Proceedings of the 2011 Pressure Vessels & Piping Division (PVPD) Conference
ASME 2011 PVP-MF20
July 17-22, 2011, Baltimore, Maryland, USA

PVP 2011-57382 (6.1)

PREDICTING MACROSCOPIC DYNAMICS IN LARGE DISTRIBUTED SYSTEMS

K. L. Mills & J. J. Filliben
NIST
Gaithersburg, MD, USA

D.-Y. Cho
NIH
Bethesda, MD, USA

E. J. Schwartz
Carnegie-Mellon University
Pittsburgh, PA, USA

ABSTRACT

Society increasingly depends on large distributed systems, such as the Internet and Web-based service-oriented architectures deployed over the Internet. Such systems constantly evolve as new software components are injected to provide increased functionality, better performance and enhanced security. Unfortunately, designers lack effective methods to predict how new components might influence macroscopic behavior. Lacking effective methods, designers rely on engineering techniques, such as: analysis of critical algorithms at small scale and under limiting assumptions; factor-at-a-time simulations conducted at modest scale; and empirical measurements in small test beds. Such engineering techniques enable designers to characterize selected properties of new components but reveal little about likely dynamics at global scale.

In this paper, we outline an approach that can be used to predict macroscopic dynamics when new components are deployed in a large distributed system. Our approach combines two main methods: scale reduction and multidimensional data analysis techniques. Combining these methods, we can search a wide parameter space to identify factors likely to drive global system response and we can predict the resulting macroscopic dynamics of key system behaviors.

We demonstrate our approach in the context of the Internet, where researchers, motivated by a desire to increase user performance, have proposed new algorithms to replace the standard congestion control mechanism. Previously, the proposed algorithms were studied in three ways: using analytical models of single data flows, using empirical measurements in test beds where a few data flows compete for bandwidth, and using simulations at modest scale with a few sequentially varied parameters. In contrast, by applying our approach, we simulated configurations covering four-tier network topologies, spanning continental and global distances, comprising routers operating at state-of-the-art speeds and transporting more than 10^5 simultaneous data flows with varying traffic patterns and temporary spatiotemporal congestion. Our findings identify the main factors influencing macroscopic dynamics of Internet congestion control, and define the specific combination of factors that must hold for users to realize improved performance. We also uncover potential for one proposed algorithm to cause widespread performance degradation. Previous engineering studies of the proposed congestion control algorithms were unable to reveal such essential information.

1 INTRODUCTION

Every month millions of users update their computers with new software components, designed to add new functionality or to address security flaws. Could one of these updates lead to widespread performance degradation? If so, could such performance degradation be predicted a priori? As we show in this paper, loading new congestion control algorithms into user computers could potentially degrade Internet performance, and, using methods we outline here, it should be possible to predict such degradations prior to their occurrence.

The focus of our investigation concerns network congestion control procedures embedded in the transmission control protocol (TCP), which is used by all computers on the Internet. TCP congestion control procedures increase and decrease transfer rate on data flows, based on feedback measurements, such as losses and delays. The rate adjustments made on individual data flows influence the measurements seen by other flows, which adjust their own rates, and so on, leading to complex interactions that drive a global pattern of network congestion. To effectively understand and predict macroscopic behavior in such systems requires one to model large, diverse, configurations, covering many users, data flows, network routes and traffic types. Heretofore, simulating such systems has proven infeasible. By combing scale-reduction methods

with multidimensional data analysis techniques, we successfully simulated a large, diverse network, allowing us to understand key properties influencing the operation of Internet congestion control procedures. We were also able to compare and contrast various proposed replacements for standard TCP, and to identify a cautionary note with regard to widespread deployment of one of the proposed replacements. The information in this paper is abstracted from our study of Internet congestion control algorithms [1].

The paper is organized in six main sections. In Sec. 2, we briefly introduce the domain of Internet congestion control. The purpose of this short introduction is to provide sufficient grounding to follow Sec. 3, where we use some domain concepts to outline the main scale-reduction methods and multidimensional data analysis techniques we adopted for our study. In Sec. 4, we explain several replacements for standard TCP. These replacements have been proposed by various researchers in an effort to provide users with higher transmission rates as the speed of the Internet increases. We close Sec. 4 with a synopsis of previous methods used to compare and evaluate proposed TCP replacements. In Sec. 5 we describe MesoNet, a 20-parameter model for simulating Internet congestion control procedures. Sec. 5 also summarizes a MesoNet sensitivity analysis, revealing significant behavioral dimensions and identifying influential parameters that drive those behaviors. In Sec. 6, we present key results from five simulation experiments we conducted to compare proposed TCP replacements. In Sec. 7 we discuss the implications of our experiment results. We conclude in Sec. 8.

2 INTERNET CONGESTION CONTROL

Computers attached to the Internet rely on the transmission control protocol (TCP) to send each flow of related data reliably as a sequence of segments. For example, a TCP flow may contain the data objects to be displayed by a Web browser when a user clicks on a Web link. In addition to ensuring reliability and sequencing of data segments on a flow, TCP also contains congestion control procedures that adapt the rate of data transfer to the conditions experienced on the flow. TCP reduces the transmission rate when congestion is detected, and increases rate when congestion is absent.

A typical TCP flow evolves through three phases: connection, transfer and close. For purposes of congestion control, we limit our discussion to the connection and transfer phases. Fig. 1 gives a high-level view of these two phases. During the connection phase, a source attempts to establish contact with an intended receiver; inability to establish contact results in connection failure, which prevents data from flowing between source and receiver, so connection establishment provides one form of congestion control implemented by TCP. During the transfer phase, a source sends data segments on the flow until the required number has been received successfully. A receiver signals receipt of data segments by sending acknowledgments (ACKs) to the source. By sending duplicate ACKs, a receiver may also indicate failure to receive specific segments, which the source must then retransmit. Further, a sender may fail to receive ACKs, which requires the sender to raise a timeout and to retransmit unacknowledged data.

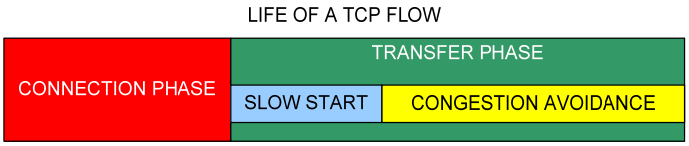

Figure 1: Main phases and congestion control procedures in the life of a TCP flow

During the transfer phase, congestion control procedures determine when a source may send data segments to a receiver. At any given time, a source may send a prescribed number of segments (known as the congestion window, or *cwnd*) prior to receiving an ACK. Thus, the size of the *cwnd* controls the rate of transmission on a flow. Using TCP slow start procedures, a source increases a flow's *cwnd* exponentially from a small initial value until either a loss is detected or until the *cwnd* reaches a threshold, known as the initial slow start threshold, or *sst*. If the *sst* is reached, the source enters congestion avoidance, subsequently increasing the *cwnd* more slowly, at a linear rate. If a segment is lost, then the *cwnd* is reduced in half and then increased linearly until another segment is lost, after which the *cwnd* is reduced in half again, and so on. This algorithm is known as additive increase, multiplicative decrease (AIMD) [14]. The resulting saw-tooth pattern (illustrated in Fig. 2) in the *cwnd* induces a corresponding variation in the rate of transmission on a flow.

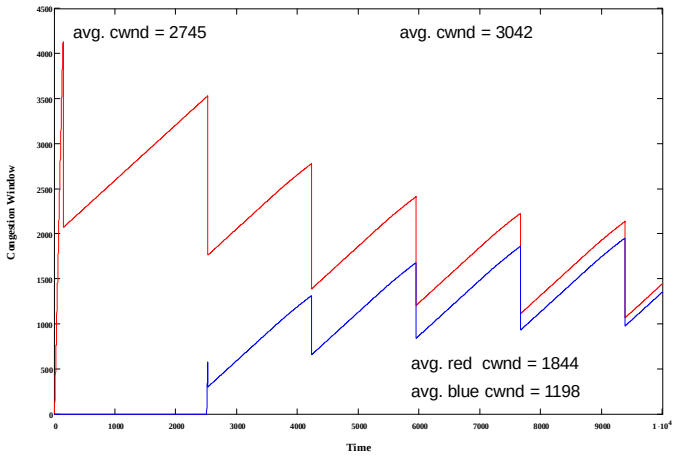

Figure 2: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two TCP flows sharing a bottleneck path with 162 ms round-trip time (*rtt*) – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

Fig. 2 shows the temporal evolution of *cwnd* for two competing TCP flows, each sourced from a separate computer, transiting a shared bottleneck path in a network. The path is a

bottleneck because it can support only 21 segments per ms (about 252 Mbps – Megabits per second – assuming 1500 byte segments, which we do throughput the paper), while each computer is capable of transmitting 80 segments per ms (about 960 Mbps). The round-trip time (*rtt*) on the bottleneck path is 162 ms, so the path can hold 3402 segments, while 680 additional segments can be held in buffers contained within routers on the path. Thus, the total number of outstanding segments can be 4082, after which segments are lost. TCP congestion avoidance procedures are designed so that flows sharing a bottleneck path use local decision procedures to adjust transmission rates so that each flow will achieve an equal share of available path capacity. Notice in Fig. 2 that the blue and red flows are moving toward such equilibrium.

Fig. 2 shows that one (red) flow, operating on the path before time 2500, has average *cwnd* of about 2745, which implies a throughput of only 80 % of the path capacity. When a second flow shares the path, aggregate throughput increases to 89 %. Lost capacity stems from the fact that TCP congestion avoidance procedures reduce the *cwnd* in half following a loss and then increase transmission rate linearly until the next loss is generated, and so on.

Lost throughput on a single TCP flow can become quite large as the product of the path capacity and round-trip time increases. For example, researchers made empirical measures [15] on a 1 Gbps (Gigabits per second) path between Chicago and Dublin, finding that the average throughput was only 218 Mbps, about 20 % of capacity. This finding, that average throughput on TCP flows is likely to decrease significantly as the capacity of the Internet backbone increases, motivated researchers to propose replacements for the standard TCP congestion avoidance procedures, so that users might experience increased throughput as Internet capacity increases. In this paper, we compare standard TCP with six proposed TCP replacements, as listed in Table 1. We explain the proposed TCP replacements in Sec. 4.

Table 1: Identifiers, acronyms and names of congestion avoidance algorithms compared in this paper

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

An entire Internet of individually adapting TCP flows makes a complex system, where individual flows adapt their transmission rate based on perceived congestion, which changes the pattern of congestion, which causes flows to readapt their transmission rates, and so on. This is why serious studies of Internet congestion control procedures must consider the macroscopic dynamics emerging from interactions among a large number of varying flows transiting across a large network.

Predicting macroscopic behavior in a large network requires mathematical models exhibiting a substantial number of input parameters and output variables. Such models can be considered abstractly in functional form, as shown in Fig. 3, where a set of parameters (e.g., $x_1 \ldots x_p$), each assigned a value from within a range (e.g., $1 \ldots \ell$), is input to a function (e.g., f) to yield a set of response variables (e.g., $y_1 \ldots y_z$). Varying the values assigned to each parameter results in different response values, which may be analyzed and plotted. The p parameters and associated range ($1 \ldots \ell$) of valid input values compose a model's stimulus state-space, while the z output variables compose a model's response state-space.

$$y_1, \ldots, y_z = f(x_{1|[1,\ldots,\ell]} \ldots, x_{p|[1,\ldots,\ell]})$$

Response State-Space — Stimulus State-Space

Figure 3: Functional model of a large network

To be more concrete, in Table 2 we introduce the 20 input parameters for a network model called MesoNet [33]. We describe MesoNet parameters more fully in Sec. 5. As shown in Table 2, MesoNet parameters can be classified into five categories: network configuration, user behavior, sources and receivers, protocols, and simulation control and measurement. The four network configuration parameters define a topology (X4) of network routers and communication links, spanning a geography that defines propagation delay (X2) for segments flowing over network links. The network routers transmit segments at a specific rate (X1). When awaiting transmission, segments may be queued in buffers of specific size (X3).

Attached to the topology are some number (X12) of sources and receivers, which can be distributed with different densities (X13 and X14) throughout the topology. Sources inject segments into the network at a specified maximum rate (X11). The behavior of sources is controlled by users who transfer Web objects of specific size (X5), while occasionally transferring files of larger size (X6). Between transfers, users think for some period of time (X7). During transfers, users may exhibit limited patience (X8), aborting transfers that take too long or progress too slowly. During specified time periods (X9) the size of user file transfers can be increased to create localized spatiotemporal congestion within a topology. In addition, specific long-lived transfers (X10) can be initiated.

Long-lived flows entail users transferring as much information as possible during a specific simulation.

Three protocol parameters influence MesoNet's congestion control procedures. One parameter (X16) sets the initial *cwnd*, while another (X17) sets the initial *sst*. The remaining parameter (X15) gives the probability that a source uses any of the congestion avoidance algorithms identified in Table 1.

Table 2: MesoNet Parameters

Category	Identifier	Name
Network Configuration	X1	Network Speed
	X2	Propagation Delay
	X3	Buffer Provisioning
	X4	Topology
User Behavior	X5	Web Object Size for Browsing
	X6	Proportion & Size of Larger Files
	X7	Think Time
	X8	Patience
	X9	Selected Spatiotemporal Congestion
	X10	Long-lived Flows
Sources & Receivers	X11	Source & Receiver Interface Speeds
	X12	Number of Sources & Receivers
	X13	Distribution of Sources
	X14	Distribution of Receivers
Protocols	X15	Congestion Control Procedures
	X16	Initial Congestion Window
	X17	Initial Slow Start Threshold
Simulation Control & Measurement	X18	Measurement Interval Size
	X19	Simulation Duration
	X20	Startup Pattern

The remaining MesoNet parameters control selected aspects of simulation and measurement, including how long a simulation executes (X19) and the probability and duration (X20) with which sources think prior to making an initial file transfer. The remaing parameter defines the interval (X18) at which MesoNet output variables are sampled.

While MesoNet can report hundreds of output variables, here we introduce a sample subset[1] in two categories: (1) 16 responses (Table 3) characterizing macroscopic network behavior and (2) six responses (Table 4) characterizing throughput of various flow classes. We arbitrarily label the responses y1 through y22, which we use below in Sec. 3, when discussing our method to reduce the response state-space.

The 16 macroscopic responses measure the temporal evolution of various network wide properties, such as number sources transmitting (y1), data segments entering (y3) and leaving (y4) the network, connection failures (y9), average fraction of segments retansmitting (y10), average *cwnd* (y11) and average *rtt* (y15). The remaining six responses report the average number of segments per second on flows transiting between pairs of routers of various speeds: D-class (very fast), F-class (fast) and N-class (typical). In general, DD flows, transiting two very fast routers, should achieve highest throughput and NN flows, transiting two typical speed routers, should achieve lowest throughput. Throughput on other flow classes (i.e., DF, DN, FF and FN) should fall between the extremes of the DD and NN flows.

Table 3: Sample Macroscopic Response Variables

Response	Definition
y1	Number of sources transmitting
y2	Proportion transmitting [y1 / all sources]
y3	Number segments entering network
y4	Number of segments leaving network
y5	Loss Rate [y4/ (y3 + y4)]
y6	Flows completing per interval
y7	Flow-completion rate [y6/(y6+y1)]
y8	Connection failures per interval
y9	Connection-failure rate [y8/(y8+y1)]
y10	Retransmission rate
y11	Congestion window (*cwnd*)
y12	cwnd increases per interval
y13	Duplicate ACKs per interval
y14	Timeouts per interval
y15	Round-trip time (*rtt*)
y16	Queuing delay

Table 4: Sample Throughput Response Variables

Response	Definition
y17	Average Segments/Second on DD flows
y18	Average Segments/Second on DF flows
y19	Average Segments/Second on DN flows
y20	Average Segments/Second on FF flows
y21	Average Segments/Second on FN flows
y22	Average Segments/Second on NN flows

3 METHODS

We adopt rigorous methods in two-main classes: scale reduction and multidimensional data analysis. Scale reduction methods guide us in limiting the number of parameter combinations that must be simulated, which can reduce significantly the time and expense required to conduct experiments. More importantly, our methods for reducing parameter combinations ensure that we vary those parameters that most significantly influence model behavior, allowing us to

[1] Elsewhere in the paper we introduce other response subsets with different arbitrarily assigned labels, beginning from y1 for each subset.

compare candidate algorithms under the most informative conditions. We also adopt scale reduction methods to limit the number of output variables we examine, while identifying essential system behaviors. Experimenters that use ad hoc selection techniques when choosing a subset of outputs variables may omit responses that characterize important model behaviors. Ad hoc selection methods may also overweight particular model behaviors, creating a skewed view of system dynamics.

Multidimensional data analysis methods enable us to identify pervasive patterns of system behavior, while also helping to find underlying causes. In many cases, the multidimensional analysis techniques we employ leverage key characteristics of the scale reduction methods we use. When combined together, our rigorous methods for scale reduction and multidimensional data analysis yield powerful insights into the behavior of large models.

3.1 Scale Reduction

For large distributed systems, mathematical models, such as the one shown in Fig. 3, present two difficult challenges. First, the stimulus state-space can take on a large size (e.g., ℓ^p), which can be infeasible to search. Second, the response state-space may be too large to examine effectively, or the response state-space may include parameters that redundantly represent similar behaviors, which will overweight the significance of those behaviors in subsequent analyses. To address these challenges, we adopt methods aimed at reducing both the stimulus state-space and the response state-space. To provide a realistic, concrete example, we consider modeling a communication network for purposes of comparing various congestion control procedures, as introduced above in Sec. 2. We begin by describing methods to reduce the stimulus state-space of a model. Then we discuss reducing the response state-space.

3.1.a Stimulus State-Space Reduction. Detailed simulation models for communication networks based on TCP and the Internet protocol (IP), or so-called TCP/IP networks [2], can require an experimenter to specify numerous parameters. Hundreds or thousands of parameters might be necessary, depending on the particulars of the simulator [e.g., 3-6] and the size of the simulated network. If we assume such a model requires us to specify 10^3 parameters and that each parameter can be specified by a 32-bit integer (i.e., has 2^{32} possible levels), then, as shown in Fig. 4, the resulting stimulus state-space would encompass about 10^{9633} parameter combinations, compared to 10^{80} estimated atoms in the visible universe. Clearly, this stimulus state-space is impossible to search fully given any present day computational capability, or even any foreseen computational capability. For practical experiments, we must somehow reduce the model to a lower dimensional stimulus state-space. We adopted a combination of techniques, as illustrated in Fig. 4. The techniques include: parameter reduction, level reduction and orthogonal fractional factorial (OFF) experiment design.

Parameter Reduction. One technique we adopt is to reduce the number of parameters (p) needed to model a network. This involves two main activities: (1) discarding parameters not germane to the focus of a particular study and (2) grouping retained parameters that can be discerned to represent aspects of the same model attribute. Conducting these activities requires significant domain expertise, and may also entail substantial trial and error. For the example discussed in this paper, we were able to construct MesoNet by first identifying 56 parameters germane to our study and then finding that 36 of those parameters could be grouped together with other parameters. See our full study for details [1]. The resulting model reduced our stimulus state-space substantially to about 10^{192}, a number of parameter combinations that is still larger than atoms in the visible universe and, thus, still infeasible to search completely with modern computers.

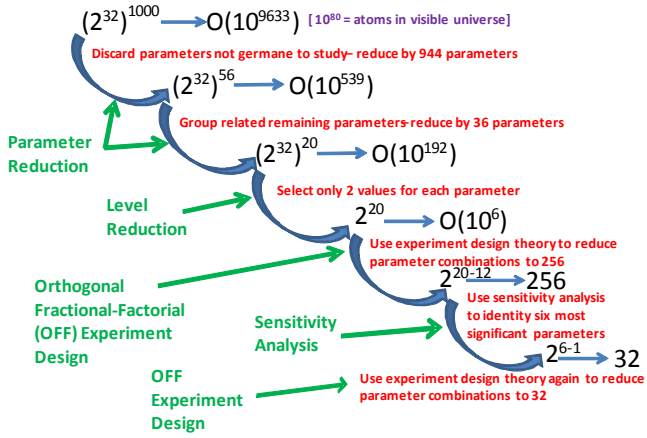

Figure 4: A sequence of techniques applied to reduce the stimulus state-space of a network model

Level Reduction. To further reduce the search space, we adopt 2-level experiment designs [7], where each parameter is assigned only two of its possible values, providing an immediate reduction in the search space to 2^p parameter combinations. Restricting parameters to only two levels has obvious limitations: only a small number of parameter values are explored and extrapolating from the results assumes a model behaves monotonically in the range between chosen levels. On the other hand, 2-level designs provide some advantages [7]: (1) requiring few runs per parameter, (2) facilitating interpretation of response data, (3) identifying promising directions for future experiments (which may be augmented with thorough local explorations), (4) fitting naturally into a sequential strategy, which supports the scientific method and (5) forming the basis for further reduction in parameter combinations through use of fractional factorial designs.

