Answering "What-If" Deployment and Configuration Questions with WISE: Techniques and Deployment Experience

Mukarram Bin Tariq, Kaushik Bhandankar, Vytautas Valancius, Amgad Zeitoun, Nick Feamster, Mostafa Ammar

***Abstract*—Designers of content distribution networks often need to determine how changes to infrastructure deployment and configuration affect service response times when they deploy a new data center, change ISP peering, or change the mapping of clients to servers. Today, the designers use coarse, back-of-the-envelope calculations, or costly field deployments; they need better ways to evaluate the effects of such hypothetical "what-if" questions before the actual deployments. This paper presents *What-If Scenario Evaluator (WISE)*, a tool that predicts the effects of possible configuration and deployment changes in content distribution networks. WISE makes three contributions: (1) an algorithm that uses traces from existing deployments to learn causality among factors that affect service response-time distributions; (2) an algorithm that uses the learned causal structure to estimate a dataset that is representative of the hypothetical scenario that a designer may wish to evaluate, and uses these datasets to predict hypothetical response-time distributions; (3) a scenario specification language that allows a network designer to easily express hypothetical deployment scenarios without being cognizant of the dependencies between variables that affect service response times. Our evaluation, both in a controlled setting and in a real-world field deployment on a large, global CDN, shows that WISE can quickly and accurately predict service response-time distributions for many practical *what-if* scenarios.**

I. Introduction

Content distribution networks (CDNs) for Web-based services comprise hundreds to thousands of distributed servers and data centers [1], [5], [8]. Operators of these networks continually strive to improve the response times for their services. To perform this task, they must be able to predict how service response-time distribution changes in various hypothetical *what-if* scenarios, such as changes to network conditions and deployments of new infrastructure. In many cases, they must also be able to reason about the *detailed* effects of these changes (e.g., what fraction of the users will see at least a 10% improvement in performance because of this change?), as opposed to just coarse-grained point estimates or averages.

Many factors affect a CDN's service response time on both short and long time scales. On short time scales, response time can be affected by routing instability or changes in server load. Occasionally, the network operators may "drain" a data center for maintenance and divert the client requests to an alternative location. Over longer timescales, service providers may upgrade their existing facilities, move services to different facilities or deploy new data centers to address demands and application requirements, or change peering and customer relationships with neighboring ISPs. These instances require significant planning and investment; some of these decisions are hard to implement and even more difficult to reverse.

Unfortunately, reasoning about the effects of any of these changes is extremely challenging in practice. Content distribution networks are complex systems, and the response time perceived by a user can be affected by a variety of inter-dependent and correlated factors. Such factors are difficult to accurately model or reason about and back-of-the-envelope calculations are not precise.

This paper presents the design, implementation, and evaluation of *What-If Scenario Evaluator* (WISE), a tool that estimates the effects of possible changes to network configuration and deployment scenarios on service response time. WISE uses statistical learning techniques to provide a way of interpreting *what-if* questions as statistical interventions. WISE uses packet traces from Web transactions to prepare a graphical causal and functional model for the factors that affect the service response time. Network designers can then use WISE to specify a *what-if* scenario that changes one or more of the factors. Using the causal and functional model for the CDN, WISE determines new statistical distributions for factors (including the response time) that are directly or transitively affected by the changes specified by the designer. Because WISE is aware of the causal functional model of the CDN, the new distributions of the variables represent the *what-if* scenario and are consistent with the operational CDN.

Although function estimation using passive datasets is a common application in machine learning, using these techniques is not straightforward because they can only accurately predict the response-time distribution for a *what-if* scenario if the estimated function receives accurate *joint distribution of all the factors that affect response time* as input. Providing this joint distribution of factors as input presents several challenges:

First, **WISE must allow the network designers to easily specify *what-if* scenarios**. To use a typical off-the-shelf function estimator for evaluating a *what-if* scenario, a designer would have to provide an accurate joint distribution of all factors that affect response time so that this distribution is representative of the scenario as well as consistent with dependencies that exist in the operational CDN. For example, packet loss and retransmit rates depend on network round-trip time and bandwidth. For consistency, if a *what-if* scenario

Mukarram Bin Tariq, Vytautas Valancius, Nick Feamster, and Mostafa Ammar with the School of Computer Science, College of Computing, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: {mtariq, valas, feamster, ammar}@cc.gatech.edu

Kaushik Bhandankar and Amgad Zeitoun are with Google Inc. Mountain View, CA, 94043 USA e-mail: {kaushikgoa, amgad}@google.com

