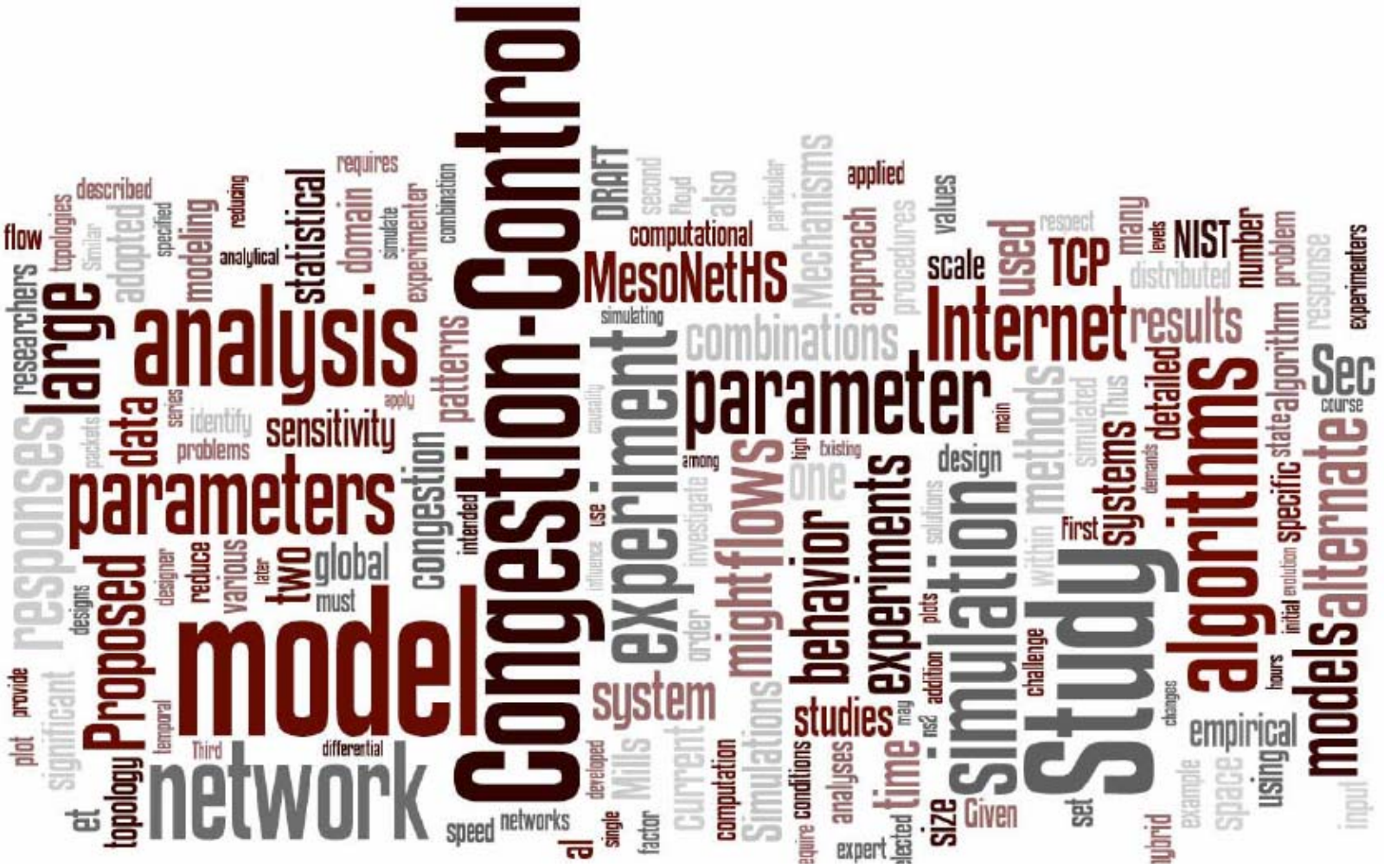
Chapter 2 – Method and Related Work



2 Method and Related Work

The work described in this report supports an overarching goal to develop and evaluate a coherent set of methods that can be applied to understand behavior in large distributed systems, such as the Internet, computational grids, service-oriented architectures and computing clouds. Large distributed systems may exhibit emergent behaviors, which are global behaviors arising from independent decisions made by many simultaneous actors, which adapt their behavior based on local measurements of system state. As a result of actor adaptations, system state shifts, influencing subsequent measurements made by the actors, which leads to further adaptations. This continuous cycle of measurement, adaptation and changing system state produces a time-varying (emergent) global behavior that influences performance experienced by individual actors within specific spatiotemporal regions of a large distributed system. For this reason, any proposed changes in decision algorithms taken by actors must be examined within the context of a large spatiotemporal scale in order to predict the effects of such algorithms on overall system behavior, as well as the resulting implication for individual actors.

In this study, we develop methods to investigate global system behavior within the context of a challenge problem: comparing selected proposed changes to the standard congestion control algorithm [9-10] for the Internet. As we show later, in Chapter 10, using our methods we were able to draw conclusions (1) about likely network-wide behaviors and user experiences that may arise if the Internet adopts any one of the algorithms we studied and (2) about the efficacy of the methods we used. In this chapter, we introduce the challenge problem, describe the current state-of-the-art techniques used to address the problem and outline a proposed advance in the state of the art. We consider some approaches that might be adopted to achieve our intended improvement in practice and then we explain the approach we adopted for the current study. We identify five hard problems we had to solve to develop our approach and we discuss some possible solutions to the problems and identify the solutions we adopted for the current study. We conclude with an argument that the methods we develop and apply in the current study should be generally applicable to a wide array of large distributed systems.

2.1 Challenge Problem

The fundamental design of the Internet protocol suite [3] assumes that network elements, such as routers, are relatively simple – receiving, buffering and forwarding packets among connected links and dropping packets when buffers are insufficient to accommodate arriving packets. Under this assumption, computers connected to the Internet must implement decision algorithms to pace the rate at which packets are injected into the network. Such decision algorithms, known typically as congestion control mechanisms, operate independently for each network flow between a source and receiver. The overall network, with a goal of achieving satisfactory service and a fair distribution of resources among all simultaneously active flows, relies upon each network source to measure congestion and then adapt the rate at which the source injects packets into the network – injecting faster when congestion is low and slower when congestion is high. Thus, congestion in the Internet is an emergent property of the simultaneous operation of many independent sources.

In current practice, congestion control mechanisms are implemented as part of the transmission control protocol (TCP) that operates within every computer attached to the global Internet. While TCP congestion control procedures have proven quite successful [2] at achieving desired global properties, numerous researchers [46-51] have postulated potential changes in relationships among bandwidth and propagation delay as the speed of network links increases toward 10s and 100s of gigabits per second (Gbps). Under the envisioned circumstances, researchers predict that TCP congestion control procedures will prove insufficient, leading to substantial underutilization in network resources and preventing end users from achieving high transfer rates, potentially reaching or surpassing one Gbps. These predictions have stimulated researchers to propose alternate congestion control procedures [52-61] that might achieve higher network utilization and better user performance as network speeds increase. Given the increasing number of proposals, interest is growing [62-68] in developing procedures to fairly and effectively evaluate properties of the various proposals. Evaluating the implications of adopting proposed changes to TCP congestion control procedures requires investigating global behaviors that result when such changes are deployed on a large scale throughout an Internet-like network. This is the challenge problem we tackle within the current study.

2.1.1 Current State of the Art

As part of proposing changes to standard congestion control procedures, researchers typically model, simulate and implement prototypes and then explore how candidate congestion control mechanisms might affect the Internet and its users. In general, researchers investigate candidate algorithms using three primary approaches: (1) empirical studies, (2) simulation studies and (3) analytical studies. In this section we outline and critique the current state of the art with respect to these approaches.

2.1.1.1 Empirical Studies. A fundamental approach to studying proposed changes to TCP congestion control procedures involves deploying a few computers, acting as sources and receivers, connected to Gigabit Ethernet switches in a dumbbell topology, where the network links are represented by a single computer that can be parameterized with specific bottleneck speed, buffer size and propagation delay. Several, typically two to ten, long-lived flows share the bottleneck path in the dumbbell topology and various measurements are made regarding traits such as fairness of resource allocation, responsiveness to changing conditions and link and buffer utilizations. Additional sources are often added to investigate the response of the proposed congestion control algorithms in the presence of background traffic, sometimes TCP flows and sometimes user-datagram protocol (UDP) streams. Usually the background traffic crosses the bottleneck link in an orthogonal direction to the long-lived flows. Several examples of such studies appear in the literature [65-68] and the basic approach is being considered by a group of researchers [62] intending to standardize procedures to characterize proposed changes to TCP.