While 2-level designs reduce the number of parameter combinations to simulate, a full search of 2^p parameter combinations may still be infeasible. For example, a full

factorial experiment design with MesoNet (p = 20) would require (2^{20} =) 1,048,576 simulations. Assuming that an average MesoNet simulation requires 28 processor hours, and assuming that 48 processors are available, a full search of parameter combinations would take (2^{20} simulations x 28 processor hours/simulation over 48 processors =) about 611,670 hours, which is around 70 years. Adding processors could reduce the latency, e.g., to 3.4 years for 1000 processors or to 4 months for 10,000 processors, but the expense would remain constant, e.g., just under $3M assuming processors cost $0.10/hour. Thus the expense of a full search would prove infeasible for most researchers, which means the number of parameter combinations must be reduced further to fit within time and budget constraints.

Orthogonal Fractional Factorial (OFF) Experiment Design. Reducing the time and cost of an experiment requires adopting a fractional factorial design [7], which simulates only a 2^{p-r} subset of parameter combinations. While many experimenters adopt ad hoc techniques, such as factor-at-a-time (FAT) design [8], to select subsets of parameter combinations, orthogonal fractional factorial (OFF) theory [7] provides a principled approach to create designs, where the choice of 2^{p-r} parameter combinations is made to achieve balance and orthogonality.

In constructing an OFF design, an experimenter must ensure exploration of a sufficient number of parameter combinations to prevent confusion about the specific parameters responsible for variations in model responses. As a rule of thumb, experimenters should strive for at least a "Resolution IV" [7] design. Resolution IV designs ensure no confusion about effects[2] attributable to individual parameters and also prevent confusion about whether effects are caused by individual parameters or by interactions among parameter pairs. Further, Resolution IV designs specify precisely which parameter pairs may be confused with which other parameter pairs. Typically, confusion involving specific parameter pairs can be resolved by a domain expert. A Resolution IV design must provide a sufficient number of simulations (n) to estimate a leading constant, each parameter (p) and each pair of parameters (p choose 2), or

$$n = 1 + p + \binom{p}{2}. \quad (1)$$

For example, MesoNet (p = 20) requires at least 211 (n = 1 + 20 + 190) simulations to construct a Resolution IV design. For a 2-level design, choose the next higher power of 2 above n, i.e., 256 runs, which identifies the need for a 2^{20-12} design. Box, Hunter and Hunter [7] give an algorithm for choosing specific combinations of parameters for various 2-level designs, where the levels are encoded as a -1 and a +1. The resulting experiment design exhibits balance, i.e., each parameter has 128 -1 levels and 128 +1 levels, and orthogonality, i.e., each pair of parameters has 64 settings at each of (-1, -1), (-1, +1), (+1, -1) and (+1, +1). Balance minimizes variance in estimated effects.

Searching the MesoNet stimulus state-space with only 256 parameter combinations requires (256 runs x 28 processor hours per run/ 48 processors =) just over 149 hours, which can be completed in one week instead of the 70 years required for a full search of parameter combinations. Further, the fixed computational cost will be reduced from nearly $3M to just under $725. And, as we discuss below (Sec. 5), using these 256 simulations to conduct a sensitivity analysis allowed us to identify six key parameters that influence MesoNet behavior, which implies that subsequent experiments would require simulating only 64 parameter combinations for each congestion control algorithm compared. In fact, applying OFF experiment design a second time, we could reduce the number of parameter combinations further, for example to (2^{6-1} =) 32. Next, we consider methods to reduce the response state-space of a model.

3.1.b Response State-Space Reduction. Typical network simulation models [e.g., 3-6] can measure system response through hundreds to thousands of outputs, which might represent aspects of fewer significant underlying model behaviors. Usually, experimenters select a subset of model outputs to analyze because considering all available responses proves too time consuming or computationally infeasible. When choosing a subset of simulation outputs, experimenters using ad hoc selection techniques may omit responses that characterize important model behaviors. Further, experimenters may select outputs in a fashion that overemphasizes particular behaviors. Overweighting or underweighting significant model behaviors can yield invalid conclusions, thus some method is required to identify precisely the model outputs that correspond to each significant behavior.

Fodor [9] identifies principal components analysis (PCA) as the best linear dimension-reduction technique in terms of mean-square error, which is a typical technique to quantify differences between values implied by an estimator and the true values of the quantity being estimated. PCA transforms a set of possibly correlated variables into a smaller set of uncorrelated variables called principal components (PCs). While PCA might be the best linear-dimension reduction technique, some difficulties arise in application. Because PCs are uncorrelated variables created from a set of possibly correlated variables, PCA guarantees no obvious domain interpretation for even the top 2 or 3 PCs. Even when a reasonable domain interpretation is possible, PCs may take both positive and negative values for which experimenters cannot determine any obvious interpretation, even after establishing a meaning for a PC itself. While one can avoid using the PCs by substituting response variables in their place, selecting specific response variables to use depends upon heuristics for which there exist no definitive criteria [11]. Different heuristics can lead to different sets of response variables. To overcome these limitations, we

[2] In a 2-level experiment, an effect is the mean model response when a parameter is set to one level minus the mean response when the parameter is set to the other level.

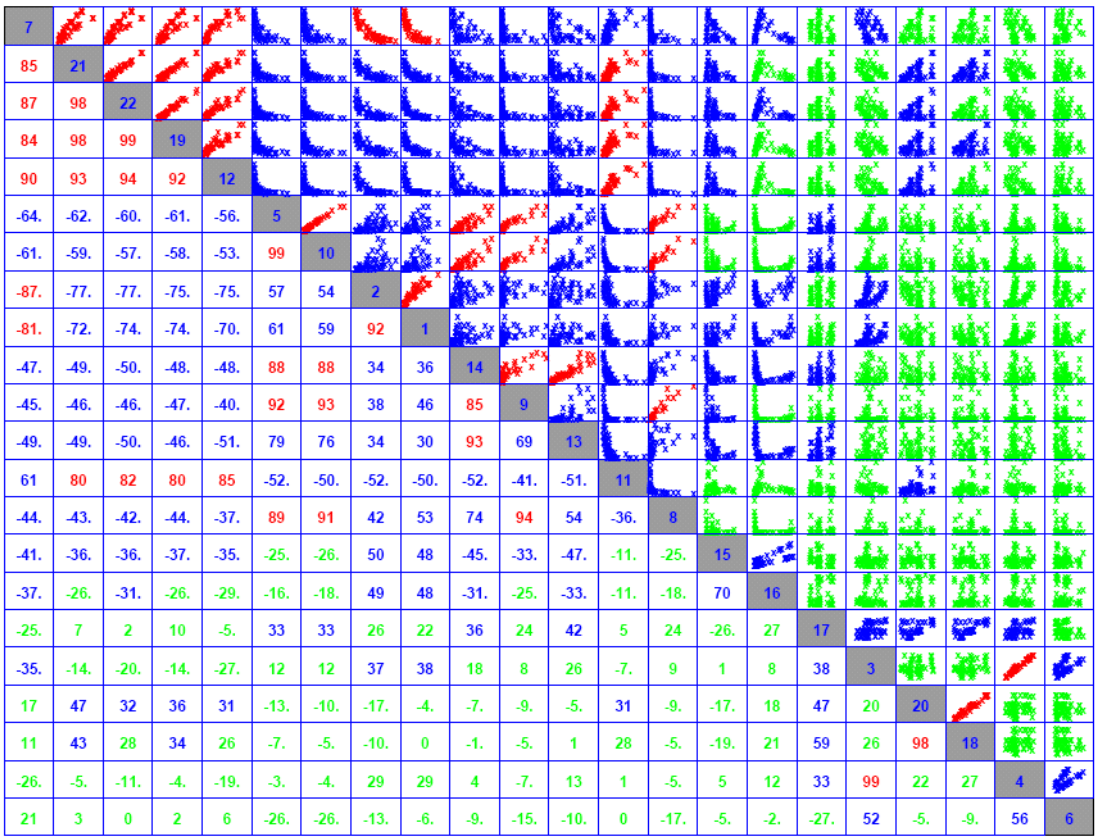

Figure 5: Pair-wise correlation matrix: scatter plots above diagonal, correlation coefficient (r x 100) below diagonal, response number on the diagonal is ordered by decreasing value of mean correlation.

developed Correlation Analysis and Clustering (CAC), an alternate approach to selecting response variables.

CAC begins with a preliminary step: computing correlation coefficients (r) for each pair of responses, and summarizing the results in a matrix, which gives pair-wise scatter plots in cells above and to the right of the diagonal and corresponding correlation coefficients (scaled x 100) below and to the left of the diagonal. Fig. 5, for example, shows 231 pair-wise scatter plots encompassing 22 responses (recall Tables 3 and 4) measured under 64 parameter combinations. We color the scaled correlations: above 80 red, below 30 green and intermediate values blue. We order the matrix by decreasing value of mean correlation for each response. For example, consider the gray cell labeled 13 (response y13) in the middle of Fig. 5. The column of cells moving upward gives scatter plots for y13 and each of the 11 responses higher on the diagonal (i.e., moving up from y9 to y7 in the top left corner). The row moving leftward gives the corresponding correlation coefficients. The row of cells moving rightward from 13 gives scatter plots for y13 and each of the 10 responses lower on the diagonal (i.e., moving down from y11 to y6 in the lower right corner). The column moving downward gives the corresponding correlation coefficients.

Given pair-wise correlations, an experimenter must decide which pairs to include in further analyses and which pairs to discard. To help with this decision, we plot a frequency distribution (Fig. 6) of the absolute values, $|r|$, of correlation coefficients for all response pairs. We use the frequency plot to select a threshold for correlations to consider further. Here, we chose $|r| > 0.65$ because the histogram shows a notable change in pattern above that value, appearing as a separate (sub) distribution of 42 pair-wise correlations centered on a different mode.

Next, we construct an index-index plot, where both the x and y axes indicate the index of corresponding responses (1 – 22). We plot a blue dot for each of the 42 $y_{i,j}$ pairs where $|r| > 0.65$. We identify the response that is correlated with the most other responses and create a self-contained subset from those responses. We then repeat the process for those responses remaining outside the subset, forming a second self-contained subset. We continue repeating the process until all responses have been allocated to a subset. Subsequently, we reorder the

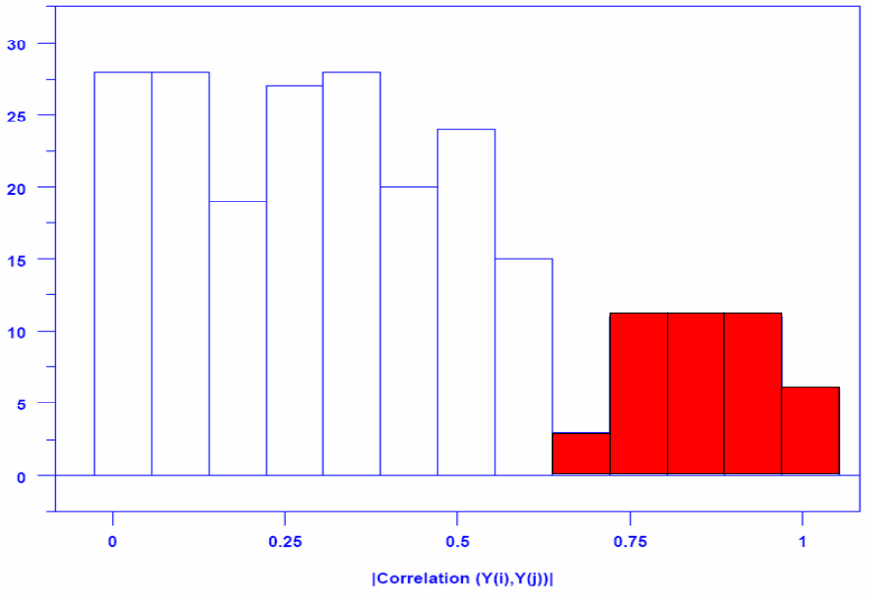

Figure 6: Frequency distribution of $|r|$ for pair-wise correlations – bins highlighted for $|r| > 0.65$

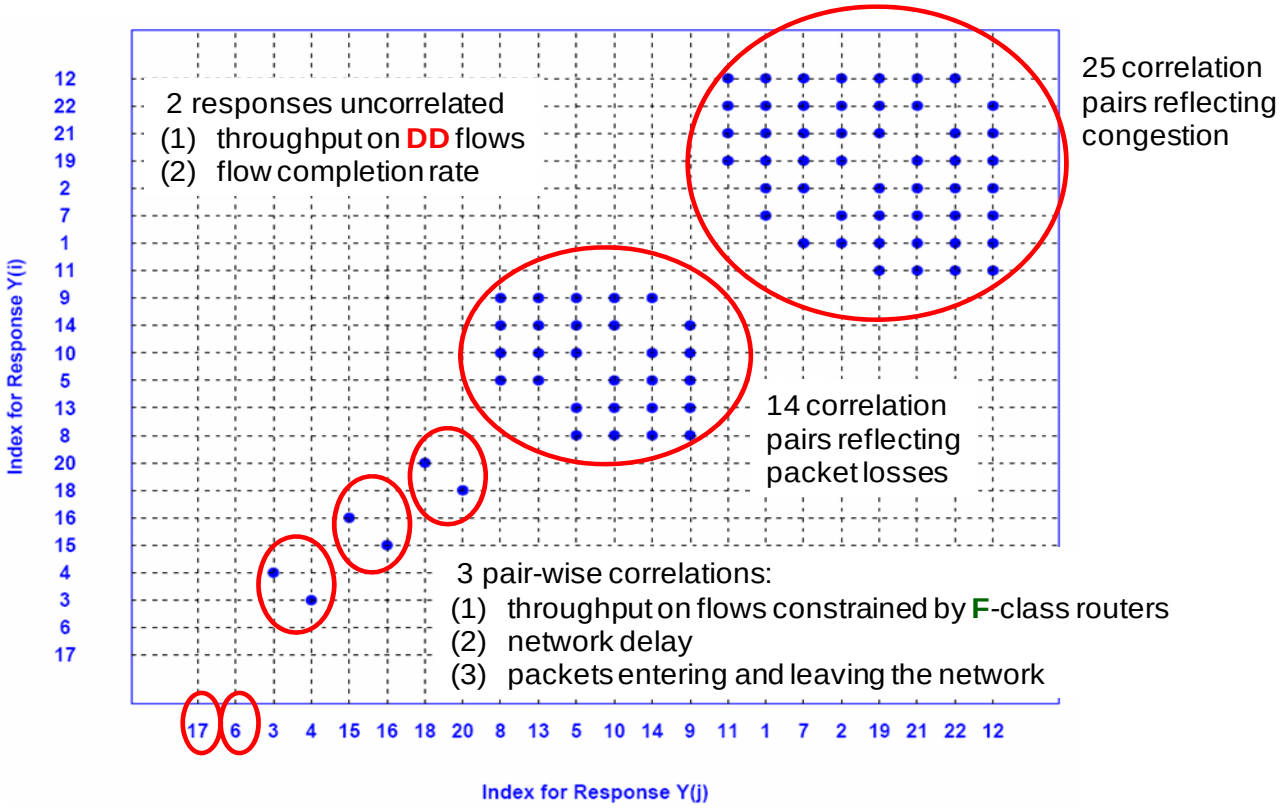

Figure 7: Index-Index plot sorted by increasing count of correlation pairs to identify clusters of mutual correlations that represent seven behavioral dimensions in the response state-space of MesoNet

axes of the index plot so that response identifiers are arrayed in increasing order of the cardinality of the subset of which they are a member. Response identifiers within each subset are ordered arbitrarily. Fig. 7 shows the resulting sorted index-index plot generated from the 42 correlation pairs selected from Fig. 6. Fig. 7 reveals seven behavioral dimensions: five subsets

(or clusters) of mutually correlated responses and two responses that were not correlated with any others. The largest cluster (25 correlation pairs) includes responses that reflect network congestion. The second largest group (14 correlation pairs) includes responses reflecting data losses. Three pair-wise correlations reflect: (1) throughput on flows constrained by F-class access routers, (2) network delay and (3) data entering and leaving the network. One u ncorrelated response (y17) measures average throughput on DD flows, while the other (y6) measures flows completed per second. An experimenter can use domain knowledge to select one variable to represent each identified behavior.

3.2 Multidimensional Data Analysis

Even after reducing a response state-space, specific experiments can produce multivariate datasets containing a large number of elements. For example, a typical experiment with MesoNet might compare eight congestion control algorithms with respect to 48 responses under 32 conditions over three time periods using two different protocols, yielding a multivariate dataset containing approximately 60,000 elements. Inferring meaning from such large datasets requires employing a variety of multidimensional data analysis methods. Here, we discuss those techniques we found to be most useful: (1) main effects analysis, (2) cluster analysis, (3) primary principal components plots and (4) condition-response summaries.

3.2.a Main Effects Analysis. Main effects analysis (MEA) [12] compares the mean value for a response variable when each model parameter is set to each level (-1 and +1) in a 2-level experiment design. During a sensitivity analysis (e.g., as discussed below in Sec. 5), MEA provides the basis for identifying significant parameters influencing key responses. Fig. 8 shows a sample MEA evaluating the influence of each of the 20 MesoNet parameters (see Table 2) on the average number of transmitting sources during a particular time period in one simulation experiment. For each parameter the plot gives two means: (1) when the parameter is set to the -1 level and (2) when set to the +1 level.

