we wished to simulate, we decided we could afford examining only 32 conditions. For this reason, we adopted a 2^{6-1} orthogonal fractional factorial (OFF) design. To generate the subset of conditions required by the design, we selected values from Table 6-2 as specified in Table 6-13, a template where each row defines a condition as a combination of the six input factors. The resulting experiment design (in Table 6-14) provides a good balance of individual factors as well as orthogonal combinations of factors. The 2^{6-1} design is a resolution VI design, which means that main effects will be confounded (explained in Sec. 2.5.1) only with five-factor interactions. In addition, two-factor interactions will be confounded only with four-factor interactions. Our previous sensitivity analysis revealed that our model is driven primarily by main effects; even two-factor interactions were not

very evident. For these reasons, we can obtain all necessary information by simulating only 32 of the 64 conditions defined by our input factors.

Table 6-13. Template Specifying a 2^{6-1} Orthogonal Fractional Factorial Design

Factor->	X1	X2	X3	X4	X5	X6
Condition	--	--	--	--	--	--
1	-1	-1	-1	-1	-1	-1
2	+1	-1	-1	-1	-1	+1
3	-1	+1	-1	-1	-1	+1
4	+1	+1	-1	-1	-1	-1
5	-1	-1	+1	-1	-1	+1
6	+1	-1	+1	-1	-1	-1
7	-1	+1	+1	-1	-1	-1
8	+1	+1	+1	-1	-1	+1
9	-1	-1	-1	+1	-1	+1
10	+1	-1	-1	+1	-1	-1
11	-1	+1	-1	+1	-1	-1
12	+1	+1	-1	+1	-1	+1
13	-1	-1	+1	+1	-1	-1
14	+1	-1	+1	+1	-1	+1
15	-1	+1	+1	+1	-1	+1
16	+1	+1	+1	+1	-1	-1
17	-1	-1	-1	-1	+1	+1
18	+1	-1	-1	-1	+1	-1
19	-1	+1	-1	-1	+1	-1
20	+1	+1	-1	-1	+1	+1
21	-1	-1	+1	-1	+1	-1
22	+1	-1	+1	-1	+1	+1
23	-1	+1	+1	-1	+1	+1
24	+1	+1	+1	-1	+1	-1
25	-1	-1	-1	+1	+1	-1
26	+1	-1	-1	+1	+1	+1
27	-1	+1	-1	+1	+1	+1
28	+1	+1	-1	+1	+1	-1
29	-1	-1	+1	+1	+1	+1
30	+1	-1	+1	+1	+1	-1
31	-1	+1	+1	+1	+1	-1
32	+1	+1	+1	+1	+1	+1

6.1.3 Responses Measured

The remainder of the experiment design addresses the system responses measured for each simulated condition. At the top level, we measured a collection of 45 instantaneous responses averaged over each time period and we aggregated 28 measures across all 25 minutes of the simulated scenario. We designate the instantaneous responses as y1 through y45 and we designate the aggregate responses as T.y1 through T.y28. We begin by defining the instantaneous average measures, which may be divided into three categories: (1) measures of macroscopic network behavior, (2) measures of user experience and (3) measures of buffer usage in designated access routers.

6.1.3.1 Measures of Macroscopic Behavior. We selected 12 responses (see Table 6-15) to represent macroscopic behavior in the simulated network. Each response is measured during each measurement interval, which forms a time series. The measured values are then averaged over the relevant time period. Five responses (highlighted in yellow) characterize the status of non-idle flows. Idle flows are those flows waiting within a think period. Non-idle flows are either connecting (y42) or active (y1). Active flows may be operating within initial slow start (y43) or within the normal TCP congestion control

regime (y44) or an alternate regime (y45). The precise nature of the alternate congestion control regime depends upon which congestion avoidance algorithm (recall Table 6-1) is adopted for a particular set of runs. As one would expect, y45 will always be zero when normal TCP Reno congestion avoidance is in use and y44 will be zero for FAST.

Table 6-14. Instantiated Robustness Conditions for 2^{6-1} Experiment Design

Factor->	X1	X2	X3	X4	X5	X6
Condition	--	--	--	--	--	--
1	4000	2500	.1/.6/.3	1	50	RTTxCapacity/SQRT(N)
2	8000	2500	.1/.6/.3	1	50	RTTxCapacity
3	4000	5000	.1/.6/.3	1	50	RTTxCapacity
4	8000	5000	.1/.6/.3	1	50	RTTxCapacity/SQRT(N)
5	4000	2500	.3/.3/.3	1	50	RTTxCapacity
6	8000	2500	.3/.3/.3	1	50	RTTxCapacity/SQRT(N)
7	4000	5000	.3/.3/.3	1	50	RTTxCapacity/SQRT(N)
8	8000	5000	.3/.3/.3	1	50	RTTxCapacity
9	4000	2500	.1/.6/.3	2	50	RTTxCapacity
10	8000	2500	.1/.6/.3	2	50	RTTxCapacity/SQRT(N)
11	4000	5000	.1/.6/.3	2	50	RTTxCapacity/SQRT(N)
12	8000	5000	.1/.6/.3	2	50	RTTxCapacity
13	4000	2500	.3/.3/.3	2	50	RTTxCapacity/SQRT(N)
14	8000	2500	.3/.3/.3	2	50	RTTxCapacity
15	4000	5000	.3/.3/.3	2	50	RTTxCapacity
16	8000	5000	.3/.3/.3	2	50	RTTxCapacity/SQRT(N)
17	4000	2500	.1/.6/.3	1	100	RTTxCapacity
18	8000	2500	.1/.6/.3	1	100	RTTxCapacity/SQRT(N)
19	4000	5000	.1/.6/.3	1	100	RTTxCapacity/SQRT(N)
20	8000	5000	.1/.6/.3	1	100	RTTxCapacity
21	4000	2500	.3/.3/.3	1	100	RTTxCapacity/SQRT(N)
22	8000	2500	.3/.3/.3	1	100	RTTxCapacity
23	4000	5000	.3/.3/.3	1	100	RTTxCapacity
24	8000	5000	.3/.3/.3	1	100	RTTxCapacity/SQRT(N)
25	4000	2500	.1/.6/.3	2	100	RTTxCapacity/SQRT(N)
26	8000	2500	.1/.6/.3	2	100	RTTxCapacity
27	4000	5000	.1/.6/.3	2	100	RTTxCapacity
28	8000	5000	.1/.6/.3	2	100	RTTxCapacity/SQRT(N)
29	4000	2500	.3/.3/.3	2	100	RTTxCapacity
30	8000	2500	.3/.3/.3	2	100	RTTxCapacity/SQRT(N)
31	4000	5000	.3/.3/.3	2	100	RTTxCapacity/SQRT(N)
32	8000	5000	.3/.3/.3	2	100	RTTxCapacity

Table 6-15. Responses Characterizing Macroscopic Behavior

Response	Definition
y42	Average number of connecting flows
y1	Average number of active (i.e., connected) flows
y43	Average number of active flows in initial slow start
y44	Average number of active flows in normal congestion-control mode
y45	Average number of active flows in alternate congestion-control mode
y3	Average packets output per measurement interval
y5	Average flows completed per measurement interval
y6	Average retransmission rate
y7	Average smoothed round-trip time (SRTT)
y8	Average round-trip queuing delay
y2	Average congestion-window increases per active flow
y4	Average congestion window per active flow

Two responses (highlighted in blue) represent network-wide throughput, either as packets output (y3) per measurement interval or as flows completed (y5) per measurement interval. Three responses (highlighted in orange) summarize network-wide congestion. One reflection of congestion is the average retransmission rate (y6). Two

other responses reflect congestion-induced delay: average SRTT (y7), from which the average round-trip propagation delay may be subtracted to reveal the average round-trip queuing delay (y8).

The two remaining responses (in green) relate to the congestion window for an average active flow. One (y2) measures the average number of window increases per flow in a measurement interval, while the other (y4) measures the average congestion window size (in packets) per flow. These measures reflect congestion but can also reflect details associated with the operation of specific congestion control algorithms.

6.1.3.2 Measures of User Experience. We use *goodput* as a fundamental measure of user experience. We define goodput as the number of packets per second (pps) received at the user level on a given flow. Thus, goodput excludes retransmissions. Since various flows transit the topology on paths that possess different characteristics, we measure user experience for flows on each path class (recall Table 6-2). Recognizing that goodput can be influenced by the number of flows sharing the same path, we measure the relevant characteristics. For example, Table 6-16 shows how we characterize user experience for flows on very fast (DD) paths. We measure not only average goodput (y9) but also the average number of active flows (y10) and the average number of completed flows (y11). We assume that completed flows finish at uniformly distributed times in a given measurement interval. We then compute (y12) the average aggregate number of pps delivered on all DD flows. This allows us to investigate average goodput in a nuanced fashion. We make similar measurements for (DF and FF) flows transiting fast paths (Table 6-17) and for those (DN, FN, NN) flows transiting typical paths (Table 6-18).

Table 6-16. Responses Characterizing User Experience on Very Fast Paths

Response	Definition
y9	Average goodput (pps) for DD flows
y10	Average number of active DD flows
y11	Average number of DD flows completed per measurement interval
y12	Average aggregate number of DD packets delivered per second = y9x(y10+(y11/2))

Table 6-17. Responses Characterizing User Experience on Fast Paths

Response	Definition
y13	Average goodput (pps) for DF flows
y14	Average number of active DF flows
y15	Average number of DF flows completed per measurement interval
y16	Average aggregate number of DF packets delivered per second = y13x(y14+(y15/2))
y21	Average goodput (pps) for FF flows
y22	Average number of active FF flows
y23	Average number of FF flows completed per measurement interval
y24	Average aggregate number of FF packets delivered per second = y21x(y22+(y23/2))

Table 6-18. Responses Characterizing User Experience on Typical Paths

Response	Definition
y17	Average goodput (pps) for DN flows
y18	Average number of active DN flows
y19	Average number of DN flows completed per measurement interval
y20	Average aggregate number of DN packets delivered per second = y17x(y18+(y19/2))
y25	Average goodput (pps) for FN flows
y26	Average number of active FN flows
y27	Average number of FN flows completed per measurement interval
y28	Average aggregate number of FN packets delivered per second = y25x(y26+(y27/2))
y29	Average goodput (pps) for NN flows
y30	Average number of active NN flows
y31	Average number of NN flows completed per measurement interval
y32	Average aggregate number of NN packets delivered per second = y21x(y22+(y23/2))

We also measure user experience individually for the three long-lived flows defined in the scenario. For these flows, we measure only average goodput, as shown in Table 6-19.

Table 6-19. Responses Characterizing User Experience on Long-Lived Flows

Response	Definition
y33	Average goodput (pps) for the long-distance flow (L1)
y34	Average goodput (pps) for the medium-distance flow (L2)
y35	Average goodput (pps) for the short-distance flow (L3)

Table 6-20. Responses Charactering Buffer Usage in Directly Connected Access Routers

Response	Definition
y36	Average buffer saturation for router B0a
y37	Average buffer saturation for router C0a
y38	Average buffer saturation for router E0a
y39	Average buffer saturation for router F0a
y40	Average buffer saturation for router I0a
y41	Average buffer saturation for router K0a

6.1.3.3 Measures of Buffer Usage. The construction of the simulated topology ensures that most (if not all) significant buffer usage occurs at the access routers, most of which have much lower speeds than the POP and backbone routers. The topology used in the simulation consists of 139 access routers. We chose to analyze buffer usage only for the six directly connected access routers, as shown in Table 6-20. For each router, we measure average buffer saturation, defined as the ratio of buffers in use to buffers available.

6.1.3.4 Aggregate Measures. We measure 28 responses over the course of the entire 25-minute scenario, including the warm-up period. These responses fall into three broad categories: (1) measures of macroscopic behavior, (2) measures of user experience and (3) measures of flow distribution among backbone routers. We discuss each of these in turn.

As shown in Table 6-21, we aggregate the number of data packets injected (T.y1) into the network as well as the number of packets delivered (T.y2) over the entire 25 minutes simulated. We provide similar measures for flows connected (T.y3) and completed (T.y4). For connected flows, we also measure (T.y5) the average number of SYN packets sent per flow. This provides some measure of the degree to which congestion impedes the ability of flows to connect.

Table 6-21. Aggregate Responses Characterizing Macroscopic Behavior

Response	Definition
T.y1	Aggregate packets input
T.y2	Aggregate packets output
T.y3	Aggregate flows connected
T.y4	Aggregate flows completed
T.y5	Average SYNs sent per flow

We characterize user experience for completed flows in each path class using two measures: (1) aggregate number of flows completed and (2) average per-flow goodput on the completed flows. We consider completed flows in aggregate for two reasons. First, we can include flows across the entire 25 simulated minutes. Second, some flows may have trouble completing, so we can view goodput for completed flows as a best case measure of user experience. Below, we identify the measures for each path class: very fast paths (Table 6-22), fast paths (Table 6-23) and typical paths (Table 6-24).

Table 6-22. Responses Characterizing User Experience for Completed Flows on Very Fast Paths

Response	Definition
T.y6	Aggregate number of DD flows completed
T.y7	Average goodput (pps) for completed DD flows

Table 6-23. Responses Characterizing User Experience for Completed Flows on Fast Paths

Response	Definition
T.y8	Aggregate number of DF flows completed
T.y9	Average goodput (pps) for completed DF flows
T.y12	Aggregate number of FF flows completed
T.y13	Average goodput (pps) for completed FF flows

Table 6-24. Responses Characterizing User Experience for Completed Flows on Typical Paths

Response	Definition
T.y10	Aggregate number of DN flows completed
T.y11	Average goodput (pps) for completed DN flows
T.y14	Aggregate number of FN flows completed
T.y15	Average goodput (pps) for completed FN flows
T.y16	Aggregate number of NN flows completed
T.y17	Average goodput (pps) for completed NN flows

The final set of responses measure the distribution of flows transiting the 11 backbone routers. As shown in Table 6-25, we simply total the number of completed flows that transit each backbone router during the 25 simulated minutes. Measuring these responses enables us to detect whether any of the congestion control regimes shift the workload experienced by backbone routers.

Table 6-25. Responses Characterizing Distribution of Flows among Backbone Routers

Response	Definition
T.y18	Aggregate completed flows transiting backbone router A
T.y19	Aggregate completed flows transiting backbone router B
T.y20	Aggregate completed flows transiting backbone router C
T.y21	Aggregate completed flows transiting backbone router D
T.y22	Aggregate completed flows transiting backbone router E
T.y23	Aggregate completed flows transiting backbone router F
T.y24	Aggregate completed flows transiting backbone router G
T.y25	Aggregate completed flows transiting backbone router H
T.y26	Aggregate completed flows transiting backbone router I
T.y27	Aggregate completed flows transiting backbone router J
T.y28	Aggregate completed flows transiting backbone router K

6.2 Experiment Execution and Data Collection

In this section, we shift gears to discuss the mechanics of executing the experiments and collecting the data. We describe the resources available for conducting the simulations and also the resource requirements. In addition, we define the format in which we collected data to capture our measured responses.

6.2.1 Experiment Execution

We simulated seven congestion control mechanisms (recall Table 6-1) under the same 32 conditions (recall Table 6-14), requiring (7 x 32 =) 224 separate simulation runs. We had six available compute servers with the characteristics defined in Table 6-26. Each compute server provided 8 processors, so we had a total of (6 x 8 =) 48 processors on which we could execute simulations in parallel. Each of the compute servers was

provisioned with 32 Gbytes of memory. Two of the servers (ws9 and ws10) had four dual-core AMD Opertron™ 8218 processors operating at 2.6 GHz, while the remaining servers (ws11-ws4) had four dual-core AMD Opertron™ 8222 SE processors operating at 3 GHz. All of the compute servers executed under the control of the 64-bit version of Microsoft Windows[2] Server 2003™.

Table 6-26. Characteristics of Compute Servers Used to Execute the Simulations

Compute Server	Physical Processors	Speed (GHz)	Memory (GB)	Operating System
ws9	8	2.6	32	Windows Server 2003 R2 x64 Edition SP2
ws10	8	2.6	32	Windows Server 2003 R2 x64 Edition SP2
ws11	8	3.0	32	Windows Server 2003 R2 x64 Edition SP2
ws12	8	3.0	32	Windows Server 2003 R2 x64 Edition SP2
ws13	8	3.0	32	Windows Server 2003 R2 x64 Edition SP2
ws14	8	3.0	32	Windows Server 2003 R2 x64 Edition SP2

Given 48 processors (also referred to as central-processing units, or CPUs), we were able to run one congestion control mechanism simultaneously against all 32 conditions and we could run another congestion control mechanism against half (16) of the conditions. As shown in Table 6-27, we ran simulations for five congestion control mechanisms (BIC, CTCP, FAST, HTCP and TCP) on the four faster compute servers (ws11-ws14) and we ran simulations for the other two (HSTCP and Scalable TCP) on the slower compute servers (ws9-ws10).

Table 6-27. Processing Requirements for Simulations Mapped to Specific Compute Servers (Units are Processor Days)

	Compute Servers ws11-ws14						Compute Servers ws9-ws10		
	BIC	CTCP	FAST	HTCP	TCP	Totals	HSTCP	Scalable	Totals
CPU time (32 runs)	91.5	97.2	93.4	96.4	94.2	472.5	108.6	110.5	219.1
Avg. CPU time (per run)	2.86	3.04	2.92	3.01	2.94	14.77	3.39	3.46	13.70 (6.85x2)
Min. CPU time (one run)	1.16	1.33	1.44	1.40	1.28		1.61	1.51	
Max. CPU time (one run)	5.94	5.85	5.17	5.84	5.63	28.42	6.57	6.61	26.37 (13.18x2)

Each simulated condition required about 1.25 Gbytes of memory, so running 8 simulations in parallel on one compute server required about 10 Gbytes, or about 1/3 of

[2] Our simulation model, MesoNet, is written in the SLX simulation language. The SLX compiler and run-time require the Microsoft Windows™ operating system.

the available memory. On the other hand, as indicated in Table 6-27, running all 224 simulations required substantial processing resources: (472.5 + 219.1 =) 691.6 processor (CPU) days. Running 48 simulations in parallel, we could potentially have finished the experiment in (691.6 processor days/48 processors =) 14.4 days. Achieving this goal required some astute management of the runs. For example, launching 32 runs for a given congestion control mechanism and then waiting for all runs to complete prior to starting the next set would advance progress at a pace congruent with the maximum processor time required among the 32 simulations run for each congestion control mechanism. As shown in the last row of Table 6-27, this naïve approach would have completed the simulations in 28.42 days, which is the time required to run the five congestion control mechanisms on ws11-ws14. Using the same naïve approach, the two mechanisms simulated on ws9 and ws10 could complete (28.42 - 26.37 =) two days sooner. Note that since only 16 of the 32 conditions could be run in parallel on ws9 and ws10 the processor time required must be doubled, e.g., (6.57 + 6.61) x 2 = 26.37 days.

To complete the simulations in about two weeks one needs to achieve a rate of progress close to the average processor time per run, shown in the second row of Table 6-27. This can be done by first estimating the relative run time required by each simulated condition, and then sorting the conditions by estimated run time into two lists: (1) shortest-to-longest and (2) longest-to-shortest. The two lists define a mapping function for scheduling simulation runs. Whenever a simulation finishes for a specific condition on the first list, select the next condition to start based on its mate from the second list. In this way, as short conditions finish they are replaced by long conditions and vice versa. This enables completing the simulations in just over two weeks, the maximum of 14.77 days and 13.7 days, as shown in the second row of Table 6-27.

Why does the simulation require so much processor time? Each experiment simulates the operation of up to hundreds of thousands of simultaneously active flows over a period of 25 simulated minutes. Each flow that starts during the simulation must be modeled, as well as every packet sent on each flow. Each packet transits several routers as it propagates through the simulated topology. As shown in Table 6-28, the average condition requires simulating just over 74 million flows during the 25 simulated minutes. This amounts to simulating around 7 billion data packets, each of which has a matching acknowledgment. Thus, in a given simulation run 14 billion packets are sent on average. For all conditions across all congestion control algorithms, more than 16.5 billion flows and 3 trillion packets (1.5 trillion data packets and 1.5 trillion acknowledgments) must be simulated. In Chapter 7 we investigate whether a scaled down network simulation can provide sufficient information while requiring less processor time.

Table 6-28. Characterization of the Number of Flows and Data Packets Simulated

Statistic	Flows Completed	Data Packets Sent
Avg. Per Condition	74.033×10^6	6.912×10^9
Min. Per Condition	40.966×10^6	3.147×10^9
Max. Per Condition	154.914×10^6	11.917×10^9
Total All Runs	16.583×10^9	1.548×10^{12}

6.2.2 Data Collection

We collected summary response measurements into four files: one file for each of three five-minute time periods (recall Fig. 6-2) and one file containing data aggregated across the entire 25-minute scenario. Table 6-29 shows the format used for each time-period file. Each file consists of (7 x 32 =) 224 rows of 47 columns. The header row, shown for clarity in Table 6-29, was not included in the data file. The first column identifies the congestion control algorithm and the second column identifies the condition. Each of the remaining columns contains the value for one of the 45 responses measured for the relevant time period (recall Sec. 6.1.3). A response represents the average value across all measurement intervals within the time period.

Table 6-29. Format Adopted for Each Time-Period Data File

Algorithm	Run	y1	y2	…	y44	y45
1	1	33473.81	9.111708	…	21090.14	1834.653
1	2	15370.5	41.35026	…	0.758	0.555333
…	…	…	…	…	…	…
1	31	107357	24.93412	…	602.5607	110.8673
1	32	28287.67	9.126458	…	23397.35	83.21733
…	…	…	…	…	…	…
7	1	34108.22	9.115773	…	23487.56	0
7	2	15333.99	41.4579	…	0.116	0
…	…	…	…	…	…	…
7	31	108421.6	1.899736	…	99872.35	0
7	32	27644.4	25.66166	…	363.9507	0

As shown in Table 6-30, we adopted a similar format for the file containing aggregate responses. In this case, the file included only 30 columns because the number of responses was limited to 28. As discussed in Sec. 6.1.3.4, most values represent an aggregation across the entire 25-minute scenario, while some values represent an average goodput or SYN rate across the scenario.

Table 6-30. Format Adopted for Reporting Aggregate Measures

Algorithm	Run	T.y1	T.y2	…	T.y27	T.y28
1	1	6141323986	5849131237	…	13251058	21098541
1	2	7569164137	7565416432	…	16627869	28921523
…	…	…	…	…	…	…
1	31	6267074928	5653265836	…	8205819	6654958
1	32	9246017312	9244473772	…	13946759	10825599
…	…	…	…	…	…	…
7	1	6132681434	5854382727	…	13255301	21163097
7	2	7564415186	7562570309	…	16634422	28934311
…	…	…	…	…	…	…
7	31	6158229285	5557616933	…	8215894	6648529
7	32	9245995194	9245121085	…	13993455	10880051

To support some detailed analyses (as discussed below in Sec. 6.3) we also used selected time-series data files as output directly by MesoNet. A time series for a particular response simply provides the raw measurement data that was used to create the summarization reported in Table 6-29.

6.3 Data Analysis Approach

In this section, we introduce and explain our approach to data analysis. For illustrative purposes, we also provide a few insights into the behavior of our simulated network. We defer a complete presentation of key results until Sec. 6.4.

We employed three main techniques for analyzing data: (1) cluster analysis, (2) detailed analysis of individual responses and (3) summary analysis of all responses across all conditions. Where advantageous, we also adopted some useful strategies to explore the data. We address each of these topics in turn, beginning with cluster analysis.

6.3.1 Cluster Analysis

We use cluster analysis to provide a comprehensive comparison of differences among all congestion control algorithms for all responses and conditions. Results from the cluster analysis establish whether any of the algorithms generate a distinctive response to the various conditions. To perform the analysis we used hierarchical clustering tools from the MATLAB™ Statistics Toolbox™ [87]. Hierarchical clustering requires selection of a function to compute distances between points in the vector space composed by the response data. We used the standardized Euclidean distance function.

$$Dist(Y_i, Y_j) \equiv \sqrt{\sum_{m=1}^{45} \frac{(Y_{im} - Y_{jm})^2}{(\sigma_m)^2}} \tag{1}$$

Equation (1) computes the inter-algorithm distance in 45-dimension space, where each dimension *m* represents one response. Here, Y_i and Y_j represent the response vectors for the *i*th and *j*th congestion control algorithms. (Note that we use a 28-dimension space when clustering aggregate results.) Distances for each response are normalized with respect to response variance. This enables distances to be placed on a similar scale. (Any response with zero standard deviation is excluded from the distance computation.) A pair of algorithms with close proximity may be linked together within a cluster.

We measure the linkage between clusters of algorithms as the average distance between responses associated with each algorithm in each cluster. The linkage function, shown in (2), uses the Euclidean-distance function from (1).

$$D(r,s) \equiv \frac{1}{n_r n_s} \sum_{k=1}^{n_r} \sum_{l=1}^{n_s} Dist(Y_{k,r}, Y_{l,s}) \tag{2}$$

Equation (2) computes the linkage between any two clusters *r* and *s*, containing n_r and n_s congestion control algorithms, respectively. $Y_{k,r}$ represents the response vector for the *k*th congestion control algorithm in cluster *r*; similarly, $Y_{l,s}$ represents the response vector for

the *l*th congestion control algorithm in cluster *s*. The linkage function is used to place binary clusters into larger clusters, forming a hierarchical tree.

The final step in hierarchical clustering is to suggest which congestion control algorithms should be included within the same cluster. For this purpose, we use the MATLAB™ dendrogram ( ) function to color the lines on the hierarchical tree whenever the linkage value between two clusters falls below 70 % of the maximum linkage value. The net result from clustering is a diagram, such as Fig. 6-3, suggesting relationships among the congestion control algorithms. Identifiers for the seven congestion control algorithms (Table 6-1) are plotted on the x axis and the y axis displays standardized distances between algorithms in the subordinate cluster(s). Here, the clustering suggests algorithms 4 and 6 give similar results and algorithms 1 and 2 give similar results. The remaining algorithms are dissimilar, with algorithm 3 being most dissimilar from the others.

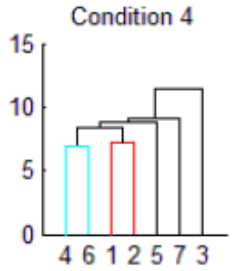

Figure 6-3. Dendrogram Illustrating Clustering Based on Responses for Condition 4 During Time Period One (TP1) – x axis gives the algorithm identifier from Table 6-1 and y axis gives the standardized Euclidean distance between algorithms or clusters of algorithms

Clustering must be performed individually on the various conditions because the conditions can yield results that are quite dissimilar. One may obtain an overall picture of clustering across conditions by plotting together 32 dendrograms, one per condition. Fig. 6-4 shows such a plot for seven congestion control algorithms and related responses covering TP1. Review of the plot reveals that algorithm 3 appears distinctive under about 23 of the 32 conditions. Further, the responses generated by the different algorithms are indistinguishable in six conditions – in fact, are identical for condition 12, where the corresponding dendrogram shows zero distance between the algorithms. The remaining three conditions (2, 27 and 32) find small distinctions among the algorithms. As Fig. 6-4 illustrates, clustering analysis can reveal some significant overall patterns in the data.

A natural next step is to consider why algorithm 3 (FAST) is distinctive in many of the conditions but not in all. In other words, can we determine properties that distinguish among the conditions and then map those properties into hypotheses regarding the operation of algorithm 3? Given the input factors (x1…x6) defining the conditions, we suspect that distinct conditions represent differing levels of congestion within the simulated network. To confirm our suspicion, we can sort the conditions using some property, such as loss rate or retransmission rate, which reflects congestion. Fig. 6-5 displays a bar chart where conditions on the x axis are sorted in order of increasing retransmission rate (response y6) on the y axis. The bar chart shows that 16 conditions have much higher retransmission rates (reflecting higher congestion) than the others. Thus, half the conditions lead to significant congestion and half do not. To quantify the

difference, we include an inset bar chart in Fig. 6-5. The inset shows that the highest retransmission rate (for condition 11) among the uncongested conditions is an order of magnitude or more lower than the lowest retransmission rate (for condition 18) among the congested conditions. Examining the uncongested conditions in detail, one can declare somewhat arbitrary distinctions between conditions with no congestion (N), little congestion (L) and moderate congestion (M). We label Fig. 6-5 accordingly.