Simple empirical studies have several merits. First, a topology involving few computing elements can be constructed conveniently in a laboratory setting. Further, the fundamental characteristics (speed, buffer size and propagation delay) of a bottleneck path can be reliably established to provide a suitable basis for head-to-head comparisons of alternate congestion control algorithms in identical, controlled situations. Third,

empirical studies investigate actual implementations of proposed algorithms as would be distributed in code used by computers on the Internet; thus, there is no room for modeling error. Of course, code bugs can exist; however, those code bugs, if undiscovered, would be actually deployed on the Internet. Fourth, as demonstrated in a recent study [67], empirical measurements in a simply topology can reveal behavioral properties that might well have significant implications. For purposes of our study, the major shortcoming of simple empirical studies is an inability to investigate the influence on global network behavior should proposed congestion control algorithms be adopted on a large scale. As we discuss below in Sec. 2.2.1, some researchers are investigating techniques to support configuration of larger empirical networks, which might be used to study global network behavior.

2.1.1.2 Simulation Studies. Another approach is to implement proposed congestion control algorithms within a simulation modeling framework and then to construct (or generate) simulated topologies representing large networks and conduct experiments to evaluate global behavior and user experience. In fact, many of the proposed congestion controls we investigate in the current study have been implemented within a widely accepted simulation framework, *ns2* [79].

Simulation provides a convenient vehicle for defining controlled experiments that can be used to compare alternate congestion control algorithms, head-to-head, with respect to potential effects on global network behavior. Further, using simulation, an experimenter can measure many network-wide properties that might be difficult or impossible to measure in a large, empirical network. Several simulation studies [46, 48, 54-56, 61, 64] have been conducted using the *ns2* framework, but all of these studies have simulated small topologies with a limited number of flows, perhaps up to hundreds. Simulations at such small scales are unlikely to reveal the influence of alternate congestion control algorithms on global network behavior. Large topologies must be simulated while simultaneously transporting up to hundreds of thousands of active flows. The computational and memory demands of *ns2* are significant, which discourages experimenters from attempting large simulations. We certainly decided that we could not achieve our goals using *ns2* simulations.

There exist many other network simulation frameworks [76-78, 80-83], both commercial and academic, that might be adopted. Most of these frameworks would need to be expanded to include simulations of the alternate congestion control algorithms. Still, at least one of these simulation frameworks has been used in an experiment transporting up to 10^5 TCP flows [69]. Unfortunately, configuring such a simulation experiment requires setting values for thousands of parameters, which must be managed by a database. To define a comprehensive set of experiments, such as we envisioned, would require significant, perhaps insurmountable effort. Further, we would need to specify settings for many parameters that require values but that would not necessarily be germane to our experiments. As we discuss below in Sec. 2.2.2, some researchers are investigating techniques to support faster simulations of large networks, which might be used to study global network behavior.

2.1.1.3 Analytical Studies. A third approach is to construct a model as a system of differential equations that represent network flows as fluids rather than as a sequence of

discrete packets. A fluid approximation is justified using an argument that when the number of packets flowing through routers is very large and when the packets move at very high speed then packet flows can be well approximated by a smoothly varying continuous data steam with an average flow rate. Continuous systems can be modeled with differential equations. Several researchers [107, 112, 114, 119, 122-127] have proposed differential equation models for standard TCP congestion control procedures.

Existing differential equation models for TCP exhibit some significant shortcomings for our purposes. First, existing models yield inaccurate results [112-113], which derive largely from difficulties in modeling the loss-estimation function in the differential equations. Existing models have tried estimating loss probability using an M/M/1/B queuing system; however, when considering many flows transiting a single router, such models give inaccurate results for many parameter combinations. Second, the difficulty of estimating loss probability increases with increasing complexity in network topology. For this reason, employing differential equations to model network flows in large topologies has received little attention. Third, current differential equation models treat only standard TCP congestion control procedures. For our purposes, such models must be augmented to include congestion control procedures for the set of alternate congestion control algorithms we studied. As we discuss below in Sec. 2.2.3, we are investigating techniques to model network flows with differential equations that have improved accuracy and that incorporate alternate congestion control procedures. When combined with fluid-flow simulators [73], our extended models might prove applicable to study global network behavior under various congestion control regimes.

2.1.2 Proposed Advance in the State of the Art

As described in the preceding overview, the current state of the art in modeling and analysis of distributed systems is limited to relatively small scales. We propose to define and apply modeling and analysis techniques that enable measurement and investigation of global behavior and actor experience in relatively large models of distributed systems. In this study, we explain and investigate our proposed modeling and analysis techniques in the context of comparing alternate congestion control algorithms suggested for use on the Internet. As a result of our investigation, we provide new insights into likely global behavior and user experience should the Internet deploy any of the alternate algorithms we study. Further, we characterize strengths and weaknesses of the modeling and analysis methods we use. Finally, we suggest additional directions for future investigations in modeling and analyzing global behavior in large distributed systems. We expect the methods illustrated in our study to advance the state of the art in modeling and analyzing large distributed systems because we believe our methods can be applied beyond the current study to consider other large-scale feedback processes, such as might arise in computational grids, computing clouds and service-oriented architectures.

2.2 Potential Approaches

In this section, we consider some potential approaches to achieve our goal to define and apply modeling and analysis techniques that enable measurement and investigation of the global behavior and actor experience in relatively large models of distributed systems. We discuss the possibility of expanding either empirical, simulation or analytical studies in order to consider larger systems than possible within the current state of the art.

2.2.1 Expanded Empirical Studies

In increasing numbers, researchers are investigating management frameworks that allow experimenters to configure collections of hardware and software components into specified, reproducible topologies that enable empirical experiments with distributed systems. Emulab [99], perhaps the earliest instance of such a framework, provides access to hundreds of configurable nodes made available to experimenters. Emulab has been replicated at several sites around the world. Perhaps inspired by Emulab and its clones, PlanetLab [101] has connected a network of sites across the globe that allocates hardware and software components for configuration into distributed-system topologies for controlled experiments. Similar management frameworks, though limited to controlling configurations within a single laboratory, have been developed by various university researhers [65, 97]. While most of these efforts include innovations with respect to system configuration and experiment control, the scale of topologies that can be constructed is limited to a few hundred nodes and often such a topology exhausts the resources of a given facility. In an effort to overcome such scaling concerns, some researchers have reported progress in emulating virtual topologies an order of magnitude larger than the available physical hardware [98].

Recognizing the limitations of current facilities for configuring topologies in support of experiments with large distributed systems, a community of researchers is pursuing GENI [100] (Global Environment for Network Innovations), a project sponsored by the National Science Foundation (NSF). GENI aims to create a virtual laboratory for exploring future distributed systems at scale. The GENI facility promises to provide a platform on which we could conduct empirical investigations into global implications of deploying various congestion control algorithms. Unfortunately, GENI is in early stages of development, so the timing is inopportune for our study. We hope that we can use GENI at some later time to validate findings from the experiments we conducted.

2.2.2 Expanded Simulation Studies

While explaining that simulation models hold a key position when attempting to understand behavior in large distributed systems, Paxson and Floyd [72] identify several impediments to simulating the Internet. First, the Internet is big; simulating a model at scale can require significant, perhaps impractical, computational resources. Second, the Internet is diverse with respect to administrative policies and technologies deployed. Third, the Internet is evolving in size, technologies, traffic patterns and applications. While these impediments might prove difficult to overcome, Paxson and Floyd discuss some possible coping strategies. First, they suggest that researchers search for invariants that can reduce the parameter space of the models. They identify two candidate invariants: (1) model session arrivals with a Poisson distribution and (2) model session sizes with heavy-tailed distributions such as log-normal or Pareto ($\alpha < 2$). Second, Paxson and Floyd suggest judiciously exploring the parameter space of a simulation model in order to identify parameters to which the model is sensitive. In a subsequent paper, six years later, Floyd and Kohler [70], critique the continued poor state of Internet modeling and admit that the research community does not know whether their models are valid. Floyd and Kohler prescribe four steps that could improve the situation. First, models should be limited in scope to the specific research questions under investigation in

particular studies. That is, the research community should abandon hopes of developing a single model of the global Internet. Second, any model used should be subjected to a sensitivity analysis in order to understand how parameter settings affect results. Third, with respect to important parameter settings, models should be compared against measurements in order to further increase confidence in real-world relevance. Fourth, a continuous program of measurements should be conducted with the aim of distinguishing between invariant and rapidly changing parameters in the real Internet. This enables models to be kept current and allows previous modeling studies to be placed in a temporal context.