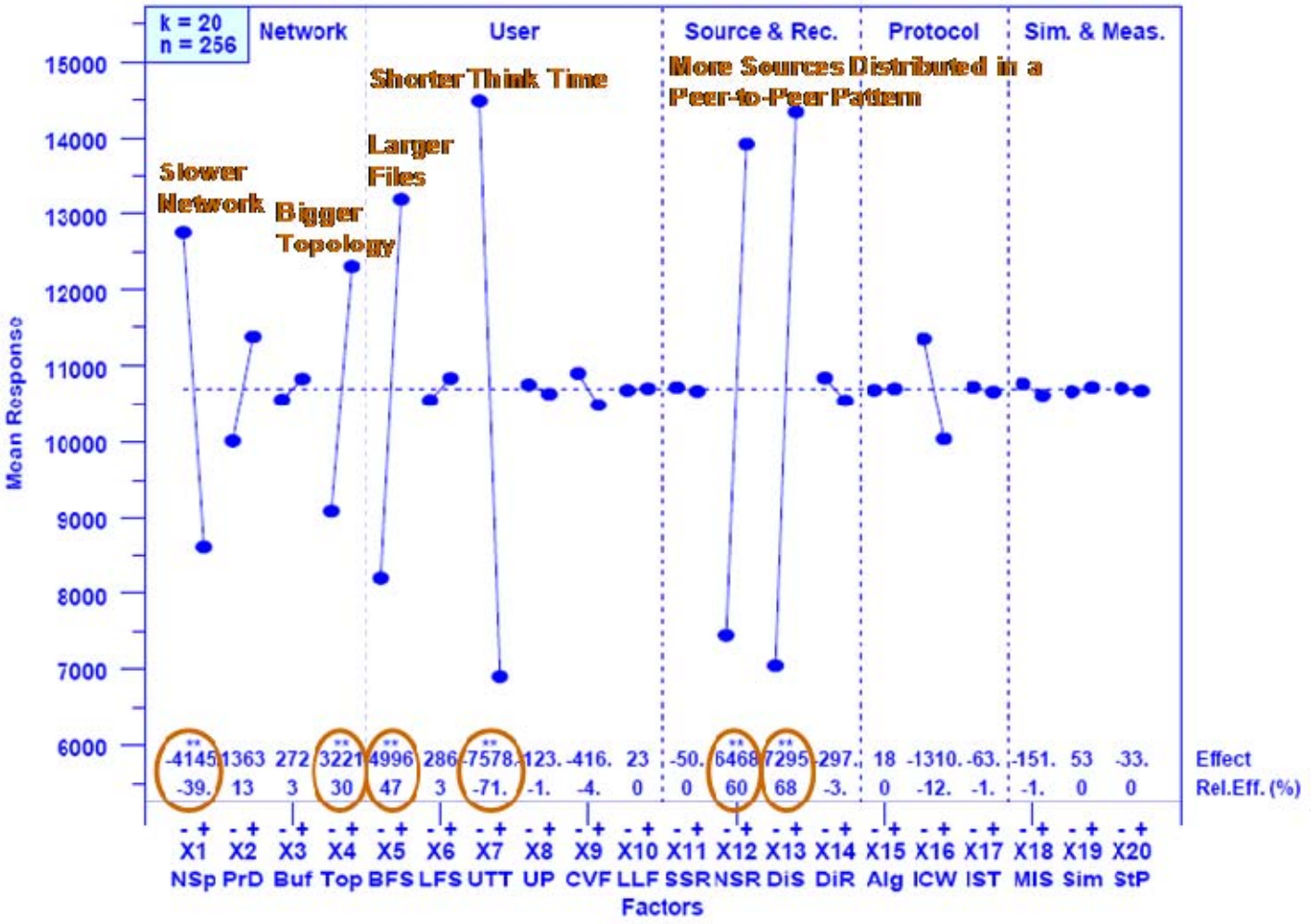

Figure 8: Main effects plot showing absolute and relative influence of each MesoNet parameter (x axis) from Table 2 on mean (y axis) number of sources transmitting (y1 from Table 3) and identifying six statistically significant parameters – network speed (X1), topology (X4), file size (X5), think time (X7) and number (X12) and distribution (X13) of sources

Fig. 8 shows that the mean number of sending sources was just under 13,000 when network speed (X1) was low (-) and was about 8500 under the higher network speed (+). For each parameter, a line connects the two means to indicate direction and magnitude of the effect when changing the parameter from its -1 level to +1 level. Two numbers are reported just above each parameter label. The top number gives the effect in raw terms (e.g., 4145 fewer sending sources under the higher network speed) and the bottom number gives the percentage change (e.g., 39 % fewer sending sources under higher network speed), which is called the relative effect.

We used a *t*-test to determine the likelihood of a true difference in effect for each parameter, inserting two asterisks (**) for parameters, such as network speed, where the estimated effect had a probability of statistical error $p < 0.01$. We inserted one asterisk (*) for effect estimates with $p < 0.05$. Interpreting Fig. 8, we find that shorter think time (X7 -) coupled with more sources (X12 +) distributed in a peer-to-peer pattern (X13 +) induce the largest of the six statistically significant effects on the number of sending sources. The effect of these parameters is followed by larger file size (X5 +) and larger topology (X4 +). The indicated values for these five parameters increase demand on the network. The remaining significant increase in sending sources arises from lower network speed (X1 -). In short, greater demand offered to a slower network increases congestion, which causes longer file transfer times for sources, leading more sources to be in the sending state. Similar analyses are possible for each MesoNet response in a specific experiment.

3.2.b Cluster Analysis. While MEA enables an experimenter to discern the influence of specific parameters on individual responses, cluster analysis can be used to discern similarities and differences in behavior when considering all responses as a multidimensional space. Cluster analysis can be accomplished using commonly available software, such as the hierarchical clustering tools from the MATLAB™ Statistics Toolbox™ [13]. 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.

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

We measure the linkage between clusters of algorithms as the average distance between responses associated with each algorithm in each cluster. In this example, we use a subset of 45 responses, not listed here. The linkage function, shown in (3), uses the Euclidean-distance function from (2).

$$D(r,s) \equiv \frac{1}{n_r n_s} \sum_{k=1}^{n_r} \sum_{l=1}^{n_s} Dist(Y_{k,r}, Y_{l,s}) \qquad (3)$$

Equation (3) 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 decide 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. 9, suggesting relationships among congestion control algorithms. Integer identifiers for the seven congestion control algorithms (see Table 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 less similar, with algorithm 3 being most dissimilar from the others.

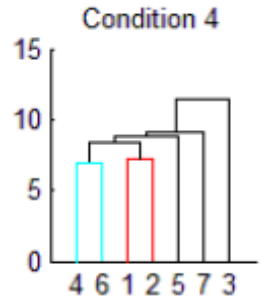

Figure 9: Dendrogram illustrating clustering based on 45 responses for one particular combination of parameters (i.e., condition 4) – x axis gives an identifier assigned to each of seven congestion control algorithms and y axis gives the standardized Euclidean distance between algorithms or clusters of algorithms.

Clustering must be performed individually on each parameter combinations because differing 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 parameter combination. Fig. 10 shows such a plot for seven congestion control algorithms and related responses. 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

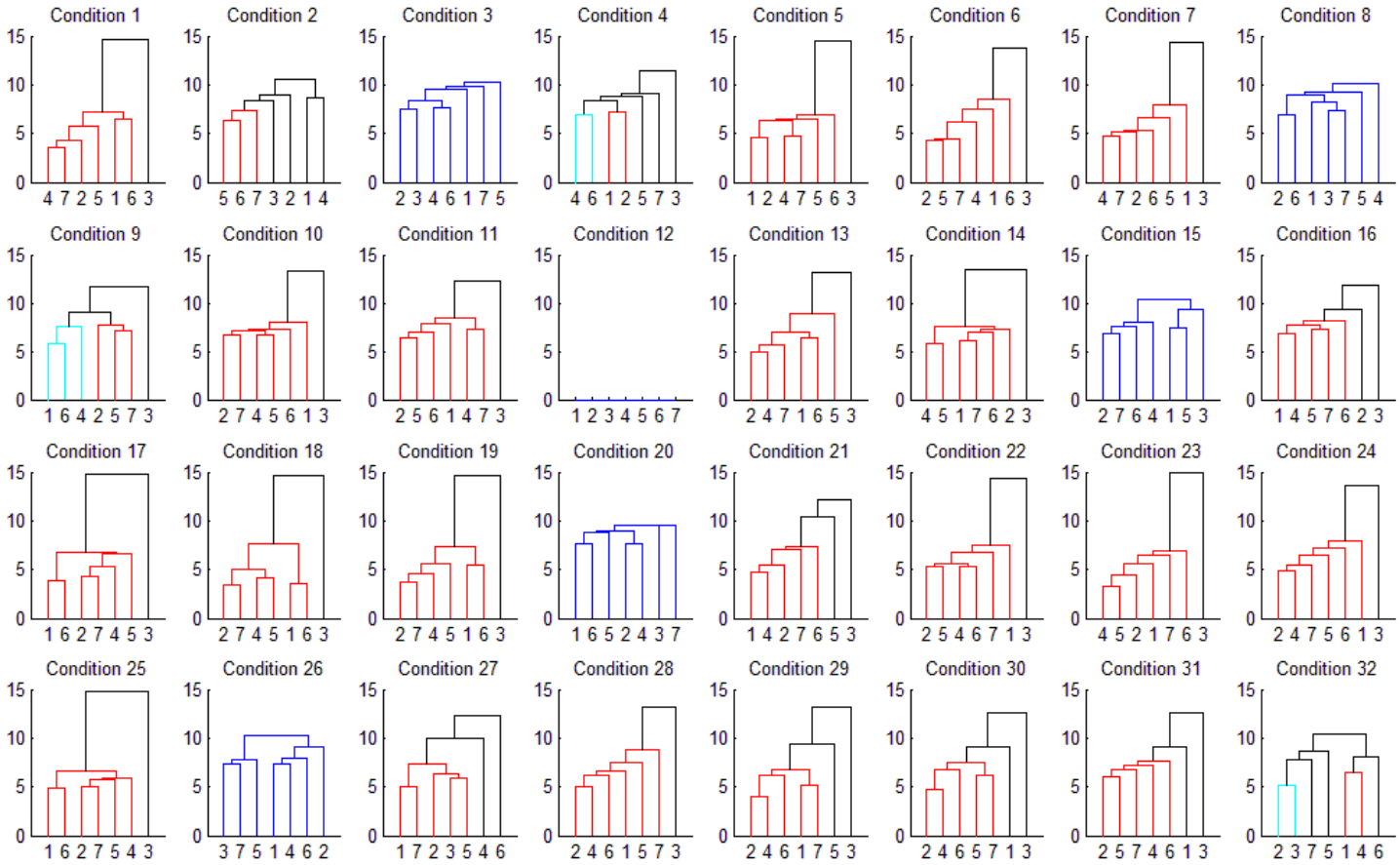

Figure 10: Cluster analysis comparing seven congestion control algorithms with respect to 45 responses under 32 parameter combinations (i.e., conditions)

algorithms. As Fig. 10 illustrates, clustering analysis can reveal significant overall patterns in multidimensional data.

3.2.c Primary Principal Components Plots. PCA can be used to explore relationships among responses, usually by plotting the primary component, PC1, which accounts for most variance in the dataset, against PC2, which accounts for second most variance in the dataset. For example, consider a multivariate dataset that contains the average throughput achieved on network connections using seven different congestion control algorithms under 32 different parameter combinations for 48 different flow types. We generate subsets of the data, where each subset, selected by congestion control algorithm and parameter combination, contains average throughput achieved on each flow type. We conduct PCA on each of the (7 x 32 =) 336 subsets, and then in Fig. 11 we plot PC1 (x axis) vs. PC2 (y axis) for each subset.

Fig. 11 shows three clear groupings of (circled) points, which can be considered with respect to the parameter combinations in common. Points within the blue circle represent cases where network speeds were lower (i.e., X1 set to level -1). The remaining points include only conditions with higher network speeds (X1 set to level +1): the red circle contains points with longer propagation delays (X2 set to +1) and the green circle contains points with shorter propagation delays (X2 set to -1). Points within the red circle can be further subdivided such that points above the dashed line represent small file sizes (X5 set to -1) and points below represent larger file sizes. Points within the green circle can also be subdivided based on relative file size. The main inference from Fig. 10 is that, under the specific parameter combinations simulated, throughput is influenced primarily by network speed, propagation delay and file size; choice of congestion control algorithm played no significant role here.

3.2.d Condition-Response Summaries. In many cases, statistical techniques can reveal not only overall behavioral patterns but can also provide evidence suggesting causality. In our case, we conducted detailed statistical comparisons for measured responses to identify conditions under which congestion control algorithms varied. Comparing the parameter combinations composing each condition allowed us to discern underlying causes. Subsequently, we extracted information from each detailed analysis to produce condition-response summaries that revealed explicit overall patterns showing under what conditions and for what responses algorithm 3 produced different behavior.

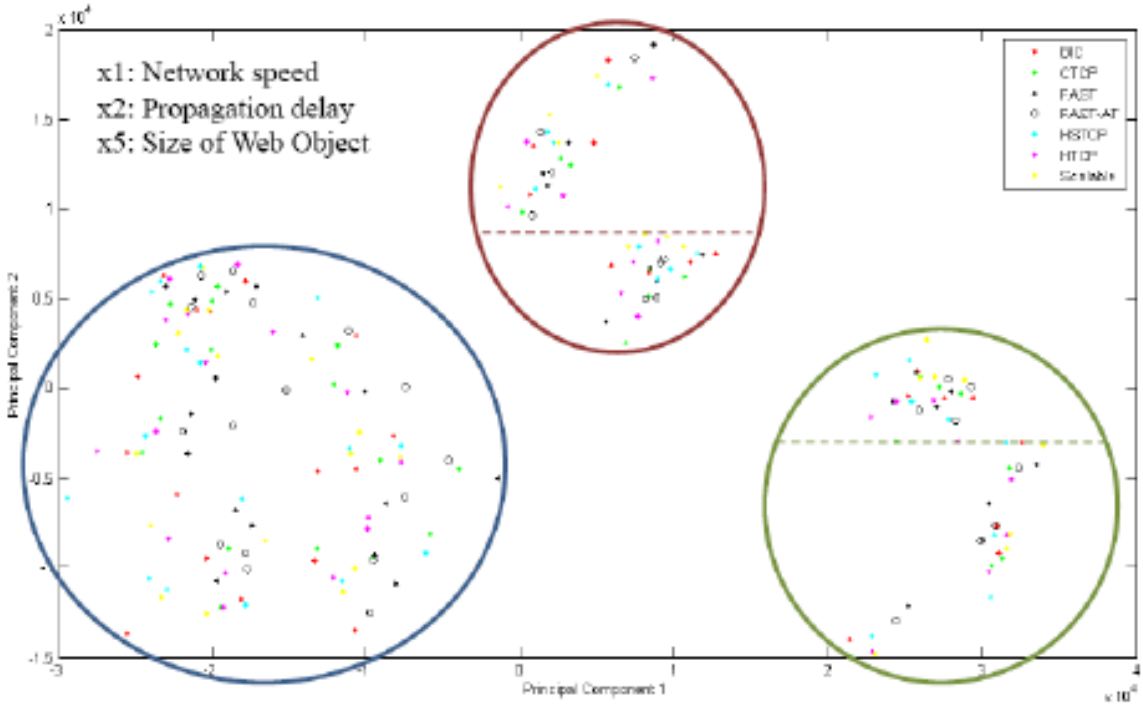

Figure 11: PC1 (x axis) vs. PC2 (y axis) from a multivariate dataset containing average throughput achieved on network connections using seven different congestion control algorithms under 32 different parameter combinations. Circles identify groupings of points and lines within circles distinguish additional subgroups.

Fig. 12 shows a plot comparing retransmission rates for seven congestion control algorithms under 32 different parameter combinations. The x axis in Fig. 12 shows the 32 conditions. Here, conditions are sorted by increasing magnitude of the largest difference in the response variable produced by 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, 8, 20 and 2) to about 0.55 (for algorithm 3 and condition 29). Below each box we display vertically the level settings (+/-) for each input factor (X1…X6) that generated the relevant condition. Here we used a subset of six input parameters identified by a sensitivity analysis (Sec. 5) to have significant influence on system behavior, which we measured with 45 responses in this particular experiment.

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. 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 11. 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 % of the mean response. A domain analyst can consider both 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 [12] associated with the extreme residual for each condition. The Grubbs' test computes *G* by dividing the largest residual by the sample standard deviation (*s*).

$$G \equiv \frac{\max\left(\left|Y_i - \mathrm{mean}(Y)\right|\right)}{s} \qquad (4)$$

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 ($p < 0.05$) when $G > 2.08$. The entire column

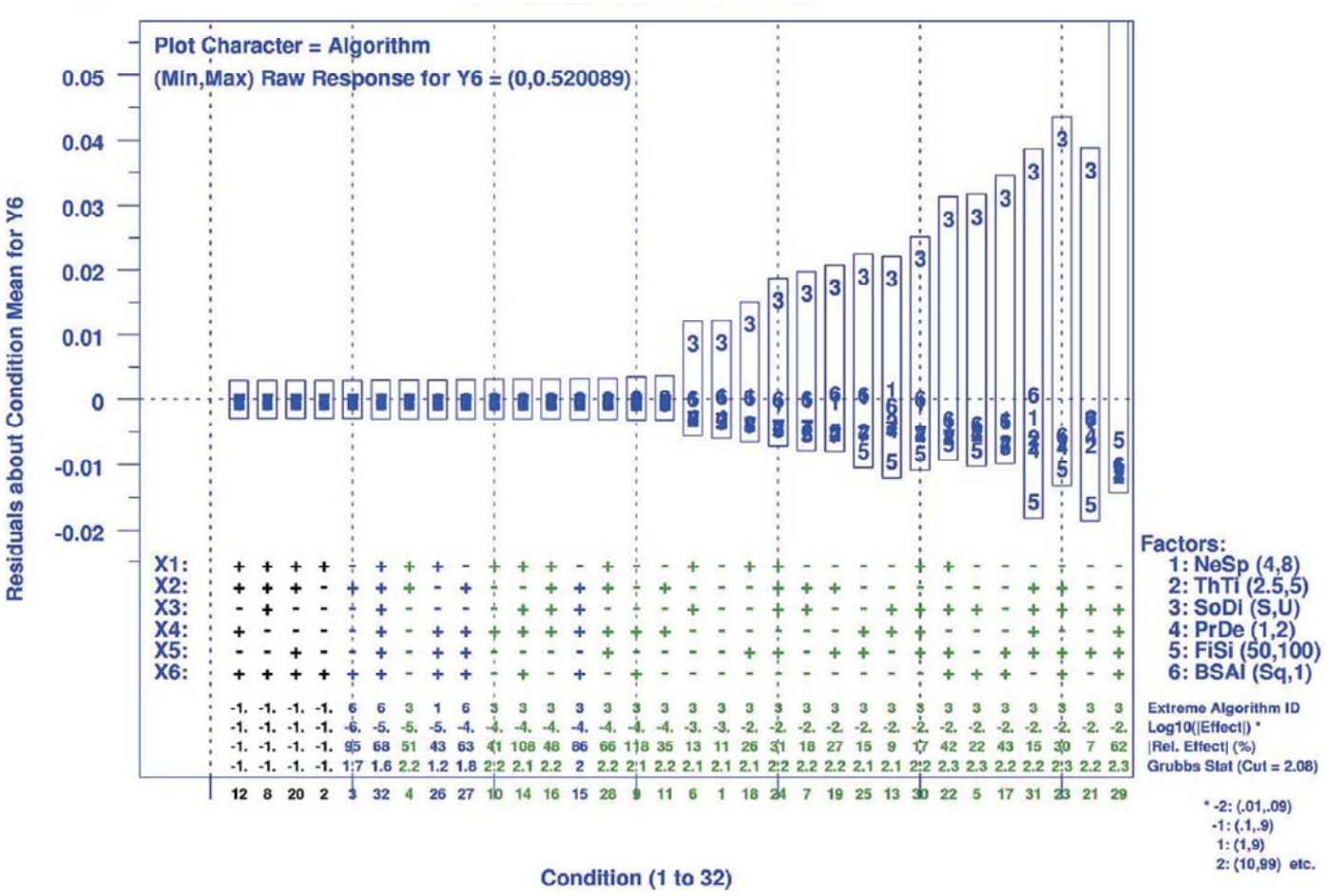

Figure 12: Sample plot analyzing the influence of condition and congestion control algorithm on the segment retransmission rate (in this experiment retransmission rate was designated as y6) – y axis gives residuals around the mean value for each condition and x axis gives conditions ordered by increasing range of residuals

(factors and statistics) is highlighted for conditions where the Grubbs' test identifies an outlier. Green identifies positive outliers (e.g., 23 conditions in Fig. 12) and red identifies negative outliers (no such conditions in Fig. 12). Columns are printed in black when no numerical difference could be detected among the responses (e.g., conditions 12, 8, 20 and 2 in Fig. 12). The remaining columns are printed in blue.

We can produce plots such as Fig. 12 for any response, and such plots can be quite revealing, but we use condition-response summaries, as shown in Fig. 13, to allow us to discern overall patterns associated with all responses under all conditions. To construct a condition-response summary, we extract information from analyzing individual plots similar to Fig. 12.

Fig. 13 shows a condition-response summary identifying any congestion control algorithms that were statistically significant outliers for 45 responses under 32 conditions. Each row in Fig. 13 corresponds to a specific condition (identified on the left). The first six columns report level settings (+/-) for the six input parameters defining the condition. The remaining columns represent individual responses. (Note the response variable number 6 in Fig. 13 corresponds to the information extracted from Fig. 12.) Vertical blue lines group related responses. For example, in this particular experiment, responses 1 through 8 relate to macroscopic behavior, responses 9 through 12 relate to throughput 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 (1 to 7) when there is a statistically significant outlier – red denotes low outliers and green denotes high outliers.