An earlier version of this paper appears in proceedings of ACM SIGCOMM 2008 [20]. This revision includes enhancements in techniques, application and results.

changes network round-trip time or bandwidth, the packet loss or retransmits should also change appropriately during evaluation of the scenario. Achieving this goal is hard in practice, because the designer wants to specify a *what-if* scenario as a change to one or a few factors relative to an existing "baseline" CDN deployment but be unaware that *how* the change might also affect other related factors. WISE's interface shields the designers from these complex details. WISE provides a *scenario specification language* that allows network designers to succinctly specify hypothetical scenarios for arbitrary subsets of existing networks and specify *what-if* values for different features. WISE's specification language is simple: evaluating a hypothetical deployment of a new proxy server for a subset of users can be specified in only 2 to 3 lines of code.

Second, because the designer can specify a *what-if* scenario without knowing the dependencies among the factors affecting response time, **WISE must automatically produce an accurate joint distribution of all factors affecting the response time** that is both *representative* of the *what-if* scenario the designer specifies and *consistent* with the causal and functional dependencies in the CDN. To enforce this consistency, WISE uses a causal dependency discovery algorithm to discover the dependencies among variables and a statistical intervention evaluation technique to transform the observed dataset to a representative and consistent dataset representing the joint distribution. Once this dataset is ready, WISE estimates the response-time distribution for the *what-if* scenario.

We have used WISE to predict service response times in both controlled settings on the Emulab testbed and for Google's global CDN for its Web search service. Our evaluation shows that WISE's predictions of response-time distribution are very accurate, yielding less than 5% error in estimating the mean response-time. Median error for estimating response-time for individual requests is between 8% and 11% for cross-validation with existing deployments and only 9% maximum cumulative distribution difference compared to ground-truth response time distribution for *what-if* scenarios on a real deployment as well as controlled experiments on Emulab.

Finally, **WISE must be fast**, so that it can be used for short-term and frequently arising questions. To achieve this, WISE uses approximations that enable fast computations without sacrificing the accuracy significantly. Further, we have tailored WISE for parallel computation and implemented it using the Map-Reduce [7] framework, which allows us to process large datasets comprising hundreds of millions of records quickly and produce accurate predictions for response-time distributions.

This paper extends our earlier work [20] by (1) presenting several enhancements to the prediction techniques; (2) applying WISE to additional applications; (3) offering a more detailed evaluation; and (4) reporting on the real-world deployment and use of WISE in Google's global CDN. With regards to prediction, we found that our initial function estimation based on Kernel Regression (KR) is accurate, but sometimes it is difficult to have training data that covers prediction for rare combinations of variables in the system. To resolve this problem, we have extended WISE to use an approximated nearest-neighbor based approach (Section V-E). This approach requires less data, is comparable in accuracy to KR for estimating distribution of variables, and is also faster in terms of computation time. We also extend WISE to predict *browser-level response time* (*brt*), in addition to the network-level response time (*nrt*) for Google's CDN. Browser-level response time is a better metric to estimate user experience, but *brt* is also more volatile than *nrt* because *brt* depends on additional factors, such as type of browser, DNS latency, cached content, and computational capability and load on client. Some of these factors are not visible to the CDN operator. We describe our extensions to WISE for estimating *brt* and show that WISE accurately predicts *brt*. This paper also presents results of applying WISE to predict network-level response time for ten additional countries and for predicting browser-level response time for seven different countries.

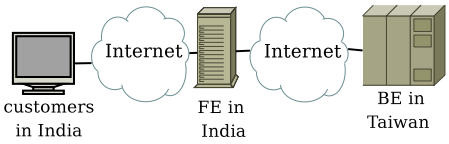
The paper proceeds as follows. Section II describes the problem scope and motivation. Section III makes the case for using statistical learning for *what-if* scenario evaluation. Section IV provides an overview of WISE, and Section V describes WISE's algorithms in detail. In Section VII, we evaluate WISE for response-time estimation for existing deployments as well as for a *what-if* scenario based on a real operational event. In particular, we demonstrate that WISE can accurately predict response time for network level transfer of data, as well as the latency for loading and rendering of a Web page when a user makes a query to Google's Web search service. Results for controlled experiments are omitted here due to space constraints but are available in [19]. In Section VIII, we discuss limitations and ongoing work for WISE. We review related work in Section IX, and conclude in Section X.