The perceptive critique from Floyd and colleagues provides a context for considering current research into network simulation. Much of the current work revolves around developing improved simulation frameworks intended to provide a model of the global Internet. For example, a group of open-source developers is working on *ns3* [80], a replacement for *ns2* that is motivated by overcoming some perceived software limitations that make *ns2* difficult to use. Another group of researchers [82] is creating a parallel version of *ns2* in an effort to allow simulation of larger systems by dividing work among multiple processors. Constructing parallel network simulators has been a research topic for some time [69, 76, 83]; unfortunately, successful use of such simulators has proven elusive for network models. Some researchers [71] have begun to investigate hybrid models as a technique to reduce the resources required by network models, which might allow larger topologies to be simulated. Various other simulators [77, 78, 81] have been developed in order to teach university students about both networks and about software development.

While parallel simulators and hybrid models hold promise to increase the size of systems that can be simulated, few existing simulation research projects are motivated specifically to address the critique of Floyd and colleagues. Under some preliminary work leading up to the current study, Yuan and Mills [74] conducted a judicious search of a model parameter space to investigate the influence of various transport protocols on correlation structure in network traffic. The study, which adopted a cellular automaton model of a network of sources and receivers interacting within a grid topology, also adopted some of the invariants identified by Paxson and Floyd. Later, Yuan and Mills [75] converted the model to include a four-tier topology, which was used to study detection methods for distributed denial of service attacks within the Internet. The cellular automaton model, including the four-tier topology, was used to conduct preliminary experiments for the current study. Specifically, the model was subjected to a sensitivity analysis (using an approach explained in Chapter 4). Unfortunately, as explained in Sec. 3.1, scaling the cellular automaton model to represent a network with Internet-like speed and size proved computationally infeasible.

A hybrid network simulation model [71], combining discrete events with continuous approximation of discrete variables, may provide an attractive alternative because published computational and memory requirements appear quite promising. Specifically, the model appears to have required about 2 hours of processor time to simulate 30 long-lived flows operating for 11 simulated hours in a partial topological model of the Abilene backbone with 10 Gbps links. The published results for the hybrid model indicated the scenario was infeasible using *ns2*. Further, the hybrid model included many of the alternate congestion control models we studied. On the other hand, the

hybrid model included only a two-tier topology with sources and receivers connected directly to the backbone. In addition, the hybrid model did not include the existing measurement and parameterization capabilities included within our cellular automaton model. We did not explore the computational implications of expanding the hybrid model to add (access and point-of-presence) tiers to the topology, to expand the number of sources to hundreds of thousands, to introduce the TCP connection phase, to add various scenarios and to incorporate addition measurement code. Based on comparing the published computation requirements of the hybrid model and with the measured computational requirements of a discrete-event simulator (see Appendix B), we conclude that the hybrid model warrants further investigation (as discussed below in 2.5.1).

2.2.3 Expanded Analytical Studies

To employ analytical methods for our study, we needed to overcome three limitations of existing fluid-flow models: (1) improve accuracy in modeling queue evolution in order to obtain realistic estimates of loss probability, (2) construct differential equation models for alternate congestion control algorithms under study and (3) determine how to evaluate the resulting models in topologies of sufficient size and complexity. We believe we can leverage existing fluid-flow simulators [73], solved using numerical methods, to scale fluid-flow models to large networks. Solving the other two problems required more research, but we believed we could make some progress. Unfortunately, to complete our study in a timely fashion we needed solutions sooner rather than later. We decided to investigate analytical solutions to more accurately model queue evolution and then to construct differential equation models for some alternate congestion control algorithms. These investigations, documented in Appendix A, ran in parallel with our main study. At a later date we could use our analytical models to repeat our current studies and compare the predictions obtained and the resources required.

2.3 Selected Approach

Lacking a sufficiently large emulation facility and without an accurate fluid-flow model for the congestion control algorithms under study, we had little choice but to adopt simulation for our study. Following recommendations from Floyd and colleagues [70, 72], we aimed to reduce model scale and improve model quality by focusing the model on the study at hand, by adopting some recommended invariants, by conducting a sensitivity analysis of model parameters and by comparing key model behaviors with empirical measurements. We did not adopt *ns2*, or similar existing simulation frameworks, because the parameter spaces of such models are too large for our needs and because the computational and memory requirements are too costly for tractable simulations of systems of the size we envisioned. We also did not adopt our existing cellular automaton model because the computational costs made it infeasible to simulate networks of very high speeds and large sizes. On the other hand, the parameter space of our cellular automaton model was quite concise and appropriate for the studies we envisioned.

We decided to convert our cellular automaton model to a discrete-event simulation (DES), which we call MesoNet (see Chapter 3 for a description of the simulator). The DES simulator proved significantly faster than the cellular automaton, especially as simulated network size and speed increased. As described in Chapter 5, we

extended MesoNet, adding six alternate congestion control algorithms to the standard TCP congestion control procedures. We also found it necessary to add TCP connection-establishment procedures to the simulator. To allow exploration of flow dynamics under a range of network conditions, we adopted a single, heterogeneous topology for our study. As explained in Sec. 3.1.2, we modeled the topology after characteristics of existing network topologies and traits revealed by studies of topologies used by commercial Internet service providers. Given a concise set of model parameters, we applied statistically-based, experiment design techniques, as used typically in rigorous scientific and engineering studies. We then leveraged statistical properties of the experiment designs to conduct statistical analyses of response data. We say more regarding our approach below in Sec. 2.5, where we outline solutions (and possible alternatives) to five hard problems we had to solve in order to develop the details of our approach. First, though, we introduce the hard problems we faced.

2.4 Hard Problems

The approach we adopted for our study could not be developed and applied without devising solutions to five hard problems. We describe these problems below.

2.4.1 Model Scale

Simulating networks of the size and speed we envisioned presents two significant impediments: substantial computation and large parameter space, which combine to create a significant scaling problem. Models with thousands of parameters usually include many details. Such detailed simulators, while perhaps easy to relate to real systems, can require substantial computation to process each packet. The more packets processed the greater the computational time required for a simulation run. For networks of large size and high speed that are simulated over minutes or hours of operation, the number of packets per simulated second can prove quite high, so reducing per-packet processing time can substantially reduce computation demands for a given simulation run. A large parameter space requires simulating many runs to cover various parameter combinations. The computational requirements for simulating a specific combination multiplied by a large number of combinations can lead to an infeasible demand for computation. Reducing the number of runs required can substantially reduce computation demands for a particular simulation study. Beyond influencing computation demands, large parameter spaces require significant intellectual and practical effort from experimenters who must design and configure experiment runs. Experimenters must select the parameters of interest to vary for a given study. For other parameters, experimenters must determine fixed values for use throughout the study. Further, under a large parameter set, configuring parameters requires significant automation aid, such as databases or scripting. Even with automation, erroneous configurations can lead to incorrect experiment executions, which not only waste precious processing resources but also require significant intellectual effort and time to detect. A similar problem (see 2.4.3 below) can arise when simulations produce a large response space. Analyzing multidimensional data can prove intellectually challenging and can consume substantial processing resources. These problems of scale should be quite familiar to experimenters who use network simulations.

2.4.2 Model Validation

Whatever solution is adopted to reduce model scale, the resulting representation is likely more abstract than that provided by a typical detailed simulation model, such as *ns2*, and certainly less realistic than a deployed network. With a reduced model in hand, an experimenter faces the problem of establishing that the model is valid for purposes of an intended study. Often, experimenters compare results from two models, one more detailed, in order to establish confidence in a reduced model. Sometimes, experimenters compare predictions from simulations against predictions from widely accepted analytical models. Such comparisons between two models leave an uneasy feeling that perhaps both models might be wrong. Of course, the widely accepted model could be wrong and the newly created model correct, so changing the newly created model until it yields results aligned with the accepted model might be the wrong course. Further, even when two models are compared, experimenters typically consider only a small subset of parameter configurations. Fundamentally, when an experimenter produces a reduced-scale model, some means must be found to determine that the model is error free and that the model represents valid behaviors for purposes of the intended study. As software developers understand, producing error-free software is quite difficult; demonstrating that software is without error even more so. System modeling adds on top of that hard problem, the challenge of demonstrating a model is valid for its intended purposes.