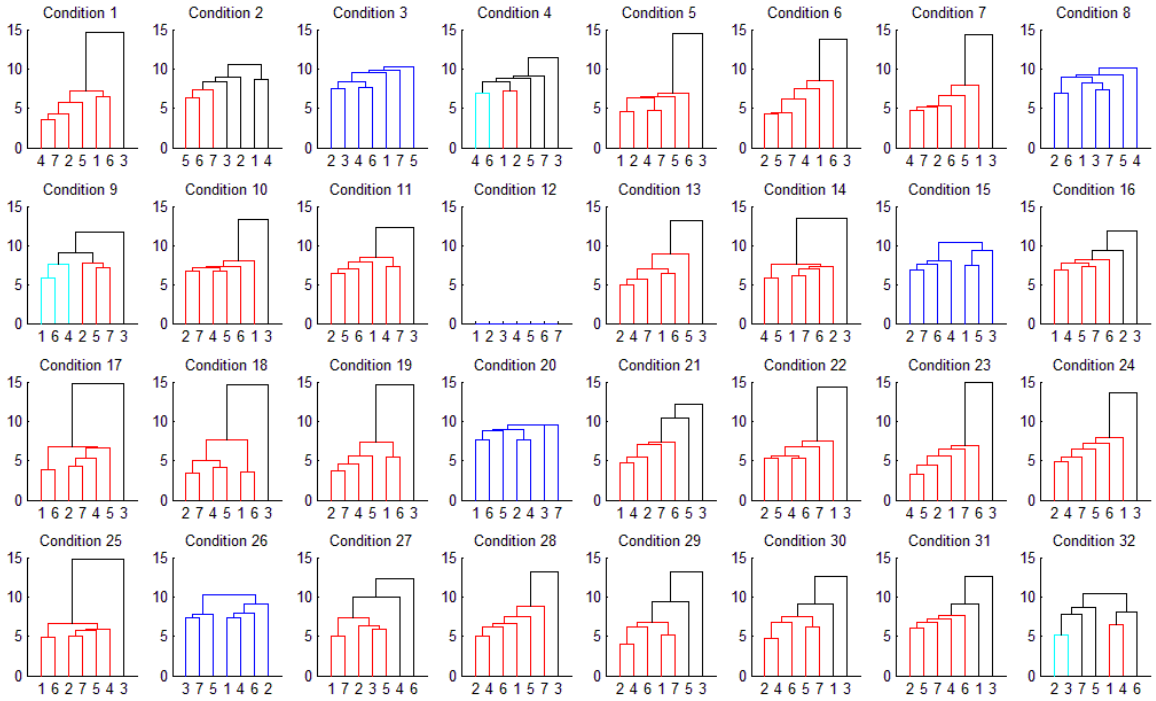

Figure 6-4. Cluster Analysis for 32 Conditions Using Data from Time Period One

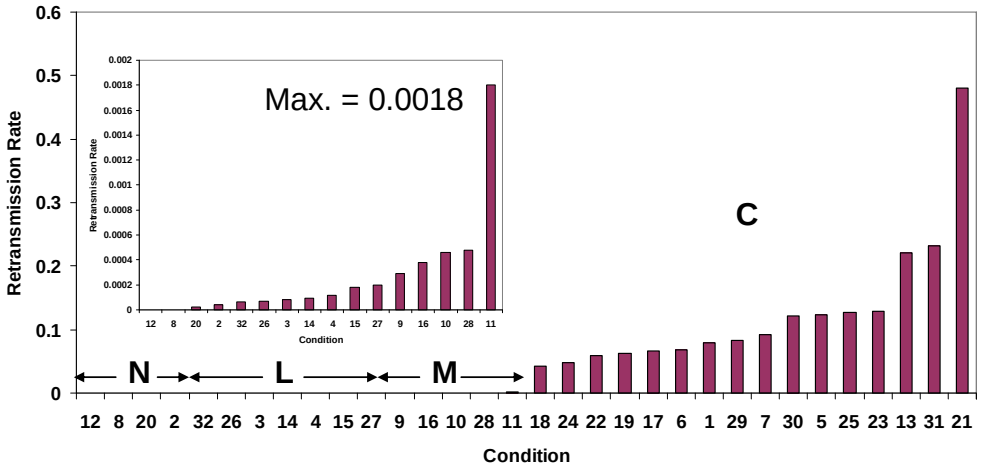

Figure 6-5. Conditions (x axis) Ordered from Least to Most Congested vs. Retransmission Rate (y axis), which is the proportion of all sent packets that were retransmissions

We can select one uncongested and one congested condition to examine more closely. Fig. 6-6 plots several time series that, taken together, show the distribution of flow states for (uncongested) condition 4 under standard TCP congestion control. The x

axis displays time over all three time periods measured for the simulated scenario. The y axis indicates the number of active (red curve) and connecting (yellow curve) flows. Additional curves decompose the active flows by congestion control state. For TP1 (3000-4500) the plot shows that most of the active flows operate in initial slow-start (green curve). This means that the network is sufficiently uncongested that most file transfers complete without a lost packet. Things change during TP2 (4500-6000) as jumbo file transfers induce congestion in the directly connected access routers. Congestion leads to losses, which increases the number of flows operating under normal congestion control procedures (brown curve). As jumbo file transfers diminish during TP3 (6000-7500), congestion ebbs so that, by time 6500, most active flows again complete file transfers without a lost packet. The same curves plotted for the other 15 uncongested conditions show similar patterns.

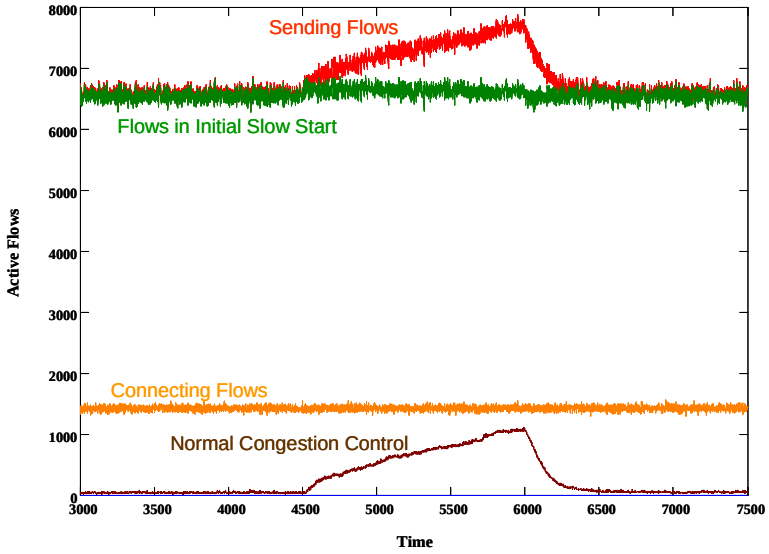

Figure 6-6. Distribution of Flow States over Three Time Periods under Condition 4 for Standard TCP – x axis shows time in 200 ms increments and y axis shows number of active flows in each state

The situation is much different for congested conditions. Fig. 6-7 plots the distribution of flow states for condition five under standard TCP congestion control. Notice that the number of active flows (red) averages about 125 x 10^3. Here, the vast majority (about 105 x 10^3) of those flows are operating under normal congestion control procedures (brown), which means these flows have suffered lost packets. Notice also that network congestion is sufficiently high so that introducing jumbo file transfers during TP 2 (4500-6000) makes very little difference in the overall distribution of flow states. The same curves plotted for the other 15 congested conditions show similar patterns.

Combining this new information with the previous cluster analysis provides substantial insight about conditions that lead to the distinctive behavior of algorithm 3. Fig. 6-8 reproduces an augmented version of Fig. 6-4. Here, we annotate the cluster plot for each condition with a character indicating the relative level of associated congestion. Reviewing the plot reveals that algorithm 3 is distinctive under conditions showing moderate to heavy congestion. The distinctiveness of algorithm 3 fades under conditions

with little or no congestion. Further, under the least congested condition (12), all seven congestion control mechanisms produced identical responses.

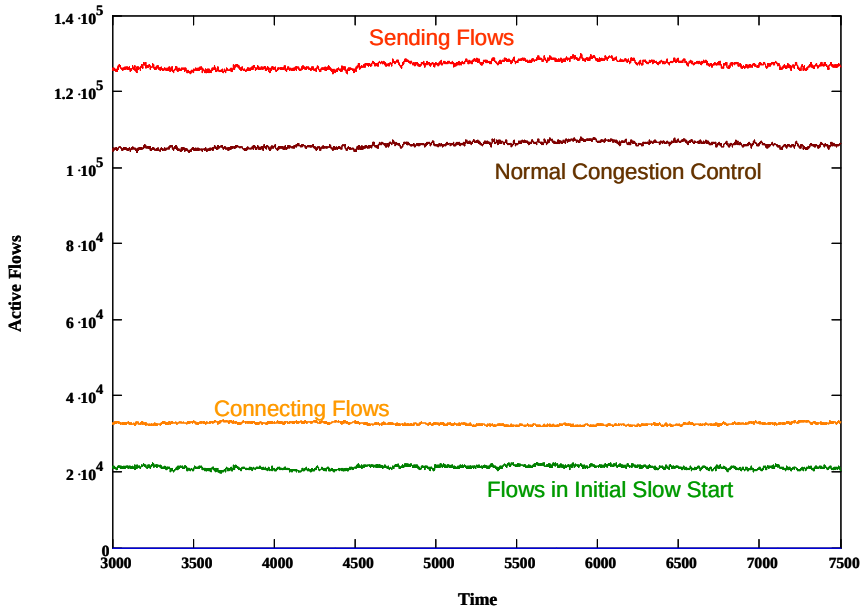

Figure 6-7. Distribution of Flow States over Three Time Periods under Condition 5 for Standard TCP– x axis shows time in 200 ms increments and y axis shows number of active flows in each state

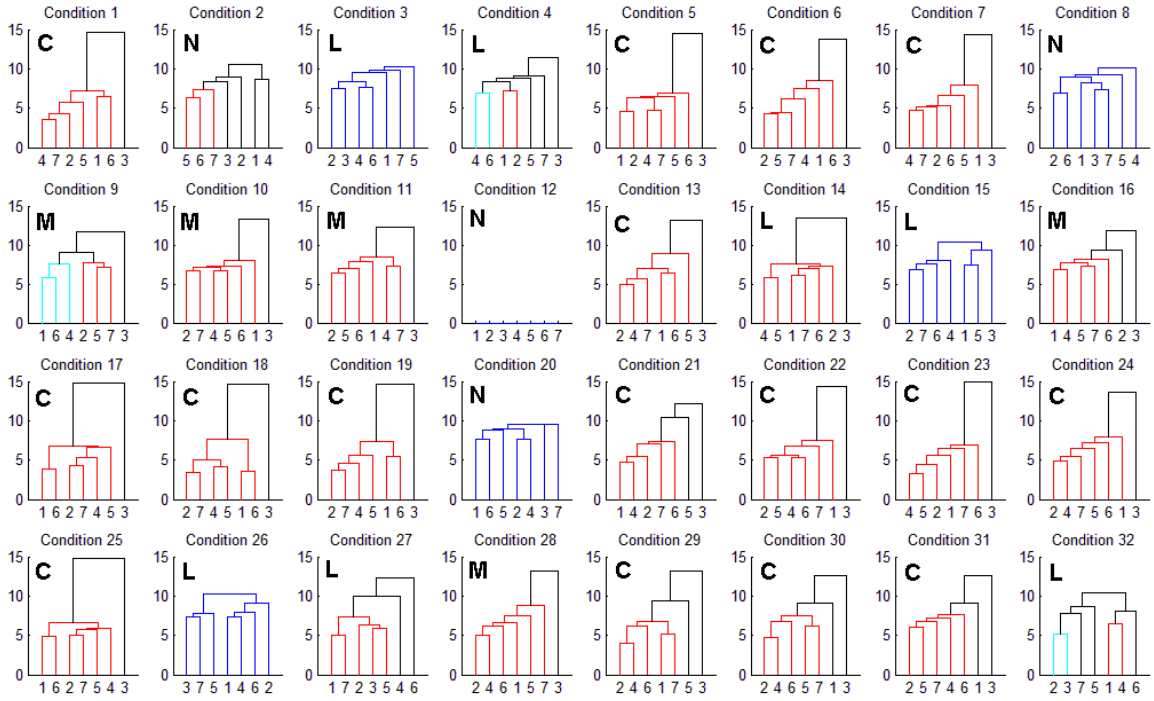

Figure 6-8. Cluster Analysis for Time Period One – Conditions Labeled with Congestion Level

Clustering, combined with some supplementary data analyses, can provide us with a useful overall view of differences among the congestion control algorithms. In our example, during TP1, algorithm 3 shows a distinctive behavior that appears tied to the level of congestion in the network. Further, under little or no congestion, the congestion control algorithms are largely indistinguishable. Unfortunately, cluster analysis does not identify the precise nature of the distinctions among the various alternative congestion control algorithms. For more insight, we need to apply a technique for the detailed analysis of individual responses. We next explain the technique we used to investigate each response.

6.3.2 Detail Analysis of Individual Responses

For each time period, we subjected each response to a statistical analysis for each of the 32 conditions simulated, and we then generated a corresponding plot displaying the relevant information. Such a plot shows, for each condition, which algorithm produced the largest difference (compared to the average for all algorithms) in the response variable. The plot also reports the results of a numerical test to determine whether the largest difference was statistically significant. In addition, the plot reports the absolute and relative magnitudes of the largest effect. We produced (45 x 3 =) 135 plots; each plot represents a single response for a single time period. The best approach to explaining the analysis is to discuss a sample plot, such as Fig. 6-9.

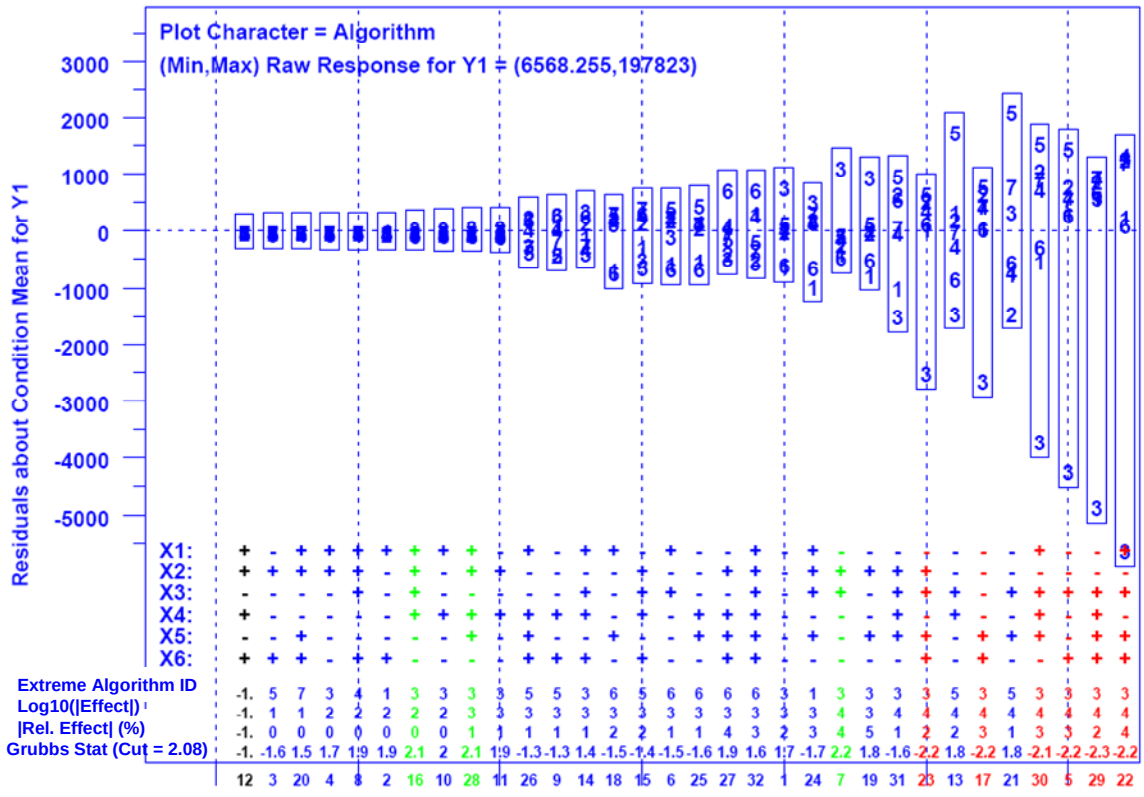

Figure 6-9. Sample Plot Analyzing the Influence of Condition and Congestion Control Algorithm on the Average Number of Active Flows (y1) – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

The x axis in Fig. 6-9 shows the 32 conditions. Here, conditions are sorted by increasing magnitude of the largest difference in the response variable produced by a congestion control algorithm. The upper left corner of the plot gives the minimum and maximum values for the raw responses when considering the data across all algorithms and conditions. The y axis gives the spreads of residuals about the mean. Here, each residual is computed by subtracting the mean response for all algorithms for a given condition from the response for a given algorithm and the same condition. For each condition, we plot a box within which we place algorithm identifiers (1-7). The location of each identifier indicates the distance of the response generated by that algorithm (i.e., the residual) from the mean response over all algorithms for the same condition. Here, the residuals range from zero (all algorithms in condition 12) to about negative 5500 (for algorithm 3 in condition 22).

Below each box we display vertically the settings (+/-) for each input factor (x1…x6) that generated the relevant condition. The remainder of the plot consists of four 32-column rows of quantitative information, where each column gives four statistics applicable to the algorithms and responses for the related condition. The first statistic identifies the extreme algorithm – that is the algorithm with the largest residual. The identifier is listed as -1 when the algorithms cannot be distinguished numerically. This arises for condition 12 in Fig. 6-9. Explicitly listing the extreme algorithm is helpful when the residuals are too close together to be visible in the box – for example in conditions 12 to 26.

The second statistic reports the absolute magnitude (log 10) associated with the maximum residual. The exponent of the absolute magnitude can be reported concisely on the plot at the cost of some numerical precision. The third statistic reports the relative effect as a percentage of the mean response. A domain analyst can consider both the absolute and relative differences when judging whether an effect is significant from an engineering view.

The fourth statistic reports *G*, which results from a Grubbs' test for outlying observations [91] associated with the extreme residual for each condition. The Grubbs' test computes *G* by dividing the largest residual by the sample standard deviation.

$$G \equiv \frac{\max\left(\left|Y_i - \mathrm{mean}(Y)\right|\right)}{s} \tag{3}$$

Assuming no significant differences among congestion control algorithms, we would expect measured residuals to be normally distributed. For this reason, residuals that deviate too far from the mean could be characterized as statistically significant outliers. For our plots we declare an outlier significant (5 %) when $G > 2.08$. The entire column (factors and statistics) is highlighted for conditions where the Grubbs' test identifies an outlier. Green identifies positive outliers (e.g., conditions 16, 28 and 7 in Fig. 6-9) and red identifies negative outliers (e.g., conditions 23, 17, 30, 5, 29 and 22 in Fig. 6-9). Columns are printed in black when no numerical difference could be detected among the responses (e.g., condition 12 in Fig. 6-9). The remaining columns are printed in blue.

What can we conclude from Fig. 6-9 alone? Not much. Algorithm 3 appears as a significant negative outlier under six conditions (all congested). This result could occur

by chance with a probability of about 0.17. Algorithm 3 also appears as a statistically significant positive outlier under three conditions (one congested). This result has about a 0.70 probability of occurring by chance. We selected Fig. 6-9 as an example because it illustrates most traits of such plots. We defer more interesting findings until we discuss pattern seeking in Sec. 6.3.4.2 and overall results in Sec. 6.4. Next, we discuss a technique to summarize our detailed analyses of individual responses. The summary considers all responses and all conditions for each time period.

6.3.3 Condition-Response Summary Analysis

The plots associated with detailed analysis of individual responses can be quite revealing, but they do not give a broad view across responses and conditions in a form similar to that provided by the dendrogram plots from cluster analysis. We can, however, extract information from analyzing individual responses and then present a condensed overview across responses and conditions. Fig. 6-10 shows such a condition-response summary plot for TP1.

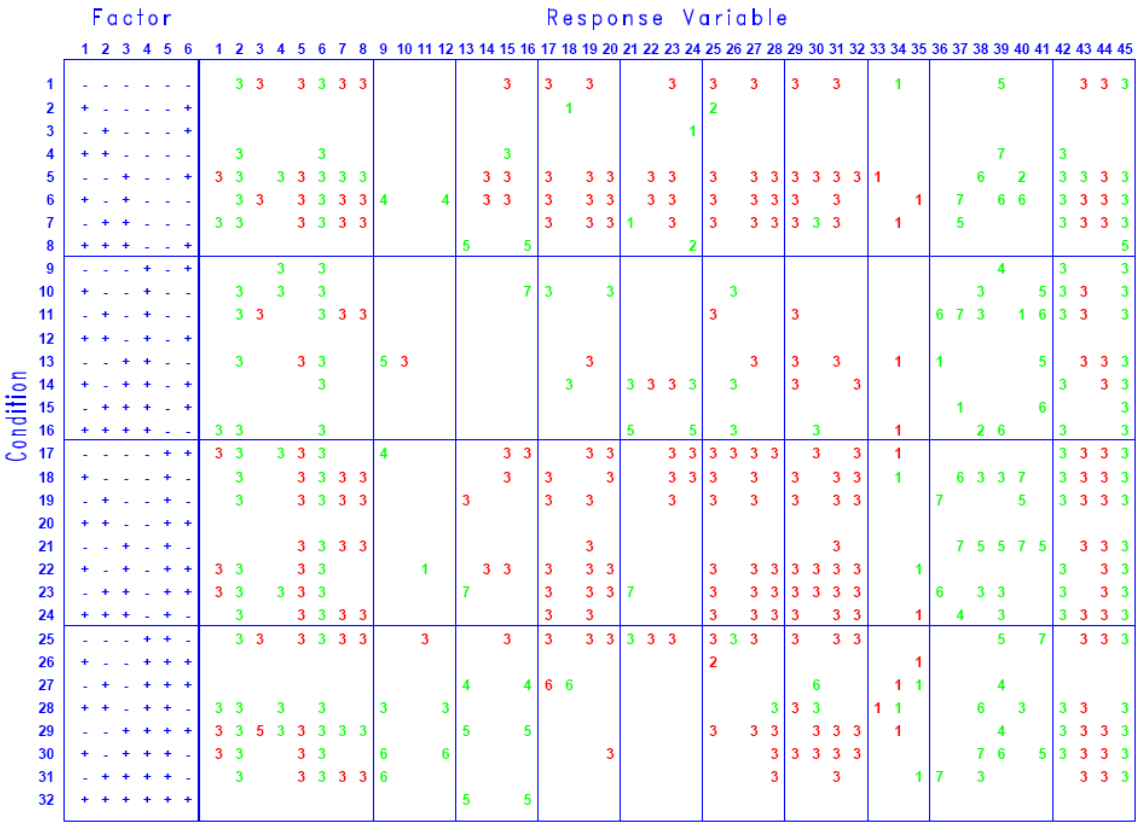

Figure 6-10. Summary of Statistically Significant Outliers in Time Period One – plot displays for each Factor Combination (row) vs. Response Variable (column) the Identifier of the Algorithm manifesting a Statistically Significant Outlier, where blank cells mean there was no significant outlier

Each row in Fig. 6-10 corresponds to a specific condition (identified on the left). The first six columns report settings (+/-) for the six input factors defining the condition. The remaining columns represent individual responses. Vertical blue lines group related responses. For example, responses 1 through 8 relate to macroscopic behavior, responses

9 through 12 relate to goodput on DD flows, responses 42 through 45 relate to distribution of flow states and so on. Cells, formed by condition-response intersections, contain an algorithm identifier when there is a statistically significant outlier – red denotes low outliers and green denotes high outliers.

A scan of Fig. 6-10 shows that algorithm 3 arises as a statistically significant outlier in many cells. The highest concentration of outliers appears for congested conditions; fewer outliers appear for less congested conditions. No algorithm appears as an outlier for condition (i.e., row) 12. These results agree with the cluster analysis (recall Fig. 6-4) for the same time period. Both analyses identify algorithm 3 as distinctive under congested conditions. Fig. 6-10 has the advantage of identifying precisely the particular responses for which algorithm 3 exhibits different behavior.

To focus analysis on the most significant behavioral differences, we can apply various filters when generating a condition-response summary plot. For example, Fig. 6-11 shows a summary plot reporting statistically significant outliers that also achieve a relative difference greater than 10%. The pattern of outliers is now sparser, so we can focus our analysis on responses y2 (congestion window increase rate), y6 (retransmission rate), y42 (average number of connecting flows), y44 (average number of active flows in normal congestion control mode) and y45 (average number of active flows in alternate congestion control mode). The responses measuring buffer usage (y36 – y41) exhibit outliers but there is no evident pattern.

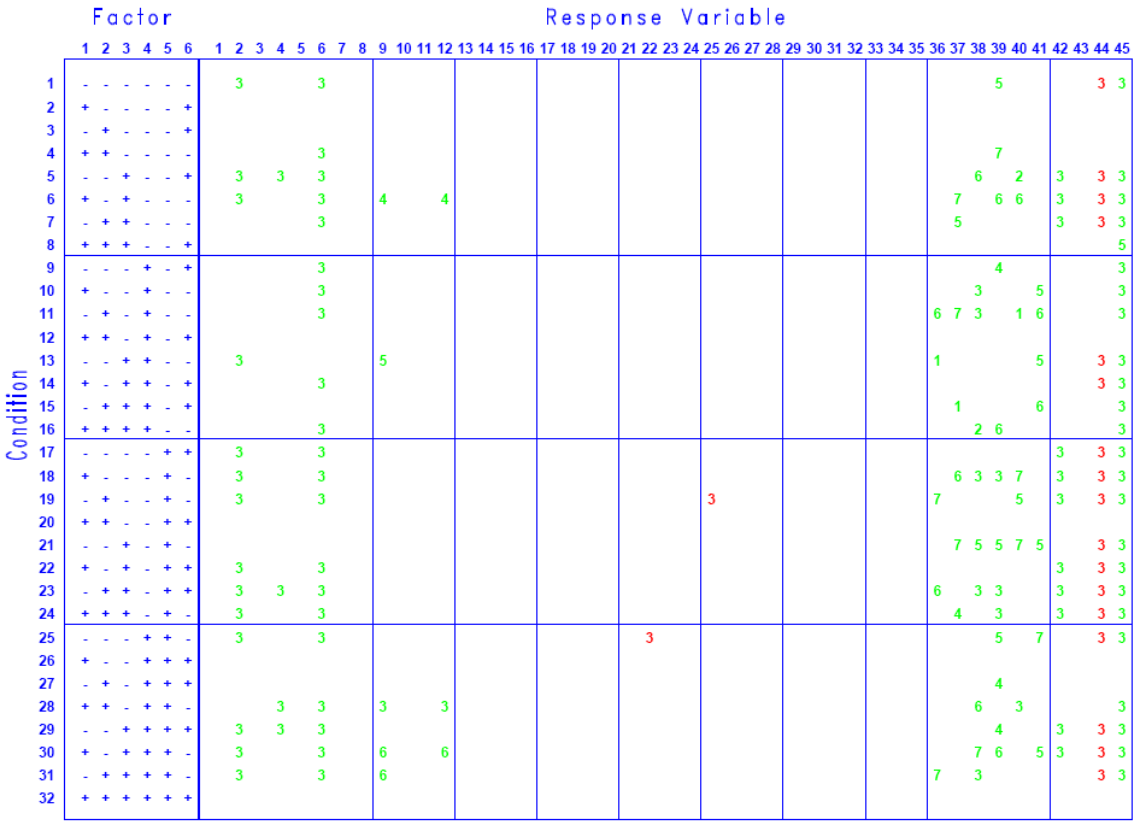

Figure 6-11. Filtered Summary Plot for Time Period One Identifying Statistically Significant Outliers with Associated Relative Effect > 10%

We can exclude y44 and y45 from further consideration because algorithm 3 (FAST) never operates in normal congestion control mode. This means that we should expect algorithm 3 to be an outlier exhibiting a large effect for responses y44 and y45. This is certainly the case in all the analyses we conducted. With this knowledge, we completed a revised cluster analysis with responses y44 and y45 excluded. The revised clustering results (reported in Sec. 6.4) continue to identify algorithm 3 as distinctive.

Fig. 6-11 suggests that algorithm 3 is most different with respect to response y6 – retransmission rate. Here, algorithm 3 produces retransmission rates more than 10 % higher than the other algorithms in 21 of the 32 conditions. In 12 conditions the retransmission rate for algorithm 3 is more than 30 % higher – more than 50 % higher in five conditions. Clearly, this is a significant finding, which we discuss more fully in later sections.

Fig. 6-11 also shows that for 14 conditions algorithm 3 (FAST) produces more than a 10 % higher rate of window increase than the other algorithms. All 14 conditions are among the most congested. Recall from Chapter 5 that FAST aims to provide a stable congestion window that reaches equilibrium, changing very little over time. The simulations in Chapter 5 also showed that when FAST had insufficient buffers a rapid oscillating behavior ensued where the congestion window was cut in half on a loss and then quickly increased up to another loss and so on. Under these rapid oscillations, FAST would tend to increase congestion windows very frequently. Thus, under FAST, the larger the retransmission rate, the higher the rate of window increases.

What about y42 (average number of connecting flows)? A high retransmission rate arises from a high loss rate. To establish flows, a source and receiver must exchange SYN and SYN+ACK packets. Since these packets are also subject to being lost, we expect that a high loss rate can impede connection establishment. This means that on average more SYNs must be sent to connect a flow. Thus, given a higher retransmission rate for algorithm 3, we should expect more flows to be pending in the connecting state.

This discussion illustrates that condition-response summary plots can be quite powerful – allowing an analyst to identify key differences separating algorithms. In Sec. 6.4 we report summary plots for all three time periods, as well as for the aggregate responses. As we will demonstrate, the summary plots impart substantial insight regarding system behavior.

6.3.4 Data Exploration

In previous sections we introduced the main techniques we used to analyze system behavior. We augmented these analysis techniques with some exploratory approaches that allowed us to investigate specific questions. In this section we briefly describe and illustrate selected augmentations.

6.3.4.1 Extrapolating from Time Series. MesoNet samples responses at each measurement interval and produces related time series. We generate our summary responses by averaging time series of interest over particular intervals. As discussed in Sec. 6.3.1, an analyst may examine raw time series as necessary to gain additional insight. Here, we give an example that illustrates pitfalls that may arise when focusing on time series for only a few selected conditions.

Fig. 6-12 plots seven time series (one for each congestion control algorithm) for condition 4, a lightly congested condition. Each time series reports the average goodput for DD flows (y9) over the final three time periods (15 minutes total) of the experiment scenario. The plot shows the general effect of the scenario on DD flows. During the first time period (3000-4500) average per-flow goodputs fluctuate in the neighborhood of 500 pps. Jumbo flows commence at time 4500, which leads to a rapid increase in average goodput up to around 10^4 pps. As additional jumbo flows arrive, average goodput falls as bandwidth must be shared among more flows. New jumbo flows cease to arrive starting at time 6000, which enables average goodput to increase as residual jumbo flows are cleared. As the mix of flows moves away from jumbo flows and back to normal Web traffic, average goodput trails off. Had the scenario continued, all residual jumbo flows would eventually clear the system and average goodput would return to levels seen in the first time period. This general behavior is representative of the time varying scenario across all conditions. Fig. 6-13 plots seven time series for the number of active DD flows (y10) over the same time periods and under the same condition.