II. Problem Context and Scope

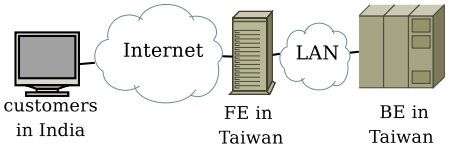
This section describes common *what-if* questions that the network designers pose when evaluating potential configuration or deployment changes to an existing content distribution network deployment.

Content Distribution Networks. Most CDNs conform to a two-tier architecture. The first tier comprises a set of globally distributed front-end (FE) servers that, depending on the specific implementation, provide caching, content assembly, pipelining, request redirection, and proxy functions. The second tier comprises back-end (BE) servers that implement the application logic, and which might also be replicated and globally distributed. The FE and BE servers may belong to a single administrative entity (as is the case with Google [5]) or to different administrative entities, as with commercial content distribution networking service providers, such as Akamai [1]. The network path between the FE and BE servers may be over a public network or a private network, or a LAN when the two are colocated. CDNs typically use DNS redirection or URL-rewriting [4] to direct the users to the appropriate FE and BE servers; this redirection may be based on the user's proximity, geography, availability, and relative server load.

An Example "What-if" Scenario. The network designers may want to ask a variety of *what-if* questions about the CDN configuration. For example, the network designers may want to determine the effects of deploying new FE or BE servers, changing the serving FE or BE servers for a subset of users, changing the size of typical responses, increasing capacity, or changing network connectivity, on the service response time. We now present an actual *what-if* scenario from Google's CDN for the Web-search service.



(a) Before the Maintenance



(b) During the Maintenance

Fig. 1. A *what-if* scenario for network configuration of customers in India.

Figure 1 shows an example of a change in network deployment that could affect server response time. Google has an FE data center in India that serves users in India and surrounding regions. This FE data center uses BE servers located elsewhere in the world, including the ones located in Taiwan. On July 16, 2007, the FE data center in India was temporarily "drained" for maintenance, and the traffic was diverted to a FE data center that is colocated with a BE in Taiwan, resulting in a change in latency for the users in India. This change in the network configuration can be described as a what-if scenario in terms of change of the assigned FE, or more explicitly as changes in delays between FE and clients that occur due to the new configuration. WISE aims to predict the response-time distribution for reconfigurations before they are deployed in practice.

III. A Case for Machine Learning

In this section, we present two aspects of *what-if* scenario evaluation that make the problem well-suited for machine learning: (1) an underlying model that is difficult to derive from first principles but provides a wealth of data; (2) a need to predict outcomes based on data that may not directly represent the desired *what-if* scenario.

The system is complex, but observable variables are driven by fundamental properties of the system. Unfortunately, in large complex distributed systems such as CDNs, the parameters that govern the system performance, the relationships between these variables, and the functions that govern the response-time distribution of the system are often complex and characterized by randomness and variability that are difficult to model as simple, readily evaluatable formulas.

Fortunately, the underlying fundamental properties and dependencies that determine a CDN's response time can be observed as correlations and conditional probability distributions of the variables that define the system, including the service response time. By observing these conditional distributions (e.g., response times observed under various conditions), machine learning algorithms can infer the underlying function that affects the response time. Naturally occurring variability in the system parameters allows observing distributions of variables under a variety of conditions. Because most production CDNs collect comprehensive datasets for their services as part of everyday operational and monitoring needs, the requisite datasets are typically readily available.

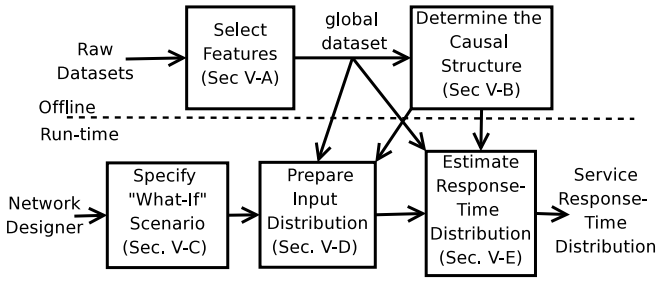


Fig. 2. Main steps in the WISE approach.

Obtaining datasets that directly represent the *what-if* scenario is challenging. Once the response-time function is learned, evaluating a *what-if* scenario requires providing this function with input data that is representative of the joint distribution of all the factors affecting the response time under the *what-if* scenario. Unfortunately, joint distribution in dataset collected from an existing network deployment only represents the current setup, and the system complexities make it difficult for a designer to manually "transform" the data to represent the new scenario.