2.4.3 Tractable Analysis

One of the significant advantages of modeling a large system using simulation is that any conceivable response can be measured at any time. Measuring responses from an empirical system can be much more difficult; impossible for some desired responses. In an analytical model, the level of detail may be insufficient even to represent various behaviors one might wish to measure. The measurement advantage of simulation can quickly introduce two challenges: identifying which responses are significant (or redundant) and analyzing large volumes of multidimensional response data. The first challenge might be rephrased as: What responses should one analyze? The second challenge might be rephrased as: Over what spatiotemporal extent should one analyze responses?

2.4.4 Causal Analysis

Analyzing measurement data from large systems allows various spatiotemporal patterns to be discerned. In general, the patterns appear from statistical analyses applied to various dimensions of the model parameter space. Existence of significant patterns can be detected and revealed to an experimenter using statistical analyses. Such patterns suggest that a model behaves in particular fashion under specified conditions. An experimenter usually desires to understand causality underlying significant patterns. Bridging the gap between statistical patterns and underlying causes represents a significant challenge with respect to any large system, including models of such systems.

2.4.5 Experiment Selection

Assuming existence of a valid, scalable simulation model where data can be analyzed tractably and causality can be established, a remaining challenge relates to selecting experiments that will probe a system in a manner needed to reveal key aspects of global

behavior that are germane to a particular study. Should an experimenter fail to include the necessary parameter values and scenarios then a simulation study may miss significant aspects of system behavior that would arise in an analogous real system.

2.5 Selected Solutions and Possible Alternatives

The modeling and analysis methods we developed and applied in the current study were intended to provide some level of solutions for the hard problems we described above. Below, we describe the solutions we adopted to address each problem. The combined solutions to the hard problems comprise the modeling and analysis methods we used, so we introduce our solutions in some detail. Where applicable, we also identify potential alternatives that might be used to address each problem. Before discussing our solutions, we provide a general mathematical introduction to the modeling state-space problem.

$$y_1, \ldots, y_m = \mathbf{f}(x_{1|[1,\ldots,k]}, \ldots, x_{n|[1,\ldots,k]})$$

Response State-Space **Stimulus State-Space**

n	Number of inputs (i.e., stimulus factors)
k	Factor range (i.e., number of values each factor can assume)
m	Number of outputs (i.e., responses)

Figure 2-1. The Modeling State-Space Problem

The mathematical transformation shown in Fig. 2-1 represents the modeling state-space problem as a mathematical equation transforming a set of input parameters into a vector of responses. This equation underlies the hard problems associated with model scale (2.4.1) and tractable analysis (2.4.3). The equation represents a simulation model as a function (**f**) returning a set of responses (y_1 to y_m) given a specified combination of input parameters (x_1 to x_n). Each input factor can take on a range (***k***) of values, often referred to in the experiment design literature as levels. The state-space transformation presents few problems when values of ***n***, ***k*** and ***m*** are small; however, for detailed network simulators the values can be large. For example, in a network model with 10^3 parameters, and assuming each parameter may be represented as a 32-bit integer[1], the stimulus state space would comprise (k^n =) $(2^{32})^{1000}$ combinations, which is on the order of 10^{9633}. Even if each simulation run can be made quite efficient, this is an infeasible parameter space to compute. The response state-space can also present a challenge when each of the ***m*** responses can be assigned to spatial partitions. For example, in a model with ***a*** actors each characterized by ***m*** responses, the response state-space becomes m^a. For a model with 10^5 actors and 20 responses, the response state-space becomes 20^{100000}. The response state-space can grow in other dimensions. For example, the ***m*** responses might be assigned to specific time intervals or to particular logical partitions. Spatial, temporal and logical partitions can be combined to further increase the response state-space.

[1] Model parameters, when represented using floating point notation, may take on even more values.

2.5.1 Scale Reduction

To constrain computational demands for our study, we adopted two main strategies: reduce the parameter space and use a two-level (or two-value) per factor orthogonal fractional factorial (OFF) experiment design method. Reducing the parameter space requires lowering the value of ***n***, while using a two-level OFF requires reducing the values of both ***n*** and ***k***. In two-level per factor experiments each parameter is assigned only 2 of its possible ***k*** values. Fig. 2-2 illustrates the theory underlying these strategies.

As a first step in reducing the scale of computational demands, a domain expert creates a model that can be specified with a reduced number (***n*** – ***r***1) of input parameters. One practical means to achieve this step is to construct a model that includes only parameters germane to an intended study. For us, this amounted to designing MesoNet (see Chapter 3), which can be parameterized with around 56 parameters, depending on how one chooses to count. Thus, assuming that a model such as *ns2* requires 10^3 parameters to configure the experiments we might design, our initial reduction factor was quite impressive (***r***1 = 944). Unfortunately, even with such a large reduction, the parameter state space remains infeasible to compute, as shown in Fig. 2-3.

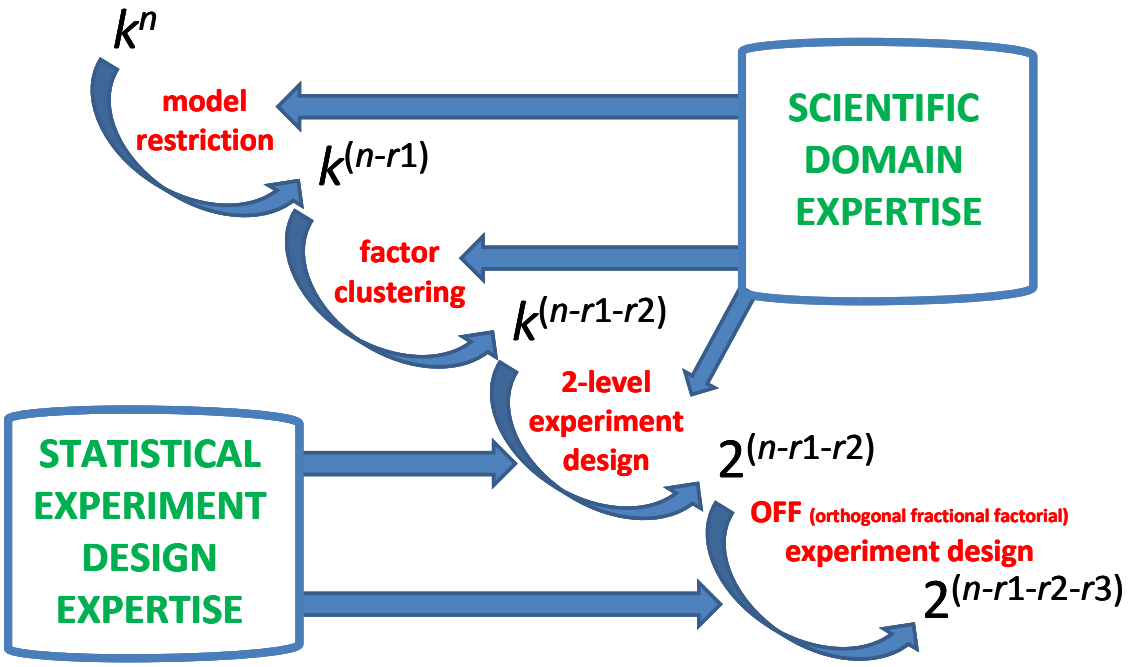


Figure 2-2. A Method to Reduce the Scale of Computational Demands

As a second step in reducing computational demands, a domain expert reexamines model parameters to identify those that can be grouped together to represent aspects of a single factor. This step requires both domain knowledge and creativity. In particular cases, parameter grouping might be guided by knowledge of the type of experiments envisioned for a study. We illustrate examples of parameter grouping in Chapter 4, where we conduct a sensitivity analysis of selected MesoNet parameters. When planning a complete sensitivity analysis of MesoNet we used parameter grouping to lower the

parameter count to 20 (***r***2 = 36). As shown in Fig. 2-3, the reduced parameter space still requires an infeasible amount of computation.

Having reduced ***n*** significantly, the next step involves reducing ***k***, the range of values each parameter may by assigned. Here, we are guided by experiment design theory [89], as developed by statistical researchers and as applied in rigorous scientific and engineering studies [95]. A major scale reduction is achieved by lowering ***k*** to a small number of levels (or values), typically 2 or 3. This reduction has two positive effects. First, the number of parameter combinations to simulate falls substantially – to a number that may be computationally feasible. Second, experiment design theory includes procedures for specifying 2-level and 3-level designs that can be subjected to statistical analyses that yield fundamental insights into system behavior. For our study, we set ***k*** to 2; as shown in Fig. 2-3, this reduces the parameter space to a point where we need to simulate only 10^6 parameter combinations.