A scan of Fig. 13 shows that algorithm 3 arises as a statistically significant outlier in many cells. The highest concentration of outliers appears for congested conditions (measured by retransmission rate); 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. 10) for the same dataset. Both analyses

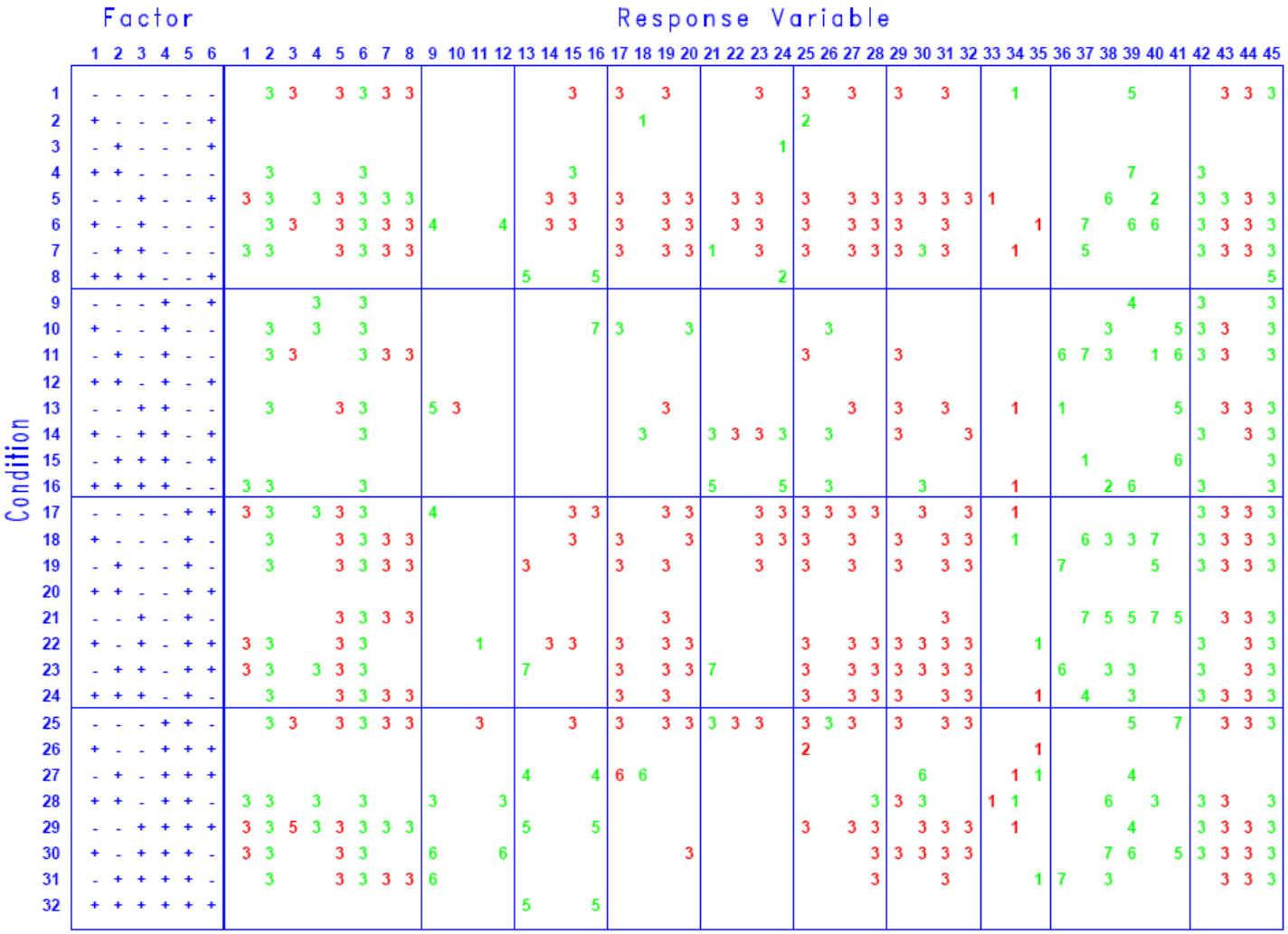

Figure 13: Condition-response summary identifying any statistically significant outliers among congestion control algorithms for each of 45 responses measured over 32 different conditions in this particular experiment

identify algorithm 3 as distinctive under congested conditions. Fig. 13 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. 14 shows a summary plot reporting statistically significant outliers from Fig. 13 that also achieve a relative effect greater than 10 %. The pattern of outliers is now sparser, so we can focus our analysis on responses y2 (congestion window increase rate, in this experiment), y6 (retransmission rate in this experiment), y42 (average number of connecting flows), y44 (average number of transmitting sources using standard TCP congestion avoidance) and y45 (average number of transmitting sources using an alternate congestion avoidance algorithm). The responses measuring buffer usage (y36 – y41) exhibit outliers but there is no evident pattern.

We also gleaned significant information from a wide range of other multidimensional analysis techniques, such as Y-Y scatter plots, condition-by-condition relative bar graphs, and rank matrices. For an explanation and application of these and other techniques, refer to our complete study [1] comparing selected Internet congestion control algorithms.

4 PROPOSED TCP REPLACEMENTS

Recall from our discussion of Internet congestion control (Sec. 2) that standard TCP achieved throughputs of about 20 % of link capacity for large files transmitted over long-distance, high-speed network paths. This implies that individual TCP users with access to such paths may not realize high throughputs and that such expensive links may be underutilized. For this reason, many researchers [16-21] have proposed improved congestion control algorithms to replace standard TCP on the Internet. The general aim of the proposed

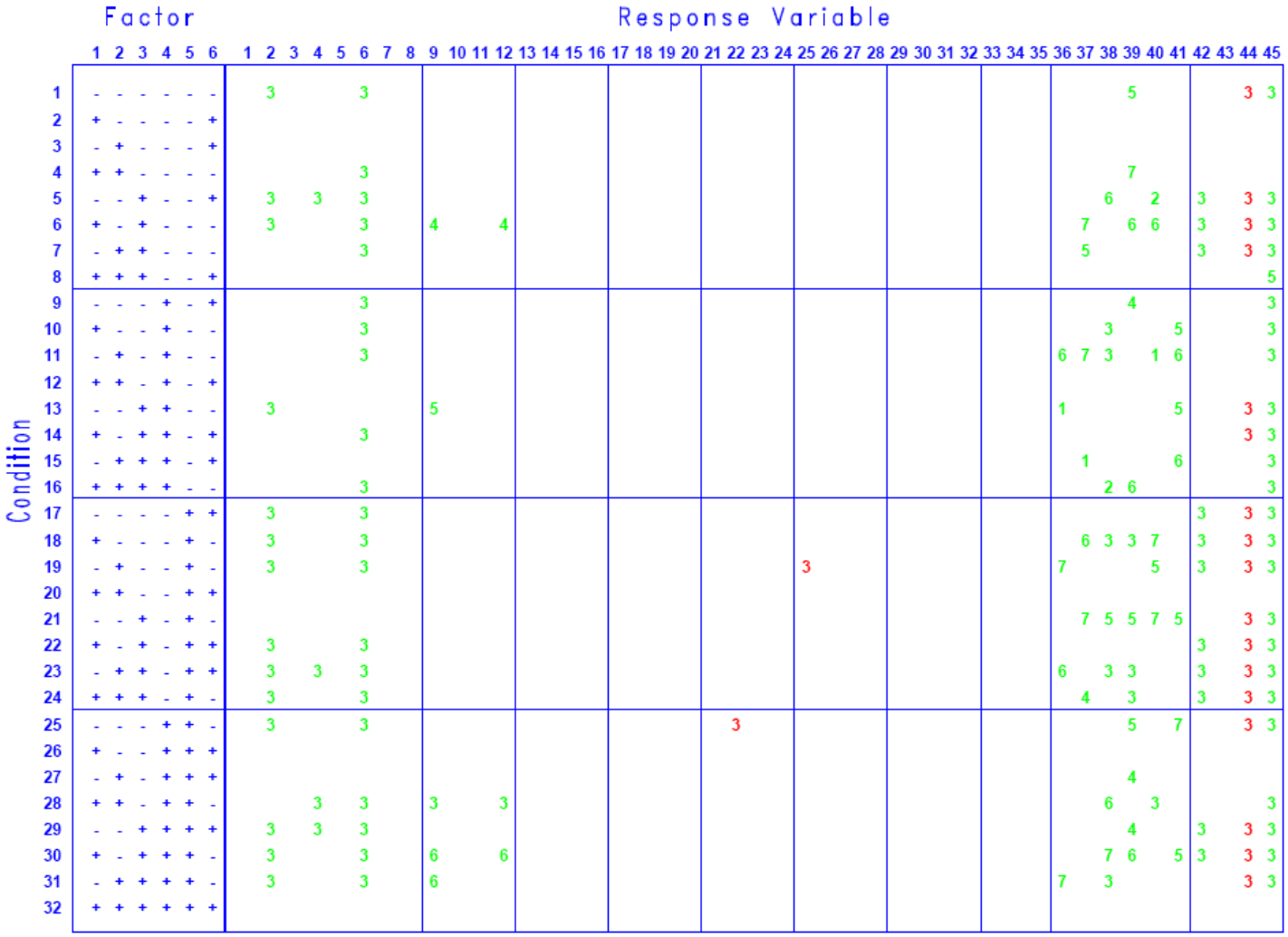

Figure 14: Filtered condition-response summary identifying any congestion control algorithms that are statistically significant outliers and that exhibit at least a 10 % relative effect for each of 45 responses measures over 32 different conditions

TCP replacements is to allow users improved throughputs when transferring large files over high-speed network paths that cover long distances. Internet users and operators may also reasonably expect that proposed TCP replacements achieve some other desirable goals: providing high link utilizations, sharing bandwidth equally under high congestion and converging quickly to equal bandwidth usage as flows come and go on network links. Most of these topics have been studied in the context of limited numbers of flows sharing bottleneck links, as we showed in Sec. 2 for standard TCP. Prior to our work, these topics have not been studied in the context of large networks, under the arrival and departure of hundreds of thousands of flows with varying characteristics. We used the methods described in Sec. 3 to conduct such a study [1] for proposed TCP replacements, as enumerated in Table 1.

While different in many significant details, most of the proposed TCP replacements share some common features. All of the proposals retain standard TCP procedures for connection establishment and initial slow start, changing only congestion avoidance. All but two of the proposals define a threshold, such that normal TCP congestion avoidance procedures are used when the *cwnd* is below the threshold. Of the remaining two, one adopts normal TCP congestion avoidance procedures for a period of time following a lost segment and the other never uses TCP congestion avoidance procedures. In general, the proposals adopt one (or both) of two main principals: (1) decrease transmission rate by less than half following a lost segment and (2) increase transmission rate more than linearly following a rate reduction. Next, we explain each proposed TCP replacement that we studied.

BIC. Binary Increase Congestion Control (BIC) [16] aims to make aggressive increases in *cwnd* when the current *cwnd* is far from a target and smaller increases as the current *cwnd* nears the target. BIC determines the target by conducting a binary search within some range around the current *cwnd*.

When the target falls beyond the search range, BIC adjusts the *cwnd* by a fixed increment and then reinitiates the binary search within the new range. Implementing this behavior requires rather complex logic. The resulting *cwnd* evolution for BIC reflects its complexity – reproducing a function that appears to change in a pattern resembling a human heartbeat.

Fig. 15 shows the temporal evolution of *cwnd* for two competing BIC flows, under conditions identical to those used in comparing competing TCP flows in Fig. 2. Here, BIC provides flows with much higher average *cwnd* than TCP, and BIC allows the flows to converge to equal sharing much more quickly than TCP does. In fact, the average *cwnd* exceeds the path capacity of 3402 segments, which means that the flows use 100 % of path capacity, carrying between 200 and 345 segments in buffers within routers on the path. These properties, high utilization and quick adaptation in sharing capacity with newly arriving flows, are exactly those that researchers hoped to achieve by proposing improvements to standard TCP.

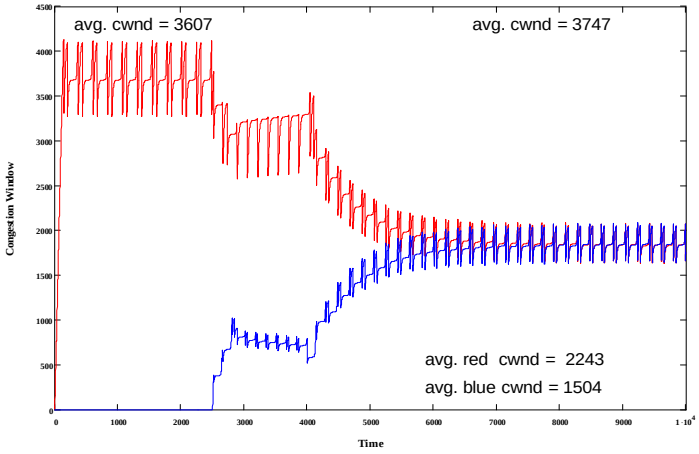

Figure 15: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two BIC flows sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

CTCP. Compound TCP (CTCP) [17] augments *cwnd* with a second component, called the delay window (*dwnd*). CTCP procedures update the *dwnd* periodically, typically once per *rtt*. The *dwnd* is added to the *cwnd* to establish the transmission rate used on CTCP flows. CTCP defines rules for increasing *dwnd* aggressively when a flow is underutilizing available transmission capacity and also defines rules for reducing *dwnd* as a flow's transmission rate nears available capacity. Upon detection of congestion, either through explicit segment losses or timeouts, CTCP reduces *dwnd* toward zero. As a flow's transmission rate nears equilibrium around some estimated available capacity, CTCP tends to cause the transmission rate to oscillate slightly by exponentially increasing the *dwnd* when the estimated number of segments queued for a flow falls below a threshold and then linearly decreasing *dwnd* when the estimated number of queued segments exceeds the threshold. On the other hand, when the transmission rate is increasing on a flow, CTCP exponentially increases the *dwnd* without exerting a countervailing linear decrease. Consequently, the CTCP transmission rate can become large relatively quickly when a transmission path exhibits no congestion.

Fig. 16 shows the temporal evolution of *cwnd* for two competing CTCP flows under the identical conditions used in Fig. 2. When only the red flow uses the path, CTCP achieves throughput comparable to TCP. On the other hand, when two flows share the path, CTCP converges to *cwnd* values that use the entire path capacity with only a few segments held in network buffers.

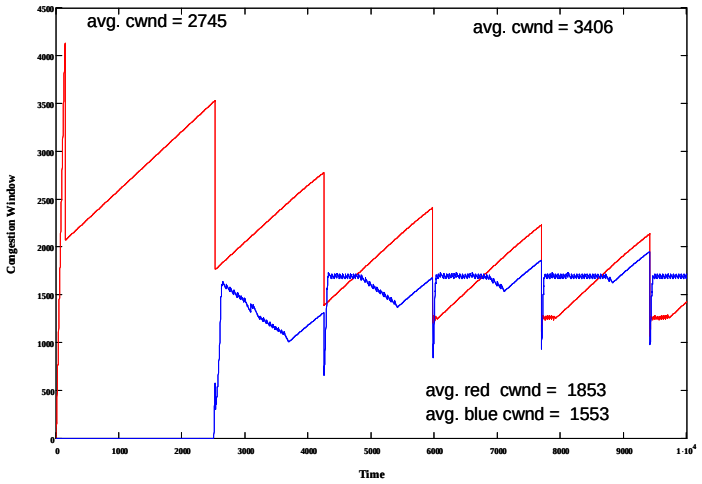

Figure 16: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two CTCP flows sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

FAST. Fast Active Queue Management Scalable TCP (i.e., FAST) [18] adopts a fundamentally different approach from the other congestion control procedures considered in this study. First, FAST aims to achieve an equilibrium *cwnd* that remains stable, while other congestion control mechanisms lead to an oscillating *cwnd*. Second, FAST updates the *cwnd* based mainly on measured changes in queuing delay, using loss signals only when congestion prevents reaching a lossless equilibrium. Third, FAST does not resort to standard TCP congestion avoidance procedures; instead, FAST uses its own procedures at all times during congestion avoidance. FAST adopts these approaches based on the idea that queuing delay can be measured quite frequently and thus accurately, while segment losses are rare events that provide insufficient information to estimate loss probability on a given flow.

The FAST congestion control procedures include α–tuning as an option. The α–tuning variant of FAST, monitors flow throughput and adjusts α up and down as various thresholds are crossed. The α term is added to the *cwnd* computation by FAST, so adjusting α can push the *cwnd* up and down with

variable thrust, compared with when α is fixed. The designers of FAST indicate [18] that α–tuning is no longer used routinely within FAST implementations. Instead, the designers suggest fixing α to a value suitable for expected network conditions. Of course, the designers recognize that fixing α is not a general solution and list α–tuning as an open issue. We studied FAST both with and without α–tuning. We call the latter alternative FAST-AT.

Fig. 17 shows the temporal evolution of *cwnd* for two competing FAST flows under the identical conditions used in Fig. 2. Here, FAST achieves its aim of a fixed *cwnd* when a single (red) flow uses the path and also adapts quickly to lower fixed *cwnd* when the blue flow arrives to share the path. The resulting average *cwnd* exceeds path capacity, so segments will be queued in network routers and the path will be utilized at 100 % capacity. Also note that both the red and blue flows achieve nearly equal shares of the path capacity. This demonstrates FAST achieving the aims of its designers.

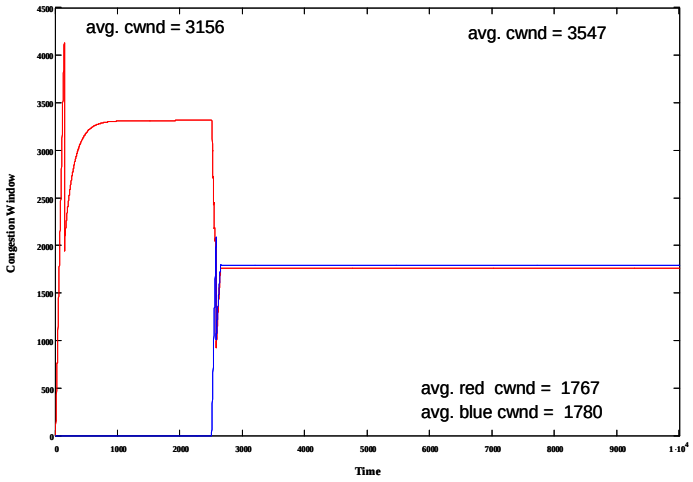

Figure 17: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two FAST flows (with α fixed at 80) sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

Fig. 18, showing the temporal evolution of *cwnd* for two competing FAST-AT flows under the identical conditions used in Fig. 17, illustrates an artifact that α–tuning can cause when flows share a network path. Notice that the two flows quickly reach equilibrium at similar *cwnd* values after the blue flow arrives. Subsequently, the *cwnd* values diverge, with the blue flow achieving significantly higher transmission rate than the red flow. Such divergence, also seen in empirical studies [15], arises because the red flow adjusts α downward as decreasing throughput crosses a threshold. The blue flow, under increasing throughput, crosses a threshold and adjusts α up. As an additional threshold is crossed, the α values are adjusted yet again. As a result, while FAST-AT achieves full utilization of link capacity, the competing flows achieve 30 % (red) and 70 % (blue) shares. This deviates significantly from the desired goal of 50 % capacity share for each of the two flows.

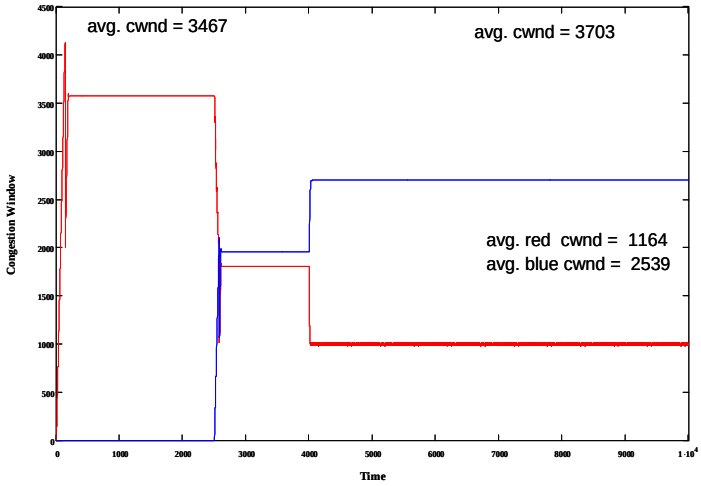

Figure 18: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two FAST-AT flows sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

HSTCP. High Speed TCP (HSTCP) [19] retains the fundamental additive-increase and multiplicative-decrease (AIMD) strategy adopted by standard TCP, but HSTCP alters the AIMD parameters to become a function of congestion window size. The altered AIMD functions result in more aggressive increases and less aggressive decreases at larger window sizes.