Fortunately, depending on the extent of the collected data and the nature of *what-if* scenario, machine learning algorithms can reveal the dependencies among variables and use the dependency structure to intelligently *re-weigh and re-sample* the different parts of the existing dataset to perform this transformation. In particular, if the *what-if* scenario is expressed in terms of the changes to values of the variables that are observed in the dataset and the changed values or similar values of these variables are observed in the dataset even with small densities in the original dataset, then we can transform the original dataset to one that is representative of the *what-if* scenario as well as the underlying principles of the system, while requiring minimal input from the network designer.

IV. WISE: High-Level Design

Figure 2 summarizes the high-level steps in WISE. They include: (1) identifying variables in the dataset that affect response time; (2) learning the dependencies among the variables; (3) constraining the inputs to "valid" scenarios based on learned dependencies; (4) specifying the *what-if* scenario; (4) estimating the response-time function and distribution. Each of these tasks raises a number of challenges, some of which are general problems with applying statistical learning in practice, and others are specific to *what-if* scenario evaluation. This section provides an overview and necessary background for these steps. Section V discuss the mechanisms in more detail.

1. Identifying Relevant Features. The main input to WISE is a comprehensive dataset that covers many combinations of variables. Most CDNs have existing network monitoring infrastructure that can typically provide such a dataset. This dataset, however, may contain variables that are not relevant to the response-time function. WISE extracts the set of relevant variables from the dataset and discards the rest of the variables.

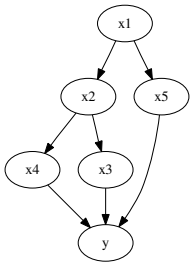
WISE can also identify whether there are missing or latent variables that may hamper scenario evaluation (Sections V-A and V-B provide more details).

The nature of *what-if* scenarios that WISE can evaluate is limited by the input dataset—careful choice of variables that the monitoring infrastructure collects from a CDN can therefore enhance the utility of the dataset for evaluating *what-if* scenarios, choosing such variables is outside the scope of WISE system.

2. Preparing Dataset to Represent the What-if Scenario. Evaluating a *what-if* scenario requires values for input variables that "make sense". Specifically, an accurate prediction of the response-time distribution for a *what-if* scenario requires a joint distribution of the input variables that is representative of the scenario and is also consistent with the dependencies that are inherent to the system itself. Change in one variable must be followed by appropriate change in all other variables that depend on the first variable. To enforce such consistency WISE learns the dependency structure among the variables and represents these relationships as a *Causal Bayesian Network (CBN)* [15]. We provide a brief background on CBNs in this section and explain the algorithm for learning the CBN in Section V-B.

A CBN represents the variables in the dataset as a Directed Acyclic Graph (DAG). The nodes represent the variables and an edge from node x_i to x_j that change in x_i "causes" a change in x_j. When conditioned on its parent variables, a variable $\mathbf{x_i}$ in a CBN is independent of all other variables in the DAG except its descendents. An optimal DAG for a dataset is one where we find the minimal parents for each node that satisfy the above property.

Consider a dataset with five input variables $(x_1 \ldots x_5)$, target variable y, and a CBN shown in the figure to the right. If a *what-if* scenario changes the value of variable x_2, then the distributions for variables x_1 and x_5 remains unchanged, but the distribution of the descendants of x_2 must change to maintain consistency. Section V-D will explain in detail how WISE updates the distributions for these descendant variables.



3. Facilitating Scenario Specification. WISE presents the network designers with an easy-to-use interface in the form of a scenario specification language called WISE-Scenario Language (WSL). The designers can typically specify the baseline setup as well as the hypothetical values for the scenario in 3-4 lines of WSL.

WSL allows the designers to evaluate a scenario for an arbitrary subset of customers. WSL also provides a useful set of built-in operators that facilitate scenario specification as relative changes to the existing values of variables or as new values from scratch. With WSL, the designers are completely shielded from the complexity of dependencies among the variables, because WISE automatically updates the dependent variables. We detail WSL and the process of scenario specification and evaluation in Sections V-C and V-D.

4. Scalable Estimation of New Distributions for Variables. Datasets for typical CDN deployments and *what-if* scenarios span a large multi-dimensional space. While non-parametric function estimation is a standard application in the machine learning literature, the computational requirements for accurately estimating a function spanning such a large space can be astronomical. To address this, WISE estimates the function in a piece-wise manner, and also structures the processing so that it is amenable to parallel processing. WISE also uses the dependency structure to reduce the number of variables that form the input to the regression function. Section VI-A will explain these scaling techniques in more detail.