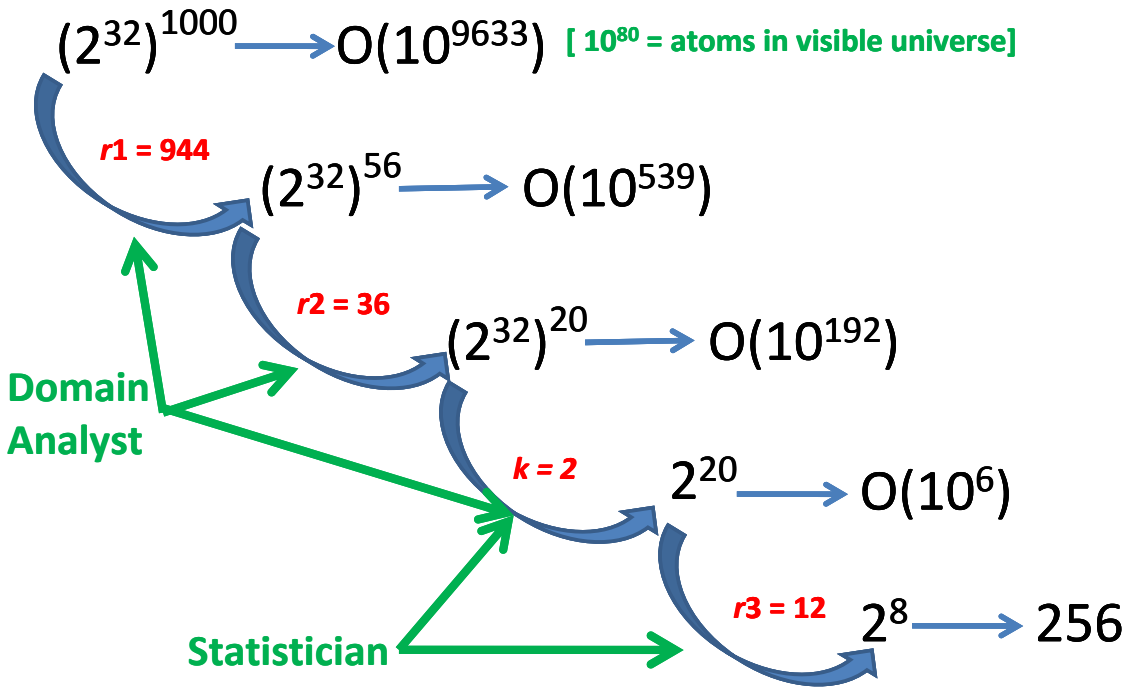


Figure 2-3. An Example Applying Scale Reduction to MesoNet Simulations – we use O( ) notation to denote "on the order of"

Suppose that using MesoNet to simulate one combination of parameters for a specified scenario requires 8 processor hours. Further suppose that we have 48 processors available for our simulations. Under those assumptions, we could simulate 2^{20} parameter combinations in about 20 years[2] (2^{20} runs x 8 processor-hours per run / 48 processors = about 1.748 x 10^5 hours). Obviously, we need to further reduce the computational demands because no one would be willing to wait two decades to learn the results of our

[2] If 10^3 processors were available, then the 2^{20} simulations could be completed in about one year, but we would still need to pay for the use of the processors.

study. We applied orthogonal fractional factorial (OFF) experiment design methods to further reduce ***n***.

To develop an OFF design, the experimenter first determines how many experiment repetitions are feasible. Suppose the experimenter decides to devote at most 2.048×10^3 processor hours to our sample experiment. Since 8 processor hours are needed to run each simulation, the experimenter can afford to investigate only ($2.048 \times 10^3 / 8 =$) 256 different combinations of parameters. Thus, given that ***k*** = 2, the experimenter requires that ***n*** = 8. This means that 2^{20} parameter combinations must be divided by 2^{12}, leaving only 2^8 combinations. In this example, as shown in Fig. 2-3, the reduction factor ***r***3 is 12.

Experiment design theorists have developed rules [89] to select which subset of parameter combinations to examine. The rules compose parameter combinations in a balanced and orthogonal form. A balanced design ensures that every factor is assigned each of its levels an equal number of times. An orthogonal design ensures that the subset of parameter combinations selected is spread evenly throughout the full set of parameter combinations. Further, the resulting design can be assessed precisely with regard to any confounding that may occur. Confounding means that given results cannot be attributed clearly to a main effect (i.e., single parameter) because the results might be due to interactions between two or more parameters. An experimenter should strive to select a Resolution IV design [89], where main effects are not confounded with two-parameter interactions; confounding between main effects and three-parameter interactions is usually acceptable because most systems are not driven by three- parameter interactions. When data analysis points to two-parameter interactions as drivers of system behavior, then a domain expert can usually determine which of the two is most likely.

A Resolution IV experiment design requires a sufficient number of simulations (n) to estimate a leading constant, each parameter (p) and each pair of parameters (p choose 2). This means the example given in Fig. 2-3 requires a least 211 ($n = 1 + 20 + 190$) simulations for a Resolution IV design. For a two-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. Reducing the number of simulations below this would lead to confounding between main effects and two-parameter interactions.

The ultimate result of statistically based experiment design is that all experiment parameters (often called factors) are varied simultaneously. This powerful technique for reducing the search space stands in stark contrast to the approach typically adopted in networking simulation studies. For example, Paxson and Floyd [72] recommend holding all factors fixed except for one element, which becomes a single factor that is varied over a range during a particular simulation experiment. Varying only one factor at a time yields little information about overall system dynamics. Instead, such experiments indicate only the influence of the single varied factor given the fixed combination of other parameters. By varying all factors simultaneously, the system being investigated is examined over a much wider range of conditions. Of course, to derive useful information the resulting system responses must be analyzed with statistical methods matched to two-level OFF designs. We say more about this aspect of our approach in Sec. 2.5.3.

As shown above, reducing the parameter space of a model and applying two-level OFF designs can significantly reduce computation demands when simulating large systems. Of course, interpreting results from such experiments entails a key assumption

that system behavior is monotonic in regions between selected parameter values. OFF experiment designs will not reveal any nonmonotonicity that occurs within such regions. For this reason, an experimenter might wish to cover more than two levels in a design. In addition, the processing cost of running each simulation might require an experimenter to select an ***r***3 reduction value that provides an insufficient number of runs for a Resolution IV design, leading to an undesirable confounding structure. Thus, there exists a tradeoff between the cost of running each simulation and the number of simulations that might be desired.

In our study, we found that experiments with MesoNet require substantial computation when simulated network speeds mirror modern Internet speeds and when network size reaches hundreds of thousands of sources. Fortunately, though costly (averaging about 420 processor hours per simulation), such large simulations were not infeasible[3] with MesoNet, as is typically the case with more detailed simulators. Further, we found that we could obtain similar results when simulating a network with an order of magnitude lower router speed and only tens of thousands of sources. The smaller scale simulations were much less costly (averaging about 24 processor hours per simulation). Even with the smaller scale simulations, simulating 256 combinations of parameters can require more than 6×10^3 processor hours. Given 48 processors, the necessary simulations can be carried out within a week. Increasing the number of processors used to 256 would allow such simulations to be completed in about a day. Thus, smaller scale simulations are quite affordable and computationally feasible. Running a larger scale simulation experiment with 256 parameter combinations would take about 3 months when 48 processors are available and about 3 weeks when 256 processors are available. Thus, running larger simulations and more parameter combinations could be made more cost effective if the computation requirements for each simulation could be reduced. An alternative modeling method, known as hybrid systems, promises to reduce computation demand for network simulations.

Lee and colleagues [71] apply a combination of continuous-time dynamics and event-based logic to model long-lived flows transiting a portion of the Abilene backbone. In addition to TCP congestion control procedures, the hybrid model includes three of the congestion control algorithms examined in our study. As described in Appendix B, we used MesoNet to replicate an experiment reported by Lee and colleagues where 30 long-lived flows transmitted packets for 11 simulated hours. While MesoNet and the hybrid model obtained similar results, the hybrid model appears to require about two orders of magnitude less computation. Thus, for a study such as ours, a hybrid model combining continuous-time dynamics with event-based logic might offer a promising alternative to reduced-scale discrete-event simulation.