Fig. 19 demonstrates that, under the conditions here, HSTCP bounds the *cwnd* within a smaller range than standard TCP and oscillates up and down more quickly. The path is fully utilized (with some segments buffered in network routers) and the two flows converge reasonably quickly to an equal share of the available link capacity.

HTCP. Hamilton TCP (HTCP) [20] differs from the other TCP replacements in two main aspects. First, HTCP determines *cwnd* increases as a function of elapsed time since the most recent segment loss. The increase is scaled by the *rtt* experienced on the network path in order to compensate for differences in feedback delay. The motive is to give larger increases in *cwnd* during periods of low network congestion, so a flow could reach higher transmission rates more quickly on uncongested, high-capacity, long-delay paths. HTCP adopts standard TCP *cwnd* increase procedures for a specified time after each loss. Second, HTCP implements an adaptive back-off procedure to determine the multiplicative decrease in *cwnd* after a loss. The back-off factor is varied based on estimating the queuing delay on a path. The motive is to prevent senders from backing off too much after packet losses. HTCP adopts standard TCP decrease procedures when flow throughput has

changed by more than a specified amount since the most recent loss.

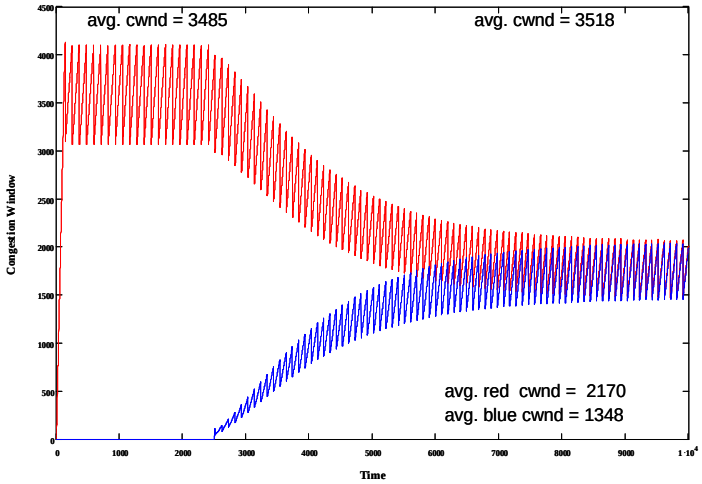

Figure 19: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two HSTCP flows sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

Fig. 20 shows that HTCP achieves full path utilization and that the *cwnd* sizes converge quickly to equilibrium when the second (blue) flow arrives. The flows achieve an equal share of the available capacity.

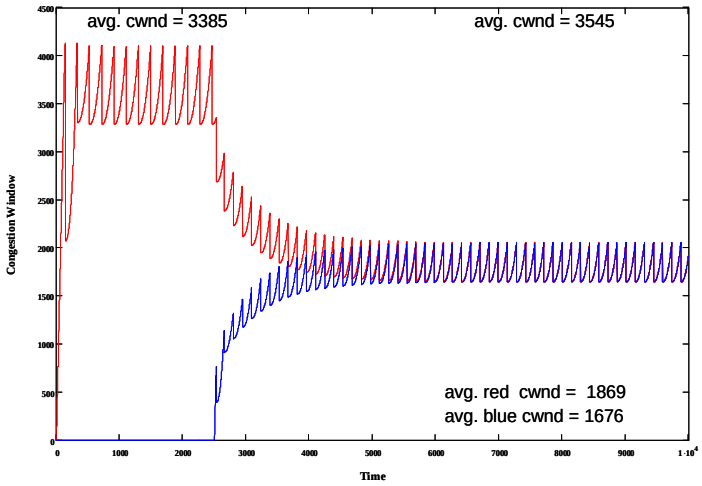

Figure 20: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two HTCP flows sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

STCP. Scalable TCP (STCP) [21] adopts a simple, fixed-increase rule aimed at allowing a flow to increase its congestion window more quickly than would be the case with standard TCP. In addition, Scalable TCP defines a decrease rule that limits a flow to a fixed multiplicative decrease that is recommended to be much less than the 50 % decrease used by standard TCP. The STCP rules are defined in an additive-increase, multiplicative-decrease (AIMD) form, but the rules actually amount to a multiplicative-increase, multiplicative-decrease (MIMD) regime. Researchers have shown [14] that MIMD algorithms are not guaranteed to converge to fair capacity sharing in networks, such as the Internet, which drop arriving segments when a queue is filled.

Fig. 21 illustrates the SCTP convergence problem. The first (red) flow achieves full utilization of the path, with 340 segments buffered in routers in the network. The second (blue) arriving flow experiences great difficulty gaining much of the path capacity, achieving an average share of only 2 % of the available capacity. Thus, SCTP achieves full utilization, while providing greatly unequal sharing of the available capacity. This behavior has been produced in empirical measurements [15], as well as in the simulations shown here.

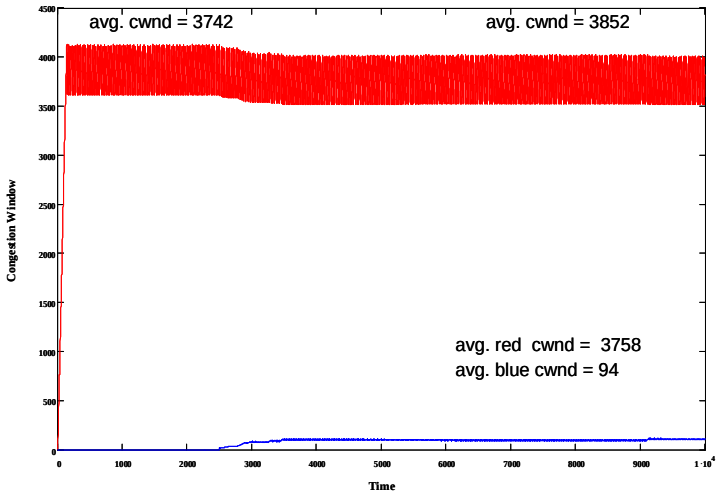

Figure 21: Change in *cwnd* (y axis in segments) vs. Time (x axis in 0.1 s units) for two STCP flows sharing a bottleneck path with 162 ms *rtt* – total average *cwnd* reported at top and average *cwnd* for the red and blue flows reported at bottom

4.1 Previous State of the Art

These, and other, proposed congestion control procedures have been studied using a variety of techniques, which constitute the state of the art in network modeling, analysis and measurement. The designers of the proposed protocols construct analytical models to study likely behavior under various situations [16-21]. In addition, some researchers [22] construct and compare analytical models of several proposals operating within the context of single, long-lived flows. Other researchers [23-24] construct analytical models of some proposals to consider the average behavior of a large number of flows sharing a single path. Such analytical studies provide insights related only to the specific, typically quite constrained, situations modeled. Modeling a single, or pair, of long-lived flows does not illuminate the more typical situation where many flows with differing characteristics come and go over

time. Even analyses with many flows idealize the situation so that all flows have the same characteristics.

Another technique entails simulation studies [25] comparing proposed congestion control procedures in small topologies. While insightful, simulations of small topologies do not exhibit the same spatiotemporal dynamics that would be experienced in a large network with varying user demands arriving and departing and spreading throughout the topology [32]. Most simulation studies use small topologies because configuring, executing and analyzing such simulations can be completed within reasonably available resources.

Another valuable technique compares proposed congestion control procedures empirically within small topologies. In fact, the graphs shown in Figs. 15-21 were simulations of scenarios that had been measured empirically by other researchers [15]. The simulations were produced with our model (MesoNet), explained below in Sec. 5. The existence of such empirical measurements on a single bottleneck path helped us to verify that we had correctly modeled the proposed TCP replacements prior to simulating them in a large topology. Researchers [15, 26] also measured the performance of small numbers of long-lived flows under each proposed TCP replacement when perturbed by cross traffic consisting of Web flows regulated by standard TCP. A few researchers [27-28] have configured topologies of a few hundred flows to measure the performance of FAST. Similar experiments should be feasible using other configurable test beds [29-31].

We can sum up some of the shortcomings of state-of-the-art modeling, analysis and measurement techniques with respect to comparing proposed TCP replacements. First, most existing techniques focus on long-lived, i.e., infinite length or very large, flows, while Internet traffic consists of a wide variety of flow sizes, where flows come and go dynamically. Second, most existing techniques focus on congestion avoidance behavior, omitting other key congestion control phases, including connection establishment and initial slow start, which might influence significantly congestion behavior in a large network. Third, most existing techniques consider from several to 100 flows, which transit a shared path, or a few shared paths, in a small topology. We will show how our methods can address these shortcomings, and advance the state of the art in modeling and analysis of large networks.

4.2 Advancing the State of the Art

Below, we demonstrate how to combine reduced-parameter modeling, 2-level OFF experiment design and multidimensional data analysis techniques to simulate more than 10^5 simultaneous flows operating at high network speeds and in realistic topologies under a wide range of congestion conditions, while measuring and analyzing high dimensional response spaces that provide new insights about proposed TCP replacements. We begin in the next section by explaining MesoNet, our 20-parameter model of a TCP network, and we show how a sensitivity analysis of the model can be used to identify key response dimensions, and the model parameters that significantly influence those dimensions. We move on in Sec. 6 to describe experiments, which leverage our methods to provide insights about Internet congestion control procedures.

5 MESONET SIMULATION MODEL

We implemented a model for each of the proposed congestion control algorithms explained above in Sec. 4. To investigate global behavior, we embedded the algorithm models within MesoNet [33], our simulation of a TCP network, which was introduced briefly in Sec. 2, where Table 2 identified the 20 parameters composing MesoNet. Here, we give more detailed information about those parameters.

A *network configuration* requires a topology (parameter X4) of routers and links, as shown for example in Fig. 22, adapted from the Abilene backbone network [34]. MesoNet supports topologies with up to three hierarchical router tiers: backbone routers (A-K in Fig. 22), point of presence (PoP) routers (A1-K2) and access routers (A1a-K2d). To model heterogeneity in network access, MesoNet allows three different types of access routers: **D**-class (e.g., six red nodes in Fig. 22, which connect directly to backbone routers), **F**-class (e.g., 28 green nodes) and **N**-class (e.g., 105 small gray nodes).

Classifying access routers enables different speeds to be assigned to each class. Sources and receivers compose a fourth tier distributed below access routers. Data segments flowing between a source-receiver pair follow a single ingress/egress path between an access router and a top-tier backbone router. In MesoNet ingress/egress paths are not subject to propagation delays. Propagation delays on backbone links are an intrinsic property of all MesoNet topologies, as are the paths taken by data segments flowing among backbone routers.

Given a cost metric for each backbone link, one can use Dijkstra's shortest-path first (or equivalent) algorithm to generate least-cost paths. Assuming a link cost equal to propagation delay, the topology in Fig. 22 generates 110 paths between backbone routers, with an average path length of 3.51 router hops. Adding in the hops for sources and receivers to reach the backbone routers increases the average path length to 9.43 hops. To scale propagation delays in a topology, parameter X2 multiplies the delays assigned to each backbone link. Unlike real networks, where links have transmission speeds and associated buffers, MesoNet assigns speeds to routers. Each router multiplexes segment forwarding from a single buffer shared among all attached links. Because MesoNet segments have no size, router speeds are assigned in units of segments/millisecond. Parameter X1 defines the base speed of backbone routers and all other router classes operate at a proportion of that speed: PoP routers 25 %, **N**-class 2.5 %, **F**-class 5 % and **D**-class 25 %. To provision router buffers, MesoNet allows buffer size (in segments) to be selected using an algorithm, specified by parameter X3.

Given a three-tier topology of routers and links, the model constructs a fourth tier, where *sources and receivers* are distributed under (and attached to) access routers. The model includes a target number of sources and receivers which should

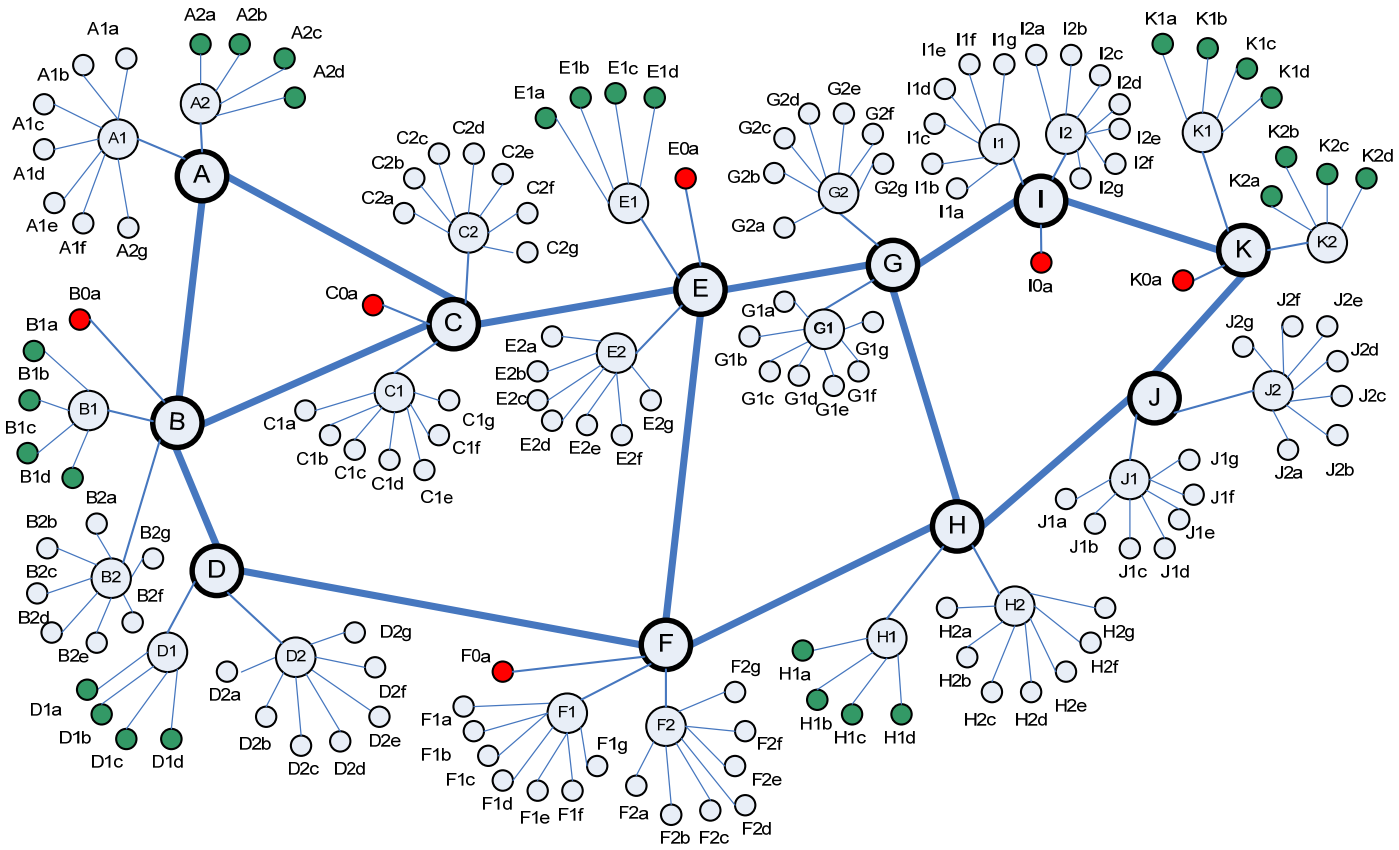

Figure 22: Three-Tier Topology with 11 Backbone Routers (A-K), 22 Point of Presence Routers (A1-K2) and 139 Access Routers (A1a-K2d) – 6 red and 28 green Access Routers may operate at different speeds from the 105 others

be set to a value appropriate for the network speed. Model parameter X12 serves as a multiplier to scale the target number of sources and receivers. Parameter X13 specifies probabilities (e.g., see Tables 11 and 13) that bias the distribution of sources so that a higher or lower proportion of the target number attaches under various classes (**D**/**F**/**N**) of access routers. Similarly, parameter X14 specifies probabilities that bias the distribution of receivers. Altering the distributions of sources and receivers adjusts the probability of flows transiting access routers of specific classes, where the slowest access router crossed by a flow determines the flow's path class: very fast (VF) for **D**-class routers, fast (F) for **F**-class or typical (T) for **N**-class. The final property of sources and receivers concerns the maximum speed at which they can transfer segments to the network. The model allows two speeds: normal (8000 segments/second, i.e., 96 Mbps) and fast (80,000 segments/second, i.e., 960 Mbps). Parameter X11 specifies the probability that a source or receiver connects at the fast speed. When a flow's receiver and source are both connected at the fast speed, a flow's maximum rate is fast (F); otherwise a flow's maximum rate is normal (N).

User behavior is modeled through periodic activity by sources, which cycle between thinking, connecting and sending. Prior to entering the thinking state, a source selects a random residence time from an exponential distribution with a mean given by parameter X7. Upon expiration of residence, the source enters the connecting state, where a connection is attempted to a randomly selected receiver. If a connection attempt succeeds, the source enters the sending state, where a flow of segments is transmitted. Once all segments in a flow are acknowledged, the source reenters the thinking state. If a connection attempt fails, the source reenters the thinking state without sending. Sources may have finite or infinite patience. Parameter X8 specifies the probability that a source has finite patience, where short flows must be completed within a reasonable time and long flows must progress at a reasonable rate or else a source aborts the flow and reenters the thinking state.

Prior to sending, a source selects a Web object size (in segments) from a Pareto distribution with a mean defined by parameter X5. Through parameter X6 the model allows sources to transmit larger files in three categories: documents, software updates and movies, with corresponding multipliers (*Fx*, *Sx* and *Mx*) that scale the selected Web object size to a larger value with a corresponding probability (*Fp*, *Sp* and *Mp*) for each category. The model also allows simulation of spatiotemporal congestion by specifying (parameter X9) a time period during which every flow transiting a VF path will have the randomly

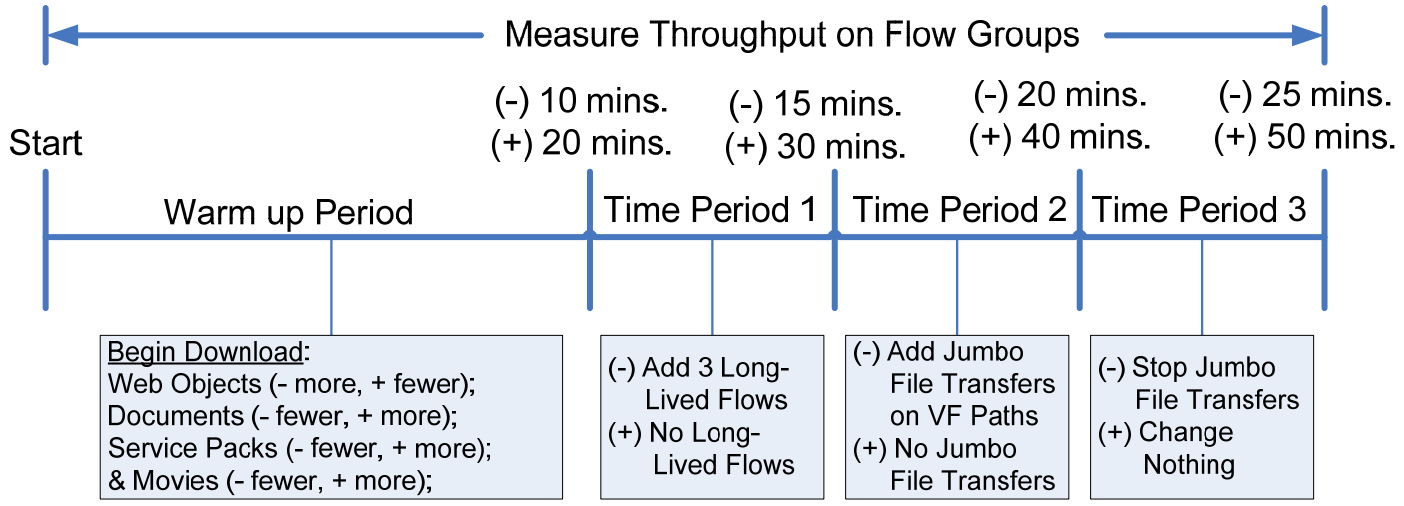

Figure 23: Possible traffic scenarios generated by various combinations of values for parameters X5, X6, X9, X10 and X19

selected file size multiplied by 10, becoming *jumbo* files that create spatiotemporal congestion. The model also permits simulation of long-lived flows that, once activated, send as many segments as possible in the course of a simulation. Parameter X10 specifies the number, location and starting time for any long-lived flows included in an experiment.