V. WISE: System

This section describes the algorithmic and design details for each of the steps in WISE. Section V-A presents the criteria that WISE uses to select the features that are useful for evaluating *what-if* scenarios. Section V-B presents the WISE Causal Discovery (WCD) Algorithm that determines the causal dependencies among the features the WISE selects. Section V-C describes the WISE Scenario Specification Language (WSL). WSL assists the network designers in scenario specification. Sections V-D and V-E explain how WISE evaluates a *what-if* scenario expressed in WSL and predicts distributions for response time and other intermediate variables.

A. Feature Selection

Traditional machine-learning applications use model selection criteria, such as Akaike Information Criterion (AIC), Mallow's C_p Test, or k-fold cross-validation [21], to determine the appropriate subset of covariates for a learning problem. WISE foregoes the traditional model selection techniques in favor of simple pairwise independence testing, because at times the conventional techniques can ignore variables that might allow the designer to more easily interpret the results.

WISE uses simple pair-wise independence tests on all the variables in the dataset with the response-time variable and discards all variables that it deems independent of the response-time variable. For each categorical variable (variables that do not have numeric meanings) in the dataset, such as country of origin of a request or AS number, WISE obtains the conditional distributions of response time for each categorical value, and discards the variable only if all the conditional distributions of response time are statistically similar. To test similarity, we used a Two-sample Kolmogorov-Smirnov (KS) goodness-of-fit test with a significance level of 10%.

For real-valued variables, WISE first tests for correlation with the response-time variable and retains a variable if the correlation coefficient is greater than 10%. For some variables, correlation may appear small because it cancels out over a large range of the variable. For example, correlation of performance with time-of-day typically cancels out over a diurnal cycle. To overcome this, we divide the range of the variable into small buckets and treat each bucket as a category. We then apply the same techniques as we do for the categorical variables to determine whether the variable is independent. There is still a possibility that we may discard a variable that is relevant, but this outcome is less likely if sufficiently small buckets are used. The bucket size depends on the variable in question; for instance, we use one-hour buckets for the time-of-day variable.

WCD $(\mathbf{V}, \mathbf{W_0}, \mathbf{\Delta})$
/*Notation
$\mathbf{V}$: set of all variables
$\mathbf{W_0}$: set of no-cause variables
$\mathbf{\Delta}$: maximum allowable cardinality for separators
$a \perp b$: Variable a is independent of variable b */
Make a complete Graph on $\mathbf{V}$
Remove all edges (a, b) if $a \perp b$
$\mathbf{W} = \mathbf{W_0}$
for c = 1 to $\mathbf{\Delta}$ /*prune in the order of increasing cardinality*/
LocalPrune (c)

LocalPrune (c)
/*Try to separate neighbors of frontier variables $\mathbf{W}$*/
$\forall w \in \mathbf{W}$
$\forall z \in N(w)$ /*neighbors of w*/
if $\exists \mathbf{x} \subseteq N(z) \backslash w : |\mathbf{x}| \leq c, z \perp w | \mathbf{x}$
then /*found separator node(s)*/
$S_{wz} = \mathbf{x}$ /*assign the separating nodes*/
Remove the edge (w, z)
Remove edges (w', z), for all the nodes $w' \in W$ that are also on path from w to nodes in W_0
/*Update the new frontier variables*/
$\mathbf{W} = \mathbf{W} \cup \mathbf{x}$

Fig. 3. WISE Causal Discovery (WCD) algorithm.

B. Learning the Causal Structure

To learn the causal structure, WISE first learns the undirected graph, then uses a set of rules to orient the edges.

Learning the Undirected Graph. Recall that in a Causal Bayesian Network (CBN), a variable, when conditioned on its parents, is independent of all other variables except its descendants. An optimal CBN requires finding the smallest possible set of parents for each node that satisfy this condition. Thus, by definition, variables a and b in the CBN have an edge between them if and only if there is a subset of *separating variables*, S_{ab}, such that a is independent of b given S_{ab}. In the general case, constructing the CBN requires searching all the possible $O(2^n)$ combinations of the n variables.

The WISE-Causal Discovery Algorithm (WCD) (Figure 3) uses a heuristic to guide the search of separating variables, given prior knowledge of a subset of variables that are "not caused" by any other variables in the dataset, or that are determined by factors outside the system model (we refer to these variables as the *no-cause* variables). WCD does not perform exhaustive search for separating variables, thus trading off optimality for lower complexity.