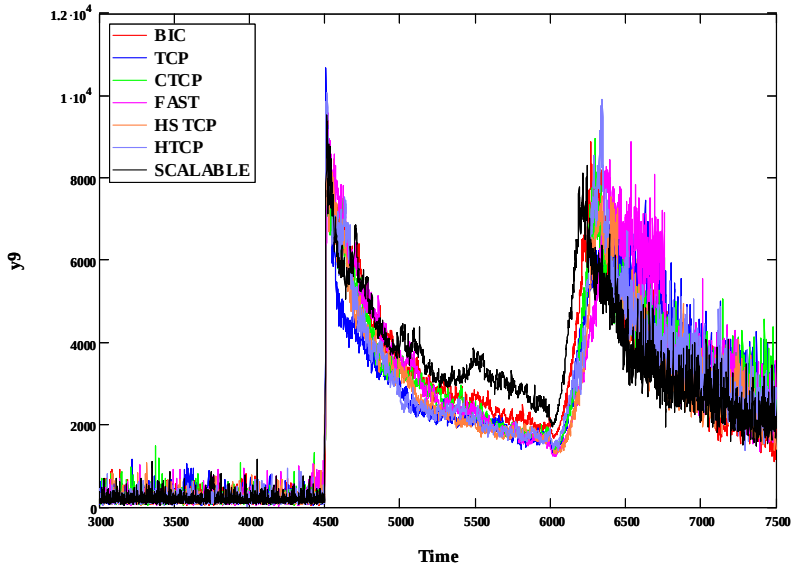

Figure 6-12. Average Per-Flow Goodput on DD Flows (y9) for Seven Congestion Control Algorithms under Condition 4 over Three Time Periods – x axis gives time in 200 ms increments and y axis gives goodput in packets per second

Fig. 6-12 indicates that Scalable TCP (black curve) provided higher average goodput during the period of jumbo file transfers. Recall that in Chapter 5 we found that under a restricted topology with few flows, Scalable TCP tended to provide unfair allocation of bandwidth. Does relative unfairness relate to the behavior shown in Fig. 6-12? The current simulation scenario was set up to ensure that a concentration of jumbo files would be transferred on DD flows between times 4500 and 6000. Yet, Fig. 6-13 reveals that Scalable TCP (black curve) has the fewest number of active DD flows in that time period; BIC (red curve) has second fewest. Given a finite (bottleneck) capacity to deliver packets, each flow will naturally receive higher average goodput when the bottleneck is shared by fewer flows. Fig. 6-14 shows that a bottleneck capacity exists, as

the total rate of packets delivered on DD flows (y12) during the second time period reaches a level of just under 2 million pps for each of the congestion control algorithms.

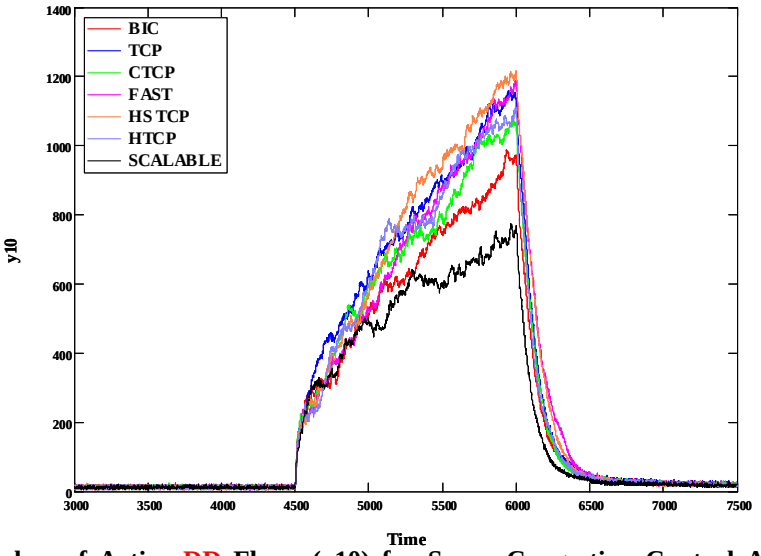

Figure 6-13. Number of Active DD Flows (y10) for Seven Congestion Control Algorithms under Condition 4 over Three Time Periods – x axis gives time in 200 ms increments and y axis number of active flows

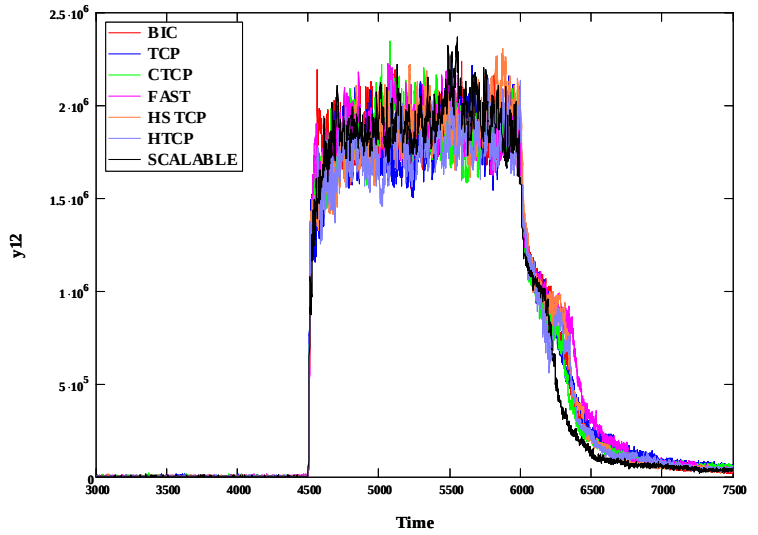

Figure 6-14. Aggregate Packet Delivery Rate DD Flows (y12) for Seven Congestion Control Algorithms under Condition 4 over Three Time Periods – x axis gives time in 200 ms increments and y axis packet delivery rate in packets per second

How can we explain the fact that fewer jumbo flows are active simultaneously under condition 4 in TP2 for Scalable TCP? The answer can be found by examining the completion rate for DD flows (y11) during TP2, as shown in Table 6-31. Scalable TCP completes slightly more (.3 to .4) DD flows per measurement interval than other congestion control algorithms. Remember that the measurement interval is only 200 ms in duration. Considered over the entire 5 minutes (1500 measurement intervals) comprising TP2, Table 6-31 shows that Scalable TCP completes 500 to 600 more DD

flows. More flows completed per unit time leads to fewer active flows, which yields higher goodput per flow.

Table 6-31. Flows Completed per 200 ms interval and Total Completions for DD Flows in Time Period Two under Condition 4

Algorithm	DD Flow Completion Rate	DD Flows Completed in Time Period 2
BIC	7.74	11.610×10^3
CTCP	7.60	11.401×10^3
FAST	7.57	11.353×10^3
HSTCP	7.42	11.133×10^3
HTCP	7.54	11.307×10^3
SCALABLE	7.88	11.814×10^3
TCP	7.53	11.296×10^3

Does this behavior repeat across a wide range of conditions? In selected uncongested conditions (such as 8 and 12) Scalable TCP provides the worst goodput for DD flows during TP2. An overall examination of y9 across all conditions (see Fig. 6-15) reveals no particular pattern, which illustrates why we must rely on comprehensive results and not focus in detail on particular conditions to the exclusion of others.

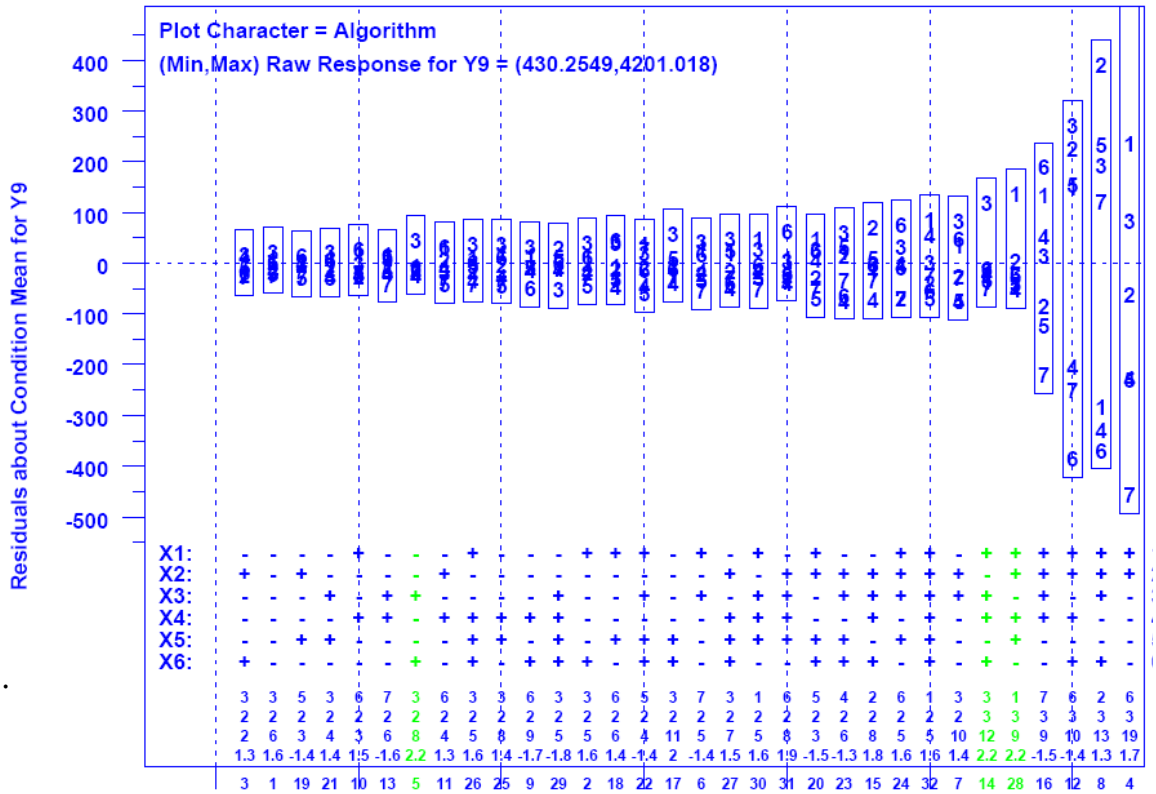

Figure 6-15. Analyzing the Influence of Condition and Congestion Control Algorithm on the Average Goodput (pps) for DD Flows (y9) during Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

In summary, examining time series for individual conditions can provide interesting and useful information regarding the movement of system responses over time. Unfortunately, one must examine a response across all conditions in order to reveal general trends and patterns. Comprehensive examination across responses and conditions can best be achieved through cluster analysis and condition-response summary plots. Examination of individual time series must be used for focused purposes, such as causality analysis, and not to draw general inferences.

6.3.4.2 Seeking Patterns. Above, we characterized Fig. 6-15 as revealing no particular pattern. This implies that we will be seeking patterns not only in the dendrogram and condition-response summary plots but also in plots giving detailed analyses of individual responses. Such patterns correspond to columns in a condition-response summary plot where identical algorithms are reported as statistically significant outliers across a substantial number of conditions. For example, Fig. 6-16 illustrates a pattern in a detailed analysis plot for congestion window size (y4) during TP3. Comparing Fig. 6-15 and Fig. 6-16 illustrates the difference between a non-pattern and a pattern. The pattern in Fig. 6-16 reports that during TP3 algorithm 2 (CTCP) yields a congestion window significantly larger than the other algorithms. This result occurs in 30 of the 32 conditions (in conditions 3 and 32 algorithm 3 is the outlier) and is statistically significant (5 %) in 28 of the 30 conditions, and nearly significant in the other two (i.e., in conditions 19 and 29 the Grubbs' test statistic is 2, which is just below the 2.08 cutoff for 5 % significance).

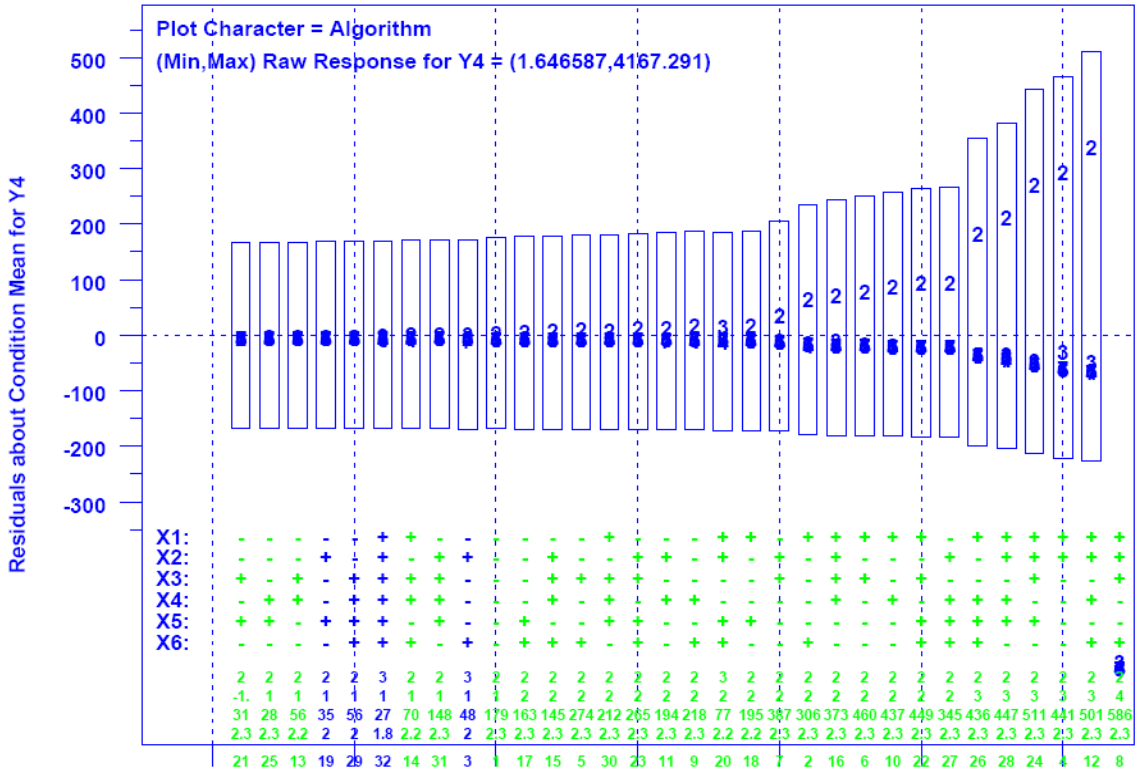

Figure 6-16. Analyzing the Influence of Condition and Congestion Control Algorithm on Congestion Window Size (y4) in packets during Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

6.3.4.3 Investigating Data Subsets. In cases where a summary plot reveals one particular algorithm as distinctive, an analyst may naturally wonder whether the distinction might be sufficient to mask more subtle distinctions among the remaining algorithms. To investigate such questions, one can exclude response data for the distinctive algorithm and then reconsider the analysis on the remaining subset of response data. For example, Fig. 6-17 gives dendrograms resulting from a cluster analysis for TP1 when response data for algorithm 3 is omitted. The resulting plot reveals that the responses are very similar across the remaining algorithms in about half the conditions. For some conditions there appears to be a slight tendency for algorithms 1 (BIC) and 6 (Scalable TCP) to cluster together, while algorithm 5 (HTCP) is somewhat distinctive under four conditions. Overall, the cluster analysis for TP1 with algorithm 3 excluded shows the behavior among the remaining algorithms to be largely indistinguishable. There appears some tendency for algorithms 1 and 6 to exhibit slightly similar behaviors somewhat different from other algorithms. A condition-response summary plot for the same subset of data identifies few statistically significant outliers.

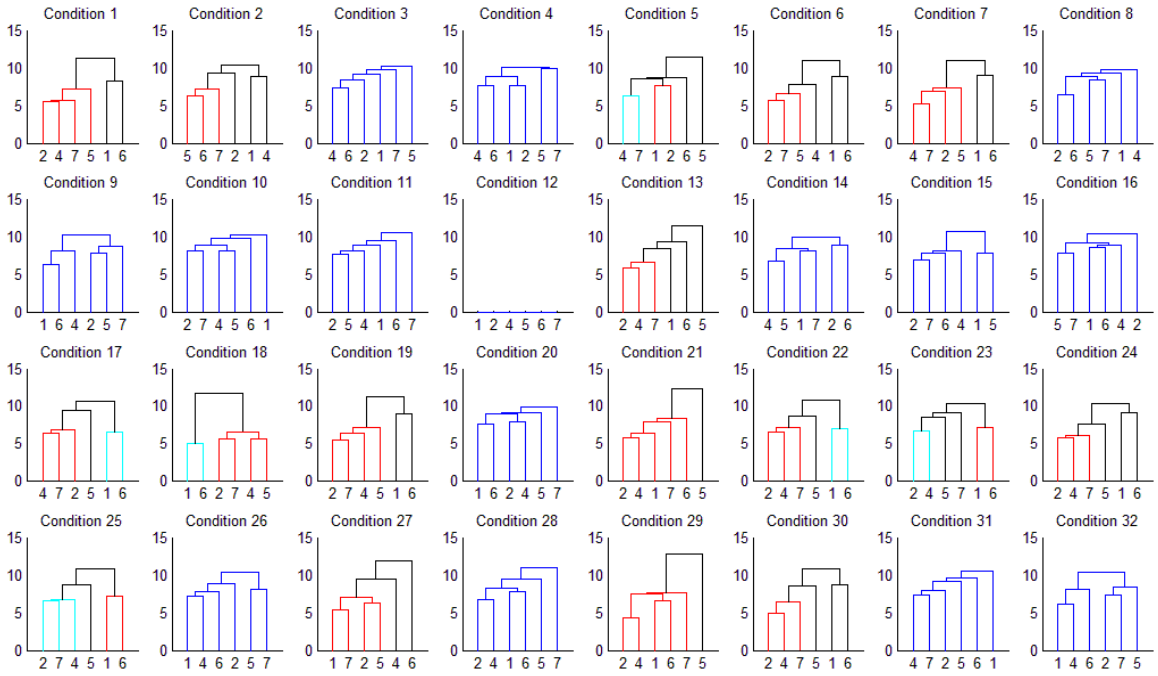

Figure 6-17. Cluster Analysis Using Data from Time Period One – Algorithm 3 Excluded

6.3.4.4 Interactive Animation. MesoNet produces a large amount of data – simulation of a congestion control algorithm under one condition can produce about 165 Mbytes of measurements. Much of the data relates to temporal behavior in individual routers in the network topology. Such data naturally lends itself to animation within a layout of the network topology. To accommodate such animation, as well as to support abstract analysis of multidimensional data, colleagues produced DiVisa [86], an interactive system for multidimensional data visualization. DiVisa, freely available for public use, requires only access to a Java™ run-time environment, so DiVisa is portable to a range of operating systems.

Fig. 6-18 depicts a sample screenshot where we used DiVisa to monitor packet losses throughout the network topology for algorithm 1 (BIC) under condition 10. The screenshot shows three main panels: a (leftmost) visualization-control panel and two plots. The control panel permits a user to define plot characteristics. In this case, we assigned the leftmost plot panel to hold the network topology (routers and links only), while the rightmost plot panel graphs packet losses over time. Further, in the topology panel we assign color to represent the rate of packet losses – from orange for minimal losses to red for high losses. The particular screenshot shows one frame from an animation of the evolution of packet losses – the animation has reached time 5510, which is within TP2. At that time, only two routers in the topology show any appreciable losses: access router I0a (yellow) and access router K0a (blue). We can select specific routers in the topology and the related curve in the time plot will be emphasized. We can also interactively explore other router characteristics, such as utilization and buffer saturation. DiVisa animations helped us discover that backbone routers could be overrun under some conditions in TP2. Using this information, we increased the simulated speed of our backbone routers. DiVisa animations also helped us to determine that access router K0a was the most heavily utilized of the access routers during TP2. In summary, availability of a data exploration tool and animator, such as DiVisa, can help an analyst gain global views of spatiotemporal patterns in a simulated system. Of course, one must remember that looking at animations of individual time series does not provide sufficient information to discern significant overall patterns across conditions and responses

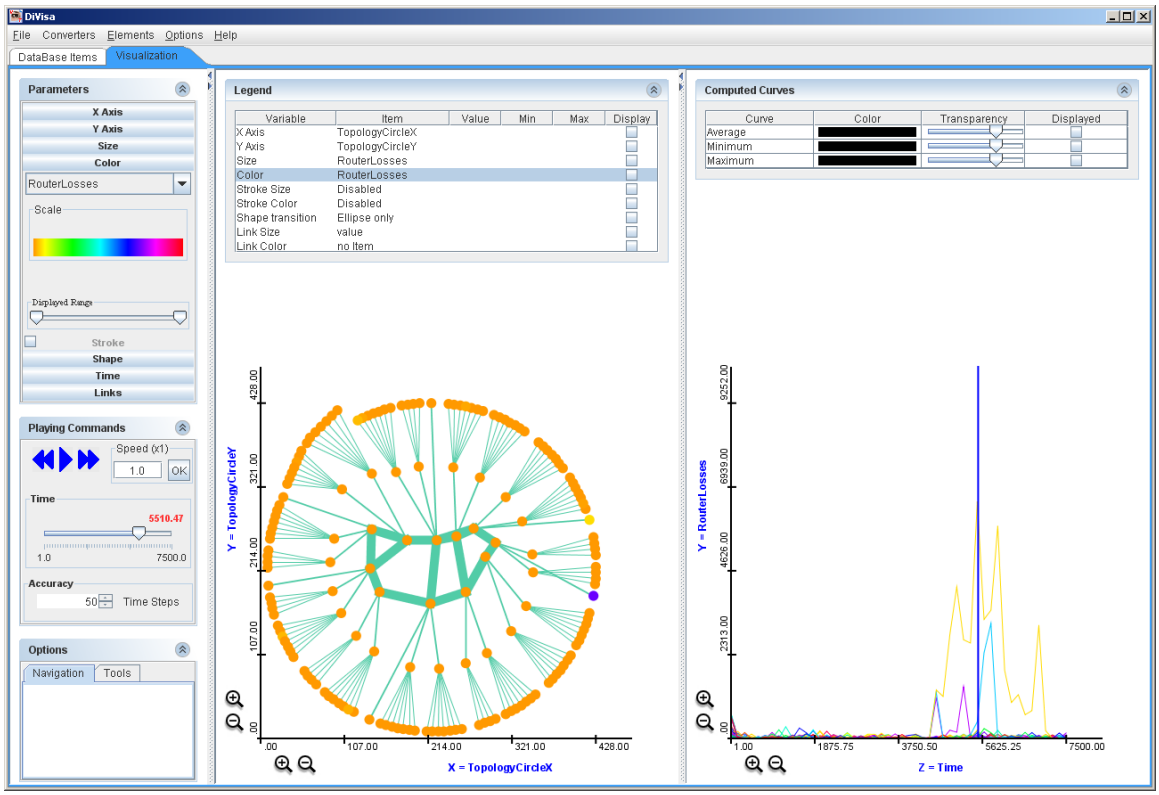

Figure 6-18. Screenshot from DiVisa Animation of the Temporal Evolution of a MesoNet Simulation

6.4 Results

In this section, we report salient results from our analysis of summarized response data (described in Sec. 6.2.2). As necessary, we provide brief commentaries to explain the results presented. We give results in four segments: one for each of the three 5-minute time periods and one for response data aggregated over the entire 25-minute scenario. We follow a similar plan for each segment: (1) present results from cluster analysis, (2) present results from condition-response summaries, (3) present detailed analysis of significant responses and (4) give a summary of the results for the segment. We defer drawing inferences from the results until Sec. 6.5, where we report our findings.

6.4.1 Time Period One (TP1)

Recall that TP1 comprises a five-minute period where three long-lived flows commence within an overall background of normal Web traffic, which includes downloading Web pages, and occasionally documents. As for any time period, we consider seven congestion control algorithms under a range of 32 conditions, where half the conditions can be considered uncongested and half congested.

6.4.1.1 Cluster Analysis for TP1. We present two dendogram plots for TP1. Fig. 6-19 gives the cluster analysis for all seven congestion control algorithms. We annotate the individual dendograms with a 3 when algorithm 3 appears distinctive. Fig. 6-20 gives the cluster analysis after omitting response data for algorithm 3.

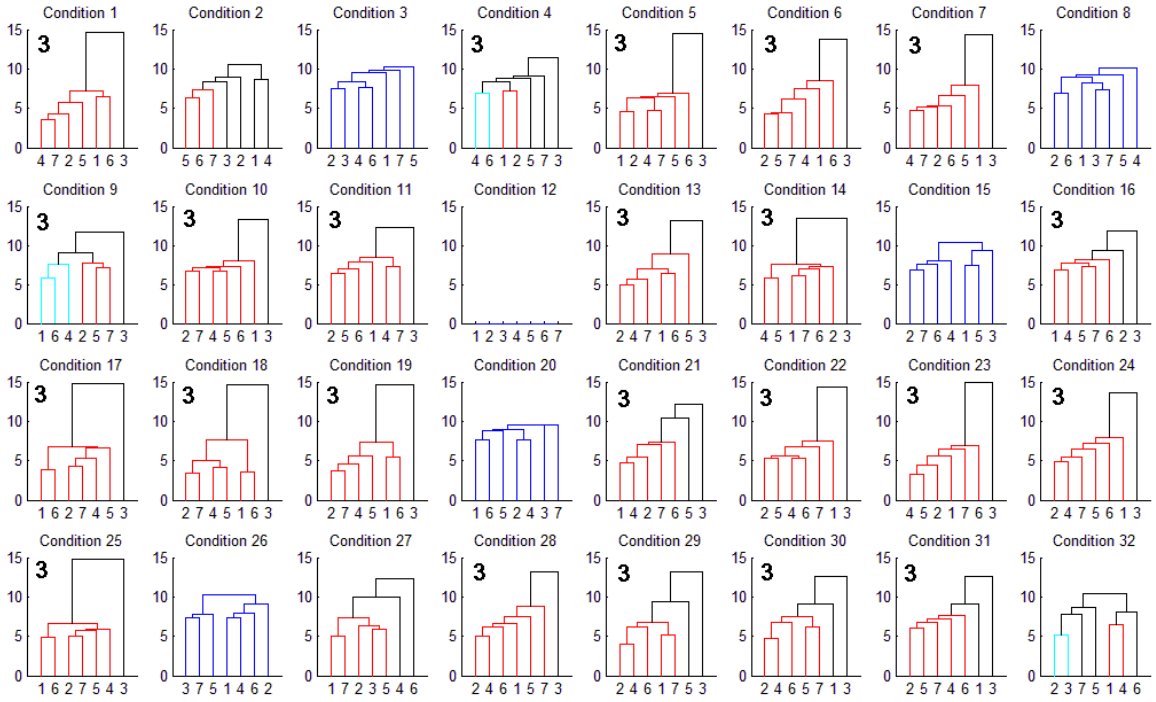

Figure 6-19. Clustering for Time Period One – Annotated to Identify Distinctive Algorithm 3

6.4.1.2 Condition-Response Summary for TP1. Fig. 6-21 gives the condition-response summary for TP1. Fig. 6-22 shows the same summary after applying a filter showing only statistically significant outliers for which the relative effect exceeds 10 %.