2.5.2 Sensitivity Analysis and Key Empirical Comparisons

To establish the validity of MesoNet for our study, we adopted two main strategies: (1) sensitivity analysis and (2) key empirical comparisons. Paxson and Floyd [72] recommend conducting a judicious exploration of a model's parameter space. Floyd and Kohler [70] repeat this advice. Floyd and colleagues also indicate that such explorations are seldom, if ever, practiced in the network simulation community. A main motivation

[3] Of course, the model code, the simulation framework, the underlying operating system and all required hardware must be highly reliable in order to run such large simulations without failures.

for exploring a model's parameter space is to understand how responses change with combinations of input parameters. Generating such knowledge can help identify parameter combinations for which experiments could yield insights and can build confidence in the operation of a model. We implemented the aims of Floyd and colleagues by conducting a sensitivity analysis of MesoNet. We designed the sensitivity analysis (described in Chapter 4) as a 2^{11-5} orthogonal fractional factorial (OFF) experiment[4], which requires 64 individual simulations. We interpreted the results of the experiment using statistical analysis methods, which we outline in Sec. 2.5.3 and explain in detail in Chapter 4. Given results from a 2^{11-5} OFF experiment, a domain expert determined if the results were as expected for a valid network model. In the case of unexpected results, the domain expert established whether the new insights were legitimate or whether the model exhibited errors. The sensitivity analysis of MesoNet uncovered both legitimate and illegitimate unexpected results. Where illegitimate results were identified, MesoNet was corrected and the sensitivity analysis was conducted again. In the end, the sensitivity analysis helped us to reduce the parameter space in later experiments so that we could focus on the top handful of factors influencing model behavior. In addition, related analyses (see Sec. 2.5.3) allowed us to reduce the response space we needed to examine in subsequent experiments. And, of course, the sensitivity analysis increased our confidence in MesoNet as a simulation platform on which to base further experiments.

One shortcoming of our sensitivity analysis arose from limiting parameter settings to only two levels. As mentioned earlier, behaviors might not be monotonic between the chosen levels. In addition, outside the range of the chosen levels a model might not exhibit the same behaviors. To address these shortcomings to some extent, we conducted a second sensitivity analysis where we chose different values to represent the two levels for each parameter. We document this second sensitivity analysis in Appendix C. We also took further steps to explore MesoNet. Some researchers [71] had conducted a hybrid simulation aimed at replicating findings expected from accepted analytical insights into TCP flows. As described in Appendix B, we repeated this published experiment using MesoNet and we compared our results to the published results. Demonstrating that MesoNet could reproduce findings predicted from accepted analytical models also raised our confidence.

Establishing MesoNet as an effective simulator of TCP flows transiting a network topology was a necessary, but insufficient, validation for the experiments we intended to conduct. Since we added six proposed alternate congestion control algorithms into MesoNet, we needed some means to establish that our models correctly implemented the algorithms. Fortunately, in a recently published paper, researchers [67] reported some empirical results from studying five of the six algorithms in a dumbbell topology. Another paper [66] provided empirical results for the sixth algorithm in a similar setting,

[4] The actual process involved multiple repetitions of sensitivity analyses, which enabled us to eliminate some model parameters from our experiments. In addition, we chose to fix selected parameters because they were not germane to the issues we intended to study. This is the reason we focused our sensitivity analysis on 11 parameters instead of 20. In later work we conducted a full sensitivity analysis using all 20 parameters in MesoNet. This later sensitivity analysis used a 2^{20-12} OFF design, which required simulating 256 parameter combinations. This later sensitivity analysis revealed that MesoNet is driven primarily by 6 or 7 of the 20 model parameters. This result confirmed our choice to use about 32 conditions for each of our experiments, described in Chapters 6-9.

though with different parameter combinations. Since MesoNet could be configured with various topologies, we were able to simulate parameter combinations from the empirical study within a dumbbell topology. As described in Chapter 5, we compared behaviors generated from our MesoNet simulations against the published empirical results. In fact, we continued to improve our models of the alternate congestion control algorithms until the simulated behavior generated by MesoNet matched the empirical behavior reported in the literature. By making these key empirical comparisons, we gained confidence that we had correctly simulated the congestion control algorithms under study.

2.5.3 Statistical Analysis Methods

By adopting two-level OFF designs as the basis for all of our experiments, we enabled the application of numerous statistical analyses that could provide insights into relationships between patterns of input parameters and observed responses. We explain the analysis methods we used in detail at each point in our study where we apply them (see Chapter 4 and Chapters 6 through 9). Here, we introduce the key analysis methods in outline form, focusing on the major contributions of each method. As a general contribution to tractable analysis, all methods we adopt either reduce the dimension of multidimensional data, concisely summarize multidimensional data in succinct form or both.

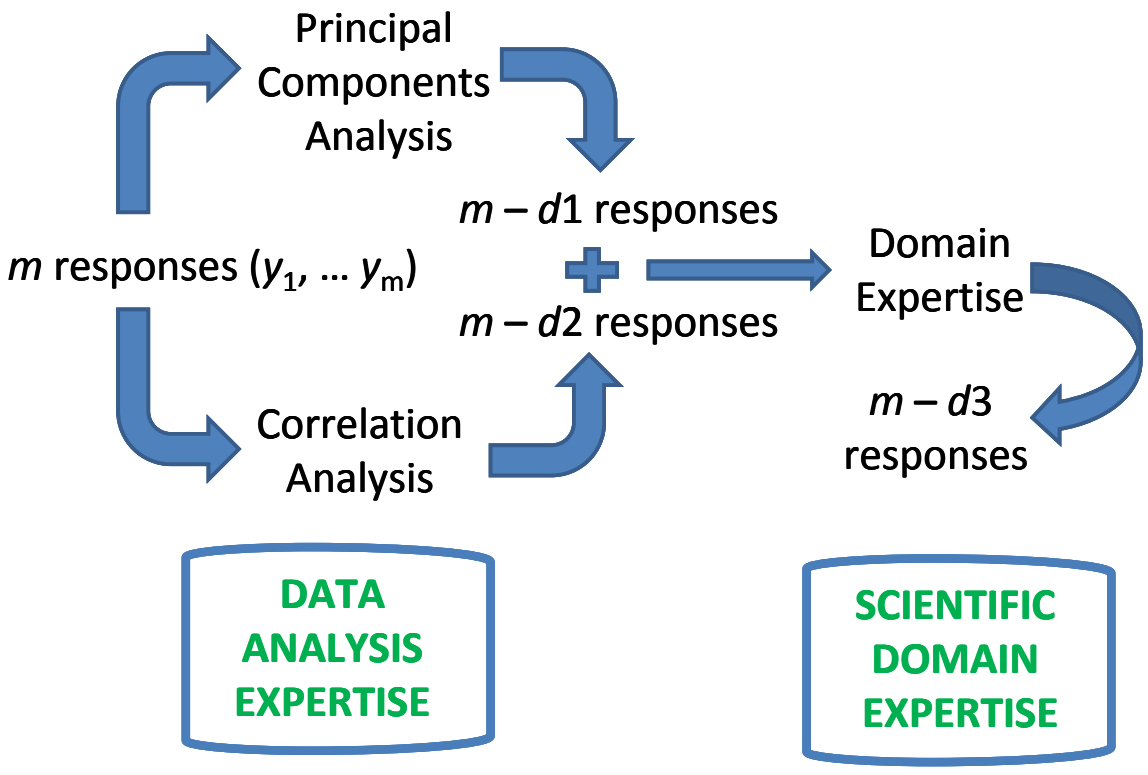


Figure 2-4. Reducing the Space of Model Responses using Correlation and/or Principal Components Analysis

One statistical method we adopted was correlation analysis, which we used to reduce the dimension of multidimensional response data. Simulation models provide

experimenters with the possibility to measure many different system responses. Correlation analysis can identify where only a subset of responses need be analyzed in order to portray system behavior. As an alternate or complementary approach one could also apply principal components analysis. Fig. 2-4 illustrates the general idea behind these two analyses. Given a set of m responses, principal components analysis can find that a lower dimension $m - d1$ can capture the variation in response data. Alternatively, correlation analysis can suggest that fewer responses $m - d2$ can suitably represent variation in system behavior. Given two different proposals for reducing the response space of a model, a domain expert may then choose which responses to analyze for subsequent experiments. Chapter 4 provides a specific example where we applied these methods to MesoNet.