WCD starts with a fully connected undirected graph on the variables and removes the edges between variables that are independent. WCD then progressively finds separating nodes between a restricted set of variables (that we call *frontier* variables), and the rest of the variables in the dataset, in order of increasing cardinality of allowable separating variables. Initially, the frontier variables comprise only the no-cause variables. As WCD discovers separating variables, it adds them to the set of frontier variables.

The algorithm terminates when it has explored separation sets up to the maximum allowed cardinality $\mathbf{\Delta} \leq n$, resulting in a worst-case complexity of $O(2^{\Delta})$ (Andre thinks its C^n_{Δ}). This termination condition means that certain variables that are separable are not separated: this does not result in false dependencies, but transitive dependencies may be considered direct dependencies. This sub-optimality does not affect the accuracy of the scenario datasets that WISE prepares, but it reduces the efficiency because it leaves the graph to be denser and the nodes having larger in-degree. Because the WISE needs values of all the parent variables to determine the value of a child variable, determining value of high in-degree child variables requires knowing value of many variables. This reduces the data required for accurate prediction. Fortunately, because WCD is part of the offline steps in WISE, we can afford a large maximum cardinality (Δ) search for the separating nodes; this will improve the efficiency for the online stage of scenario evaluation.

When the set of no-cause variables is unknown, WISE relies on the PC-algorithm [18], which also performs search for separating nodes in the order of increasing cardinality among all pair of variables, but not using the frontier variables.

Orienting the Edges. WISE orients the edges and attempts to detect latent variables using the following simple well-known rules. We reproduce these rules, slightly adapted to accommodate heuristics in WCD, and refer the reader to Pearl's textbook [15] for further details.

1) Add outgoing edges from the no-cause variables.
2) If node c has nonadjacent neighbors a and b, and $c \in S_{ab}$, then orient edges $a \rightarrow c \leftarrow b$ (unmarked edges).
3) For all nonadjacent nodes, a, b, with a common neighbor c, if there is an edge from a to c, but not from b to c, then add a marked edge $c \overset{*}{\rightarrow} b$.
4) If a and b are adjacent and there is directed path of only marked edges from a to b, then add $a \rightarrow b$

In the resulting graph, any unmarked, bi-directed, or undirected edges signify possible latent variables and *ambiguity in causal structure*. In particular, $a \rightarrow b$ means either a really causes b or there is a common latent cause L causing both a and b. $a \leftrightarrow b$, signifies a definite common latent cause, and undirected edge between a and b implies either a causes b, b causes a, or a common latent cause in the underlying model.

We address issues related to missing variables or causal relationships that may arise due to limitations of dataset in Section VI-B. Section V-D discusses how WISE deals with ambiguities in causal structure.

C. Specifying the "What-If" Scenarios

Figure 4 shows the grammar for WISE-Specification Language (WSL). A scenario specification with WSL comprises a *use*-statement, followed by optional scenario *update*-statements.

The *use*-statement specifies a condition that describes the subset of present network for which the designer is interested in evaluating the scenario. This statement provides a powerful interface to the designer for choosing the baseline scenario: depending on the features available in the dataset, the designer can specify a subset of network based on location of clients (such as country, network address, or AS number), the location of servers, properties of service sessions, or a combination of these attributes.

The *update*-statements allow the designer to specify *what-if* values for various variables for the service session properties.

```
scenario = use_stmt {update_stmt};
use_stmt = "USE" ("*" | condition_stmt)<EOL>;
update_stmt = ("ASSUME"|"INTERVENE")
    (set_directive | setdist_directive)
    [condition_stmt]<EOL>;
set_directive = "SET" ["RADIAL"* | "FIXED"]
    var set_op value;
setdist_directive = "SETDIST" feature
    dist_name([param])| "FILE" filename);
condition_stmt = "WHERE" condition;
condition = simple_cond | compound_cond;
simple_cond = compare_clause | (simple_cond);
compound_cond = (simple_cond ("AND"|"OR")
    (simple_cond|compound_cond));
compare_clause = (var rel_op value) |
    member_test;
member_test = feature "IN" (value {,value});
set_op = "+=" | "-=" | "*=" | "\=" | "=";
rel_op = "<=" | ">=" | "<>" | "==" | "<" | ">";
var = a variable from the dataset;
```

Fig. 4. Grammar for WISE Specification Language (WSL).

Each scenario statement begins with either the INTERVENE, or the ASSUME keyword and allows conditional modification of exactly one variable in the dataset.