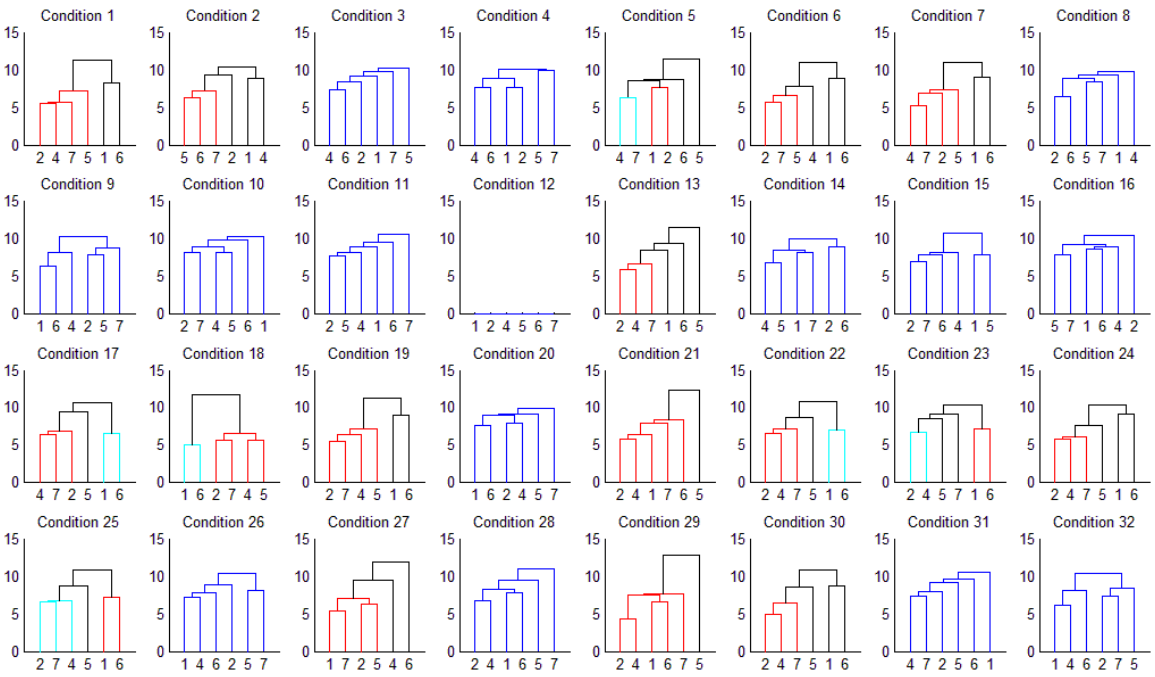

Figure 6-20. Clustering for Time Period One – Algorithm 3 Omitted

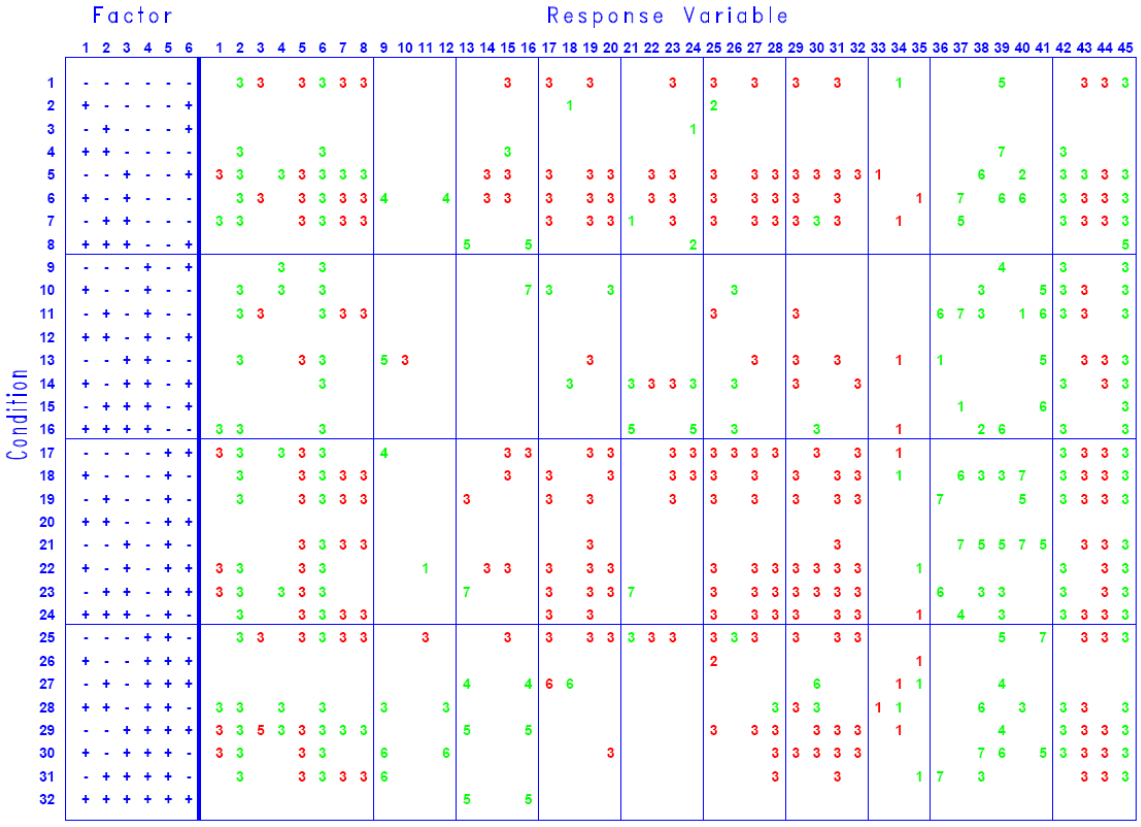

Figure 6-21. Condition-Response Summary for Time Period One – plot displays for each Factor Combination (row) vs. Response Variable (column) the Identifier of the Algorithm manifesting a Statistically Significant Outlier

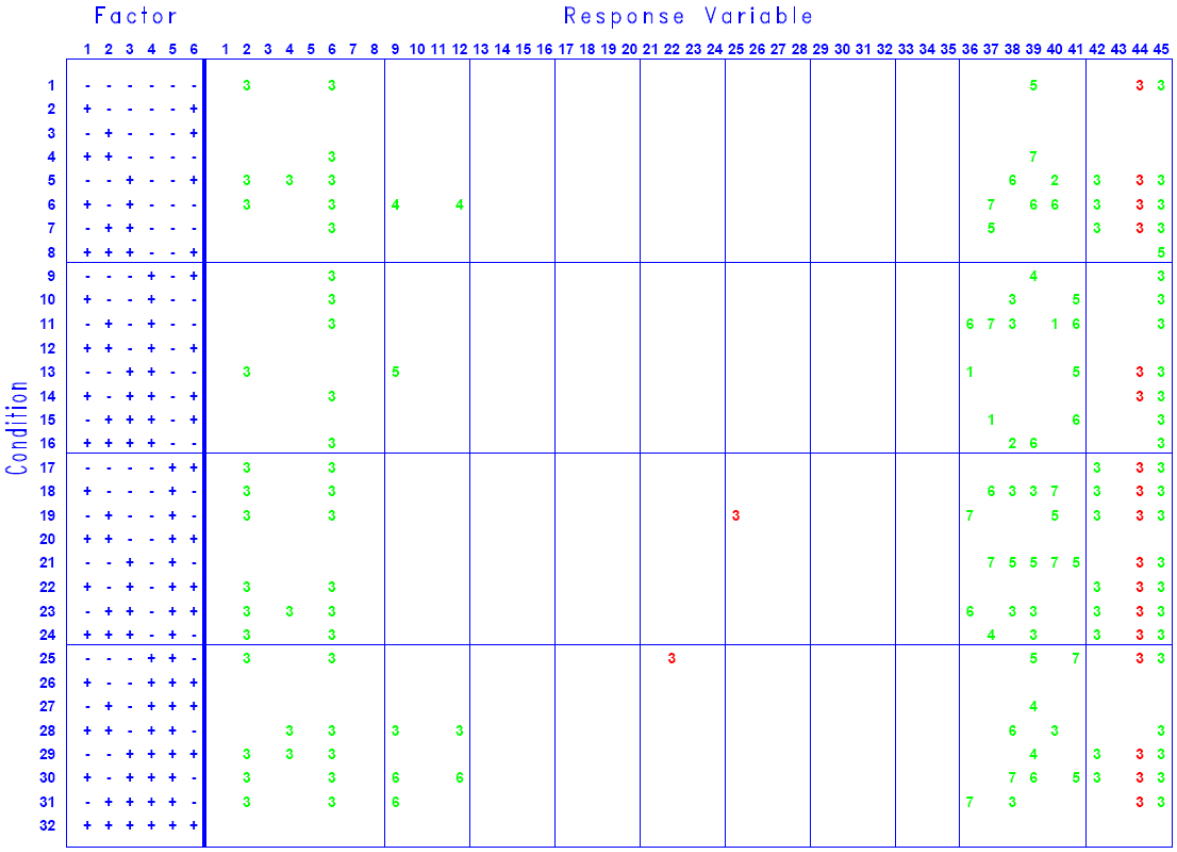

Figure 6-22. Filtered Summary Plot for Time Period One Identifying Statistically Significant Outliers with Associated Relative Effect > 10%

6.4.1.3 Analysis of Significant Responses for TP1. Based on Figs. 6-21 and 6-22 we selected several responses for more detailed analysis. Specifically, in Figs. 6-23 to 6-27, we report analyses for congestion window increase rate (y2), flow completion rate (y5), retransmission rate (y6), completion rate for NN flows (y31), and average number of connecting flows (y42). We omitted y44 and y45 because (as we explained earlier) they provide little insight into differences in behavior among the congestion control algorithms. We selected y5 and y31 based on Fig. 6-21 even though they did not pass the 10 % filter required for reporting in Fig. 6-22. We made these additional selections because the absolute magnitude of each effect within an individual measurement interval appears large enough to influence system behavior when accumulated over time. While we could have chosen completion rates for other flow classes (e.g., y19 or y25), NN flows make up the largest proportion of all flows active at any given time, so the significance of this flow class appears to be highest.

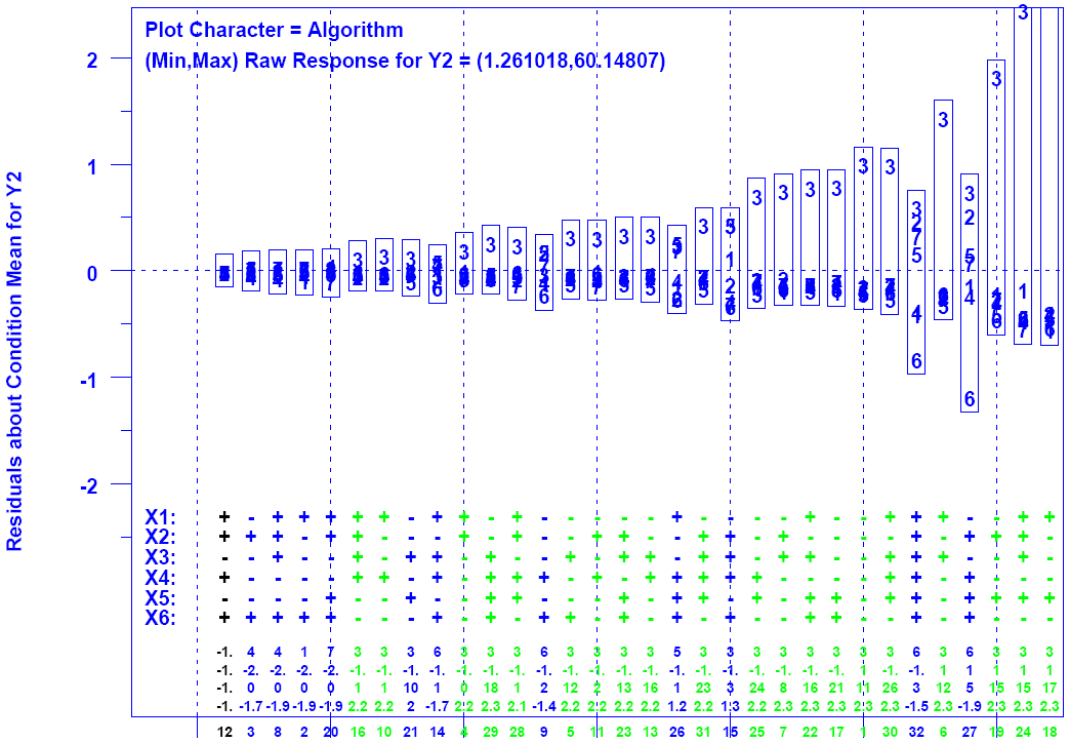

Figure 6-23. Detailed Analysis for Congestion Window Increase Rate Per Flow (increases per 200 ms) in Time Period One – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

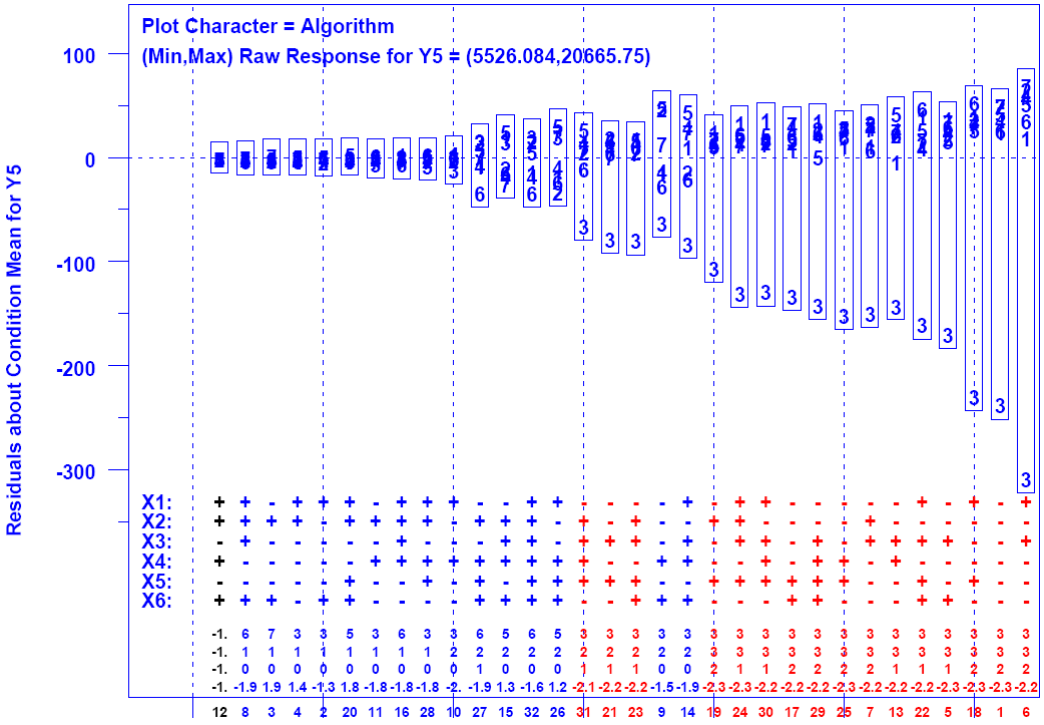

Figure 6-24. Detailed Analysis for Flow Completion Rate (flows per 200 ms) in Time Period One – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

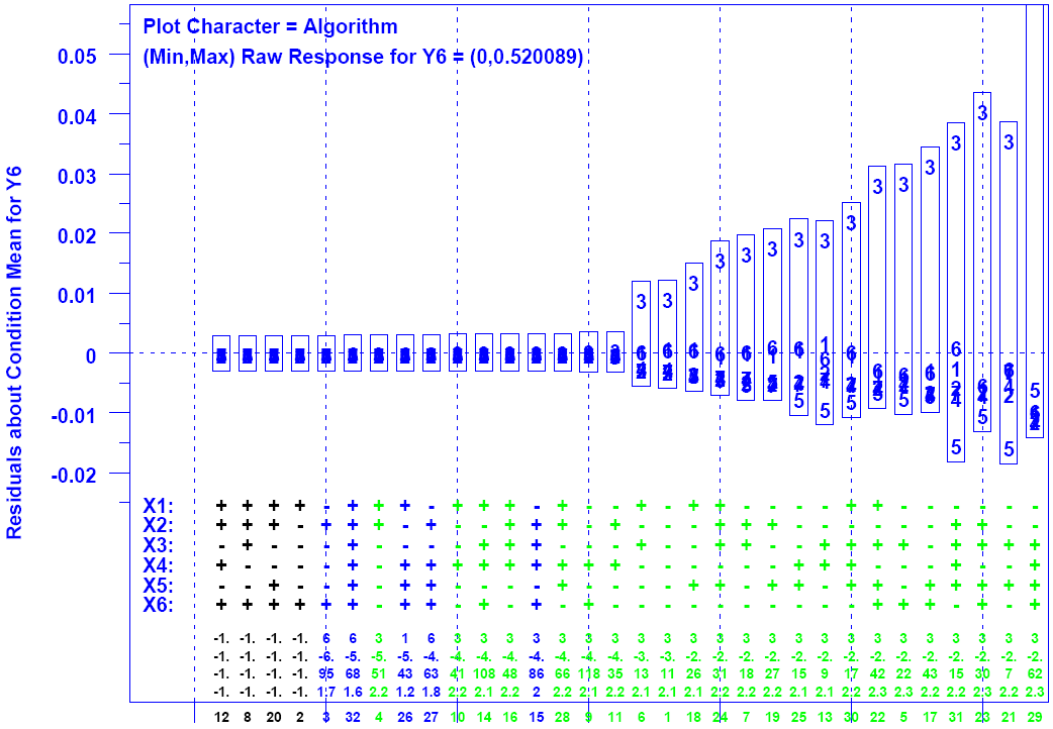

Figure 6-25. Detailed Analysis for Retransmission Rate (proportion of packets retransmitted) in Time Period One – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

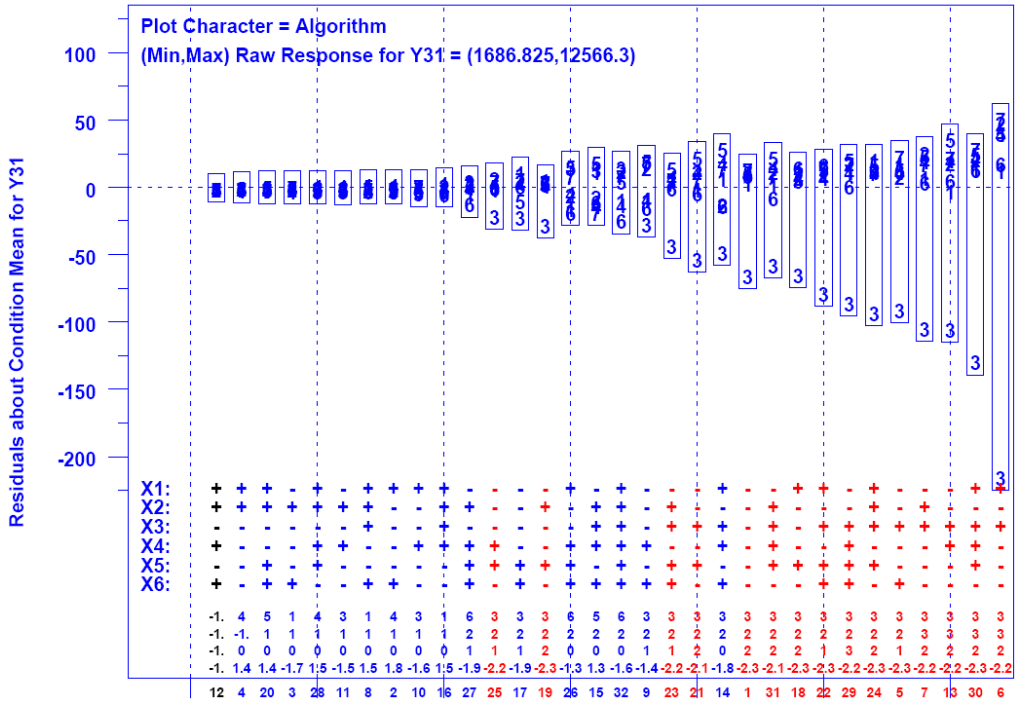

Figure 6-26. Detailed Analysis for NN Flow Completion Rate (flows per 200 ms) in Time Period One – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

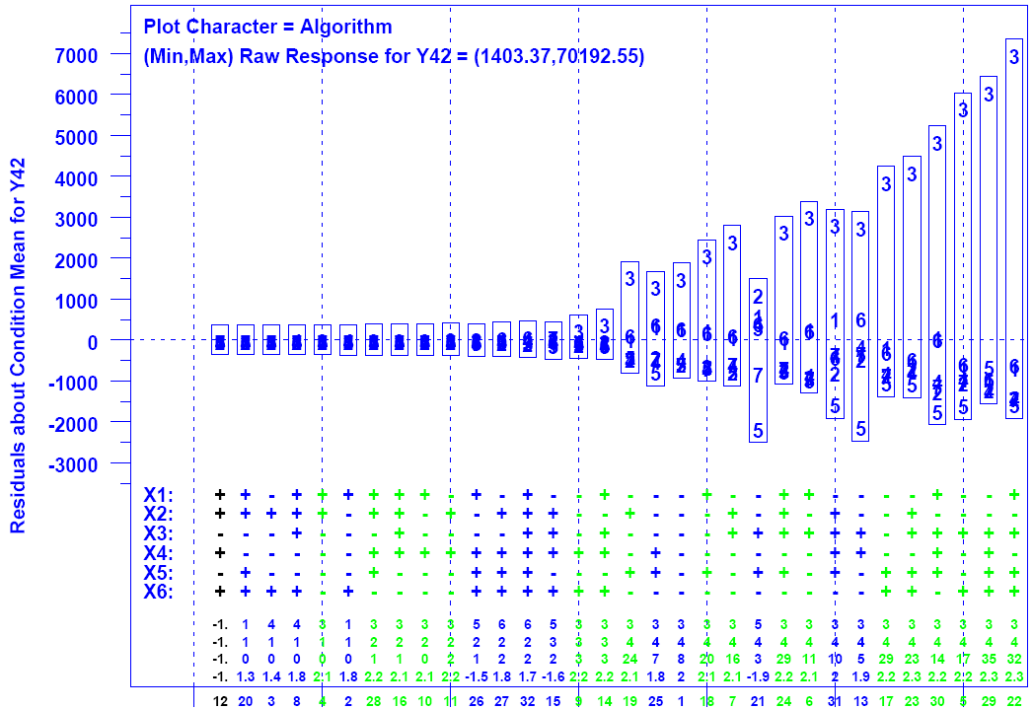

Figure 6-27. Detailed Analysis for Number of Connecting Flows in Time Period One – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

6.4.1.4 Summary of Results for TP1. Given normal Web traffic, FAST (algorithm 3) exhibits distinctive behavior, which appears to grow more distinctive with increasing congestion. The other algorithms behave quite similarly under most conditions, though BIC (algorithm 1) and Scalable TCP (algorithm 6) appear to cluster together under some conditions. When faced with congestion, FAST exacerbates the situation, as shown by the higher rate of increase in congestion windows, which leads to more packet losses and then to a higher rate of retransmissions. Increased losses under FAST also appear to increase the difficulty for establishing flows because more SYN and SYN+ACK packets are lost – as a result, on average more flows are pending in the connecting state. Increased retransmissions also cause flows to send more packets in order to ensure all data is successfully received. This means that flows take longer to finish, as shown by the lower completion rate for flows in general and for NN flows in particular.

6.4.2 Time Period Two (TP2)

During TP2 DD flows become jumbo file transfers, which lead to increased congestion within directly connected routers and also increases packet load on the network backbone. The remaining flow classes continue to generate normal Web traffic during TP2, but the net effect of adding the jumbo flows is to increase network-wide congestion.

6.4.2.1 Cluster Analysis for TP2. Fig. 6-28 shows an annotated set of 32 dendrograms for TP2. Since the level of congestion has increased throughout the network and algorithm 3 appears sensitive to congestion, one might expect the behavior of algorithm 3 to become more distinctive. Note that algorithm 3 now appears as distinctive in 28 of the conditions

– versus only 23 conditions in TP1. Fig. 6-29 gives dendrograms for TP2 but with the data for algorithm 3 omitted – none of the other algorithms stand out.

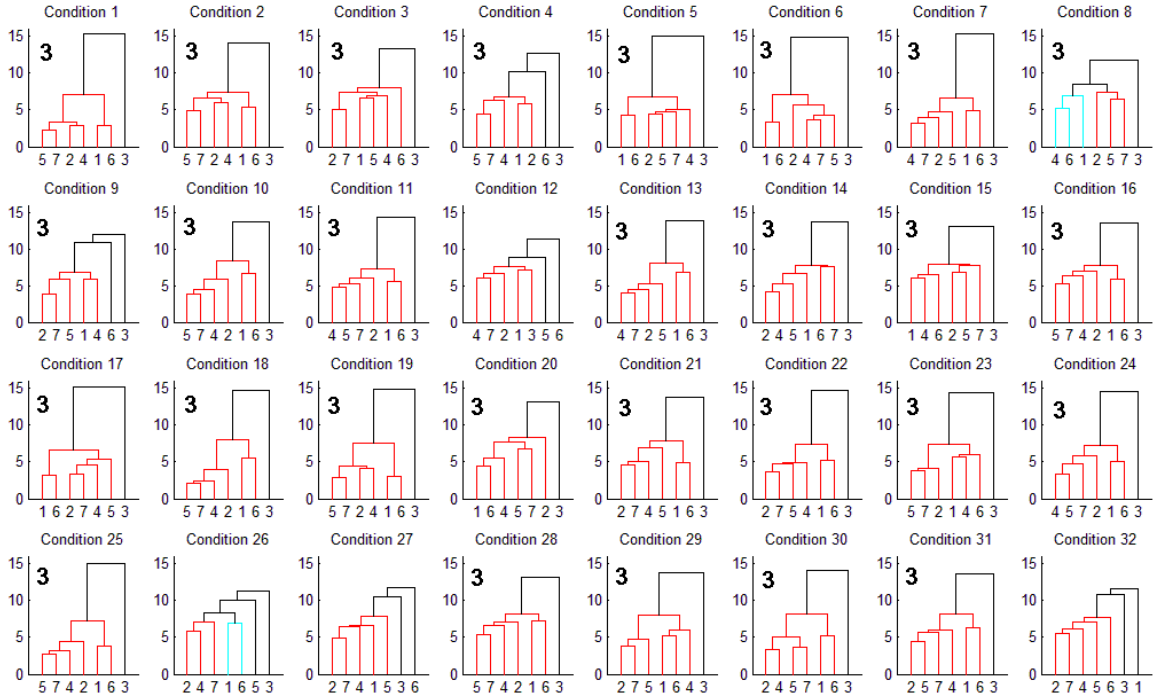

Figure 6-28. Clustering for Time Period Two – Annotated to Identify Distinctive Algorithm 3

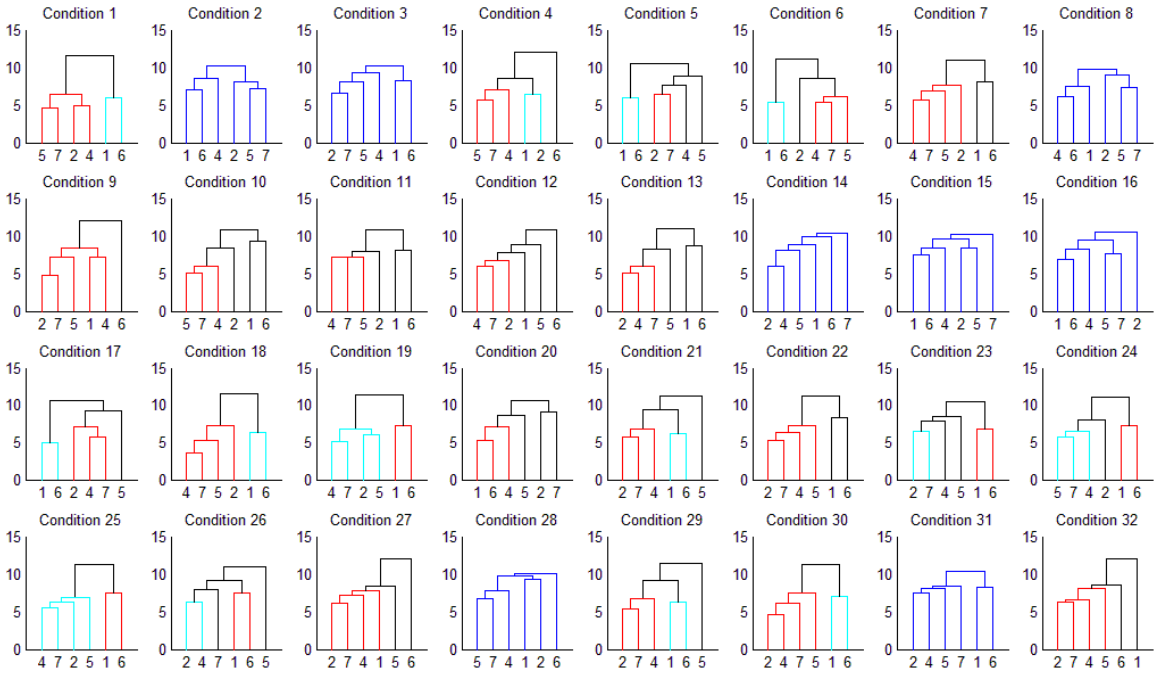

Figure 6-29. Clustering for Time Period Two – Algorithm 3 Omitted

6.4.2.2 Condition-Response Summary for TP2. Fig. 6-30 gives the condition-response summary for TP2. Fig. 6-31 shows the same summary after applying a filter showing

only statistically significant outliers for which the relative effect exceeds 30 %. Algorithm 3 stands out in both figures.

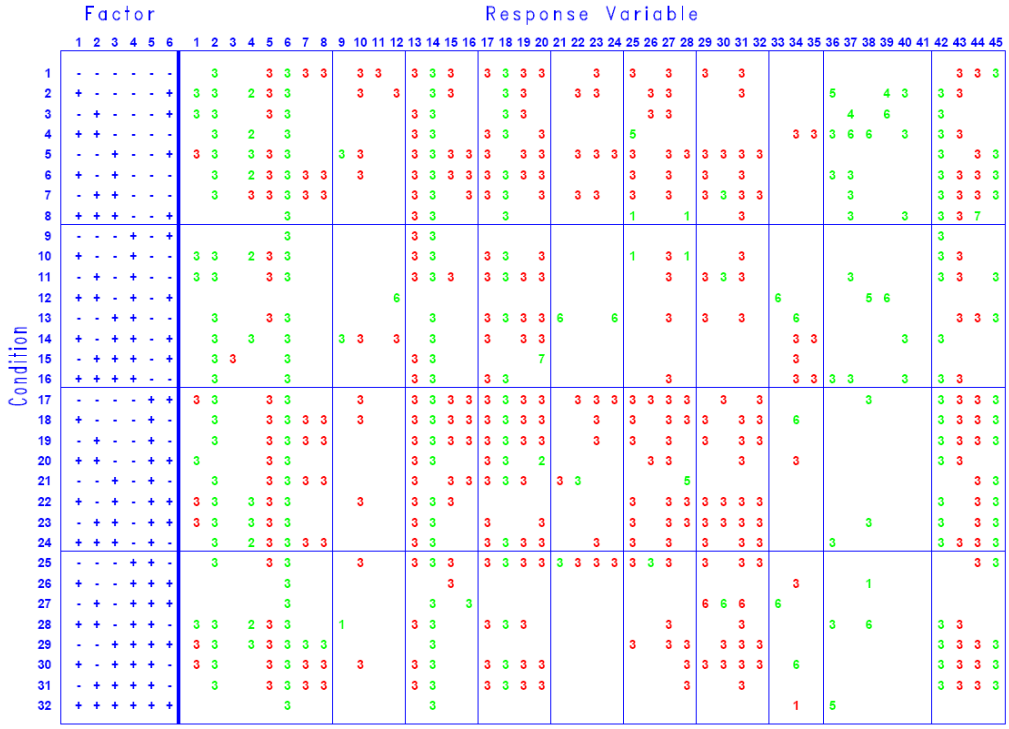

Figure 6-30. Condition-Response Summary for Time Period Two – plot displays for each Factor Combination (row) vs. Response Variable (column) the Identifier of the Algorithm manifesting a Statistically Significant Outlier

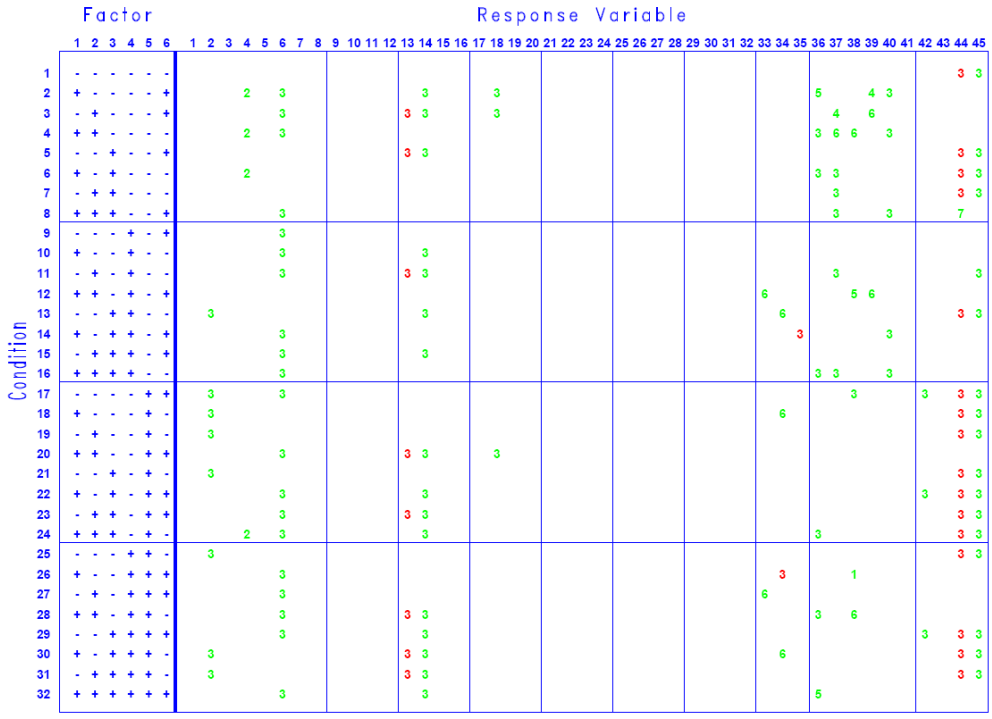

Figure 6-31. Filtered Summary Plot for Time Period Two Identifying Statistically Significant Outliers with Associated Relative Effect > 30%

6.4.2.3 Analysis of Significant Responses for TP2. Based on Figs. 6-30 and 6-31 we selected several responses for more detailed analysis. Specifically, in Figs. 6-32 to 6-37, we report analyses for congestion window increase rate (y2), flow completion rate (y5), retransmission rate (y6), average goodput for DF flows (y13), average number of active DF flows (y14), and average number of connecting flows (y42). In Figs. 6-38 and 6-39 we show the analyses for average goodput on the long (L1) and medium (L2) distance long-lived flows. We selected y5 based on Fig. 6-30 even though it did not pass the 30 % filter required for reporting in Fig. 6-31. We made this additional selection because the absolute magnitude of the effect within an individual measurement interval appears large enough to influence system behavior when accumulated over time.