Another reason to apply correlation and principal components analyses is to help validate a model. A domain expert likely has expectations that selected responses mirror similar aspects of system behavior. Correlation analysis can provide clusters of similar responses that an expert can verify. Surprising correlations may indicate errors in a model or else new findings. Similarly, a domain expert may have expectations about general network characteristics that drive behavior. By examining results from a principal components analysis, an expert can verify these overall aspects of system behavior, as represented by a model.

We also adopted a 10-step technique developed at NIST [92] to support analysis of data generated by two-level experiments designed using OFF. Each of the 10 steps produces a graphic (or plot) aimed at concisely summarizing some aspect of response data. The plots include: (1) ordered data plot, (2) scatter plot, (3) main effects plot, (4) interaction effects matrix, (5) block plot, (6) Youden plot, (7) |effects| plot, (8) half-normal probability plot of |effects|, (9) cumulative residual standard deviation plot and (10) contour plot of two dominant factors. We explain each of the plots in detail in Appendix D, and we apply selected plots in Chapter 4 to support of our sensitivity analysis of MesoNet. We also demonstrate in Chapter 4 how two-level OFF designs can aid other exploratory data plots to reveal insights about system behavior.

As demonstrated in Chapters 6 through 9, we used cluster analysis in many of our experiments to provide a concise summary of multidimensional response data. Cluster analysis computes multidimensional distances between sets of responses associated with specified parameters, or combinations of parameters, and then groups the combinations based upon similarities in distances. For example, we might cluster responses associated with each congestion control algorithm for each combination of experiment parameters (i.e., experiment condition) to produce a set of dendrograms[5], one per condition (e.g., Fig. 6-4). Clustering can help us identify patterns among responses. For example, we might find that two algorithms always cluster near each other (i.e., have similar responses). Or we might find that three different algorithms cluster together only under selected conditions. We sometimes combine cluster analysis with other methods to reveal additional patterns. For example, we use an ordered data plot to classify the relative levels of congestion associated with each condition and then overlay the classification onto a set of clustering dendrograms (e.g. Fig. 6-8) to identify how congestion level influences clustering.

[5] A dendrogram is a tree diagram frequently used to depict the arrangement of clusters, as produced by some hierarchical clustering algorithm.

In addition to clustering, we also used time series plots (e.g. Fig. 6-6) to investigate particular response dimensions highlighted by other analyses. Given a time series plot, we could determine if specific responses can be adequately summarized with an average value over particular time periods of interest. In addition, comparing time series of aggregate flow states allowed us to see how changing parameter combinations influence the pattern of flow states. We also applied time series of selected responses in selected situations (i.e., time periods and conditions) to investigate particular detailed behaviors.

Aside from the standard available statistical analyses, we combined various statistical methods into custom analysis visualizations aimed at revealing patterns for our particular experiments. We constructed such custom visualizations to provide concise summaries of multidimensional response data. We used a specific visualization (e.g., Fig. 6-9), which we designated a detailed analysis plot, to compare residuals about the mean (y axis) for each congestion control algorithm under each experimental condition, sorting the conditions on the x axis in increasing magnitude of difference. We generated one such plot for each response of interest. We labeled each condition (x axis) with a multidimensional set that included: experiment factor settings, algorithm identifier, order of magnitude in dispersion among the residuals, percentage difference in absolute response and a discriminating statistic derived from a test for outliners. We introduce and explain the details of our custom analysis plots in Chapter 6. We also used custom plots in Chapters 7 through 9.

Though each custom detailed analysis plot provided a significant amount of information about a single response, the plots, when taken together, obscured any overall pattern that might arise across responses. To overcome this limitation, we designed a second custom visualization, which we called condition-response summaries – introduced in Chapter 6 and applied also in Chapter 7. Condition-response summaries (for example see Fig. 6-10) are constructed as matrices of responses (columns) by conditions (rows), where each cell is either blank or contains the identifier assigned to a specific congestion control algorithm under test. To generate a condition-response matrix, we analyzed (automatically) each detailed response to identify cases where some algorithm was identified as a significant outlier under the corresponding combination of parameters. In such cases, we placed the identifier assigned to the outlying algorithm into the corresponding condition-response cell of the summary matrix. If the algorithm was a high outlier then the identifier was colored green. If the algorithm was a low outlier then the identifier was colored red. Further, we could apply filtering so an outlier would be included in the summary only where additional criteria were satisfied. For example, as shown in Fig. 6-11, we might specify that the outlier must show at least a 10% absolute difference from the mean of all responses for the same condition. Condition-response summaries enabled us to identify patterns where algorithms became outliers under specific conditions.

We generated several other custom visualizations, introduced in Chapter 8 and also applied in Chapter 9, to support our analyses comparing relative goodput[6] of TCP flows and competing flows running alternate congestion control algorithms. One

[6]Goodput is application level throughput, i.e. the number of useful packets per unit of time forwarded by the network from a certain source to a certain destination, excluding protocol overhead, and excluding retransmitted data packets.

visualization (e.g. Fig. 8-34) comprised a set of scatter plots (one per alternate congestion control algorithm) of goodput on alternate flows (x axis) vs. goodput on TCP flows (y axis), where each data point represented the goodputs associated with a specific combination of input parameters for a particular flow group[7]. We used these plots to identify algorithms that outperformed competing TCP flows. We augmented the scatter plots with a set of bar graphs (e.g., Fig. 8-35), one per combination of input parameters, ordered by increasing congestion. Each bar graph contained seven bars, one per alternate congestion control algorithm. One end of the bar represented the higher of the average goodputs (either on TCP flows or alternate flows) and the other end represented the lower. The bar was colored green when the alternate algorithm gave flows higher goodput and colored red when TCP flows gave higher goodput. These bar graphs enabled us to discern patterns associated with conditions under which specific alternate congestion control algorithms gave better goodputs than TCP. Finally, we designed a custom visualization (e.g., Fig. 8-12), which we called rank matrices[8], to explore patterns associated for relative goodput among alternate congestion control algorithms and among TCP flows competing with alternate algorithms under the same conditions. We generated a pair of rank matrices for each alternate congestion control algorithm. One matrix in the pair analyzed flows running the alternate algorithm and the second matrix in the pair analyzed TCP flows competing with the alternate algorithm. Each matrix intersected flow groups (columns), sorted by decreasing file size, with conditions (rows), sorted by increasing congestion. Each cell in a matrix contained a number representing the relative rank in goodput when considering all alternate congestion control algorithms under the same flow group and combination of input parameters. We used rank matrices to compare goodput among the alternate congestion control algorithms and to judge their relative influence on competing TCP flows.

2.5.4 Data-Supported Domain Expertise

The statistical analysis methods we adopted proved quite effective at identifying overall patterns from summarizations of experiment data. Further, the analysis methods that incorporated level settings for experiment input parameters could sometimes yield information that suggested causality. Unfortunately, not all of the statistical analysis methods we adopted considered input parameters. Further, in many cases specific input parameters did not directly identify causes. For example, while we inferred that varying levels of congestion in our simulations caused behavioral differences with respect to some measured responses, various parameter combinations tended to influence observed congestion. Thus, we sometimes established causality by classifying combinations of parameters with respect to responses indicating congestion (e.g., packet loss or retransmission rates). In that way, we could match patterns in other response data to changes in congestion. A similar approach could be used to classify combinations of parameters with respect to other macroscopic patterns, such as delay and demand from packets or flows. Domain expertise was required to decide which larger patterns to classify and which responses represented such patterns. Similarly, results from statistical

[7] A flow group is defined by three attributes: the relative potential for congestion on a path through the topology, the file size and the maximum achievable goodput on the path.

[8] Rank matrices report the relative ordering from lowest (1) to highest (7) goodput achieved by a particular congestion control algorithm when compared against the competing congestion control algorithms.

methods, such as correlation and principal components analyses, provided a basis for patterns on which domain experts needed to superimpose interpretations.

Establishing causality in our study always required a domain expert to interpret data. In many cases, the summarized data we used for statistical analyses was insufficient to establish causality. For this reason, we instrumented our simulation model to capture data at more detailed levels. For example, most responses reported by MesoNet were captured as time series[9], so that we could observe temporal evolution. In addition we were able to capture spatiotemporal evolution by producing time series of many characteristics (e.g., queue size, utilization, losses and count of transiting flows) for every router in our simulated topology. We did not capture spatiotemporal evolution for every source or flow. The reason we did not is twofold: (1) the model could potentially simulate hundreds of thousands of sources and (2) billions of flows could come and go over the course of a simulation. In order to gather insight into detailed behavior of flows and sources, we took two steps. First, we enabled experimenters to define long-lived flows between specified pairs of routers. The experimenter also specified when each long-lived flow would start and stop. We coded MesoNet to capture detailed temporal information (such as congestion window size, goodput, window increases, losses and timeouts) for each long-lived flow. Second, we enabled experimenters to capture message and state changes for a random sample of flows. We also instrumented MesoNet to capture temporal evolution with respect to logical classifications such as flow types. Finally, we included the option for an experimenter to capture time series of packet counts on selected links in a simulated topology. We explain these detailed measurements in Chapter 3.