The transmission rate of each flow is regulated by *protocols*. Upon connecting to a receiver, a source first sends a number of segments, known as the initial *cwnd*, specified by parameter X16. As ACKs arrive from the receiver, the source increases *cwnd* exponentially. Upon the first lost segment, the source adopts procedures specified by a designated congestion avoidance algorithm. Model parameter X15 defines the probabilities that a given source uses each of the congestion avoidance algorithms simulated by MesoNet. If there are no losses, a source switches to its congestion avoidance algorithm once the *cwnd* reaches an initial slow *sst*, defined by parameter X17.

MesoNet measures numerous aspects of model behavior during each *simulation* run. Most *measurements* are made as time series, which sample system states at periodic intervals defined by parameter X18. Model parameter X19 *controls* the duration of a simulation run. Model parameter X20 determines the rate at which sources initially enter the sending state.

5.1 MesoNet Sensitivity Analysis – Design

To understand the characteristics of MesoNet, we employed a 2-level OFF experiment design to conduct a sensitivity analysis, which allowed us to identify parameters that significantly influence model behavior. We used these findings to guide the design of subsequent experiments (see Sec. 6) to compare the proposed TCP replacements explained above in Sec. 4.

We adopted a 2^{20-12} OFF design, requiring 256 simulations, where each simulation has a specified combination of parameters that are set to one of two levels, which we refer to as the MINUS (-1) and PLUS (+1) levels. Table 5 gives the two level values we selected for each of MesoNet's 20 parameters. Most of the parameter mappings from Table 5 are straightforward. Here, we discuss a few that merit more explanation. We also introduce the response variables used in the sensitivity analysis.

Table 5: Level Settings Used in Sensitivity Analysis

Category	Identifier	MINUS (-1) Level	PLUS (+1) Level
Network Configuration	X1	800 segments/ms	1600 segments/ms
	X2	1x	2x
	X3	*rtt* x *C*/sqrt(*n*)	*rtt* x *C*
	X4	Abilene – delay	ISP – costs
User Behavior	X5	75 segments	150 segments
	X6	*Fp* = 0.02 *Sp* = 0.002 *Mp* = 0.0002	*Fp* = 0.04 *Sp* = 0.004 *Mp* = 0.0004
	X7	2 seconds	5 seconds
	X8	Infinite	Finite
	X9	4th Time Period	None
	X10	3 begin 3rd Period	None
Sources & Receivers	X11	0.2	0.8
	X12	2x	3x
	X13	Web centric	Peer-2-Peer Centric
	X14	Web centric	Peer-2-Peer Centric
Protocols	X15	TCP = 0.8 CTPC = 0.2	TCP = 0.2 CTPC = 0.8
	X16	2 segments	8 segments
	X17	43 segments	> 10^9 segments
Simulation & Measurement Control	X18	200 milliseconds	1 second
	X19	25 minutes	50 minutes
	X20	Exp. (mean X7)	50% start early

Table 6: Responses measured during a sensitivity analysis of MesoNet

Macroscopic Responses			Flow Groups for Throughput Averages			
Category	Identity	Definition	Number	File Size	Path Class	Max. Rate
Flow State	Y1	Average # sources connecting	1	Movie	VF	F
	Y2	Average # sources sending	2		VF	N
	Y3	% sending flows in initial slow start	3		F	F
	Y4	% sending flows in standard congestion avoidance	4		F	N
	Y5	% sending flows in alternate congestion avoidance	5		T	F
			6		T	N
Congestion	Y6	Retransmission rate	7	Software Service Pack	VF	F
	Y7	Average congestion window size (segments)	8		VF	N
	Y8	Aggregate # connection failures	9		F	F
			10		F	N
Delay	Y9	Average round-trip time (ms)	11		T	F
	Y10	Average queuing delay (ms)	12		T	N
			13	Document	VF	F
Work	Y11	Average # flows completed per second	14		VF	N
	Y12	Average # segments output per second	15		F	F
			16		F	N
Long-Lived Flows	Y13	Average throughput on long-lived flow #1	17		T	F
	Y14	Average throughput on long-lived flow #2	18		T	N
	Y15	Average throughput on long-lived flow #3	19	Web Object	VF	F
			20		VF	N
Flows by Path Class	Y16	Average throughput on flows transiting VF paths	21		F	F
	Y17	Average throughput on flows transiting F paths	22		F	N
	Y18	Average throughput on flows transiting T paths	23		T	F
			24		T	N

The -1 level for parameter X4 entails using the Abilene topology shown in Fig. 22. For the +1 level of X4 we used a larger topology adapted from a commercial Internet Service Provider (ISP). The ISP topology has more routers (16 backbone, 32 PoP, 8 **D**-class, 40 **F**-class and 122 **N**-class), more backbone links (24) and thus additional least-cost paths (240) in the backbone. The increased number (170) of access routers implies that the +1 topology will also have more sources and receivers than the -1 topology. Backbone paths in the +1 topology are determined based on costs assigned by the ISP in order to achieve specific traffic engineering objectives. Both the -1 and +1 topologies have propagation delays corresponding to the physical length of backbone links. Values for the X1 parameter scale all router speeds in the selected topology. Values for the X2 parameter scale propagation delays on all backbone links in the topology. Values for the X3 parameter scale buffer sizes for all routers in the topology. The +1 value for X3 selects a buffer provisioning algorithm that corresponds to the recommended practice [35], i.e., a router's buffer size in segments is the average *rtt* in a topology multiplied by the router's speed (*C*). Following the suggestion of some researchers [36], the -1 value for X3 divides a router's computed buffer size by the square root of the expected number (*n*) of flows transiting the router.

Several parameters influence network traffic generated by sources, as illustrated in Fig. 23. Each simulation run can be viewed through a time line with length corresponding to the simulation duration assigned via parameter X19: 25 (-1 level) or 50 (+1 level) minutes. The simulation begins with sources sending files of various sizes, as determined by the values of parameters X5 and X6. The -1 level for X6 denotes transfer of fewer large files, i.e., documents, software service packs and movies, which implies the transfer of more Web objects. The +1 level for X6 increases the proportion of transfers of large files and decreases the proportion of Web objects. After a warm up period of either 10 (-1 for X19) or 20 (+1 for X19) minutes, the scenario unfolds over three additional time periods, each with a duration of either 5 (-1 for X19) or 10 (+1 for X19) minutes. At onset of the first time period three long-lived flows are started if X10 is -1. The long-lived flows are not started if X10 is +1. At onset of the second time period transfer of jumbo files may be started (-1) or not (+1) on VF paths, depending on the level of X9. At the onset of the third time period no further jumbo files will be initiated.

A few other parameters merit mention. Parameters X13 and X14 vary the distribution of sources and receivers in a topology, which influences the proportion of flows transiting specific access router classes. The -1 level for these parameters creates Web-centric traffic, which means an increase in the proportion of flows transiting **D**-class and **F**-class access routers. The +1 level for these parameters increases the proportion of flows that transit **N**-class access routers, which is more consistent with peer-to-peer traffic. For this experiment, sources may regulate flow transmission rate using one of two congestion avoidance algorithms: standard TCP or compound TCP (CTCP). A -1 level for parameter X15 deploys more TCP sources in a topology, while a +1 level deploys more CTCP sources. Finally, a -1 level for parameter X20 causes sources to leave the initial thinking state after exponential delays with a mean determined by parameter X7. The +1 level for X20 causes 25 % of sources to start in the connecting state and 25 % to leave the initial thinking state early, while the remaining 50 % leave after a normal delay.

As shown in Table 6, in this sensitivity analysis we characterized MesoNet behavior by measuring 18 macroscopic responses, summarizing network state in six categories, and by averaging throughput (in segments/second) for each of 24 flow groups, where a flow group is defined by three dimensions: (1) file size, (2) path class and (3) maximum transfer rate. We averaged each macroscopic response separately in the three time periods identified in Fig. 23, yielding a total of (3 x 18 =) 54 macroscopic responses. We computed throughput per flow group separately for sources using TCP and for those using CTCP, yielding a total of (2 x 24 =) 48 flow-group throughput measurements. Thus the number of computed responses totaled (54 + 48 =) 102.

A few responses from Table 6 require brief explanation. Recall that sources cycle through three states: thinking, connecting and sending. We measured the average number of connecting (Y1) and sending (Y2) sources; other sources are thinking. Sending flows begin operating under initial slow start rules and may then move to congestion avoidance, where sources implementing CTCP may cycle between standard and alternate rules. We used responses Y3, Y4 and Y5 to measure the proportion of sending flows operating under each rule set. Since lost segments must be resent, we computed retransmission rate (Y6) as a ratio: file size to data segments sent on a flow before receiving the last acknowledgment. We measured the average work/second accomplished in flows (Y11) and segments (Y12) for each time period. With responses Y13, Y14 and Y15 we estimated instantaneous throughput in each time period for individual long-lived flows transiting specified paths in the network. Similarly, we used responses Y16, Y17 and Y18 to estimate instantaneous throughput on each path class in each time period regardless of differences in file size and maximum transfer rate. To estimate instantaneous throughput we divided the number of ACKs sent in a measurement interval by the interval size. For flow groups, we computed throughput measures by dividing file size (in segments) by the time interval between sending the first segment and receiving an ACK for the last segment.

Table 7: Significance of influence of 20 MesoNet parameters (columns) on 18 macroscopic responses (rows) during 2nd time period: blue cells indicate significance $p < 0.01$ and orange cells indicate significance $p < 0.05$, where a – or + in highlighted cells indicates the parameter setting that causes an increase in the corresponding response

		Network				User Behavior						Source/Receiver				Protocol			Sim. Control & Meas.		
Metric Class	Y#	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20
Flows	Y1	-**		+**	+*			-**					+**	+**			+**				
	Y2	-**			+**	+**		-**					+**	+**							
	Y3	+**	+**	+**		-**		+**					-**	-**				+**			
	Y4	-**	-**	-**		+**		-**					+**	+**		-*					
	Y5	+**		+**				+**					-**	-**		+**	+*	-**			
Congestion	Y6	-**	-**	-**		+**		-**					+**	+**			+**				
	Y7	+*																			
	Y8	-**	-**	-**		+**							+**	+**			+**				
Delay	Y9	-**	+**	+**									+*	+*							
	Y10	-**	+**	+**				-*					+**	+**							
Aggregate TP	Y11	+**	-*		+**	-**		-**					+**	+**							
	Y12	+**		+**	+**	+**		-**		-**			+**								
Long-Lived Flow TP	Y13	+**		+*	+*					+**	-**										
	Y14	+*			+**					+**	-**										
	Y15	+**								+**	-*				-*						
Other Flow TP	Y16	+**	-**		-**			+*		+**			-**		-*			+**	-*		
	Y17	+**	-**		+*			+**		+**			-**	+**	-**		+**		-*		
	Y18	+**	-**		-**			+**					-**	-**			+**				

5.2 MesoNet Sensitivity Analysis – Results

We analyzed the main effects of each MesoNet parameter on all 102 measured responses. We conveyed these analyses through main effects plots, such as Fig. 8, which plots response y2 (average number of sending sources) in time period 2. We created five (only one shown here) summary tables: three tables (one per time period) report statistically significant effects of parameters on the 18 macroscopic responses and two tables (one for TCP and one for CTCP) report statistically significant effects of parameters on throughput for each of the 24 flow groups. For example, Table 7 provides a summary for the 18 macroscopic responses in the 2nd time period – the row for y2 was created from the main effects plot in Fig. 8. Cells in Table 7 highlighted in blue and annotated with ** denote effects significant at the $p < 0.01$ level and those highlighted in orange and annotated with * denote effects significant at the $p < 0.05$ level. Each highlighted cell also includes either a + or – to indicate which level for the corresponding parameter (column) led to a higher value in the response (row). For y2 for example, we find slower (-) network speed (X1), larger (+) topology (X4), bigger (+) file sizes (X5), shorter (-) think times (X7) and more (+) sources (x12) distributed in a peer-to-peer (+) pattern (X13) led to a larger number of sending sources. This corresponds to the information given in Fig. 8. A quick scan of Table 7 shows that network speed had significant influence on all 18 responses during the 2nd time period. Other significant parameters can be identified, as well as those that had little or no significant influence on the responses. Other patterns can also be discerned, such as the influence of particular sets of parameters on responses associated with network congestion.

Table 8 condenses our five tables summarizing main effects for macroscopic responses (in each of three time periods) and for throughput per flow group (under each of two congestion avoidance algorithms). For each parameter (X1 to X11), we computed the fraction of responses influenced (Ψ), weighting $p < 0.05$ at ½ and $p < 0.01$ at 1, as shown with the following equation.

$$\Psi = (|\{y \mid p < 0.01\}| + ½\,|\{y \mid p < 0.05\}| \,/\, |\{y\}| \qquad (5)$$

In Table 8 we multiply these fractions by 100 to generate the percent of responses influenced. All percentages are rounded. Table 8 displays the resulting Ψ x 100 for each parameter from each of the five response sets. The bottom row gives a weighted average Ψ x 100 for each parameter. We weighted each time period at 1/6 and each congestion avoidance algorithm at 1/4, which amounts to weighting the macroscopic and flow throughput responses equally.

The main influences on model behavior included physical network parameters: speed (X1), propagation delay (X2), buffer provisioning (X3) and topology (X4), with network speed being most influential. The number (X12) and distribution (X13) of sources, along with file size (X5) and user think time (X7) also showed significant influence on model behavior. Lesser influence arose from the initial *cwnd* size (X16) and *sst* (X17). Other parameters exhibited little or no influence on MesoNet behavior.

To confirm these findings, we conducted additional sensitivity analyses [1] using different level settings, extending the range between the chosen values. The findings reported here were robust across the sensitivity analyses we conducted. The information obtained from the sensitivity analyses confirmed the understanding of domain experts regarding key factors influencing behavior in real networks. This boosted confidence in our MesoNet model. Further, our sensitivity analyses identified the seven most significant parameters to vary in subsequent experiments to compare congestion control procedures proposed for the Internet. We address these experiments next.

6 CONGESTION CONTROL EXPERIMENTS

To compare proposed TCP replacements, we conducted five experiments, in two categories, as identified and described in Table 9. While we present these experiments as an integral whole, in reality we adopted an incremental experiment design process, where results from one experiment guided design of the next experiment and so on. In this way, interesting or significant behaviors arising in one experiment could be more fully explored in subsequent experiments.

We began comparing proposed TCP replacements with an experiment designed to investigate how the algorithms react to

Table 8: Summary of significance of influence (Ψ x 100) of 20 MesoNet parameters (columns) on 18 macroscopic responses for the three time periods (first three rows) identified in Fig. 24 and average throughput for the 24 flow groups identified in Table 5 under TCP and CTCP congestion avoidance procedures (rows four and five), and the weighted average Ψ x 100, with each time period weighted 1/6 and each congestion avoidance algorithm weighted 1/4

	Network				User Behavior						Source/Receiver				Protocol			Control & Meas.		
Ψ x 100	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20
Period #1	95	53	64	50	50	0	67	0	0	17	0	75	67	14	6	44	14	8	0	0
Period #2	94	53	53	42	39	0	61	0	33	14	0	75	64	11	8	31	17	6	0	0
Period #3	97	56	61	42	53	0	69	0	25	17	0	75	67	11	6	28	17	11	0	0
TCP	79	71	56	40	56	0	50	0	4	0	0	21	67	0	0	33	67	4	0	0
CTCP	79	75	46	44	65	0	60	0	4	0	0	46	67	8	0	31	50	0	0	0
Weighted Average	87	64	55	43	54	0	60	0	12	8	0	54	67	8	3	33	37	5	0	0

and recover from spatiotemporal congestion. We decided to model a large fast network. This decision proved expensive, as it required over 16,500 processor hours (> 2 years on a single computer) to simulate each of the seven proposed TCP replacements. By using 48 processors in parallel, we completed the required simulations in only about 2 weeks.

Table 9: Summary description of five experiments grouped into two categories that each focused on one particular question

How do the proposed TCP replacements react to and recover from spatiotemporal congestion?	
Experiment #1a	Compared seven proposed TCP replacements (excluded FAST-AT) in a large (up to 278,000 sources), fast (up to 192 Gbps backbone) network; Web browsing users; 25 minutes simulated; three 5-minute time periods; large (> 1 billion segments) initial *sst*; all sources use the same congestion control algorithm
Experiment #1b	Same as #1a, except; smaller (up to 27,800 sources) and slower (up to 28.8 Gpbs backbone) network and low (100 segments) initial *sst*. (Added FAST-AT)
How do the proposed TCP replacements improve flow throughputs and affect competing TCP flows?	
Experiment #2a	Compared eight proposed TCP replacements in a small (up to 26,085 sources), slow (up to 38.4 Gpbs backbone) network; Web browsing users and interspersed users who download software and movies; 60 minutes simulated; large (> 1 billion segments) initial *sst*; some sources use standard TCP congestion control procedures and some sources use one of the proposed congestion control procedures
Experiment #2b	Same as #2a except for low (100 segments) initial *sst*.
Experiment #2c	Same as #2a except for a larger (up to 261,792 sources) and faster (up to 384 Gbps backbone) network.

Guided by the results from our sensitivity analysis, we selected six parameters to vary in experiment #1a. Table 10 shows the chosen parameters and the two level settings for each. We fixed the remaining parameters, as shown in Table 11. All of our experiments used the topology (X4) shown in Fig. 22. In experiment #1a we chose to omit FAST-AT because the designers of FAST indicated that the α-tuning variant was not used generally. Among the seven most significant parameters identified in our sensitivity analyses, we decided to exclude the multiplier (X12) on the number of sources because varying the source distribution pattern also varied (from 174,600 to 278,000) the number of sources. We selected 1000 as the target number of sources under each access router, and we fixed the multiplier for number of sources (X12) to 2, which boosted the target number of sources under each access router to 2000, which was appropriate for the network speeds simulated.

Table 10: Parameters varied in experiment #1a and the values selected for the PLUS and MINUS levels

	Parameter Definition	PLUS (+1)	MINUS (-1)
X1	Network Speed	8000 s/ms	4000 s/ms
X2	Prop. Delay Multiplier	2	1
X3	Buffer Provisioning	*rtt* x *C*	*rtt* x *C*/sqrt(*n*)
X5	Avg. File Size	100 segs.	50 segs.
X7	Avg. Think Time	5 s	2.5 s
X13	Source Distribution	.33/.33/.33	.1/.6/.3

Table 11: Fixed parameters and values for experiment #1a

	Parameter	Fixed Value
X4	Topology	Abilene – SPF Delay
X6	Large Files	*Fp* = 0.1 & *Fx* = 10
X8	Patience	infinite
X9	Selected Spatiotemporal Congestion	4th Time Period
X10	Long-lived flows	3 begin in 3rd Time Period
X11	Prob. Fast Source	0.4
X12	Number Sources	2 x 1000
X14	Dist. of Receivers	0.6/0.2/0.2
X15	Congestion Control	Algorithm Under Test
X16	Initial *cwnd* size	2 segments
X17	Initial *sst*	2**32/2 segments
X18	Meas. Interval size	200 ms
X19	Duration	25 minutes
X20	Startup Pattern	50 % start early

We set the probability of large file sizes to reflect a 10 % chance that Web browsing users would download a document of interest. Somewhat unrealistically, we assigned users infinite patience, which factored out user behavior as a means of congestion control, allowing us to focus directly on the congestion avoidance algorithms we were investigating. We set up spatiotemporal congestion to occur in the 4th of five simulated time periods because we were investigating the ability of congestion control procedures to react to and recover from congestion.