When the statement begins with the INTERVENE keyword, WISE first updates the value of the variable in question. WISE then updates the values of dependent variables. For this WISE uses a process called *Statistical Intervention Effect Evaluation*, which we describe in more detail in Section V-D.

Advanced designers can override the automatic updates to dependent variables by using the ASSUME keyword in the update statement. In this case, WISE updates the distribution of the variable specified in the statement but does not update dependent. WISE allows this function when the designers believe that the scenario that they wish to evaluate involves changes to the underlying invariant laws that govern the system. Examples of scenario specification with WSL will follow in Section VII.

D. Preparing the Input Distribution

This section describes how WISE uses the dataset, the causal structure, and the scenario specification from the designer to prepare a meaningful dataset for the *what-if* scenario.

Filtering Dataset to obtain baseline scenario. WISE filters the global dataset for the entries that match the conditions specified in the *use-statement* of the scenario specification to create the *baseline* dataset.

Evaluating Scenario for Baseline Dataset as Statistical Intervention. WISE executes the update-statements, one statement at a time, to change the baseline dataset. To ensure consistency among variables after every INTERVENE update statement, WISE employs a process called *Statistical Intervention Effect Evaluation*; the process is described below.

Let $\mathbf{set}(\mathbf{x_i})$ denote the variable $\mathbf{x_i}$ intervened in the *update*-statement. Also let $\mathcal{C}(\mathbf{x_i})$ and $\mathcal{P}(\mathbf{x_i})$ denote the children and parents, respectively, of $\mathbf{x_i}$ in the CBN. If the causal structure is correct and complete, then the new distribution of children of $\mathbf{x_i}$ is given as: $\mathbf{Pr}\{\mathcal{C}(\mathbf{x_i})|\{\mathcal{P}(\mathcal{C}(\mathbf{x_i})), \mathbf{set}(\mathbf{x_i})\}\}$. WISE estimates this distribution using *approximated nearest neighbor search* algorithm to find samples in the dataset that satisfy the condition $\{\mathcal{P}(\mathcal{C}(\mathbf{x_i})), \mathbf{set}(\mathbf{x_i})\}\}$. We describe this algorithm next.

Emulating Statistical Intervention as Approximate Nearest-Neighbors Search. To estimate the empirical distribution $\mathbf{Pr}\{\mathcal{C}(\mathbf{x_i})|\{\mathcal{P}(\mathcal{C}(\mathbf{x_i})), \mathbf{set}(\mathbf{x_i})\}\}$, WISE conditions the *global* dataset on the new value of the intervened variable, $\mathbf{set}(\mathbf{x_i})$, and the existing values of the all the other parents of the children of the intervened variable, $\mathcal{P}(\mathcal{C}(\mathbf{x_i}))$, in the baseline dataset. WISE chooses a randomly drawn sample from this distribution as *correct* value for child variable after intervention.

Unfortunately, the number of samples in global dataset whose values match exactly with the condition $\{\mathcal{P}(\mathcal{C}(\mathbf{x_i})), \mathbf{set}(\mathbf{x_i})\}$ may be limited, making it difficult to estimate the conditional distribution. As an alternative, WISE also considers samples whose value is approximately same as the condition. To find such samples, WISE uses nearest-neighbors (NN) search centered around the point specified by the above condition. Because NN search is expensive on large datasets, WISE approximates NN search by limiting the scope of search around the specified center, using a technique that we call *tiling*. Tiling is described in Section VI-A.

Dealing with Ambiguities in Causal Structure. When the causal structure has ambiguities, WISE proceeds as follows.
(a) Bi-directed or Undirected Edges. When the edge between two variables is bi-directed or undirected, WISE maintains the consistency by always updating the distribution of one if the distribution of the other is updated.
(b) Unmarked Directed Edges. Cases of unmarked directed edge, $a \rightarrow b$ are quite common. Recall that this implies that either variable a causes b, or there is a latent variable that causes both a and b. WISE updates the distribution of b assuming that a causes b. The intuition is that if there is indeed a latent variable that causes a and b, and if a scenario specification requires changing variable a, then the scenario specification is implicitly stating a change in that latent variable, and thus changes in a and b are in order. Further, the only way that we can know about the effect of change of the latent variable on b is what we observe in the dataset as effect on b as a result of changes in a.

E. Estimating Response-Time Distribution

Finding the new distribution of response time is just another case of intervention effect evaluation. WISE uses the techniques in Section V-D considering response time as the affected child variable. To predict response time for individual requests WISE uses a weighted average of all NN for prediction using a standard Kernel Regression (KR) method, with a radial basis Kernel function [22] to estimate the response time for each request in the dataset. We describe details in an extended technical report [19].