Given patterns revealed by statistical analyses and armed with detailed temporal and spatiotemporal data captured from the same simulations, we investigated causality using the scientific method. When statistical analyses revealed a significant pattern, we developed a hypothesis regarding the cause. We then used detailed data to test the hypothesis; usually positing evidence that should exist if the hypothesis proved correct. If detailed data provided supporting evidence, then we considered the hypothesis confirmed. Where detailed data did not provide supporting evidence, we developed a different hypothesis and sought supporting evidence among the detailed data. For a given pattern of interest, we iterated the approach until we found a hypothesis supported by the data. For the patterns investigated in the current study we were able to find detailed data providing evidence of causality.

In addition to supporting the findings in the study, our approach to establishing causality also proved useful in identifying occasional errors within our simulation. For example, during one experiment, statistical analysis of summary data revealed that under lightly loaded conditions one of the alternate congestion control algorithms exhibited both a higher retransmission rate and a larger average congestion window[10] size. These two patterns seemed unlikely to occur simultaneously, so we needed to determine a cause. Here, we adopted an exploratory approach. We turned to detailed data mapping temporal evolution of average congestion window size for a particular flow type under a

[9] Data subjected to statistical analyses were derived from summarizations of time series captured by the simulation model.

[10] Congestion window defines the number of packets that may be sent prior to receiving an acknowledgment. A larger congestion window generally means a higher potential goodput.

specific combination of conditions. We compared related time series for eight congestion control algorithms. In seven of the algorithms, the temporal evolution of the average congestion window was flat. For the eighth algorithm, congestion window increased linearly with time. The eighth algorithm was the same algorithm identified by the statistical analysis. The comparison of time series indicated that something was wrong with respect to congestion window adjustment in the eighth algorithm. Armed with this information we examined the specific temporal evolution of the congestion window for a long-lived flow managed by each of the algorithms under the same conditions. Comparing these time series revealed that, early in the flow's life, the offending algorithm increased the congestion window much more quickly than the other algorithms. Detailed examination of the related time series identified the exact time when the incorrect algorithm began its unexpectedly sharp rate of increase in the congestion window. The time was coincident with the point where the flow had reached the initial slow-start threshold, which initiated congestion avoidance procedures even in the absence of a loss. Examination of the related code revealed that the model failed to record the time of the transition into a variable intended to hold the time of the last congestion event. After correcting the error, the experiment was rerun and the results (both summary and detailed) were compared with the previous erroneous results, which established that the cause had been identified and corrected.

2.5.5 Domain Expertise and Incremental Design

We adopted an incremental approach to experiment selection. We relied on domain expertise to design the initial experiment (described in Chapter 6). Designing an experiment involves three main activities: (1) deciding which input parameters to vary and which input parameters to fix, (2) selecting values for both the variable and fixed input parameters and (3) specifying any spatiotemporal scenario embodied within the experiment. In deciding which input parameters to vary and fix, we were guided initially by findings from the sensitivity analysis of the simulation model. The factors that most influenced model behavior were selected as input variables for the first experiment; less influential factors were assigned fixed values. In selecting values for fixed and variable parameters, we were guided by our understanding of networks, by the intended aims of the study and by results from the sensitivity analysis. For the initial experiment, we identified an interest in investigating: (1) how congestion control algorithms behave under normal Web-browsing traffic, (2) how congestion control algorithms respond to the onset of heavy spatiotemporal congestion caused by a period of large file transfers and (3) how well congestion control algorithms recover as congestion eases when traffic transitions back toward normal Web-browsing. This naturally led to an experiment scenario encompassing three separate time periods. In addition, we wished to show that MesoNet could simulate networks of significant speed and size. This led to selecting parameters to simulate a large, fast network. To investigate how congestion control algorithms behave on various types of paths within a network, we constructed a simulated topology that permitted heterogeneity in network paths.

The design for the second experiment (described in Chapter 7) was influenced by two main factors: (1) determining whether similar findings (to the first experiment) could be obtained when simulating a smaller, slower network and (2) investigating the influence of the initial slow-start threshold. Given these incremental objectives, we

simply repeated the first experiment while specifying lower values for network speed, network size and initial slow-start threshold.

After reflecting on results from the first two experiments, we decided to extend the range of simulated user traffic in the third experiment (described in Chapter 8). This decision tests the intended purpose of the alternate congestion control algorithms: to improve user performance on large files. We specified four sizes for user flows: increasing from Web objects to documents to service packs to movies. Of course, users could transmit such files over paths with various congestion profiles, and constrained by the maximum interface speed of a source or receiver. For this reason, we introduced new code to measure flows in groups, based upon three dimensions: (1) file size, (2) network path type and (3) interface speed. This enables goodputs to be compared with respect to flows sharing similar characteristics. Given that the previous two experiments exercised the algorithms under relatively heavy congestion, we also decided to significantly reduce overall congestion in the parameter combinations specified for the third experiment. We eliminated variation in temporal congestion; the scenario considered the network operating under the same traffic mix for a single, one-hour time period. Spatial congestion could still arise due to many flows transiting particular areas of the network, but the overall level of congestion was much reduced from previous experiments. To investigate the influence of various alternate congestion control algorithms on competing TCP flows, we decided to include a mix of flows: some operating under TCP and others operating under one of the alternative algorithms. To represent a network that might be moving incrementally toward replacing TCP, we introduced a new model parameter and selected two settings: (1) more TCP flows and (2) more alternate flows. Finally, because the first two experiments suggested that the initial slow-start threshold exerted significant influence on network behavior, we chose to replicate the third experiment twice: once with a high initial slow-start threshold and once with a low initial slow-start threshold.

After reflecting on results from the third experiment, we decided on a fourth experiment (described in Chapter 9) to increase the size and speed of the network by an order of magnitude and to retain other parameters from the third experiment. This decision reflects two main purposes: (1) to investigate behavior of alternate congestion control algorithms under networks of speed and size comparable to modern Internet-based networks and (2) to demonstrate that simulating large, fast networks with large files may be computationally feasible under MesoNet. Of course, the computational requirements proved substantial, so we chose to repeat only one instance of the third experiment: the case with a high initial slow-start threshold. The choice of a high initial slow-start threshold was motivated by desire to focus on the influence of loss/recovery procedures in the alternative congestion control algorithms.

Only a domain expert can decide on the specific experiments to run and the parameters and values to fix and vary. No general method exists for making these decisions. By designing experiments incrementally, the motives and results of preceding experiments can be considered when selecting the aims and designs for subsequent experiments.

2.6 Conclusions

In this chapter, we described the motivation underlying our goal to develop and evaluate a coherent set of methods that can be applied to understand behavior in large distributed

systems. We introduced a challenge problem: comparing alternative congestion control algorithms proposed for the Internet. We described and critiqued current state-of-the-art approaches adopted by researchers to compare congestion control algorithms. We outlined how our research aims to advance the state of the art. We considered several approaches that might be used to achieve our intended advances. We described the approach we developed, which required solving five hard problems. We explained the solutions we adopted to address each problem. In the remainder of this study, we use the challenge problem to develop and evaluate our methods.

While the current study investigates a specific challenge problem, the methods we apply should be generally applicable to a wide array of large distributed systems. Discrete-event simulation (DES) is applied to study a diverse range of scientific and engineering problems. Though DES requires substantial computation to represent large systems, computing power is becoming more cost-effective as system designers adopt multi-core, multiprocessor (MCMP) designs. Orthogonal fractional factorial (OFF) designs have a long history of application in rigorous scientific and engineering studies. Further, OFF experiment designs construct simulation runs that each considers a specified combination of input parameters. Such designs provide a good match for MCMP computer systems because each simulation can be run in parallel. Thus, the more processors available, the faster the simulation campaign can be completed. The statistical analysis methods we adopted are largely independent of the details of specific system models. Even our custom visualizations are based on general analysis approaches. Of course, the methods we developed and applied in the current study have limitations that must be considered. We discuss these limitations in Sec. 10.2, where we evaluate the methods we used to compare alternate congestion control algorithms.