In MesoNet, the state of long-lived flows is recorded in quite some detail. In order to access those details, we started three long-lived flows, transiting different portions of the topology, in the third time period. We selected a rather high probability (0.4) of sources and receivers connecting to the network at high speed because the motivation of the proposed TCP replacements was to enable corresponding users with fast

connections to realize better throughput than standard TCP. The choice of congestion control algorithm (X15) was not fixed; instead, we simulated each of the 7 proposed congestion control algorithms against all of the parameter combinations created from Table 10.

For the variable parameters in Table 10, we used a 2^{6-1} OFF design, which reduced the number of simulated parameter combinations from 64 to 32. The 32 simulated conditions created: network backbone speeds of either 192 or 96 Gbps, average propagation delays of 41 or 81 ms (200 ms maximum) and a wide range of buffer size combinations, influenced by both network speed and propagation delay. These parameter combinations generated varying levels of network congestion, ranging from no segment loss to 50 % segment loss. The balance and orthogonality of the experiment design ensured that half the combinations led to a congested network and half did not.

While simulating the parameter combinations, we measured 45 responses, many of which we described above. Of particular note, we added some aggregate measures to record the total number of segments flowing into and out of the network and the total number of flows competed. We also measured buffer utilization in selected access routers and average throughput on the three long-lived flows.

Applying cluster analysis to the 45-dimensional response space for each of the 32 conditions identified algorithm 3 (FAST) as an outlier in many conditions (e.g., see Fig. 10). The condition-response summary shown in Fig. 13 also shows algorithm 3 was an outlier for many responses under many conditions. Further, as shown in Fig. 14, response y6 (retransmission rate from Table 6) was particularly high for FAST under many conditions, and the retransmission rate disparity for FAST increased with increasing network congestion. Examination of *cwnd* traces on long-lived flows revealed that the FAST congestion avoidance algorithm oscillates *cwnd* size under spatiotemporal congestion, with the oscillations increasing with increasing congestion.

Results from this experiment also revealed other valuable information. Under low or no congestion, all congestion control procedures, including standard TCP, provided identical throughputs. This result occurred because in the absence of segment losses flows can complete transfer of all segments during the initial slow start phase, provided the initial *sst* is large enough, as was certainly the case in experiment #1a. Further, under high congestion, most of the proposed TCP replacements (excepting FAST) exhibited similar rates of segment loss. This result occurred because these proposed procedures contained a mode switch that triggers the use of standard TCP congestion avoidance procedures under periods of heavy congestion, which tend to drive the *cwnd* size below the mode switching thresholds.

With this information in hand, we designed experiment #1b, which entailed three main changes from experiment #1a: (1) we reduced the initial *sst* to 100 segments, (2) we reduced the network size and speed by an order of magnitude, and (3) we included FAST-AT. We took the first step to confirm whether the setting of the initial *sst* was important under low network congestion. We took the second step to confirm that we could generate useful information with fewer computation resources (because slower, smaller networks could be simulated more quickly). We took the third step because we wanted to determine if FAST-AT produced the same macroscopic behavior as FAST. We made no other changes to the experiment parameters adopted in experiment #1a.

Results for experiment #1b showed that FAST-AT exhibits indistinguishable behavior from FAST, and quite distinct from the other proposed TCP replacements we investigated. Results from experiment #1b also found that the behavior of the other proposed TCP replacements were largely indistinguishable from each other, even when the initial *sst* was reduced to 100 segments. This, perhaps unexpected, result can be attributed to the fact that average file sizes on most flows were at or below 100 segments, and so could still be transferred within the initial slow start phase, where changes to congestion avoidance procedures would make no difference. This attribution was verified by comparing performance on long-lived flows, which have unlimited length. On these flows, during time period 3, before the onset of spatiotemporal congestion, all the proposed TCP replacements provided much greater throughput than standard TCP. This occurs because once the *cwnd* reaches 100, TCP moves from the exponential increase of slow start to the linear increase of congestion avoidance, while all the proposed TCP replacements increase transmission rate much faster than linear. Among the proposed procedures, FAST reaches high throughput most quickly. On the other hand, when the 4^{th} time period arrives and spatiotemporal congestion begins, the throughput for all congestion control procedures is driven to a very low level. However, upon cessation of spatiotemporal congestion, our results showed that some of the proposed congestion control procedures recovered high transmission rate more quickly than others, and all of them recovered more quickly than standard TCP.

Results from experiments #1a and #1b led us to investigate two other questions. What benefit can users expect from adopting each of the proposed TCP replacements, and what will be the cost to users who continue using standard TCP? To investigate these questions we used a 2^{9-4} OFF design to construct 32 parameter combinations composing experiment #2a. We used the same design for experiment #2b, changing only the *sst*, as indicated in Table 13. In Table 12 we specify the two levels for each of the nine variable parameters.

Since experiments #1a and #1b showed that users could achieve throughput improvements only on files larger than Web objects, we increased the variety of file sizes to include not only documents but also software service packs and movies. We fixed the multipliers for these larger files to Fx = 100, Sx = 1000 and Mx = 10,000. Table 13 gives other fixed parameters for the experiment.

In these experiments we did away with long-lived flows and with spatiotemporal congestion, and we considered only a

single time period, lasting 60 minutes. We adopt a small, slow network in order to ease the computation burden from simulating one hour of network operation. As in all experiments, we set the initial *cwnd* to 2 segments (a common value in the Internet [37]) and the measurement interval to 200 ms (the maximum *rtt*). We also start half the sources early in order to limit the startup transient. We chose probabilities for the distribution of sources and receivers that gave interesting flow patterns, consistent with a Web-centric Internet.

Table 12: Parameters varied in experiments #2a and #2c, along with the values selected for the PLUS and MINUS levels

	Parameter	PLUS (+1)	MINUS (-1)
X1	Network Speed	1600 s/ms	800 s/ms
X2	Prop. Del. Multiplier	2	1
X3	Buffer Provisioning	*rtt* x *C*	*rtt* x *C*/sqrt(*n*)
X5	Avg. File Size	150 segs.	100 segs.
X6	Large Files	*Fp* = 0.04 *Sp* = 0.004 *Mp* = 0.0004	*Fp* = 0.02 *Sp* = 0.002 *Mp* = 0.0002
X7	Think Time	7.5 s	5 s
X11	Prob. Fast Interface	0.7	0.3
X12	Num. Src./Rcv. Mul.	3	2
X15	Prob. Not Std. TCP	0.7	0.3

Table 13: Fixed parameters for experiments #2a and #2b

	Parameter	Value
X4	Topology	Abilene – SPF Delay
X8	Patience	infinite
X9	Spatiotemporal Congestion	none
X10	Long-lived flows	none
X13	Dist. Srcs.	0.1/0.6/0.4
X14	Dist. Rcvs.	0.6/0.2/0.2
X16	Initial *cwnd*	2 segments
X17	Initial *sst*	2**31/2 (#2a) or 100 (#2b)
X18	Meas. Int. Size	200 ms
X19	Sim. Dur.	60 minutes
X20	Startup Pattern	50 % start early

Experiments #2a and #2b considered a network with a mixture of flows using standard TCP competing with flows using each of the proposed TCP replacements, while transferring files of quite varying sizes. This allows us to investigate the conditions under which the proposed TCP replacements will give users better throughput and also to determine how the proposed TCP replacements will influence the throughput for users of standard TCP. For this reason, we measure the throughput obtained on each of 24 flow groups (see Table 6) when using standard TCP or a proposed TCP replacement.

Experiment #2a revealed (see Fig. 11) that, under large initial *sst*, throughputs for the 24 flow groups were differentiated based on three main factors: (1) network speed, (2) propagation delay and (3) file size. These results agreed with results from experiment #1a, but expanded the results to larger file sizes, including download of software updates and movies.

Experiment #2a did discern some throughput differences for the largest files, i.e., movies, flowing over the fastest (VF) paths with the fastest interface speeds (i.e., 1 Gbps). For example, Fig. 24 shows seven biplots comparing throughputs achieved with each proposed TCP replacement (x axis) against throughputs achieved on competing standard TCP flows (y axis). Each biplot includes a diagonal line representing equal throughput for flows using standard TCP and flows using the proposed TCP replacement. Four of the proposed TCP replacements (CTCP, FAST, FAST-AT and HTCP) achieved approximately equal throughput with standard TCP. On the other hand, under some conditions, three of the proposed TCP replacements (BIC, HSTCP and Scalable TCP) achieved higher throughputs than standard TCP. Detailed examination revealed that BIC, HSTCP and Scalable TCP achieved higher throughputs than standard TCP under congested conditions, with the throughput differential increasing with increasing congestion.

Experiment #2b lowered the initial *sst* to 100 segments, which again revealed the significant influence of the *sst* on throughput for large files traversing very fast, uncongested network paths. Fig. 25 shows two sets of seven bars. The leftmost set, labeled y2(u), graphs throughputs achieved on flows using each of the proposed TCP replacements. The rightmost set, labeled y16(u), graphs throughputs achieved on competing flows using standard TCP. The legend to the right of the graph indicates each of the seven bars represents a specific TCP replacement. In Fig. 25 the flows being compared were transferring movies at potentially fast speed (up to 1 Gbps) over very fast, uncongested (VF) network paths. The y axis plots the throughput achieved divided by the maximum achievable throughput. Examining the rightmost set of bars shows that standard TCP flows provide only 20 % of the available throughput, the same empirical performance observed by researchers measuring TCP throughput on high speed, high delay network paths [15]. Examining the leftmost set of bars shows that all of the proposed TCP replacements provide significantly higher throughput for movies transferred on VF paths, with FAST and FAST-AT providing highest throughput, followed by CTCP.

To construct a fuller picture comparing throughput for flows using proposed TCP replacements and competing flows using standard TCP, we conducted a rank analysis across all 24 flow groups identified in Table 6 and all 32 parameter combinations. For each pair of flow group and condition, we ordered throughputs achieved by each proposed TCP replacement from high (7) to low (1). Similarly, we ordered throughputs achieved by standard TCP when competing against

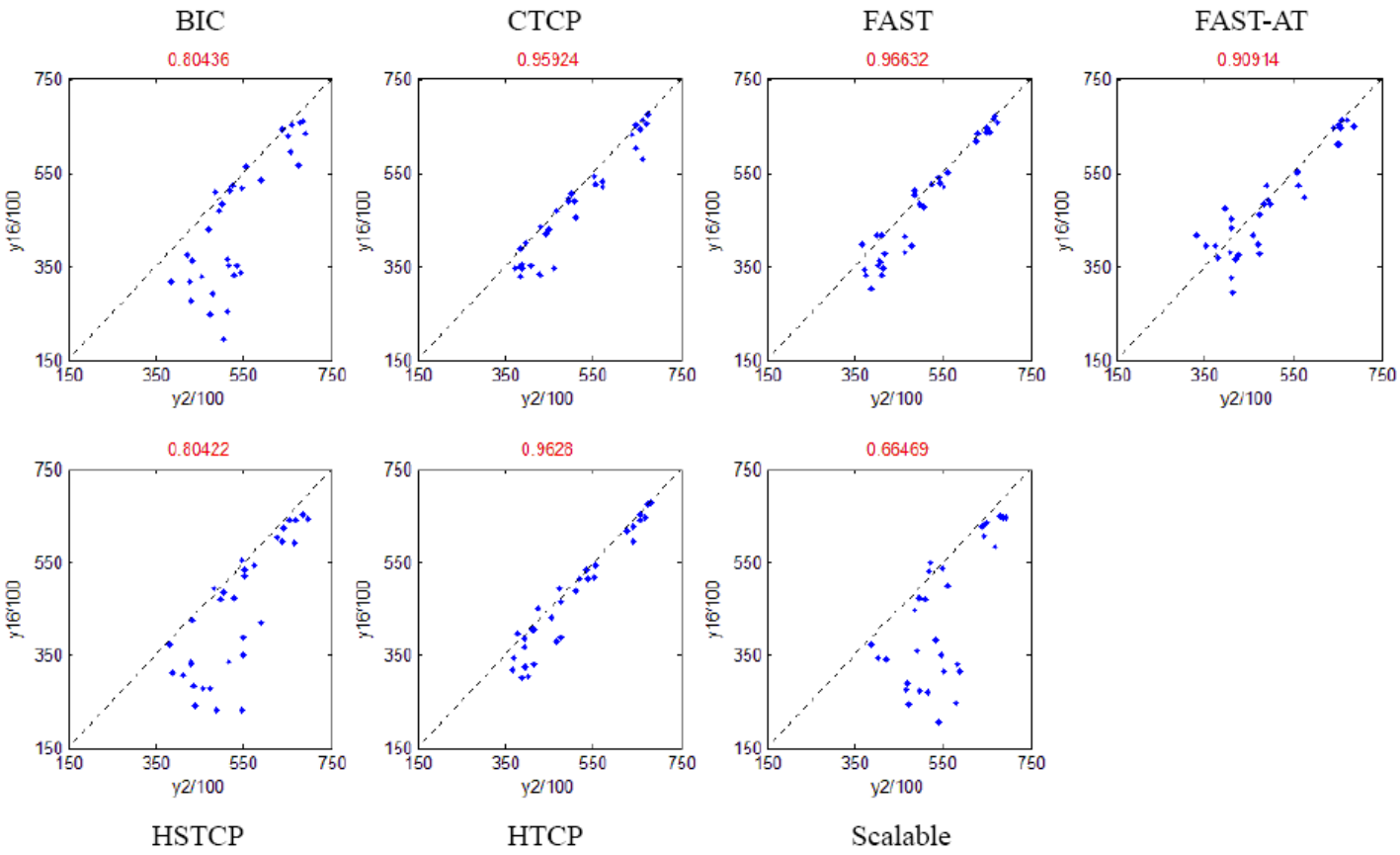

Figure 24: Seven biplots, where each biplot compares throughput (in units of 100 segments per second) for movie flows transferred over very fast paths with fast interface speeds using a proposed TCP replacement (x axis) with throughput on identical competing flows using standard TCP

each proposed TCP replacement. Then we averaged the rank related to each proposed TCP replacement, and we also computed the standard deviation in rank. We would then expect an ideal proposed TCP replacement to achieve high ranking (i.e., give good throughput for flows using the replacement and for standard TCP flows competing with the replacement) and low standard deviation (i.e., provide good throughputs across all flow groups and conditions). An ideal TCP replacement would be located in the lower right corner on a plot of average rank (x axis) vs. standard deviation in rank (y axis).

We conducted the rank analysis separately for experiment #2a (high initial *sst*) and #2b (low initial *sst*). We then plotted the average rank (x axis) against the standard deviation (y axis) for each of the experiments. For experiment #2a, Fig. 26 plots the average and standard deviation in rank across all flow groups and conditions for each proposed TCP replacement. Here the news is mixed: HTCP and CTCP provided the highest ranking throughputs, but also exhibit relatively high variability in standard deviation. This indicates the HTCP and CTCP provided good throughputs for many flow groups under most conditions, but there are some flow groups or conditions for which they do not perform very well. A look at the detailed rankings indicates that HTCP and CTCP provided highly ranked throughputs for small file sizes and for competing TCP flows, but they do not rank as high as some of the other proposed TCP replacements on larger files.

For experiment #2b Fig. 27 plots the average and standard deviation in rank for each proposed TCP replacement. Here, CTCP continued to rank quite highly, while also reducing the standard deviation in rank over the case of a high initial *sst*. FAST-AT, which was third highest ranked in Fig. 26, ranked nearly as high as CTCP, but incurred a higher standard deviation in rank. BIC, HSTCP and Scalable TCP exhibited low ranks in both Figs. 26 and 27.

In experiment #2c, we repeated experiment #2a, but increased network speed and size by an order of magnitude. We took this step to ensure that the results would not change with increasing network size and speed. The results for experiment #2c generally matched those from experiment #2a. FAST-AT, CTCP and HTCP provided the three highest ranking throughputs, but in experiment #2c FAST-AT proved highest

ranking of the three and had the lowest standard deviation in rank.

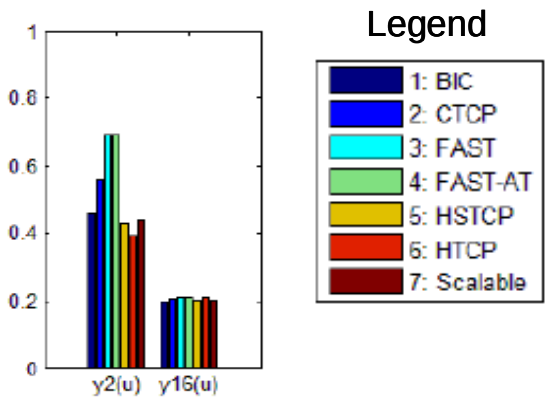

Figure 25: Fraction (y axis) of maximum available throughput achieved when transferring movies over uncongested paths with a maximum rate of 1 Gbps – each of the leftmost seven bars – y2(u) – represents flows using one of the proposed replacements for TCP (see legend), while each of the rightmost seven bars – y16(u) – represents flows using standard TCP and competing with flows using one of the proposed TCP replacements (see legend)

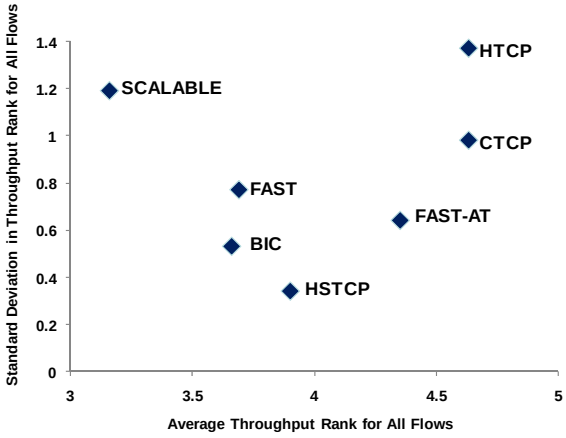

Figure 26: Average throughput rank (x axis) vs. standard deviation in throughput rank (y axis) for flows using each proposed TCP replacement and for competing flows using standard TCP – high initial *sst*

Here, we presented only some key results from our congestion control experiments. For more details, especially with respect to questions of causality, we refer the reader to our full study [1]. In the discussion that follows we draw on information presented in that study, as well as the subset presented in this paper.

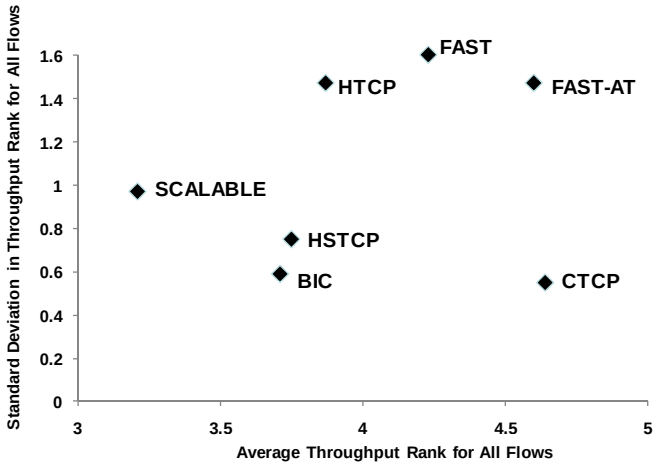

Figure 27: Average throughput rank (x axis) vs. standard deviation in throughput rank (y axis) for flows using each proposed TCP replacement and for competing flows using standard TCP – low initial *sst*

7 DISCUSSION

As demonstrated above, our methods advanced the state of the art in evaluating Internet congestion control procedures, allowing exploration of proposed TCP replacements under a wide range of conditions for a network of substantial size. Our methods also yielded another significant advance over the previous state of the art, where researchers decided a priori on what basis proposed TCP replacements would be compared. Our investigation began with a single, somewhat vague, question: would it be safe to deploy various proposed TCP replacements on the Internet? Subsequently, the multidimensional data analysis methods we applied enabled us to identify key characteristics that differentiate proposed TCP replacements. We found these characteristics from analysis of experiment data, and without a priori intent. Thus, our analysis methods led us to uncover the measures on which proposed congestion control procedures *should* be compared. Our analysis methods also enabled us to identify a specific combination of conditions that must hold for users to realize improved throughput from proposed TCP replacements. We discuss these outcomes below.