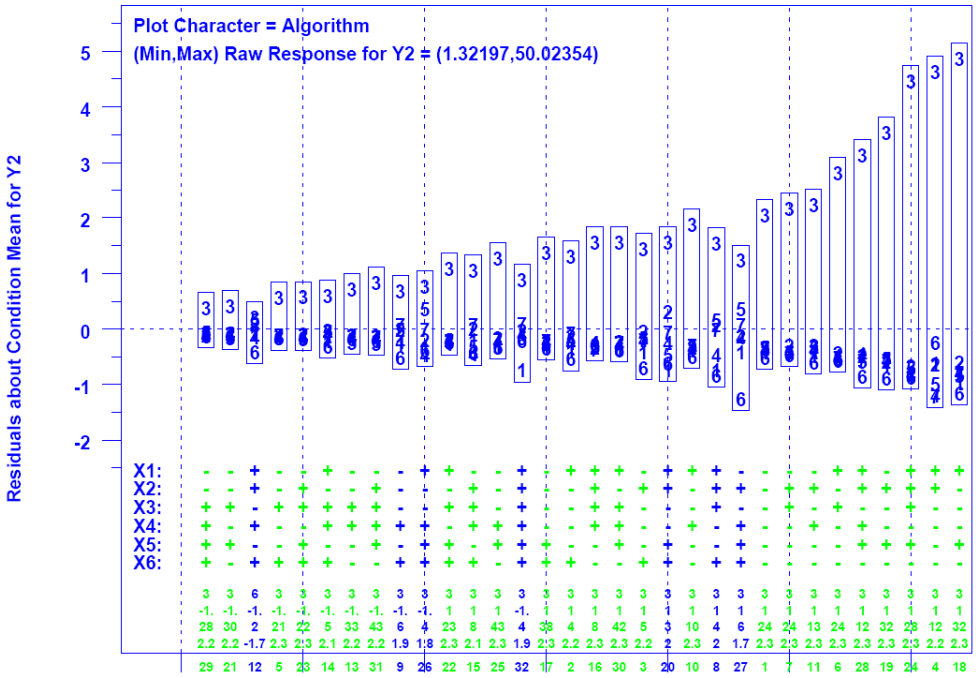

Figure 6-32. Detailed Analysis for Congestion Window Increase Rate (increase per 200 ms) in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

6.4.2.4 Summary of Results for TP2. FAST (algorithm 3) exhibits most of the same distinctive behaviors seen during TP1. The increased congestion in TP2 seems to enhance these effects, most of which now show up as relative differences of 30 % or more. A new pattern of behavior arises with respect to DF flows. The number of active DF flows accumulates for algorithm 3 during TP2, which leads to lower average goodput on those flows. We can again attribute this to the congestion sensitivity demonstrated by FAST. Under normal Web traffic, network parameter settings for the experiment tend to generate congestion at fast access routers. During TP2, DD flows experience jumbo file transfers, so DF flows are affected by the normal congestion pattern as well as increased congestion due to jumbo files. Given this increased congestion, algorithm 3 has more trouble completing DF flows than the other algorithms – increased retransmissions on DF flows lead to longer holding times to complete the flows.

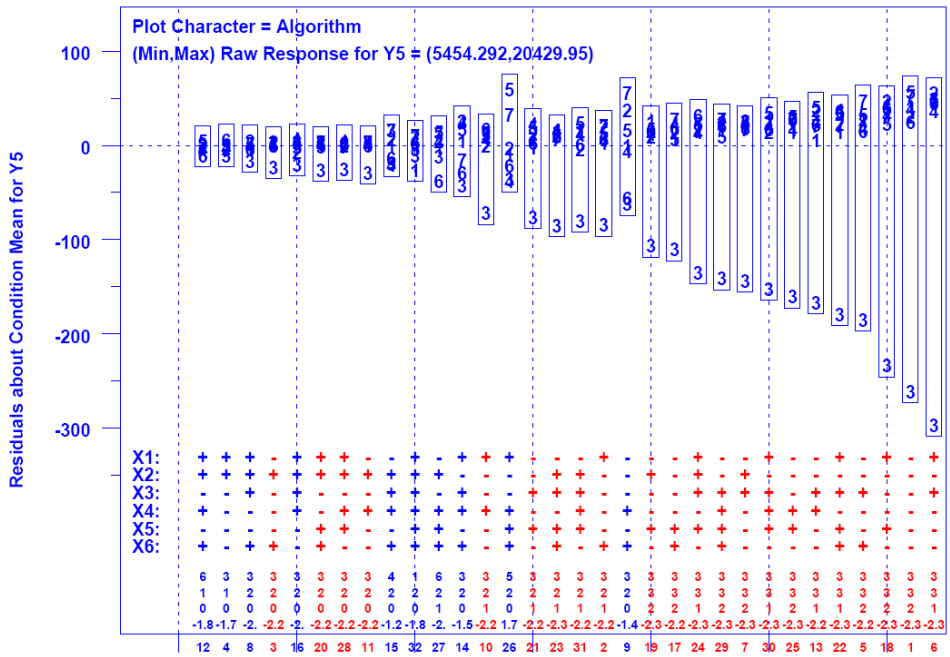

Figure 6-33. Detailed Analysis for Flow Completion Rate (flows per 200 ms) in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

A new behavior appears in TP2 with respect to the long and moderate distance long-lived flows. Algorithm 6 (Scalable TCP) tends to achieve higher average goodput. We consider this a tendency because there is no widespread pattern of statistical significance. We attribute this tendency to unfairness inherent in Scalable TCP. Under Scalable TCP (during TP1) long-lived flows establish a high congestion window. DD flows arising during TP2 have difficulty claiming a fair share of bandwidth from the entrenched long-lived flows.

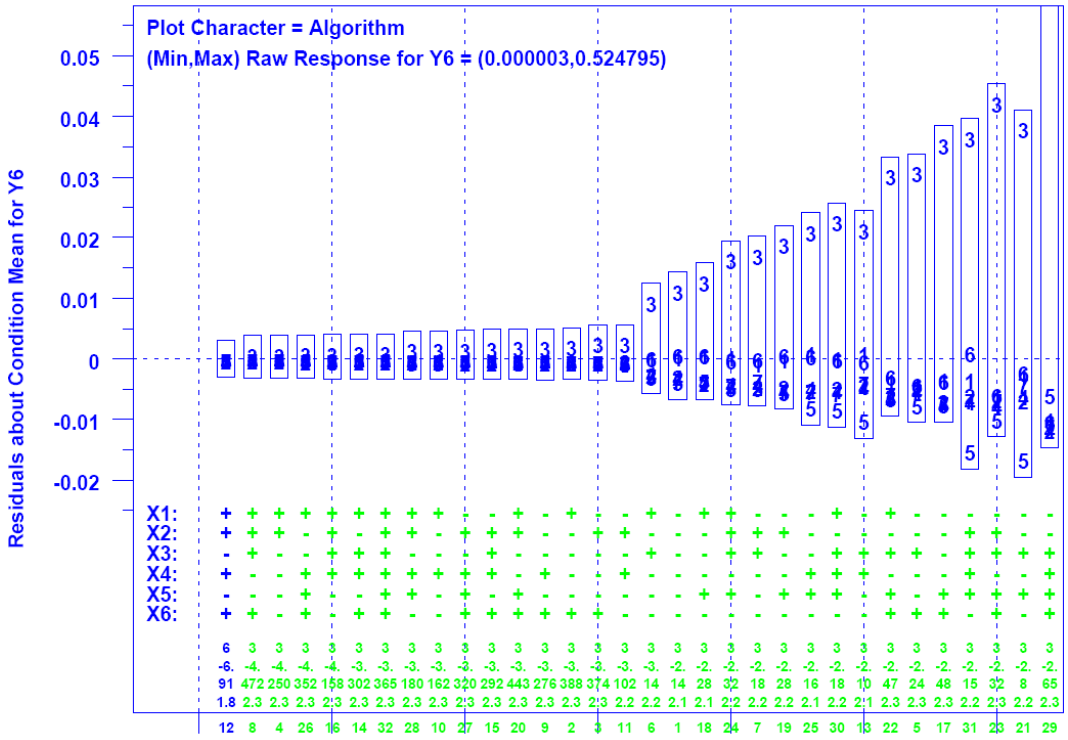

Figure 6-34. Detailed Analysis for Retransmission Rate (proportion of packets retransmitted) in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

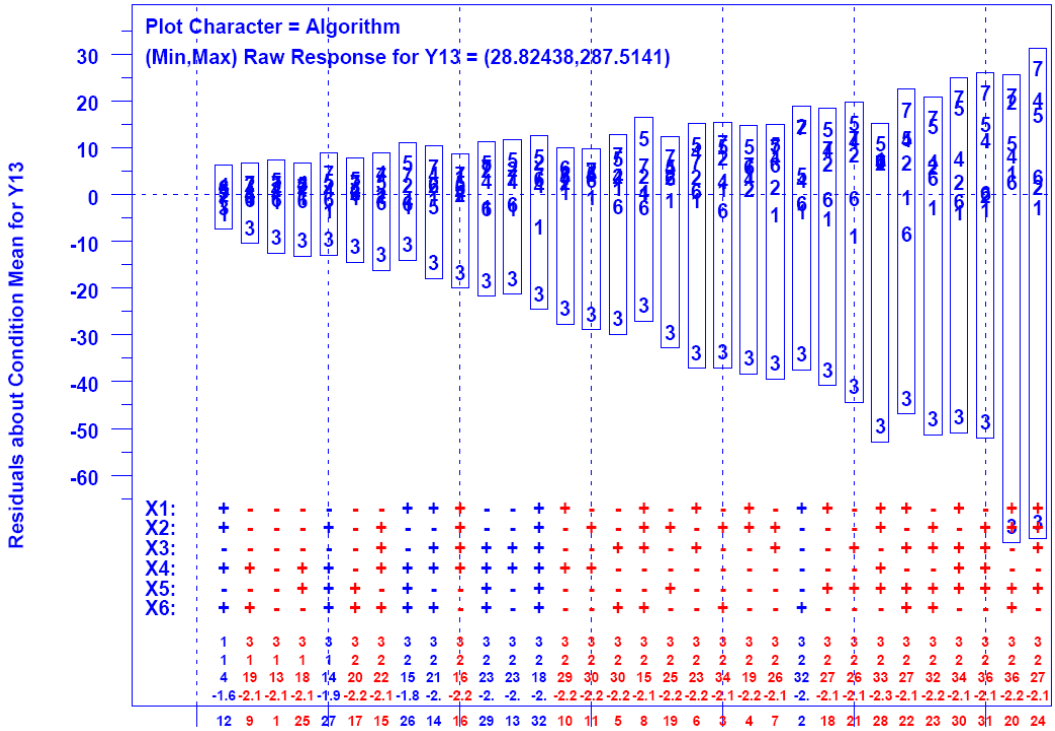

Figure 6-35. Detailed Analysis for Average Goodput (pps) on DF Flows in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

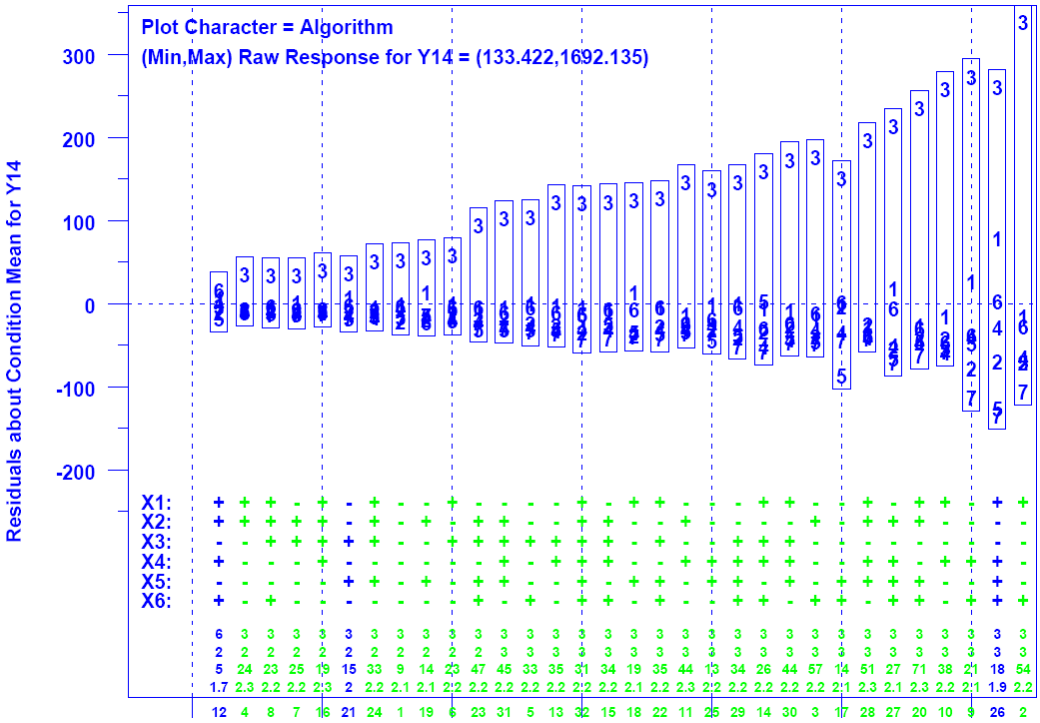

Figure 6-36. Detailed Analysis for Average Number of Active DF Flows in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

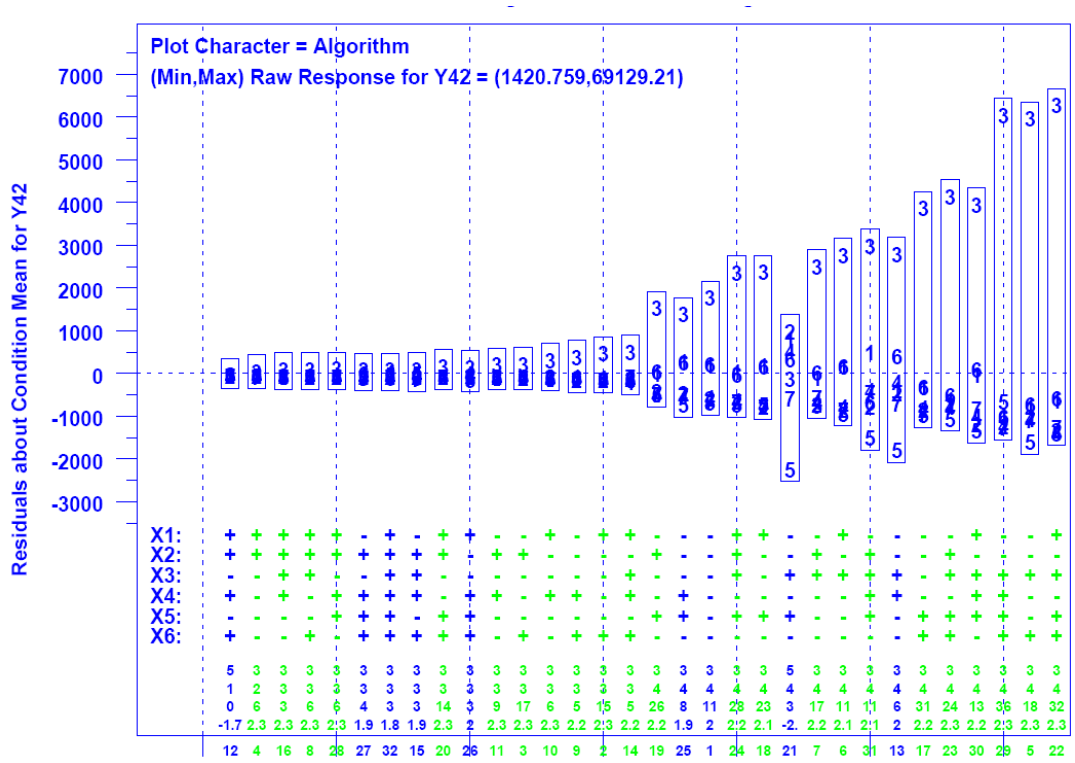

Figure 6-37. Detailed Analysis for Average Number of Connecting Flows in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

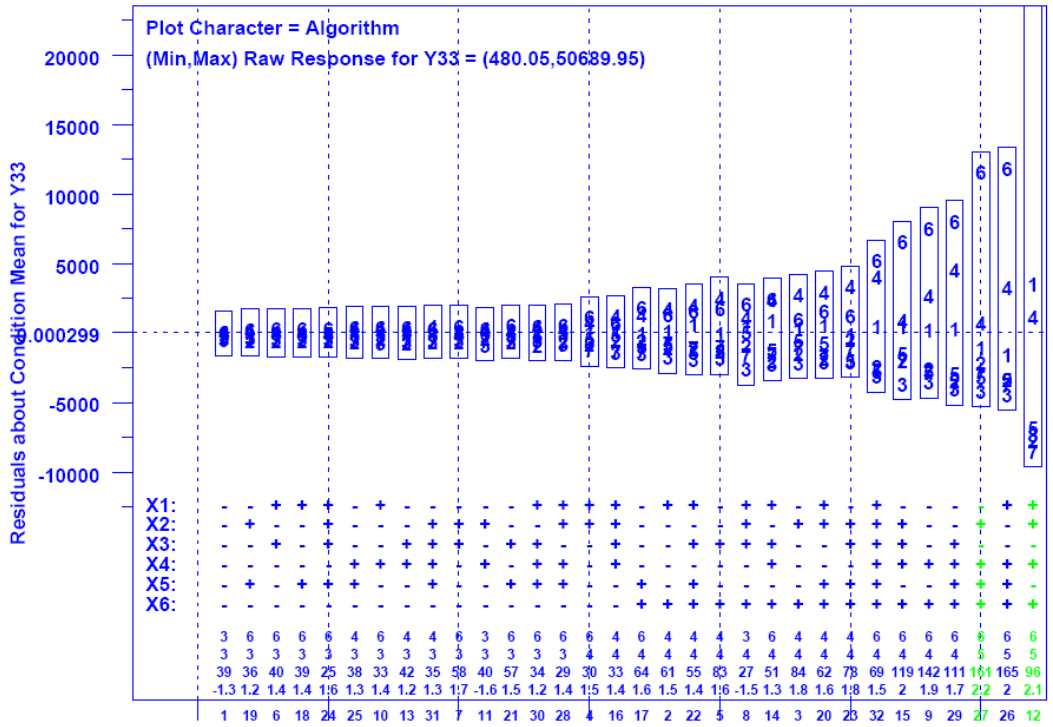

Figure 6-38. Detailed Analysis for Average Goodput (pps) on Long-lived Flow L1 in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

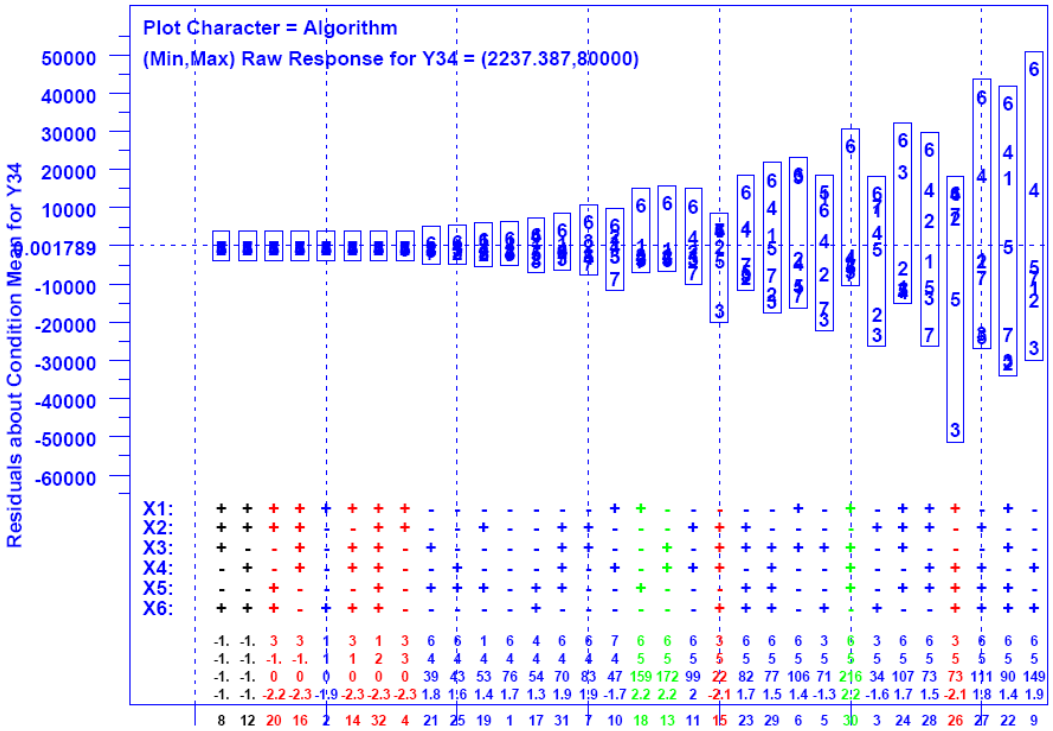

Figure 6-39. Detailed Analysis for Average Goodput (pps) on Long-lived Flow L2 in Time Period Two – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

6.4.3 Time Period Three (TP3)

During TP3 no new jumbo file transfers are initiated on DD flows; what remains is for residual jumbo transfers to complete as the network transitions back toward normal Web traffic. The degree to which normal conditions can be restored depends upon the number and size of jumbo transfers created during TP2.

6.4.3.1 Cluster Analysis for TP3. Fig. 6-40 shows an annotated set of 32 dendrograms for TP3. Since the level of congestion stays relatively high, as residual jumbo file transfers drain from the system, algorithm 3 remains distinctive. When omitting responses for algorithm 3, cluster analysis (Fig. 6-41) identifies no distinctive algorithm.

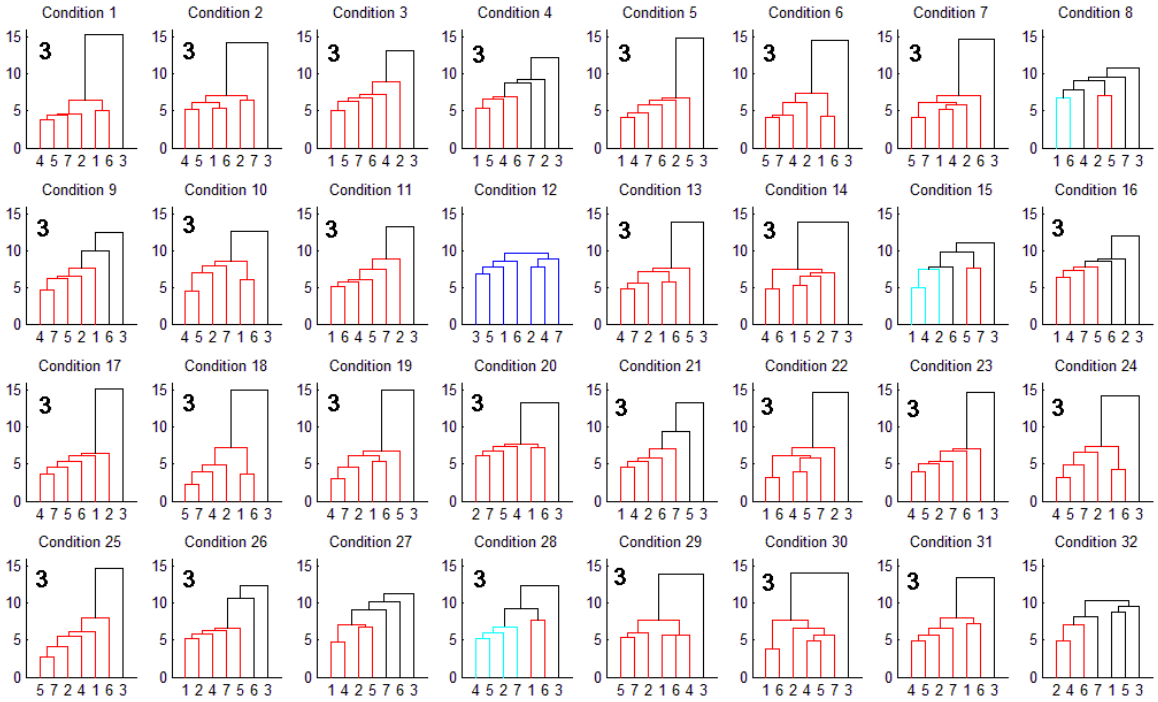

Figure 6-40. Clustering for Time Period Three – Annotated to Identify Distinctive Algorithm 3

6.4.3.2 Condition-Response Summary for TP3. Fig. 6-42 gives the condition-response summary for TP3. Fig. 6-43 shows the same summary after applying a filter showing only statistically significant outliers for which the relative effect exceeds 30 %. Algorithm 3 stands out in both figures – the distinctiveness is quite similar to that seen for TP2. Fig. 6-43 also reveals two new patterns. First, algorithm 2 (CTCP) shows a large increase in the average congestion window, which is pervasive over many conditions during TP3. Second, average goodput lags on the higher propagation, long-lived TCP flows (L1 and L2) as the DD paths recover from the period of jumbo file transfers.

6.4.3.3 Analysis of Significant Responses for TP3. Based on Figs. 6-42 and 6-43 we selected several responses for more detailed analysis. Specifically, in Figs. 6-44 to 6-49, we report analyses for congestion window increase rate (y2), flow completion rate (y5), retransmission rate (y6), average goodput on DF flows (y13), number of active DF flows (y14) and number of connecting flows (y42). Fig. 6-50 illustrates the substantial increase

in congestion window size (y4) for algorithm 2. Fig. 6-51 illustrates (with flow L2) how goodput on long-lived flows tends to lag under standard TCP (algorithm 7).

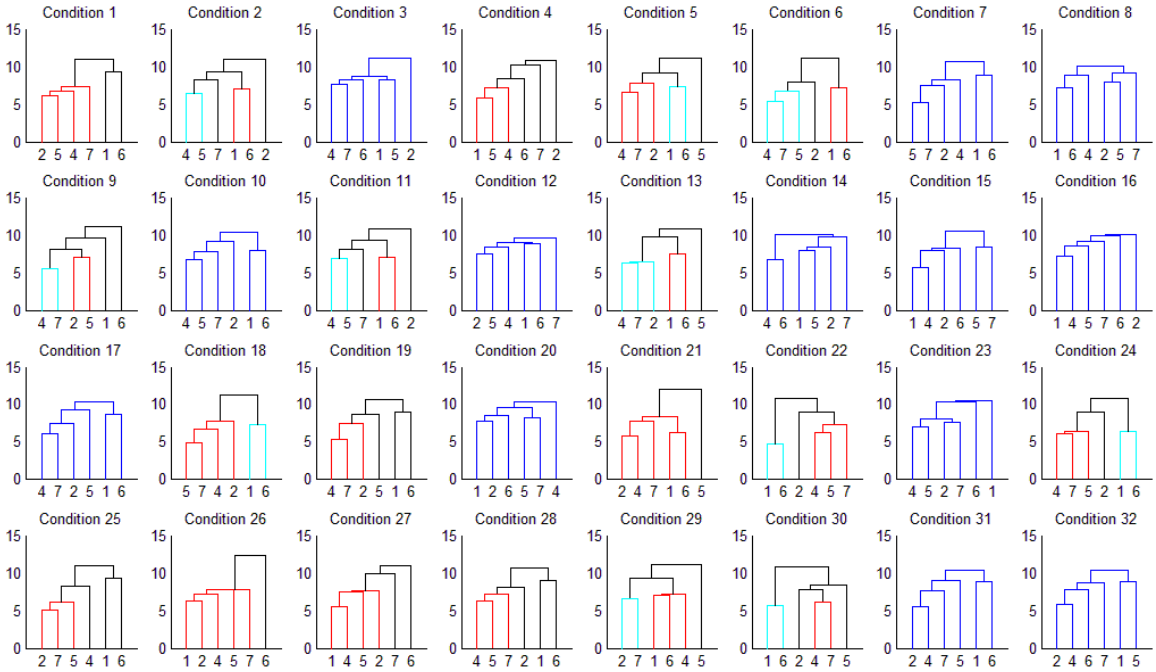

Figure 6-41. Clustering for Time Period Three – Algorithm 3 Omitted

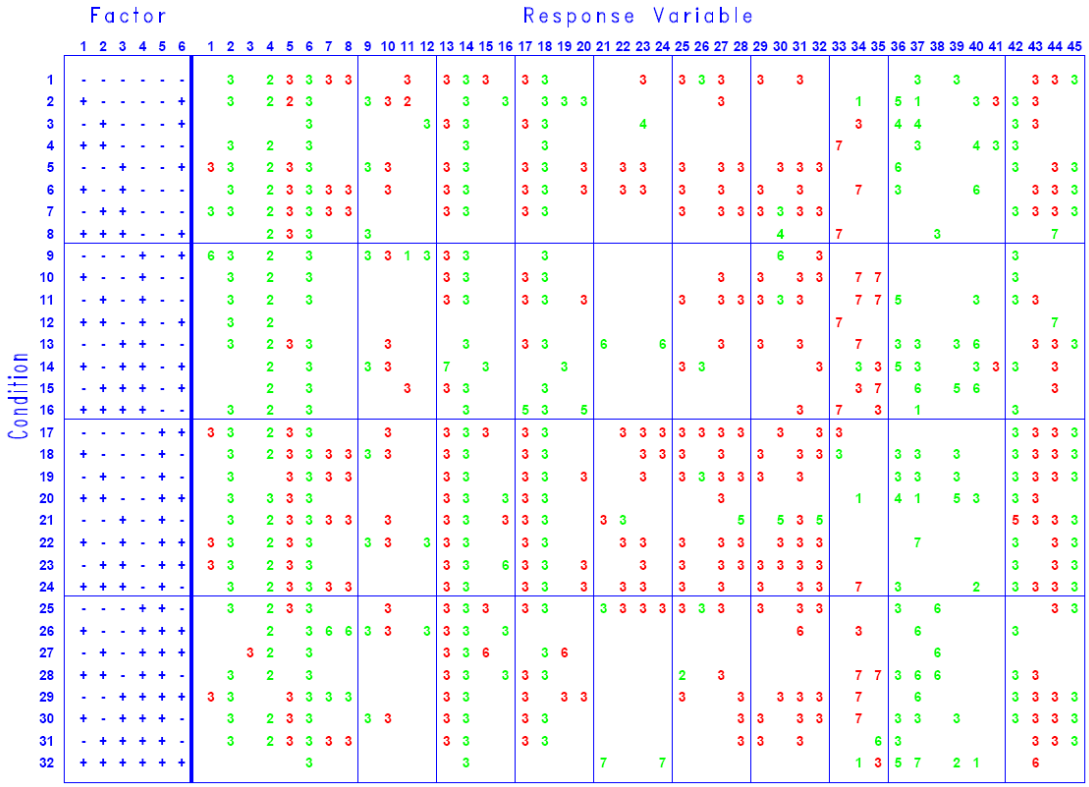

Figure 6-42. Condition-Response Summary for Time Period Three – plot displays for each Factor Combination (row) vs. Response Variable (column) the Identifier of the Algorithm manifesting a Statistically Significant Outlier

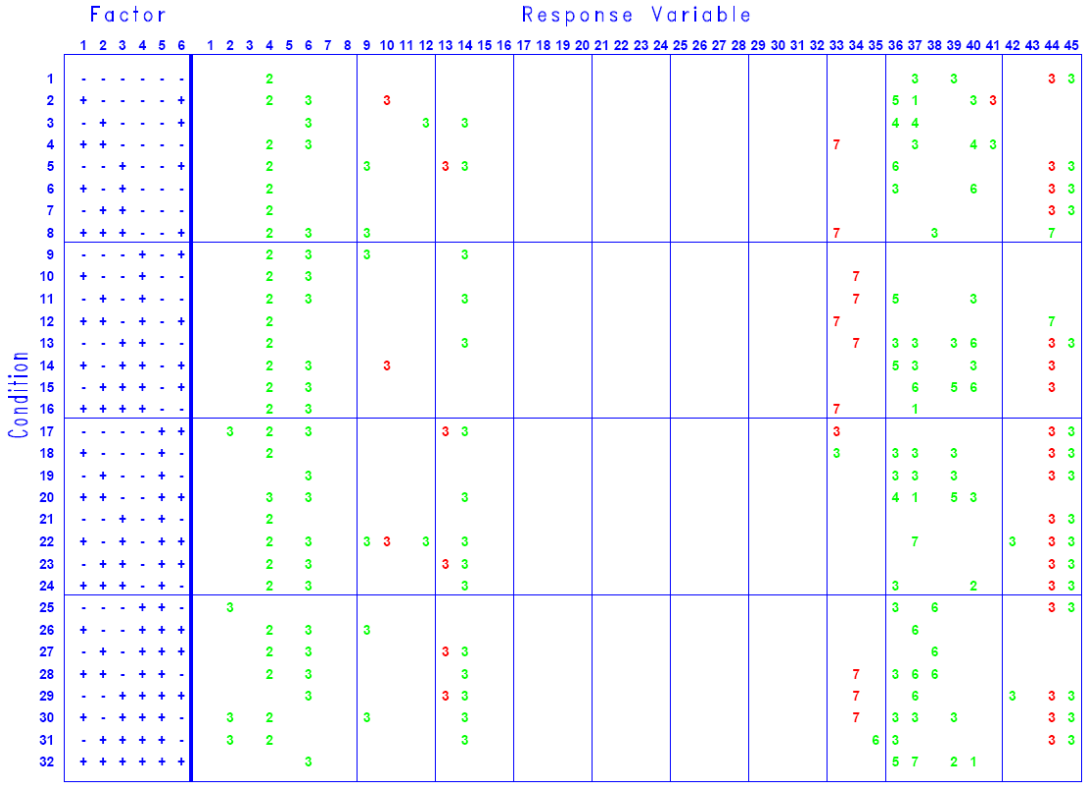

Figure 6-43. Filtered Summary Plot for Time Period Three Identifying Statistically Significant Outliers with Associated Relative Effect > 30%

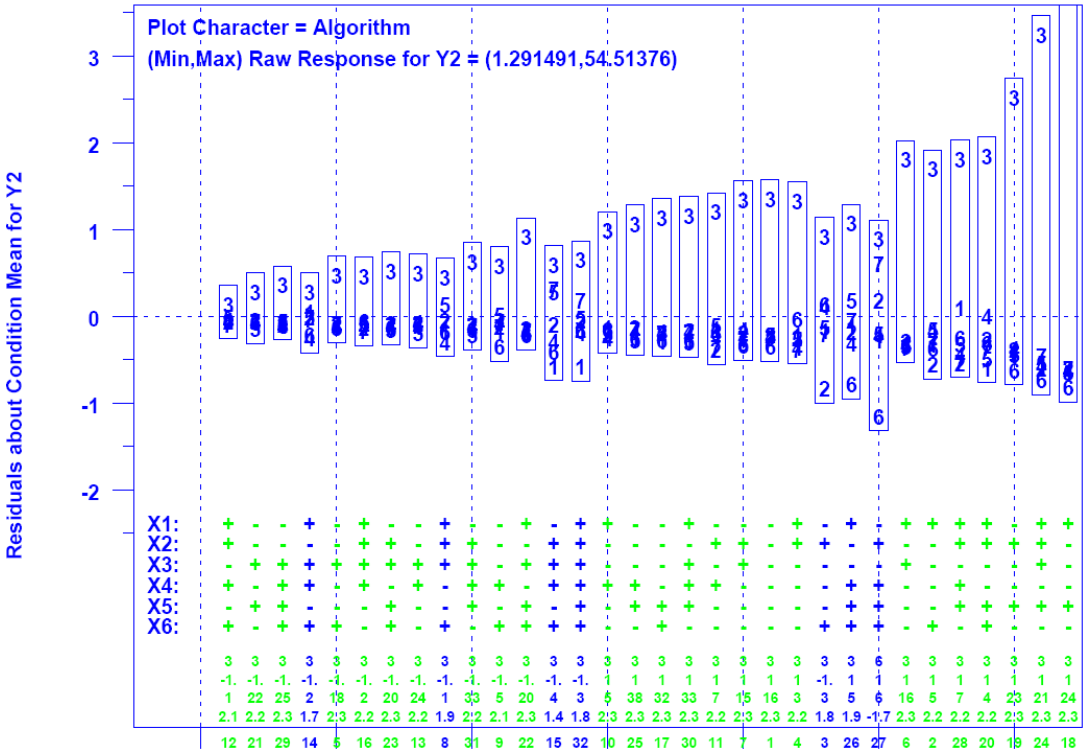

Figure 6-44. Detailed Analysis for Congestion Window Increase Rate (increases per 200 ms) in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

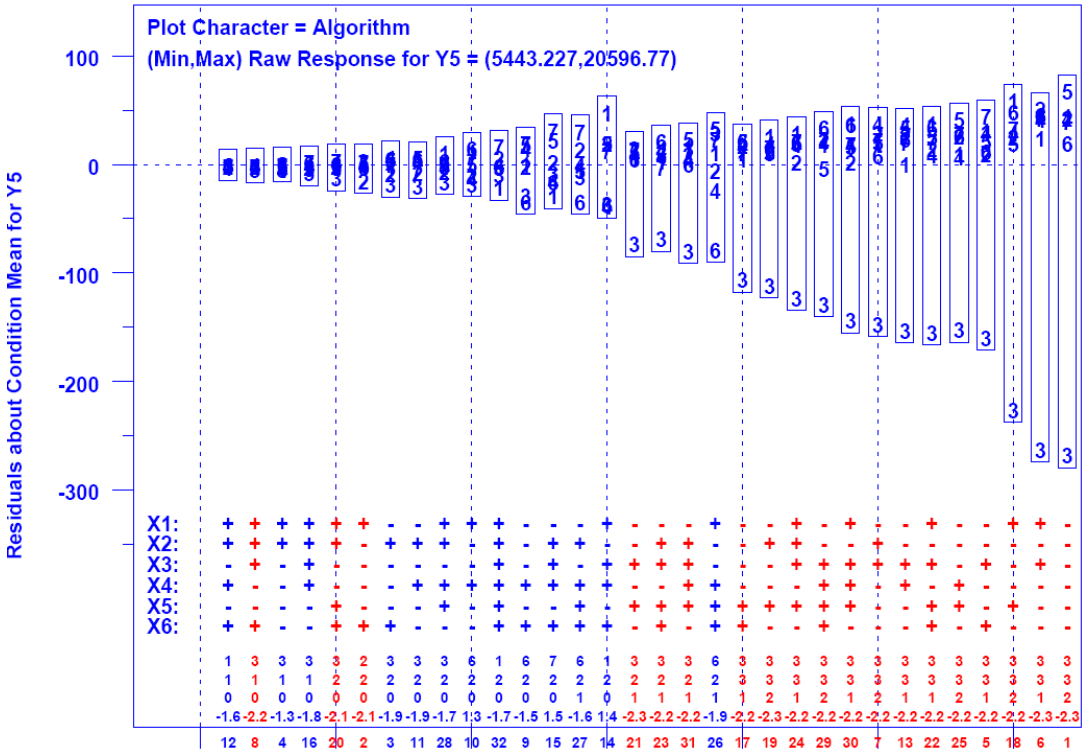

Figure 6-45. Detailed Analysis for Flow Completion Rate (flows per 200 ms) in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

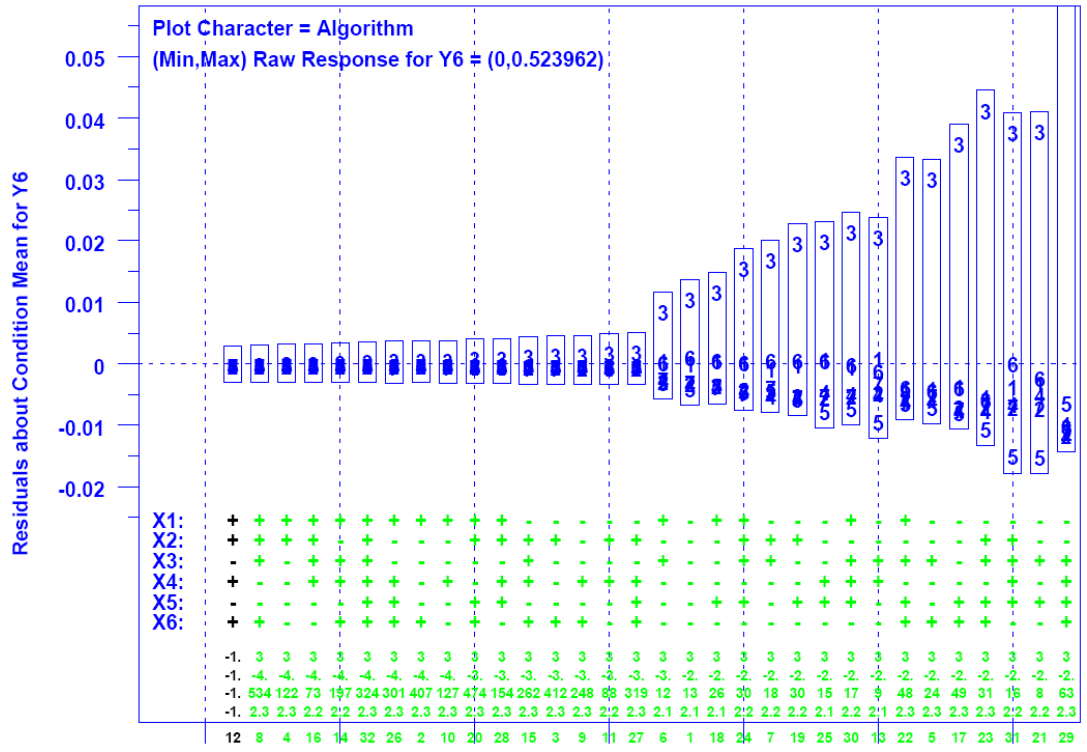

Figure 6-46. Detailed Analysis for Retransmission Rate (proportion of packets retransmitted) in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

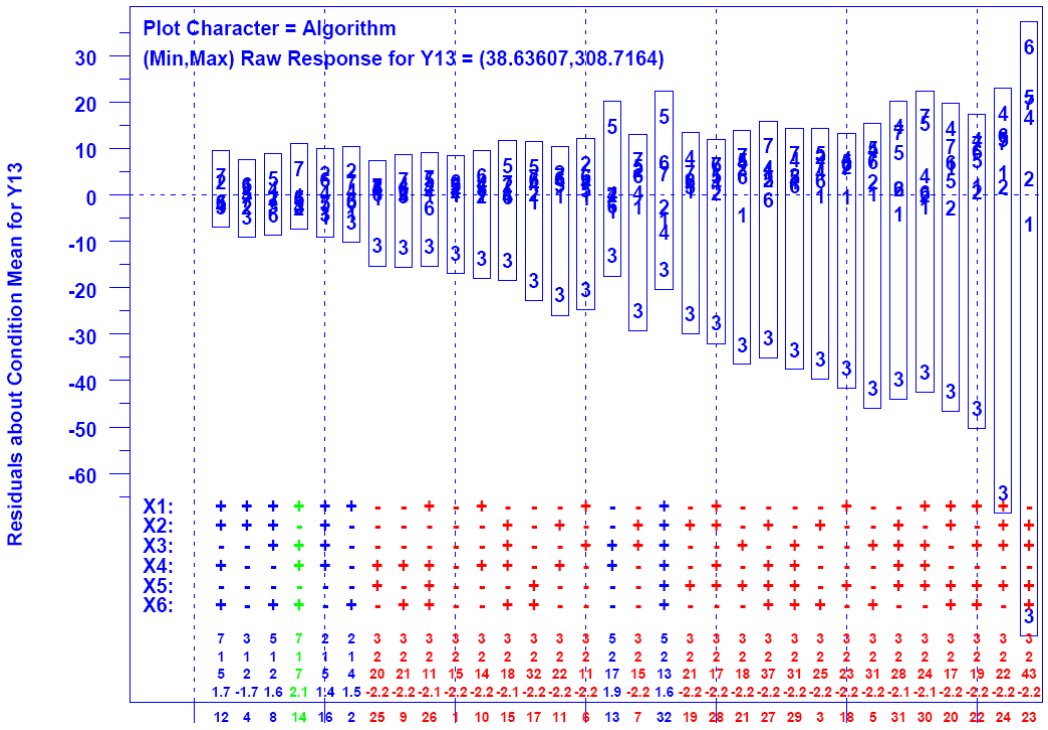

Figure 6-47. Detailed Analysis for Average Goodput (pps) on DF Flows in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

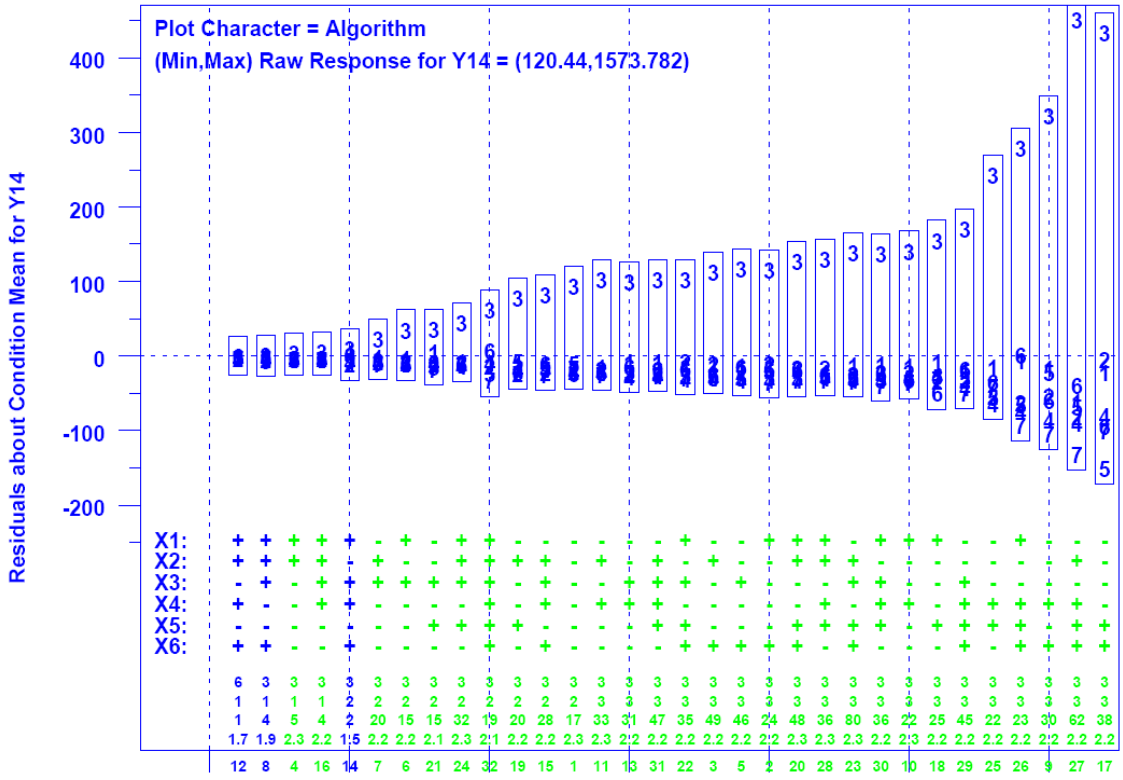

Figure 6-48. Detailed Analysis for Average Number of Active DF Flows in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

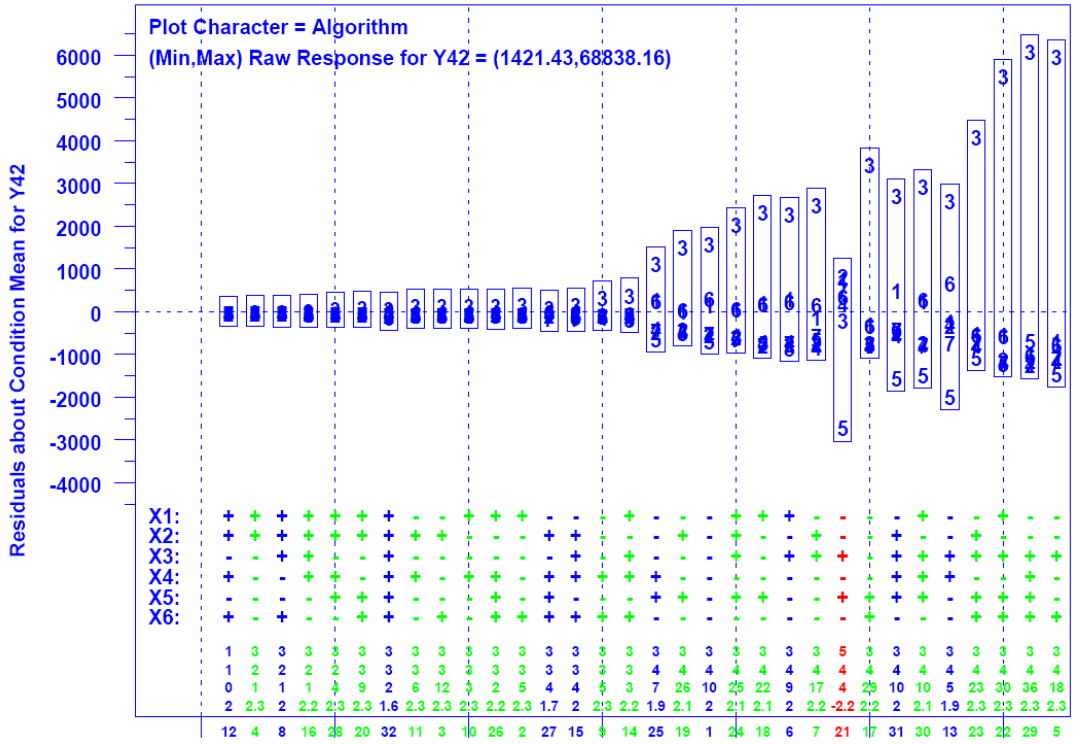

Figure 6-49. Detailed Analysis for Average Number of Connecting Flows in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

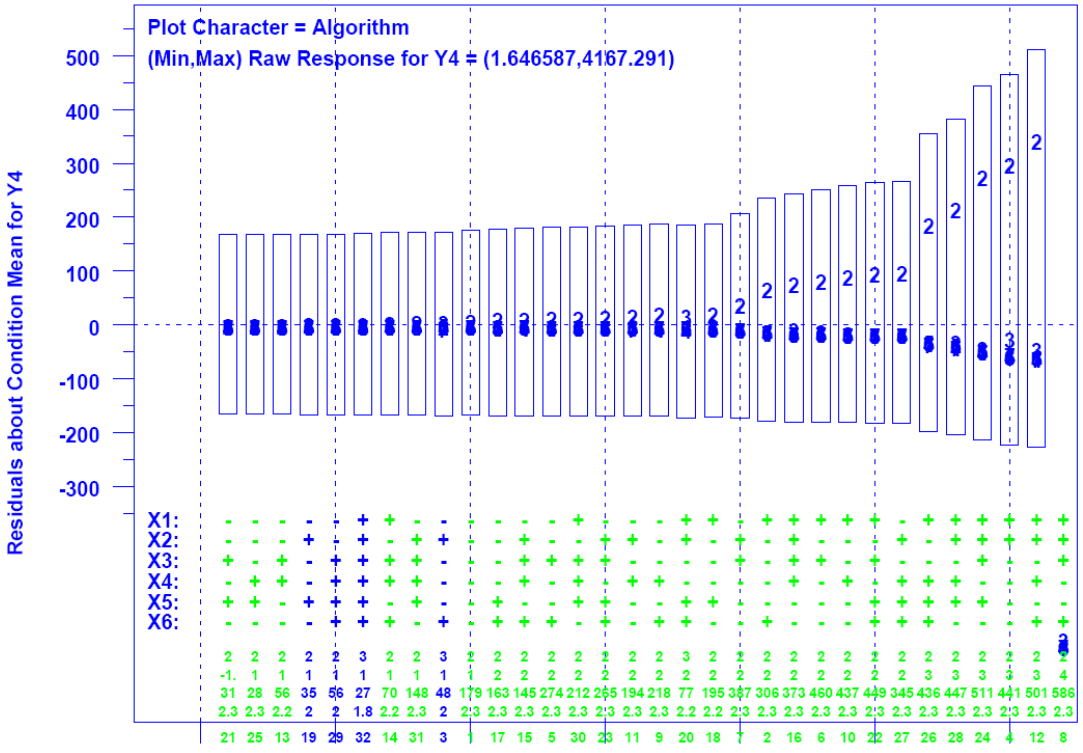

Figure 6-50. Detailed Analysis for Average Congestion Window Size (packets) in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

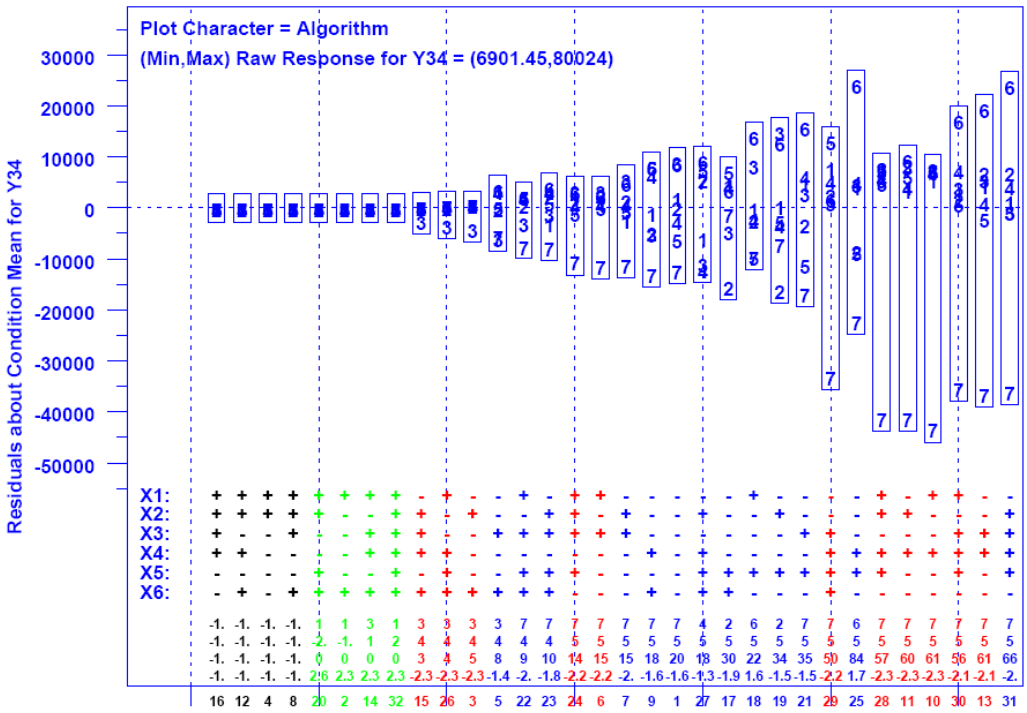

Figure 6-51. Detailed Analysis for Average Goodput (pps) on Long-lived Flow L2 in Time Period Three – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

6.4.3.4 Summary of Results for TP3. FAST (algorithm 3) exhibits the same distinctive behaviors seen during TP1 and TP2. Residual congestion from TP2 maintains these effects at an enhanced level; most effects continue to show relative differences of 30 % or more. A new pattern of behavior arises with respect to average congestion window size, where algorithm 2 (CTCP) shows a substantial increase over other algorithms – this difference is limited to TP3. Results also show that average goodput for long-lived flows using algorithm 7 (TCP) tends to lag when recovering from the congested period (TP2). This trait of standard TCP congestion control was an initial stimulus for many of the proposals for alternate congestion control mechanisms.

6.4.4 Aggregated Responses (Totals)

Here we present analyses for the 28 responses collected over the entire 25-minute scenario. Recall that most of these responses are aggregated counts, but SYN rate on connected flows and goodput on completed flows are averages. Whereas the previous analyses focused on differences in instantaneous behavior averaged over 5-minute intervals, the current analysis examines the effects of behavioral differences viewed over a longer period.

6.4.4.1 Cluster Analysis for Totals. Fig. 6-52 shows the usual annotated set of 32 dendrograms, but this time clustering based on the 28 aggregate responses. Similar to the cluster analyses for the three time periods, algorithm 3 appears distinctive in many (24) of the conditions. Fig. 6-53 shows the results from clustering with algorithm 3 responses excluded. No significant difference appears among the remaining algorithms, though algorithms 1 and 6 exhibit some tendency to be paired.

6.4.4.2 Condition-Response Summary for Totals. Fig. 6-54 gives the condition-response summary for the aggregate responses. One finding from the figure is that algorithm 3 (FAST) tends to input more packets but not to output more packets – this is congruent with a higher loss rate, and consequent increased retransmission rate. For path classes prone to congestion, algorithm 3 (FAST) provides lower average goodput, which means these flows require more retransmissions and take longer to complete. In addition, algorithm 3 (FAST) connects and completes fewer flows – among a wide range of flow classes and across the entire set of backbone routers. As expected, based on analysis of the time periods, algorithm 3 (FAST) shows a higher average SYN rate over most conditions. This is congruent with a larger number of flows pending in the connecting state, and with a higher retransmission rate due to lost packets.