7.1 Key Differentiators

Our results showed that proposed congestion control procedures for the Internet can be distinguished by three main characteristics: (1) increase rate, (2) loss/recovery processing and (3) fairness. We address each of these topics in turn.

7.1.a Increase Rate. One of the key questions for any data transport protocol is: How fast can the maximum available transfer rate be achieved on a network path? Assuming no congestion protocols that can quickly attain the maximum rate will spend the largest portion of a file transfer at that rate. Each TCP flow begins without any knowledge of the maximum available transfer rate. For this reason, TCP specifies an initial slow start process where the source transmits slowly but then,

as feedback arrives from a receiver, quickly increases the transmission rate until reaching a specified (initial) *sst* or encountering a loss. This initial slow start process is not altered by any of the proposed TCP replacements that we studied.

Assuming no (or low) congestion, the setting of the initial *sst* can be quite important when comparing throughputs experienced by users on TCP flows with throughputs for users on flows operating under other congestion control procedures.[3] When initial *sst* is set arbitrarily high, on average all flows achieve maximum transfer rate with the same quickness. Under such situations, the throughput seen on TCP flows and flows running alternate procedures appears quite comparable. Flows transporting short files (e.g., Web objects and document downloads) tend to complete while in the initial slow start phase, which means that alternate congestion control procedures (restricted to the congestion avoidance phase) do not operate. Even flows conveying long files can operate for extended periods under initial slow start because such flows do not enter congestion avoidance until encountering a segment loss.

When initial *sst* is set low (e.g., 43 segments) all of the proposed TCP replacements that we studied increase transmission rate more quickly than the linear increase provided by standard TCP. Thus, under low congestion, when *sst* is set low compared to the size of files transferred users on TCP flows will see much lower throughput than users on flows operating under the proposed TCP replacements. The larger the file sizes being transferred the larger the throughput advantage of the proposed TCP replacements, which each provide varying degrees of improvement over standard TCP. These throughput variations can be tied directly to the speed with which each of the proposed procedures reaches maximum transmission rate.

Under conditions of heavy congestion the setting of initial *sst* matters less because initial slow start terminates upon the first segment loss and then a flow enters congestion avoidance, which is where the proposed TCP replacements differ from standard TCP. In such situations, the main difference in throughput experienced by users relates to the loss/recovery procedures defined for each of the proposed TCP replacements. We turn to this topic next.

7.1.b Loss/Recovery Processing. Two key questions arise when a data transport protocol experiences congestion. (1) How much should the protocol reduce transmission rate? (2) How quickly should the protocol increase transmission rate after a reduction? Standard TCP congestion avoidance procedures reduce transmission rate by one-half on each segment loss. Subsequently, TCP congestion avoidance procedures linearly increase transmission rate. The proposed TCP replacements that we studied specify various algorithms for transmission rate reduction and for subsequent rate increase.

One group of proposed procedures (Scalable TCP, BIC[4] and HSTCP) reduce transmission rate less than standard TCP after a segment loss. As a result, these procedures tend to retain a higher transmission rate and associated buffers than standard TCP. Smaller rate reduction can allow these procedures to provide established flows with higher throughputs following segment losses. We found this effect to increase with increasing loss rate and also file size. In addition, Scalable TCP, BIC and HSTCP can be somewhat unfair (as explained below) to algorithms (such as TCP) that exhibit a more reduced transmission rate following a loss, as well as to flows that have had insufficient time to attain a high transmission rate prior to a loss.

A second group of proposed procedures (CTCP, FAST and FAST-AT) reduce transmission rate in half following a loss. HTCP reduces transmission rate variably, between 20 and 50 %, depending on conditions. To obtain higher throughput, these algorithms increase transmission rate more quickly than standard TCP following a rate reduction. The rate of increase varies among the procedures. Typically, HTCP and CTCP are less aggressive than FAST and FAST-AT when increasing transmission rate after a reduction. Though, FAST-AT will be less aggressive than FAST when sufficient congestion exists to force a reduction in the α parameter. An aggressive rate increase following a rate reduction can induce additional losses on a path. Where such losses affect TCP flows, the TCP linear recovery procedures lead to lower throughputs. Under severe congestion, CTCP and HTCP can provide better throughput than FAST and FAST-AT, which underperform standard TCP.

In areas and at times of extreme congestion, most of the proposed TCP replacements we studied include rules to adopt standard TCP congestion avoidance behavior. These rules appear motivated by the theory that, when congestion is sufficiently severe, existing TCP behavior provides the best approach to fairly share limited available transmission capacity. The most typical technique employed is to set a low-window threshold. When the *cwnd* is below the threshold then standard TCP congestion avoidance procedures are used. When *cwnd* is above the threshold then replacement congestion avoidance procedures are used. Specific values for the threshold vary among the proposed TCP replacements. The combination of different thresholds with different file sizes can lead to modest variation in user throughputs among the proposed TCP replacements.

HTCP handles adaptation to TCP procedures somewhat differently than the other proposed TCP replacements we investigated. After a loss, HTCP adopts linear rate increase for

[3] Note that on real TCP flows receivers may convey a receiver window (*rwnd*) that can restrict throughput quite severely because sources pace transmission based on the minimum of the congestion window (*cwnd*) and *rwnd*. The following may hold: *rwnd* < *cwnd*. In our studies, we assume an infinite *rwnd* in order to compare the effects of congestion control procedures adjusting the *cwnd*. The throughput on many TCP flows in a real network might well be constrained by *rwnd*. In such cases, the proposed TCP replacements we studied would provide little advantage over standard TCP.

[4] Note that on repeated losses occurring close in time, BIC can reduce *cwnd* substantially more than standard TCP; thus, on paths with very severe congestion BIC can actually provide lower throughput than TCP and can also occupy fewer buffers.

a time. The time period is an HTCP parameter, set in these experiments to one second. We found that HTCP then adapts to TCP linear increase after every loss, regardless of file size or *cwnd* value. For larger files, which tend to have higher *cwnd* and to experience more losses during transmission, this approach tends to lower throughput significantly relative to other proposed TCP replacements, which do not adopt periods of linear increase after every loss.

FAST and FAST-AT do not use standard TCP congestion avoidance procedures under any circumstances. In times and areas of heavy congestion, failure to adopt less aggressive rate increase can lead to oscillatory behavior and to an associated increase in loss rate. Increased losses lead to lower user throughputs. FAST-AT does somewhat better under heavy congestion because the α parameter can be lowered, causing less aggressive rate increases. Still, under many conditions, FAST-AT exhibits a similar increased loss rate to FAST.

7.1.c TCP Fairness. TCP fairness denotes the effect where competing flows transiting a shared bottleneck path in the Internet all receive an equal share of available throughput. Comparing proposed TCP replacements with respect to TCP fairness can be somewhat difficult because the proposed replacements are designed to give better throughput than standard TCP for large file transfers on high-speed, long-delay paths. Thus, for example, all of the proposed TCP replacements can increase transmission rate more quickly than standard TCP given low initial *sst* and large file sizes. Further, all proposed TCP replacements take steps to provide loss/recovery improvements over standard TCP congestion avoidance procedures. On the other hand, most of the proposed TCP replacements take steps to adopt TCP congestion avoidance procedures when congestion is sufficiently high. Given these factors, one would expect all the proposed TCP replacements to provide better throughput than standard TCP under optimal conditions, and to perform no worse than standard TCP under suboptimal conditions. The usual measures of TCP fairness do not apply in such circumstances because they would tend to measure how much of a throughput advantage given proposed replacements provide over standard TCP. Instead, we measured *relative* TCP fairness by ranking the average throughput achieved by standard TCP flows when they competed with each proposed TCP replacement under the same conditions. We considered the average rank across four file sizes: Web objects, documents, software service packs and movies. In this way, we could elicit the relative TCP fairness of the proposed TCP replacements.

We found that CTCP and HTCP were most fair to TCP flows. We found FAST-AT third fairest to TCP flows under high initial *sst*. Under low initial *sst*, FAST-AT, because of its quick increase in transmission rate after passing the initial *sst*, proved more unfair to TCP flows. Injecting more FAST-AT segments into the network induced more losses in TCP flows, which could not recover as quickly.

We found Scalable TCP, BIC and FAST to be most unfair to standard TCP flows. Established Scalable and BIC flows for large files tended to maintain higher transmission rates after losses, while competing TCP flows cut transmission rates in half. By maintaining higher transmission rates and, thus, more segment buffers, Scalable and BIC flows induced more losses in TCP flows. FAST could recover more quickly from losses than TCP flows and so FAST flows could occupy more buffers and induce more losses in TCP flows. In addition, because of its quick increase in transmission rate upon entering congestion avoidance, FAST exhibited unfairness under low initial *sst*.

HSTCP appeared moderately fair to TCP flows, especially under conditions of lower congestion and under low initial *sst*. HSTCP showed TCP unfairness, similar to Scalable TCP, under conditions of heavy congestion.

We believe that Scalable TCP, BIC and HSTCP could also be unfair to flows that are newly arriving. Given that some large flows operating under Scalable TCP, BIC and HSTCP have established relatively high transmission rates and associated large buffer states and given that newly arriving flows induce losses, the established flows will not reduce transmission rate very much and will maintain large buffer states. The newly arriving flows will be forced into congestion avoidance on the loss. Further, Scalable TCP and HSTCP do not increase transmission rate very fast early in a flow's life, so newly arriving flows of these types can face difficulty increasing transmission rate.

7.2 Utility Bounds of Proposed TCP Replacements

We showed that proposed TCP replacements could provide increased throughput for users, but only under specific, bounded circumstances. First, the *rwnd* must not be constraining flow transmission rate. Second, a flow must be using a relatively low initial *sst*. Third, a flow must be transmitting a large file. Fourth, a flow's segments must be transiting a relatively uncongested path (i.e., experiencing only sporadic losses) or else users must be willing to accept marked unfairness (e.g., as seen with Scalable TCP) in trade for increased throughput.

7.3 Safety

Are there significant costs that might offset the modest benefits associated with deploying proposed TCP replacements? We can answer this question only in part because we simulated networks where sources used either a single congestion control regime or where some sources used a selected TCP replacement while other sources used standard TCP. There could be additional cautionary findings that arise from a heterogeneous mixture of proposed TCP replacements. We postpone such investigations to future work.

For most proposed TCP replacements, under most conditions we found little significant change in macroscopic network characteristics. One exception relates to FAST and FAST-AT. In spatiotemporal regions with high congestion, where there were insufficient buffers to support the flows transiting specific routers, FAST and FAST-AT exhibited oscillatory behavior where the flow *cwnd* increased and

decreased rapidly with large amplitude. Under these conditions, the network showed increased loss and retransmission rates, a higher number of flows pending in the connecting state and a lower number of flows completing over time. We recommend additional study of FAST and FAST-AT prior to widespread deployment and use on the Internet.

8 CONCLUSIONS

The Internet consists of millions (someday billions) of interconnected components that may be changed independently. For example, every time vendors of major operating systems introduce software updates, millions of users download new software modules into computers connected to the Internet. As another example, users may download software to support new functions, such as social networking or distributed gaming. At the current state of the art, system designers lack techniques to predict global behaviors that may arise in the Internet as a result of interactions among existing and altered software components. Similarly, hardware faults and unexpected usage patterns may occur within the Internet. Engineers have insufficient methods and tools available to forecast global behaviors and resulting effects on individual users. The study described here aimed to improve existing knowledge about a combination of methods and tools that could be applied to understand and predict behavior in such complex information systems.

To give our study a concrete context, we selected a challenging problem of current interest and relevance for the Internet at large. Specifically, we studied likely consequences for macroscopic behavior and for individual users should any of several proposed mechanisms be introduced to augment or replace congestion control procedures in standard TCP, which is currently deployed to regulate the rate of information transfer among computers connected to the Internet. Previously, such proposed changes have been studied on individual long-lived flows using analytical methods and also studied using simulation and empirical measurements in small topologies with limited types of data traffic. Though researchers and engineers would like to predict the effects of such changes on macroscopic behavior and on individual users, no techniques were previously available to make such extrapolations to large, fast topologies transporting hundreds of thousands of simultaneous data transfers of various sizes under a wide range of network conditions. The study discussed here applied modeling and analysis techniques to make such extrapolations for seven proposed replacements for standard TCP congestion control procedures.

We applied techniques often used by scientists at NIST when studying physical systems. First, we proposed an abstract simulation model, in this case representing a data communications network. Second, we adopted 2-level-per-factor experiment designs, which considered each parameter at only two values, as compared with the billion or so values that each parameter could possibly take on. Third, we leveraged orthogonal fractional factorial (OFF) experiment designs that enabled us to model a sparse but balanced set of parameter combinations spread widely throughout the space of possible combinations. Reducing the number of parameters, parameter levels and combinations enables feasible simulation of large networks under a wide range of conditions. Third, we used a variety of statistical analysis and visualization techniques designed to explore multidimensional data sets. We demonstrated that our combination of modeling and analysis techniques allowed us to predict the influence of seven proposed TCP replacements on macroscopic network behavior and on individual user experience.

Future work remains to apply our methods to large distributed systems in other domains, as we are doing currently with respect to infrastructure Cloud computing systems. Early returns suggest that the methods described here transfer quite readily among various types of large distributed systems.

ACKNOWLEDGMENTS

The authors thank the Complex Systems program in the NIST Information Technology Laboratory for funding this work.

REFERENCES

[1] Mills, K, Filliben J., Cho D., Schwartz E.and Genin D., (2010) Study of Proposed Internet Congestion Control Algorithms, NIST Special Publication 500-282, May 2010, 534 pages.

[2] Stevens, W. R. (1994) TCP/IP Illustrated, Volume 1: the Protocols, 1st edition, Addison-Wesley Professional, 600 pages.

[3] Cowie, J., Liu, H. Liu, J., Nicol, D. and Ogielski, A. (1999) "Towards Realistic Million-Node Internet Simulations", *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, June 28-July 1, 1999, Las Vegas, Nevada.

[4] Zeng. X., Bagrodia, R. and Gerla, M. (1998) "GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks", *Proceedings of the 12th Workshop on Parallel and Distributed Simulations* (PADS '98), pp. 154-161.

[5] Fall, K. and K. Varadhan, eds. (2009) The *ns* Manual. Available via <http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf> [accessed December 2, 2009].

[6] Yaun, G., D. Bauer, H. Bhutada, C. Carothers, M. Yukel and S. Kalyanaraman. (2003) Large-Scale Network Simulation Techniques: Examples of TCP and OSFP Models. In *SIGCOM Computer Communications Review*, 33:3, 27-41.

[7] Box, G., Hunter, J. and Hunter, W. (2005) Statistics for Experimenters, 2nd edition, Wiley, 639 pages.

[8] Frey, D., Engelhardt, F. and Greitzer, E. (2003) "A role for 'one-factor-at-atime' experimentation in parameter design", *Research in Engineering Design*, Volume 14, Number 2, 65-74, DOI: 10.1007/s00163-002-0026-9, pp. 65-74.

[9] Fodor, I.K. (2002) A Survey of Dimension Reduction Techniques. Lawrence Livermore National Laboratory Technical Report no. UCRL-ID-148494.

[10] Jolliffe, I. T. (2002). Principal Components Analysis. 2nd ed. Springer Series in Statistics.

[11] Mardia, K.V., Kent, J.T., and Bibby J.M. (1995) *Multivariate Analysis.* Probability and Mathematical Statistics. Academic Press.

[12] Croarkin, C. Tobias, P., Filliben, J., Hembree, B. Guthurie W., Trutna, L., and Prins, J. (2006) "Grubbs' Test for Outliners", NIST/SEMATECH e-Handbook of Statistical Methods,

[13] The MathWorks. (2008) MATLAB® Statistics Toolbox™ 7 Users' Guide, 1749 pages.

[14] Chiu, D.-M. and Jain, R. (1989) "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", *Computer Networks and ISDN Systems*, 17:1, pp. 1-14.

[15] Li, Y.-T., Leith, D. and Shorten, R.N. (2007) "Experimental Evaluation of TCP Protocols for High-Speed Networks", *IEEE/ACM Transactions on Networking*, 15:5, pp. 1109-1122.

[16] Xu, L., Harfoush, K. and Rhee, I. (2004) "Binary Increase Congestion Control for Fast, Long Distance Networks", *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2004), vol. 4, pp. 2514- 2524.

[17] Tan, K., Song, J., Zhang, Q. and Sridharan, M. (2006) "A Compound TCP Approach for High-Speed and Long Distance Networks", *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2006), pp. 1-12.

[18] Wei, D. X., Jin, C., Low, S. H. and Hegde, S. (2006) "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Transactions on Networking*, 14:6, pp. 1246-1259.

[19] Floyd, S. (2003) HighSpeed TCP for Large Congestion Windows, RFC 3649, The Internet Society, 34 pages.

[20] Leith, D. and Shorten, R. (2004) "H-TCP: TCP for High-speed and Long-distance Networks", *Proceedings of the 2nd International Workshop on Protocols for Fast Long-Distance Networks*, 16 pages.

[21] Kelly, T. (2003) "Scalable TCP: Improving Performance in Highspeed Wide Area Networks", *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83-91.

[22] Blanc, A., Avrachenkov, K. and Collange, D. (2009) Comparing some high speed TCP versions under bernoulli losses. In *Proceedings of the International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNet 2009)*, 59-64.

[23] Genin, D. and V. Marbukh. (2009) Bursty fluid approximation of TCP for modeling Internet congestion at the flow level. In *Proceedings of the 47th Annual Allerton Conference on Communication, Control and Computing*, Paper ThD4.3.

[24] Baccelli, F., McDonald, D. and Reynier, J. (2002) "A mean-field model for multiple TCP connections through a buffer implementing RED", *Performance Evaluation*, 49(1/4):77–97.

[25] Jackson, T. and Smith, P. (2008) "Building a Network Simulation Model of the TeraGrid Network", *Proceedings of TeraGrid'08*.

[26] Leith, D., Andrew, L., Quetchenbach, T., Shorten, R., Lavi, K. (2008) "Experimental Evaluation of Delay/Loss-based TCP Congestion Control Algorithms", *Proceedings of the 6th International Workshop on Protocols for Fast Long-Distance Networks*.

[27] Lee, G., Lachlan, A., Tang, A. and Low, S. (2007) "WAN-in-Lab: Motivation, Deployment and Experiments", *Proceedings of the 5th International Workshop on Protocols for Fast Long-Distance Networks.*

[28] Arora, P., Wang, Y. and Rhee, I. (2009) "Netset: Automating Network Performance Evaluation", *Proceedings of the 7th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports*, pages 25-30, Tokyo, Japan, May 2009.

[29] Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K. and Lepreau, J. (2008) "Large-scale Virtualization in the Emulab Network Testbed", *Proceedings of the 2008 USENIX Annual Technical Conference*, pages 113–128, Boston, MA, June 2008

[30] White, B., Lepreau, J. Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C. and Joglekar, A. (2002) "An Integrated Experimental Environment for Distributed Systems and Networks", *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, pages 255-270, Boston, MA, December 2002.

[31] GENI – Exploring Networks of the Future
http://www.geni.net/

[32] Yuan, J. and Mills, K. (2006) "Simulating Timescale Dynamics of Network Traffic Using Homogeneous Modeling", *The NIST Journal of Research*, Volume 111, No. 3, May-June 2006, pp. 227-242.

[33] Mills, K., E. Schwartz and J. Yuan. (2010) "How to model a TCP/IP network using only 20 parameters", *Proceedings of the 41st Winter Simulation Conference*, IEEE, pp. 849-860.

[34] Kratz, M., Ackerman, M., Hanss, T. and Corbato, S. (2001) "NGI and Internet2: Accelerating the Creation of Tomorrow's Internet", in MEDINFO 2001, IOS Press, Vol. 84, pp. 28-32.

[35] Bush R. and Meyer D. (2003) Some internet architectural guidelines and philosophy, RFC 3439.

[36] Appenzeller G., Keslassy, I. and McKeown, N. (2004) "Sizing Router Buffers", *Proceedings of ACM SIGCOMM*.

[37] Microsoft® Help and Support. (2006) "TCP/IP and NBT configuration parameters for Windows XP", Article ID 314053, Revision 3.2,
http://support.microsoft.com/kb/314053.