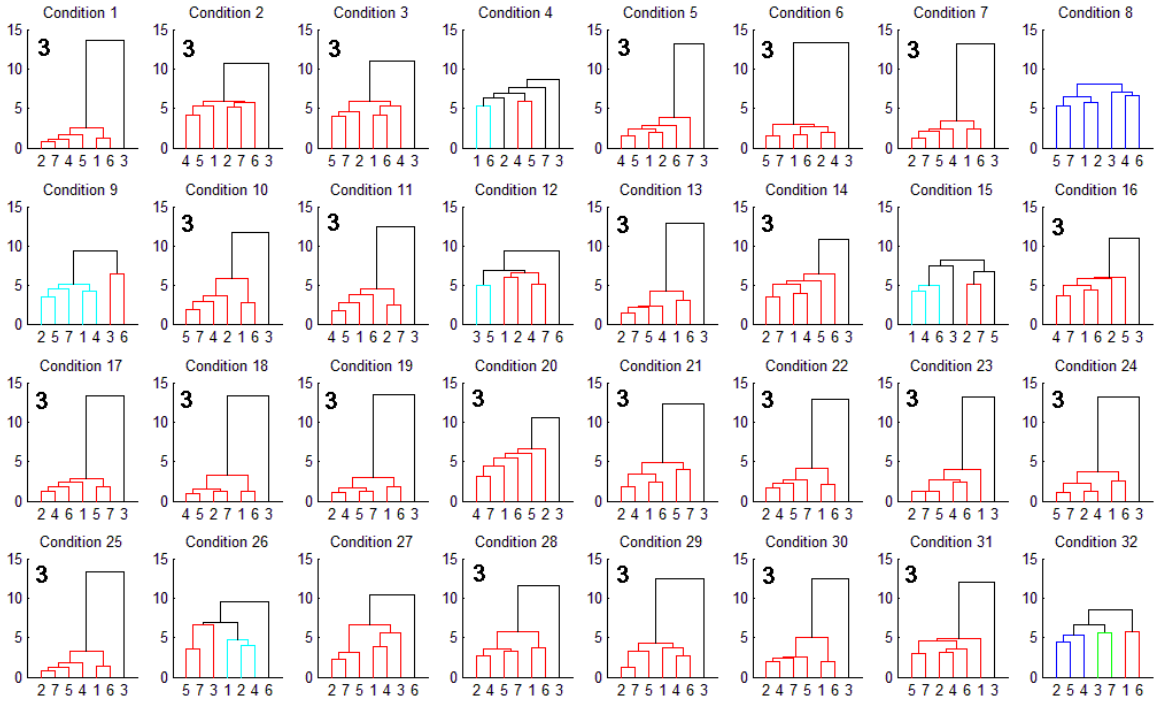

Figure 6-52. Clustering for Totals – Annotated to Identify Distinctive Algorithm 3

6.4.4.3 Analysis of Significant Responses for Totals. Based on Fig. 6-54 we selected two responses for detailed analysis. Algorithm 3 completed fewer flows under most conditions for most flow classes – including DF flows (T.y8), DN flows (T.y10), FF flows (T.y12), FN flows (T.y14) and NN flows (T.y16). For these flow classes, algorithm 3 also usually exhibited lower average goodput. Algorithm 3 completed fewer flows across all backbone routers in the network. Rather than show detailed analyses for all of these categories, we present, in Fig. 6-55, an analysis of the aggregate number of flows completed (T.y4), where algorithm 3 underperforms under most conditions. We also show, in Fig. 6-56, a detailed analysis of the average SYN rate. In all but two conditions (the least and most congested), algorithm 3 leads to more SYNs being sent on average to establish flows. This supports earlier observations that algorithm 3 tends to have substantially more flows pending in the connecting state at any instant in time.

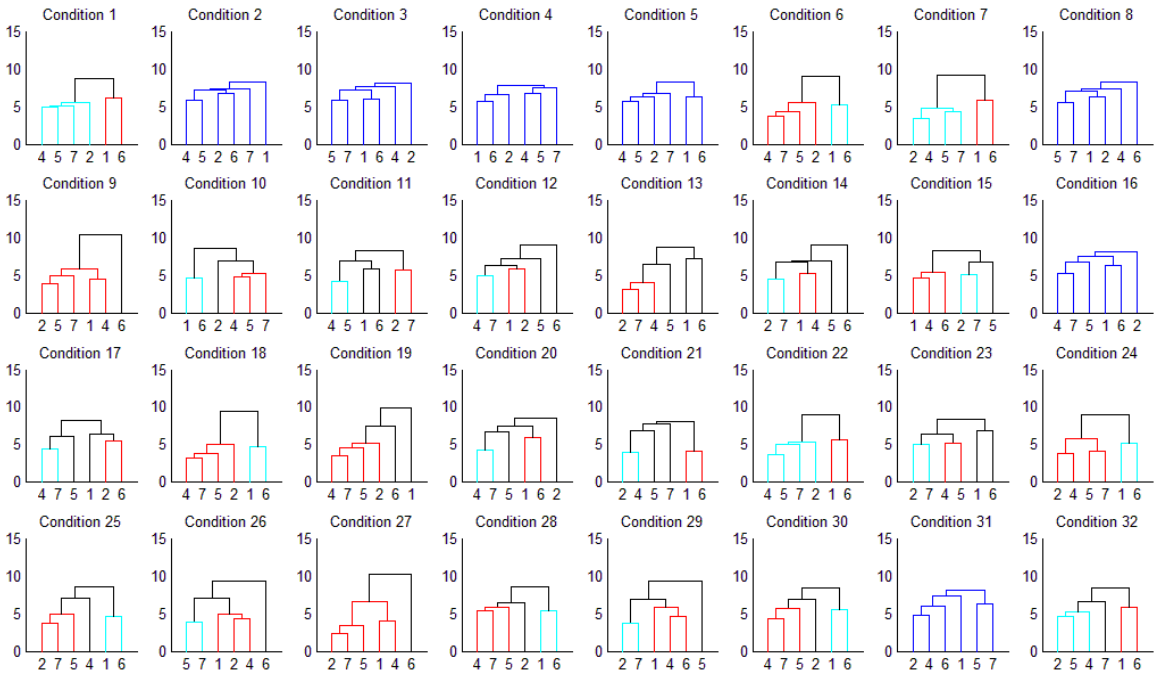

Figure 6-53. Clustering for Totals – Algorithm 3 Omitted

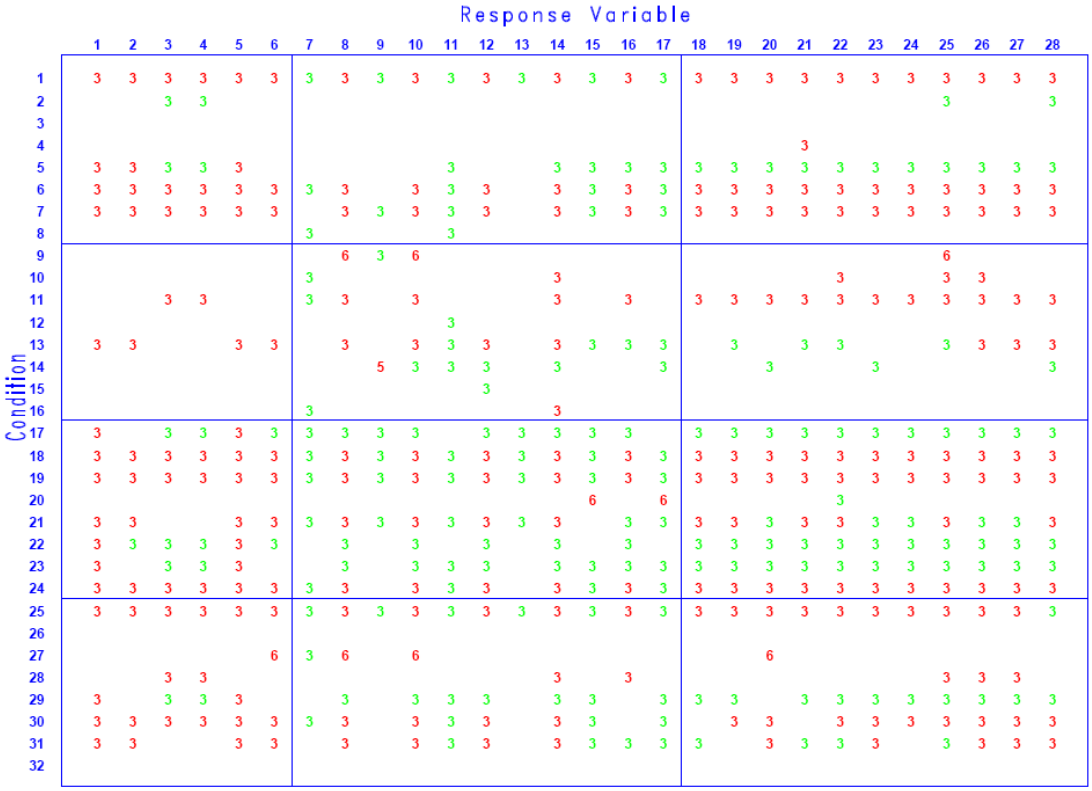

Figure 6-54. Condition-Response Summary for Totals – plot displays for each Factor Combination (row) vs. Response Variable (column) the Identifier of the Algorithm manifesting a Statistically Significant Outlier

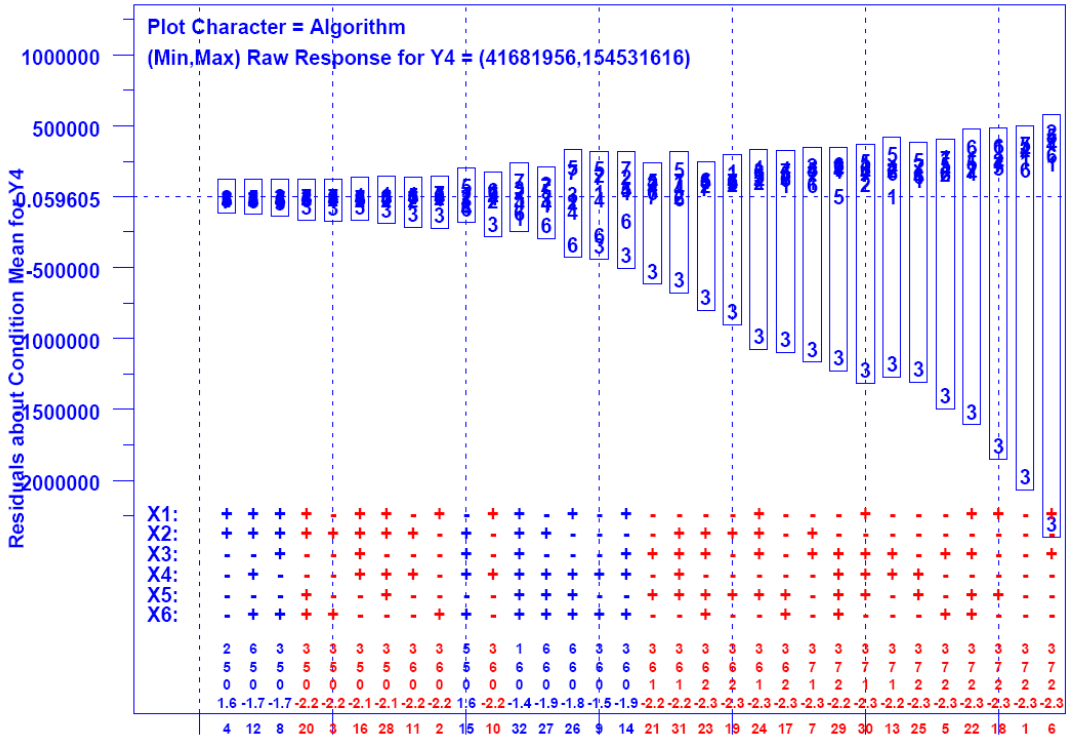

Figure 6-55. Detailed Analysis for Aggregate Number of Flows Completed over 25-minute Scenario – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

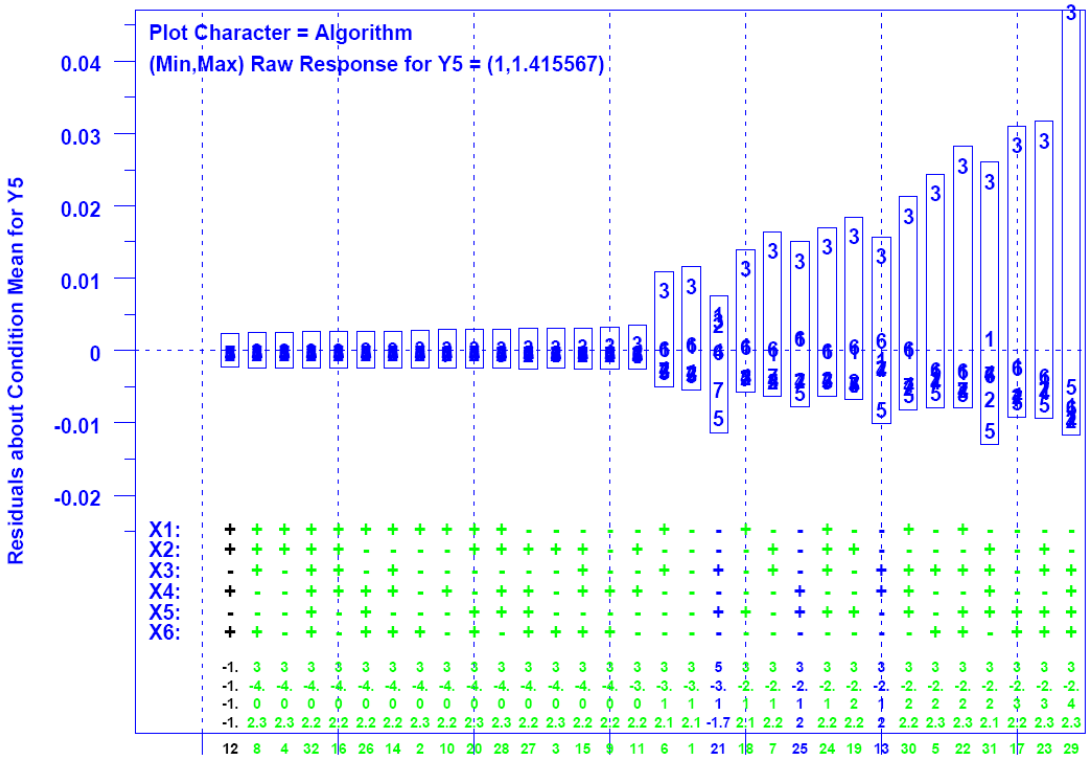

Figure 6-56. Detailed Analysis for Average SYN Rate (count of SYNs sent per connection attempt) for Connecting Flows over 25-minute Scenario – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

6.4.4.4 Summary of Results for Totals. Under the conditions investigated in this experiment, FAST (algorithm 3) completes fewer flows during the 25-minute scenario. While FAST completes only up to 2 % fewer flows, this amounts to between a million and 10 million flows over 25 minutes – 40×10^3 to 400×10^3 flows a minute. In addition, FAST impedes the ability of flows to establish connections.

6.5 Findings

From the results reported in Sec. 6.4, we identified four main findings, as discussed below. In addition, detailed analysis of individual responses, when excluding algorithm 3, identified some tendencies, which we outline in Sec. 6.5.5.

6.5.1 Finding #1

Setting aside algorithm 3 (FAST), for the experiment scenario and conditions examined in this section, the alternate congestion control algorithms exhibited indistinguishable macroscopic behavior and modest differences in user experience. In other words there was no overall advantage to be gained in switching the entire network to a particular alternate congestion avoidance scheme, nor was there any overall disadvantage in switching. (Remember we are excluding FAST from this finding.) Selected users could experience somewhat higher throughputs when using alternate congestion control algorithms during periods of competing large file transfers, but no widespread improvement in user experience should be expected.

To understand this finding, recall that slow-start procedures are unaffected by alternate congestion control mechanisms, which define replacements only for the TCP congestion avoidance phase. No matter what congestion control mechanism is used, a flow commences operating in initial slow-start and switches to congestion avoidance only after a packet loss (because we used a high initial slow-start threshold). Aside from FAST and TCP Reno, the alternate congestion avoidance procedures specify an activation threshold (either a certain congestion window size or duration since the most recent loss). Below that threshold, a flow adopts standard TCP congestion avoidance procedures; above that threshold the flow adopts alternate congestion avoidance procedures.

Recall that in our experiment we simulated 32 conditions covering a range of congestion patterns, which could be classified roughly into 16 uncongested and 16 congested conditions. Condition 12 created the least congestion, while condition 21 created the most congestion. Of course, even uncongested conditions include localized congestion arising from the onset of jumbo file transfers during TP2, as well as from hot spots appearing from time-to-time at particular access routers. For example, in Fig. 6-57, we plot data under condition 12 for algorithm 1 (we chose to plot BIC because it has the lowest activation threshold: congestion-window > 14 packets). Note that most of the 1.2×10^4 or so active flows (red) in TP1 (3000 – 4500) and TP3 (6000 – 7500) operate in initial slow start (green). This means that these active flows complete their file transfers without packet loss. For flows of this nature, congestion avoidance is never activated, so one would expect alternate congestion avoidance procedures to make no difference. During TP2 (4500 – 6000), jumbo file transfers on DD flows cause concentrated congestion at directly connected access routers. As Fig. 6-57 shows, even during TP2 the number of flows operating in congestion avoidance reached a level of around 10^3 (under 10 %) out of 1.3×10^4 active flows. Half of the flows operated in normal (brown)

congestion control mode (i.e., congestion window $\leq$ 14) and half operated using BIC (blue) congestion avoidance procedures. One would expect DD flows operating in alternate congestion control mode to achieve higher throughput than the DD flows operating in normal congestion control mode. So, selected users could experience improved throughput over others during TP2.

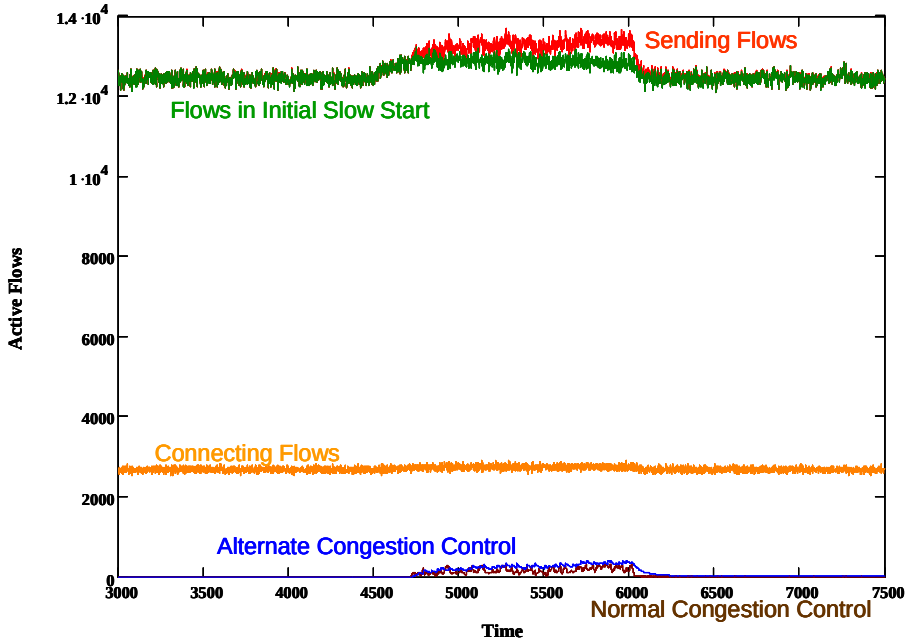

Figure 6-57. Five Time Series Showing the Distribution of Flow States over Three Time Periods for Algorithm 1 (BIC) under Condition 12 – x axis shows time in 200 ms increments and y axis shows number of active flows in each state

In Fig. 6-58, we plot the equivalent distribution of flow states for BIC under the most congested condition: 21. Of the 1.45 x 10^5 active flows (red) in TP1 and TP3 about 1.4 x 10^5 flows (brown) operate in normal congestion control mode and the rest (green) operate in initial slow start. Under these conditions, alternate congestion avoidance procedures are not activated. During TP2, the onset of jumbo file transfers leads to about 1.5 x 10^3 flows (blue) (around 1 %) using alternate congestion avoidance procedures. This small proportion of flows adopting alternate procedures cannot be expected to make a large difference in macroscopic network behavior.

What about user experience? Most flows in a heavily congested network, or in heavily congested portions of a network, will be sharing paths with many other flows. For this reason, one should expect most flows to be operating within normal congestion control mode; these flows cannot achieve a large enough congestion window size (or avoid losses for long enough) to activate alternate congestion avoidance procedures. On the other hand, flows transiting very fast (DD) paths may be able to benefit from alternate congestion control procedures. Overall pattern analysis found that average goodput on DD flows in TP2 showed statistically significant improvement for the extreme algorithm in only three (4, 15 and 28) of 32 conditions; the three conditions were all uncongested. On the other hand, Table 6-32 gives, for each congestion control algorithm, the average goodput on DD flows when averaged across all conditions during TP2, as well as the

minimum and maximum average goodputs. The figures in Table 6-32 suggest that the alternate congestion control mechanisms do, on average, provide better user experience on DD flows during TP2. In fact, during TP2 TCP yields lowest average goodput, but this is only 1 % to 7 % lower than for the other algorithms. The detailed analysis of average goodput on DD flows (y9) during TP2 also shows that a particular alternate congestion control algorithm can improve goodput by 2 % to 19 % over the average for specific conditions. However, there is no particular pattern as to which alternate congestion control algorithm provides best goodput. From this, we conclude that under some conditions users can experience higher goodput when using alternate congestion control algorithms on DD flows that compete to complete large file transfers. The overall improvement when averaged across a wide range of conditions would, however, likely be below 10 %.

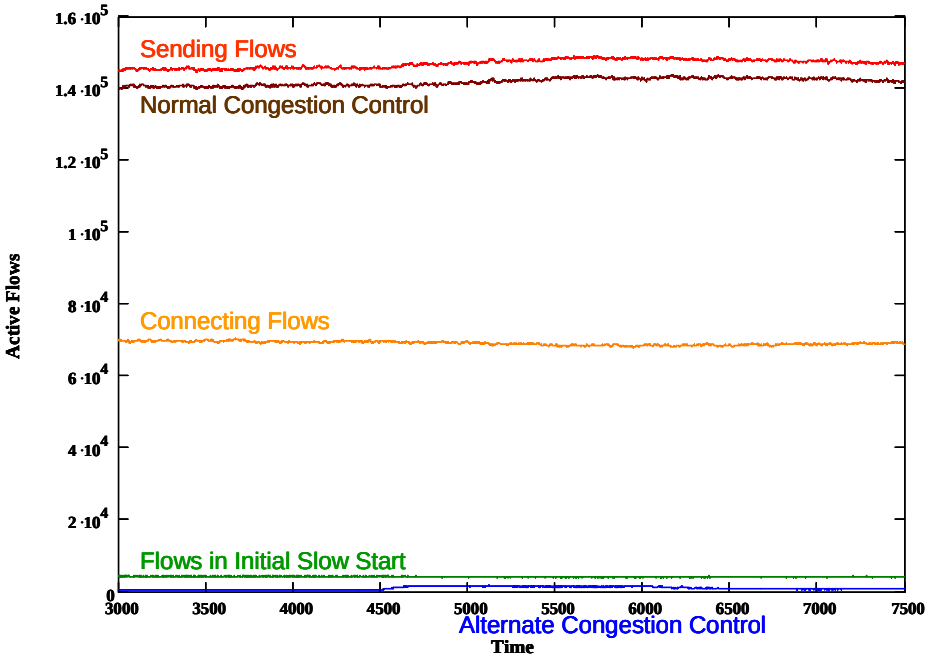

Figure 6-58. Five Time Series Showing the Distribution of Flow States over Three Time Periods for Algorithm 1 (BIC) under Condition 21 – x axis shows time in 200 ms increments and y axis shows number of active flows in each state

Table 6-32. Average, Minimum and Maximum Goodput (pps) on DD Flows for Each Congestion Control Algorithm during TP2 when Averaged over All 32 Conditions

		BIC	CTCP	FAST	HSTCP	HTCP	Scalable	TCP
Average Goodput	Average	1260.62	1242.59	1277.23	1193.47	1216.96	1241.56	1184.58
	Minimum	461.27	436.93	474.47	438.84	431.58	437.13	430.25
	Maximum	4079.63	4155.30	4201.02	3724.64	4085.80	3907.04	3680.50

In summary, switching the entire network from standard TCP congestion control to BIC, CTCP, HSTCP, HTCP or Scalable TCP should not cause large shifts in macroscopic network behavior. Further, Web-browsing users would see little difference in their experience. Under uncongested conditions typical file transfers complete in initial slow start. Under heavily congested conditions typical file transfers enter normal congestion avoidance mode. On the other hand, switching to an alternate congestion control mechanism could modestly benefit selected users with high capacity access paths during periods where large file transfers compete for bandwidth on shared, high-capacity paths. These findings are limited to cases where all users on the network: (a) have a high initial slow-start threshold and (b) adopt the same congestion control mechanism. In Chapter 7 we investigate the case of a lower initial slow-start threshold. We address the case of heterogeneity among congestion control mechanisms in Chapters 8 and 9.

6.5.2 Finding #2

When deployed network wide, alternate congestion control algorithm 3 (FAST) can produce macroscopic changes in network behavior at congested places in the topology and during congested periods. Further, these changes can present Web-browsing users with lower average goodputs and longer connection times. The influence of these effects increases with increasing congestion. These findings suggest that deploying FAST on a wide scale could incur significant risk.

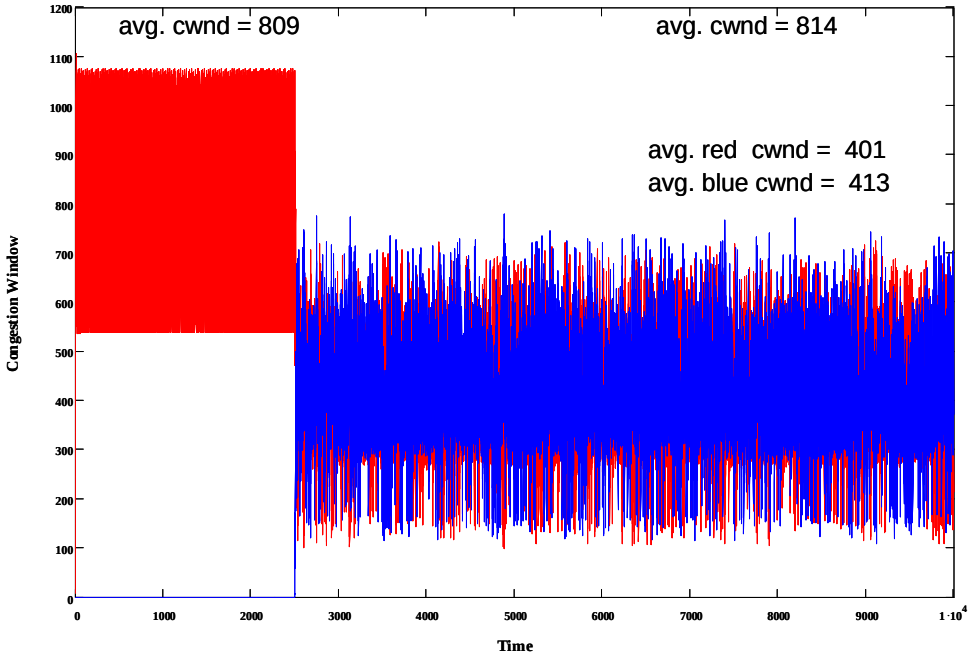

Figure 6-59. Reproduction of Fig. 5-22, Showing Change in *cwnd* for Two FAST Flows (α_F = 200, *rtt* = 42 ms) – x axis gives time in 200 ms increments and y axis gives congestion window in packets ranging from 0 to 1200

To understand this finding, recall from Sec. 5.4.4 that FAST exhibits rapid oscillations in congestion window size when a path has insufficient buffers to contain the packets that the FAST algorithm attempts to maintain queued at a bottleneck. The resulting behavior is illustrated by Fig. 5-22, which, for convenience, we reproduce here

as Fig. 6-59. In this figure, two FAST flows are attempting to maintain 100 packets each through a bottleneck router that has buffers for only 176 packets. Insufficient buffer space results in packet losses, followed by (50 %) window reduction, followed by rapid increase in congestion window. This cycle repeats quite rapidly because FAST flows update their target congestion window frequently (every 20 ms here). This rapid oscillation in congestion window appears to be the source for the deleterious behavior exhibited by FAST in congested locations and at times of significant network-wide congestion, as we elaborate below.

When a large number of flows simultaneously transit a network router, the overall effect can be to flood the router with many packets. When the number of flows is sufficient to overrun the available buffers in the router, FAST flows exhibit an oscillatory behavior that can create additional congestion that causes the flows to remain in oscillation for an extended time. For example, Fig. 6-60 shows the evolution of the congestion window for long-lived FAST flow L2 during 500 measurement intervals within TP2 under (the most congested) condition 21. For comparison, Fig. 6-61 gives the behavior of standard TCP Reno under the same circumstances. Faced with congestion, the other alternate congestion control algorithms we simulated oscillate with a frequency closer to TCP than to FAST. Figs. 6-62 through 6-66 show the behavior for the remaining congestion control algorithms under condition 21 for the same measurement intervals in TP2.

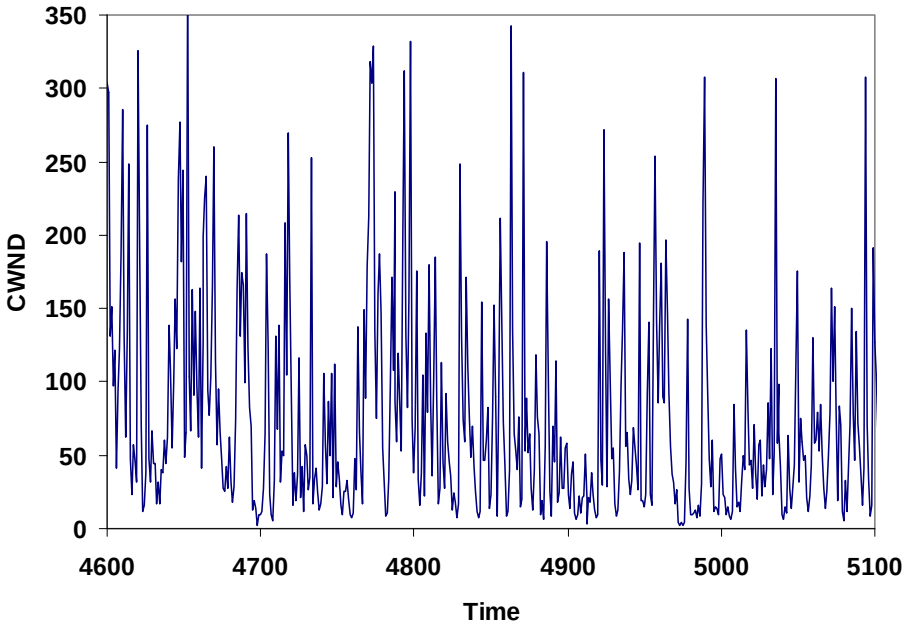

Figure 6-60. Change in Congestion Window (packets) under FAST for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

The rapid oscillatory behavior of FAST results in numerous packet losses, which leads to a larger rate of congestion window increase (as shown in Figs. 6-23, 6-32 and 6-44) and to a higher retransmission rate (as shown in Figs. 6-25, 6-34 and 6-46). The higher loss rate also causes a higher SYN rate (as shown in Fig. 6-56), which leads to a larger number of flows pending in a connecting state (as shown in Figs. 6-27, 6-37 and 6-49) because flows take longer to connect. Flows also take longer to complete because a larger number of packets must be retransmitted. This effect can be seen in Figs. 6-24, 6-

26, 6-33 and 6-45, which show that FAST flows have a significantly lower completion rate. The net effect of a lower completion rate appears in Fig. 6-55, which shows that FAST completes many fewer flows (than other algorithms) over a 25-minute period of network operation. A lower rate of flow completions also means that more flows can be active simultaneously in congested locations in the topology. See, for example, Figs. 6-36 and 6-48. As a result, the average goodput will be lower for flows transiting congested areas, as shown in Figs. 6-35 and 6-47.

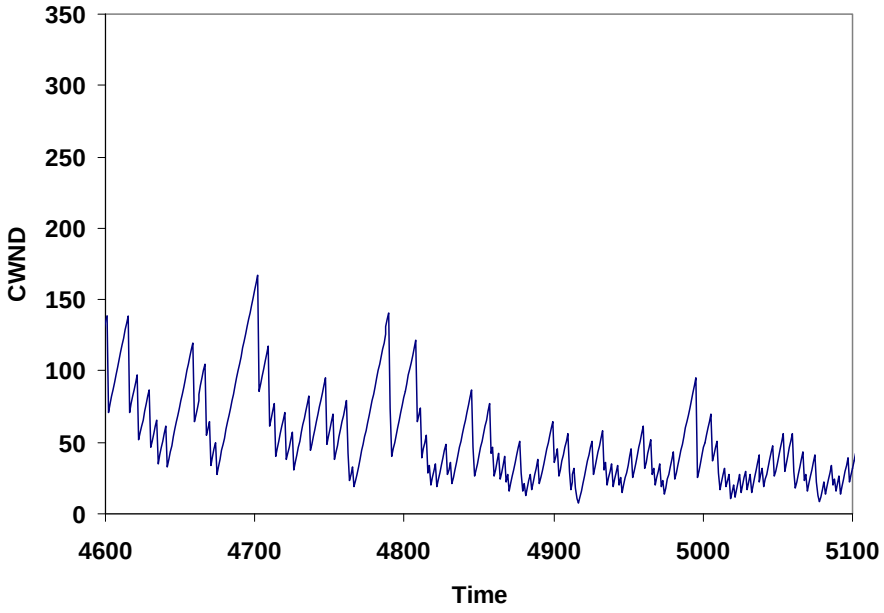

Figure 6-61. Change in Congestion Window (packets) under TCP Reno for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

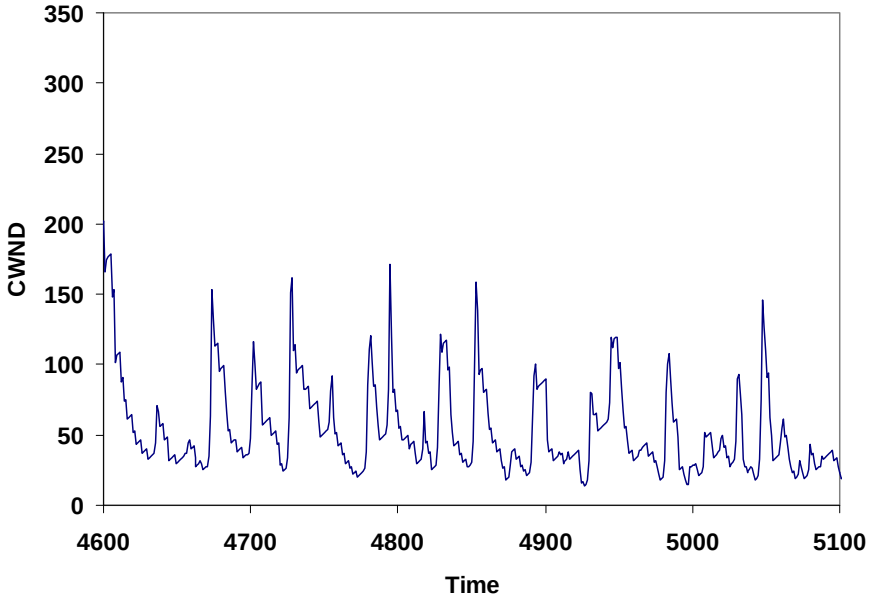

Figure 6-62. Change in Congestion Window (packets) under BIC for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

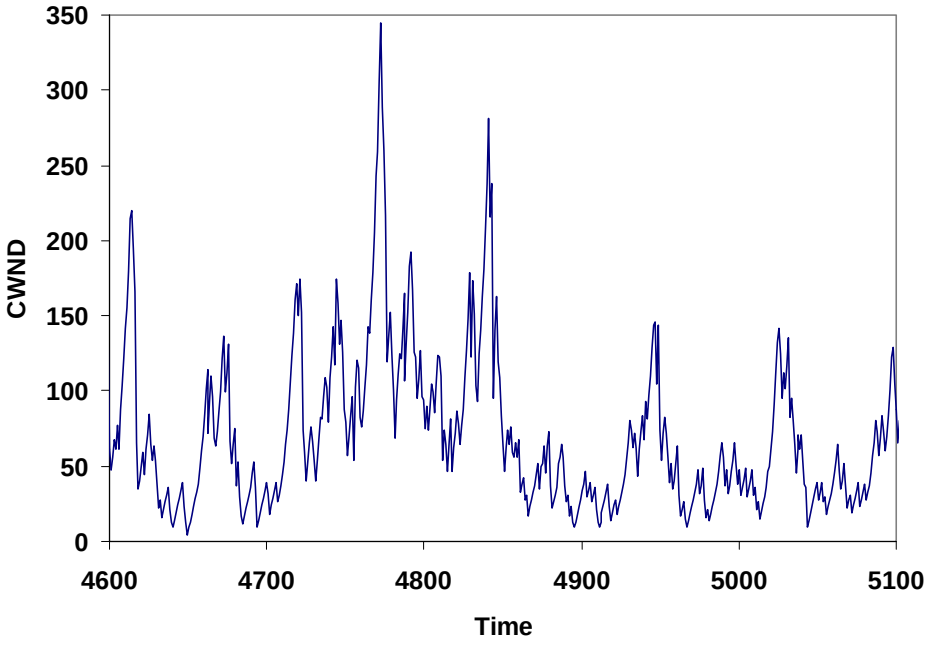

Figure 6-63. Change in Congestion Window (packets) under CTCP for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

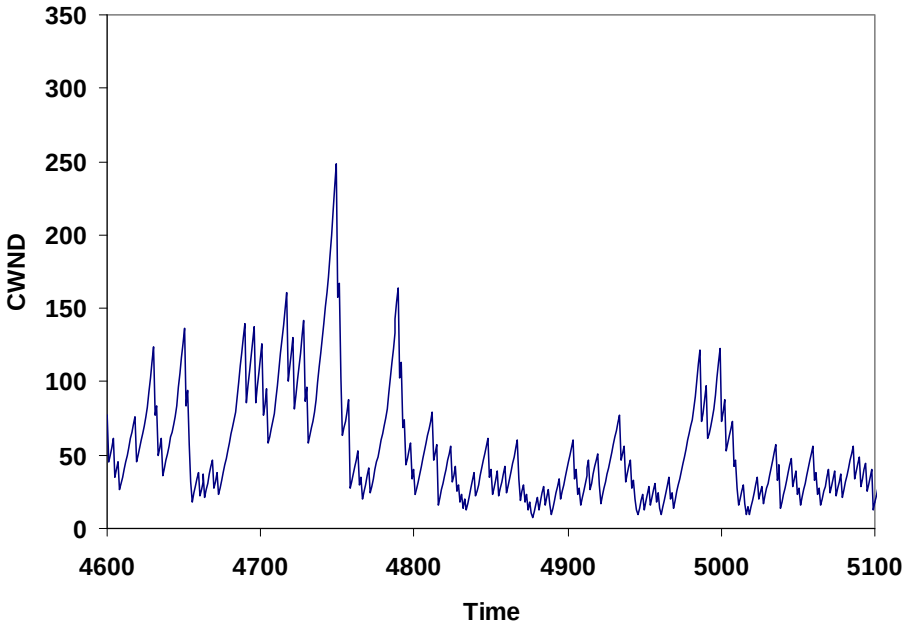

Figure 6-64. Change in Congestion Window (packets) under HSTCP for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

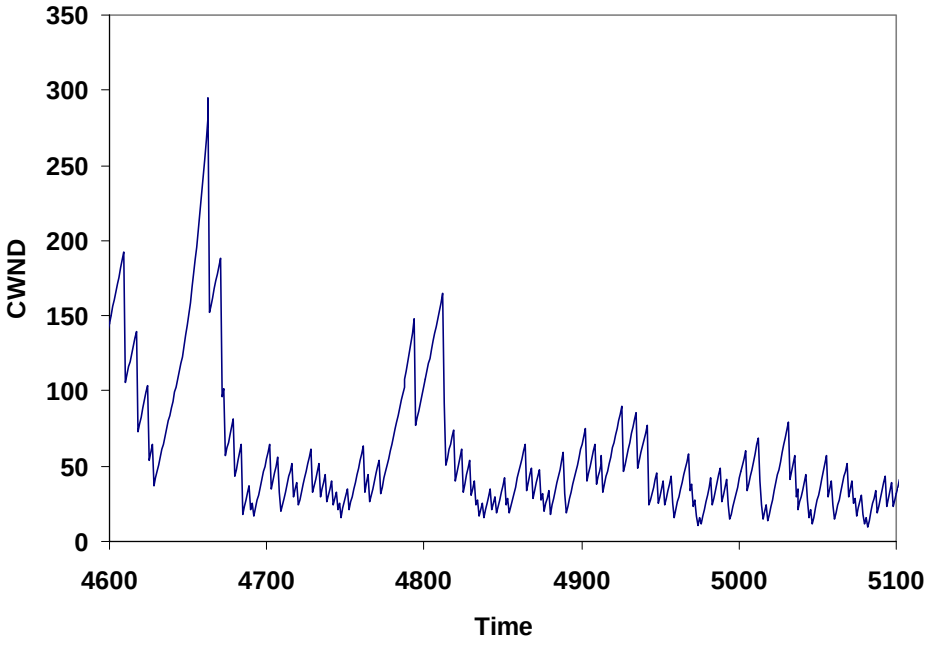

Figure 6-65. Change in Congestion Window (packets) under HTCP for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

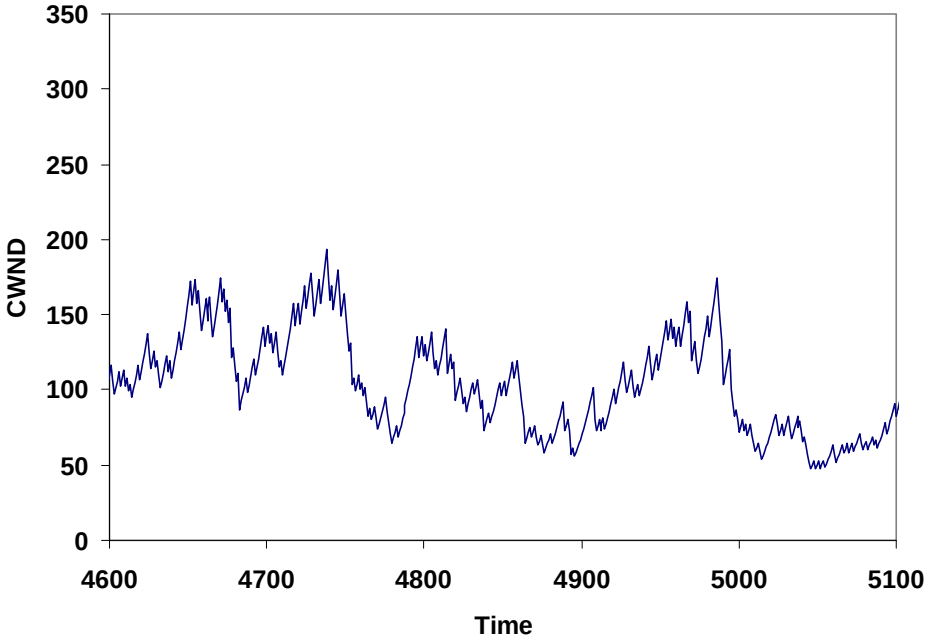

Figure 6-66. Change in Congestion Window (packets) under Scalable TCP for Long-Lived Flow L2 during 500 Measurement Intervals (200 ms each) within TP2 under Condition 21

In summary, a large network with many simultaneously active flows can induce congestion at various times and locations within the topology. When congestion is sufficient to induce losses, flows using the FAST algorithm can enter a rapid oscillatory behavior that exacerbates congestion. As a result, the network can exhibit a higher overall loss rate with consequent increase in retransmissions. Flows can take longer to connect and complete. The number of flows completed in such a network can be significantly reduced over long time spans. Should FAST be deployed throughout a network, typical Web-browsing users could experience lower average goodput on flows transiting through congested areas. These findings are limited to cases where all users on the network: (a) adopt FAST and (b) FAST is configured as discussed in Sec. 5.2.3 with fixed αF = 200. In Chapter 7 we also investigate the case of FAST configured with α-tuning enabled.

6.5.3 Finding #3

Under certain conditions, CTCP (algorithm 2) can drive congestion window size to substantially higher values than the other congestion control algorithms we simulated. In our experiment, this behavior arose during TP3, as shown in Fig. 6-50, which analyzes average congestion window size. Detailed examination of the relevant time series revealed that this increase in congestion window size can be attributed solely to DD flows.

Recall that during TP2 jumbo file transfers were initiated on DD flows, which introduced substantial congestion within directly connected access routers. At the onset of TP3 no further jumbo transfers are initiated and congestion eases as residual jumbo transfers complete. During this easing period, the congestion window on DD flows can increase – the rate of increase depends upon the level of congestion created during TP2. For example, Fig. 6-67 plots, for six congestion control algorithms, the increase in average congestion window for DD flows during TP3 under condition 12.

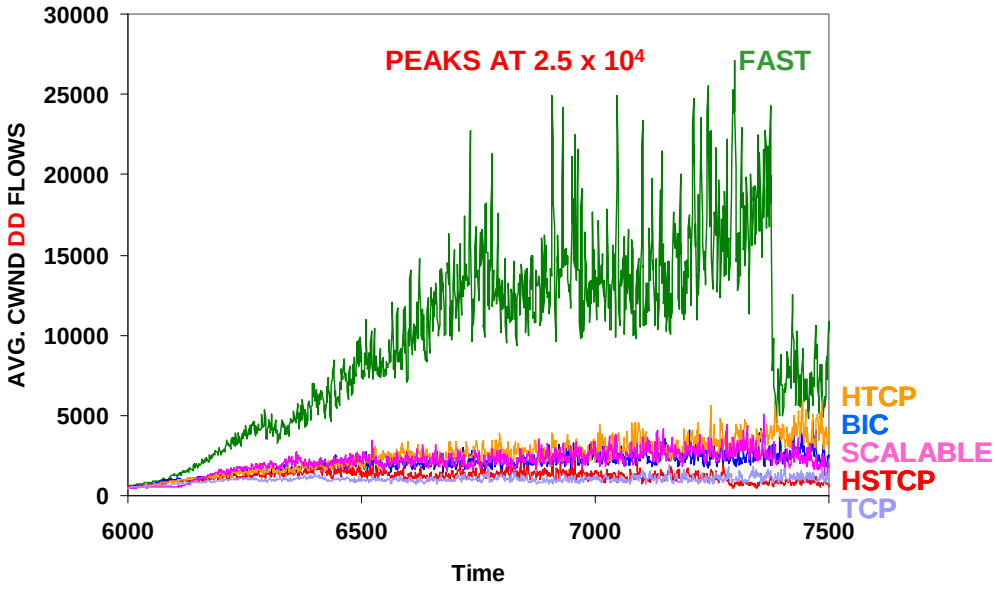

Figure 6-67. Average Congestion Window Size (packets) of DD Flows during TP3 (spanning 1500 200 ms measurement intervals) under Condition 12 for BIC, FAST, HSTCP, HTCP, Scalable TCP and TCP Reno

Fig. 6-67 shows that five of the congestion control algorithms provide a linear increase (with a small slope) in average congestion window size, up to a maximum of about 4 x 10^3 packets. The increase for FAST, which also appears approximately linear but with larger slope, peaks at around 25 x 10^3 packets. The situation for CTCP is much different, as shown in Fig. 6-68, where under the same conditions the average congestion window size increases exponentially, reaching a peak of about 1 million packets.

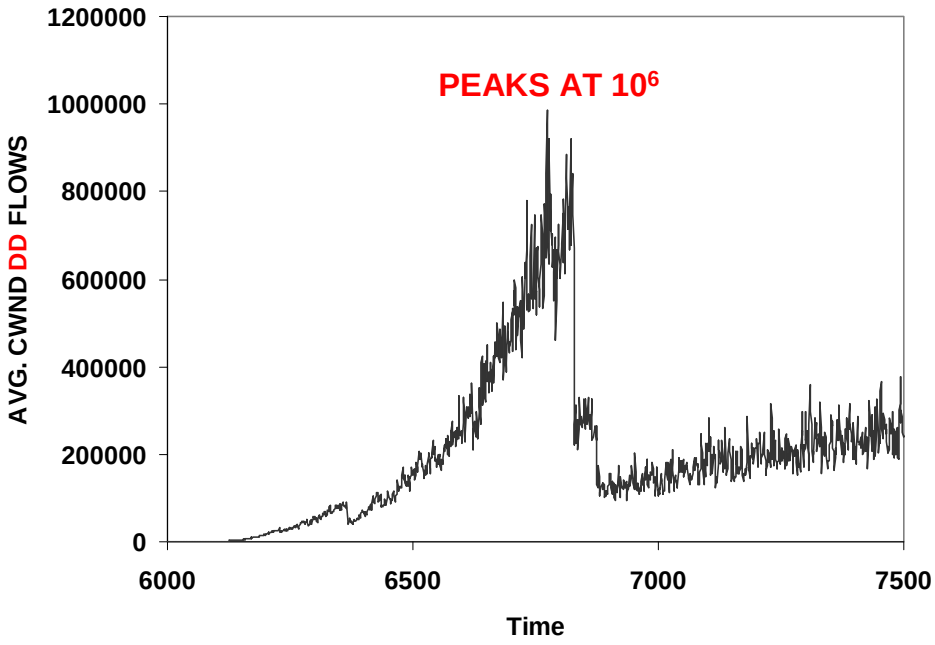

Figure 6-68. Average Congestion Window Size (packets) of DD Flows during TP3 (spanning 1500 200 ms measurement intervals) under Condition 12 for CTCP

To understand this distinctive behavior we must revisit the window management algorithm used by CTCP. Specifically, we are interested in equation (17) from Sec. 5.2.2. We repeat the equation here for convenience.

$$
\begin{aligned}
\textbf{every}(SRTT_C) \equiv\ & E_C \leftarrow \frac{cwnd}{minRTT_C} \\
& A_C \leftarrow \frac{cwnd}{SRTT_C} \\
& D_C \leftarrow (E_C - A_C) \times minRTT_C \\
& \textit{if}\ \ CD_C = \textbf{true} \\
& \quad dwnd \leftarrow \min\left[0, cwnd \times (1 - \beta_C) - \frac{cwnd}{2}\right] \\
& \quad CD_C \leftarrow \textbf{false} \\
& dwnd \leftarrow dwnd + \min\left(0, \alpha_C \times cwnd^{K_C} - 1\right) \ \ \textit{if}\ \ CD_C = \textbf{false} \wedge D_C < \gamma_C \\
& dwnd \leftarrow \min\left[0, dwnd - (\zeta_C \times D_C)\right] \ \ \textit{otherwise} \\
& cwnd \leftarrow \max(\textbf{int\_max}, cwnd + dwnd)
\end{aligned}
\qquad (17)
$$

Equation (17) specifies a periodic algorithm used by CTCP to adjust the delay window as needed every round-trip time (RTT). The CTCP delay window augments the congestion window. The highlighted line in (17) shows that CTCP will increase the delay window exponentially when no congestion has been detected and the actual congestion window is within (γ_C =) 30 packets of the expected congestion window. In other words, if there is no congestion and the actual window is close to what is expected from previous measurements, then perhaps the window can be increased because congestion is easing.

In our scenario, DD flows that start during TP2 are likely to face stiff congestion, which implies that the initial minimum RTT for these flows will be somewhat high. At the onset of TP3, congestion eases as residual jumbo transfers complete. Easing congestion causes measured SRTT (smoothed RTT) to fall, thus minimum RTT recorded on these flows will be driven down. As a result, the minimum RTT and the measured SRTT will be identical, or nearly so. Thus, the difference in expected and actual congestion window, as computed by the CTCP algorithm, will be around zero. As SRTT continues to fall, and minimum RTT falls with it, the highlighted line in (17) will be executed during each RTT. Naturally, this leads to an exponential increase in the congestion window.

Under our scenario, this exponential congestion window increase has little practical implication because a source cannot transmit faster than its maximum interface speed (or the maximum interface speed of a slower receiver). Note, however, that under easing congestion and no packet losses the CTCP congestion window continues to increase exponentially until a transfer completes even though the source is unable to increase its transmission rate. This situation is analogous to initial slow start, which also increases the congestion window exponentially. Given an arbitrarily high initial slow-start threshold, a large file transfer that proceeds without packet loss will likely remain in initial slow start until the transfer completes. Under these circumstances the congestion window grows exponentially even though the source is unable to increase its transmission speed beyond a physical maximum. In theory, a CTCP flow (or any flow operating within initial slow start) could achieve a very high window (e.g., millions of packets). A subsequent loss on a flow that has achieved such a high window could require many losses to reduce the window (by 50 % per loss) to a point where the transmission rate is throttled sufficiently to respond to the congestion signal. The possibility for such an outcome suggests that some practical upper limit should be placed on delay window size, though most TCP implementations place an upper limit on the size of the congestion window.

6.5.4 Finding #4

Focusing on longed-lived flows reveals several points of interest. First, during TP1 all congestion control algorithms showed nearly identical goodput on the three long-lived flows – the less the congestion, the closer the goodput. This occurs because the initial slow-start threshold was set to an arbitrarily high value. During TP1, when the long-lived flows commenced amid a background of Web traffic, initial slow-start was typically able to carry the long-lived flows to the maximum achievable transmission rate (960 Mbps). Since all congestion control algorithms adopted identical initial slow-start procedures, this finding should not be surprising. (In Chapter 7 we investigate effects from a lower initial slow-start threshold.)

When heavy congestion strikes, as jumbo file transfers commence in TP2, algorithm 6 (Scalable TCP) exhibited a tendency to provide higher goodput on the long-lived flows than did the other congestion control algorithms. This comparative advantage of Scalable TCP tended to increase with increasing propagation delay and decrease with increasing congestion. Detailed analyses of long-lived flows (e.g., 6-38 and 6-39) did not find the goodput advantage of Scalable TCP to be statistically significant (5 %) under many conditions, but this appears influenced by the wide range of goodputs exhibited. The reason that Scalable TCP tended to provide higher goodputs on long-lived flows during TP2 is that newly arriving flows have more difficulty claiming their share of bandwidth when the competing flows are all using the Scalable congestion avoidance algorithm. This difficulty was illustrated in Chapter 5 (see Figs. 5-31 to 5-33). Further evidence of this effect is shown in Fig. 6-69, which compares algorithms 3 (FAST) and 6 (Scalable TCP) with respect to decrease in congestion window size for all three long-lived flows at the onset of TP2 under condition 27 (light-to-moderate congestion).

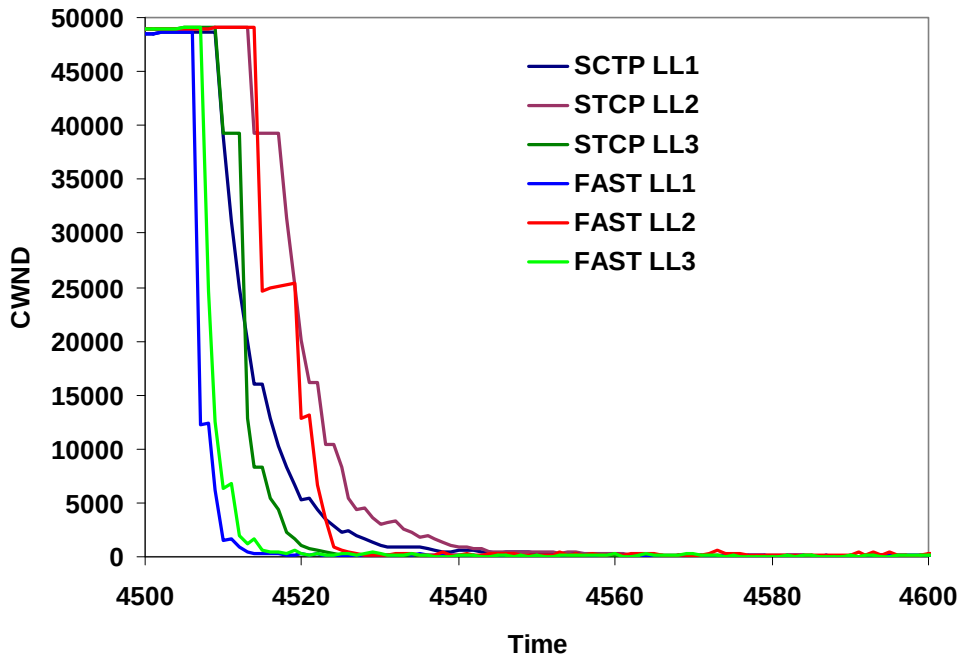

Figure 6-69. Comparing Congestion Window Size (packets) of Scalable TCP (STCP) and FAST with respect to Falling Congestion Window for Three Long-Lived Flows during the First 100 measurement intervals (200 ms each) of TP2 under Condition 27

The comparative advantage of Scalable TCP vanished in TP3 as congestion abated and most of the alternate congestion control algorithms recovered well. The most notable effect for long-lived flows during TP3 is that standard TCP lags in recovering peak goodput. This finding is as expected. In fact, the sluggishness shown by standard TCP when recovering from congestion provides motivation for researchers to propose alternate congestion control algorithms.

6.5.5 Tendencies

Given that algorithm 3 showed several distinctive behaviors, we discarded its response data and then conducted our detailed analyses a second time on the remaining congestion control algorithms. As already demonstrated, the remaining algorithms could not be distinguished using tests for statistically significant differences. On the other hand, we noted earlier that algorithms 1 (BIC) and 6 (Scalable TCP) showed some tendency to behave similarly to each other and distinctly from other algorithms. Based on our supplementary analyses, we identified some tendencies that, though they cannot be considered findings, might illuminate differences among alternate congestion control algorithms (excluding FAST). The tendencies we identify are from rather small differences in relative and absolute effect. If nothing else, these tendencies help to explain why BIC and Scalable TCP clustered together under many conditions.

The first observation to note is that BIC and Scalable TCP behaved more similarly under congested conditions. In part this is due to the fact that all the algorithms tended to behave similarly under uncongested conditions, so behavioral distinctions appeared only with increasing congestion. We note that BIC and Scalable TCP tended to push more packets through the network, while completing fewer flows. Algorithms 5 (HTCP) and 7 (TCP) exhibited the opposite tendencies (i.e., fewer packets pushed through and more flows completed). One factor affecting these trends is that BIC and Scalable TCP tended to complete fewer NN flows, which were most numerous and also had the lowest potential for goodput, while CTCP (algorithm 2), HTCP (algorithm 5) and TCP tended to complete more of such flows. From this, we conclude that BIC and Scalable TCP showed a tendency to push more packets through the network for flows that could achieve higher goodputs (e.g., long-lived flows and other flows over fast and very fast paths). Another way to look at this is that (in this experiment) CTCP, HTCP and TCP provided fairer bandwidth sharing under heavy congestion than either BIC or Scalable TCP. This confirms differences demonstrated earlier in Sec. 5.4. These differences led to some distinctions in network-wide behavior.

The average congestion window size tended to be higher under BIC and Scalable TCP; this higher average was due largely to bigger windows on advantaged flows. Pushing more packets into the network also led BIC and Scalable TCP to have higher retransmission rates, larger queuing delays and higher SYN rates (along with more flows pending in the connecting state). While not statistically significant in this experiment, the differences we highlight provide some tendencies that might separate BIC and Scalable TCP qualitatively from the other congestion control algorithms.

6.6 Conclusions

In this section we described an experiment comparing alternate congestion control algorithms deployed in a large, fast network with typical Web traffic, a few long-lived flows and a period of large file transfers between selected locations, followed by easing congestion. The specific experiment design we described follows a general approach that we will apply repeatedly in subsequent chapters to compare congestion control algorithms under various circumstances. In addition, we defined a data analysis approach that allowed us to find key differences, where they existed, among various congestion control algorithms. We applied the experiment design and data analysis approaches to compare seven alternate congestion control algorithms within a simulated network. In a

given simulation, all sources in the network used the same congestion control algorithm. This unrealistic assumption of homogeneity aided our analysis and allowed us to identify differences among the algorithms we compared. We subjected each congestion control algorithm to the same 32 conditions, which provided a range of congestion levels.

We demonstrated that (aside from FAST) under our scenario and conditions the alternate congestion control algorithms exhibited indistinguishable macroscopic behavior and modest differences in user experience. We showed that the behaviors were more similar in uncongested conditions. We also explained why this was the case. We showed that FAST can exhibit distinctive, undesirable network-wide behavior, which grows more distinctive under increasing congestion. We described the root cause of this distinctive behavior, and we argued that deploying FAST throughout the Internet might entail significant risk. We identified an element of the CTCP delay-window adjustment algorithm that can lead to an exponential increase in congestion window under particular circumstances associated with easing congestion. We showed that Scalable TCP tends to retain a higher congestion window for a longer time on long-lived flows under periods of increasing congestion. We identified some tendencies for BIC and Scalable TCP to provide higher goodputs on large flows with high available bandwidth, while providing lower quality of service on more numerous, typical flows with lower available bandwidth.

In the next section, we repeat the current experiment while changing only a few parameters. We scale down the network by one order of magnitude in size (number of sources and receivers) and speed. We intend to show that a scaled-down simulation, which requires much less computing resources, can reveal findings similar to a larger simulation. We also lower the initial slow-start threshold to a relatively small number of packets. Decreasing the initial slow-start threshold will allow flows to enter congestion avoidance earlier. More frequent activation of congestion avoidance under low congestion might reveal additional information about differences among congestion control algorithms. Finally, we add the α-tuning variant of FAST as an eighth congestion control algorithm to consider. Here, we seek to understand whether activating α-tuning might lead to improved behavior for FAST.