VI. Additional Challenges

This section describes additional challenges. Section VI-A presents how WISE addresses scalability challenges that arise due to the size of the required training data. Section VI-B discusses how WISE copes with missing variables or missing causal dependencies that may arise due to insufficient data.

A. Computational Scalability

Because CDNs are complex systems, response time may depend on many variables, and the dataset might comprise hundreds of millions of requests. To efficiently evaluate the *what-if* scenarios, WISE must efficiently organize and utilize the dataset. In this section, we discuss our approach to these problems.

Approximating Nearest-Neighbor Search. WISE uses nearest-neighbors (NN) search for estimating the distribution of variables after intervention. Unfortunately, NN search can be expensive: finding k nearest-neighbors for all the samples in a dataset is $\mathcal{O}(n^2)$. WISE overcomes this problem by limiting the scope of NN search. WISE divides the dataset space into *small* buckets, that we refer to as *tiles* and all the samples that lie in a bucket are considered equally likely nearest neighbors.

To create tiles, WISE uses fixed-size buckets for each dimension in the dataset. The bucket size presents a tradeoff: If the bucket sizes are sufficiently small, the samples in the bucket are a good approximation of the NN, but the likelihood of having many samples lying in a bucket decreases with the size of the bucket. If the bucket is large, the likelihood of finding samples in the bucket increases, but some of the samples may be far away from the center of the bucket and will not be a good representation of NN. Fortunately, having more samples beyond a certain threshold does not contribute appreciably to the accuracy of estimating the empirical distribution of variables in question with either NN or KR techniques.

To balance the tradeoff and maintain sufficient samples per tile, WISE uses a *multi-layer tiling* scheme. For the lowest layer of tiles, WISE uses small bucket sizes; for higher layers, WISE increases the size of buckets. WISE maintains up to n_{max} training data samples per bucket. This reduces storage requirement without compromising accuracy. For NN search, WISE maps the test point to its tile in the lowest layer. If the tile has more than n_{min} samples, all the samples in the tile are considered the NN for the test sample. If there are fewer than n_{min} points in the tile, then WISE maps the test sample to a tile in next higher layer and repeats the process. If all the layers are exhausted but WISE is not able to find a suitable NN, the search aborts. In practice, two or three layers of tiles suffice to provide reasonable accuracy. The n_{min} and n_{max} parameters are tunable. Our current implementation of WISE uses $n_{min} = 1$ for NN search and $n_{min} = 10$ for KR. n_{max} is set to 200.

Retrieval of Data. Because the datasets that WISE uses are large (often comprising several hundred million samples), retrieving the data quickly is critical. WISE expedites data retrieval by indexing the training dataset offline and indexing the test dataset as it is generated during the scenario evaluation. Tiles are used here as well: Each tile is assigned a *tile-id*, which is simply a string formed by concatenating the tile's layer number and tile's boundaries in each dimension. Each data samples is assigned keys that are the tile-id of the tiles in which data samples lie. WISE then uses these keys to index the data samples. Because WISE uses fixed-size buckets, mapping a sample to its tile can be performed in constant time using simple arithmetic operations. Also, because the approximated NN search algorithm in WISE simply matches training and test data samples that share a tile, pre-indexing the data allows WISE to easily collate training and test data and perform NN search.

Parallelization and Batching. We have carefully designed various stages in WISE to support parallelization and batching of jobs that use similar or the same data. In the training data preparation stage, each entry in the dataset is independently assigned its *tile-id* based key because WISE uses fixed-sized tiles. Similarly, for NN search, WISE can find NN for all the test data samples that share a tile in one go because NN samples for one test sample are also NN for other test samples that share the tile.

WISE also addresses problems arising due to many factors that contribute to response time, and the regression-to-mean problem that can arise due to the uneven density of datasets obtained from operational CDN. We describe details for WISE approach in our technical report [19].

B. Missing Variables and Causal Relationships

False or missing causal relationships can arise if the population in the dataset is not independent of the outcome variables. Unfortunately, because WISE relies on passive datasets, this limitation is fundamental. Fortunately, we expect that because the basic principles of computer networks are similar across the Internet, and the service providers use essentially the same versions of software throughout their networks, the bias in the dataset that would significantly affect the causal interpretation is not common. If such biases exist, they will likely be among datasets from different geographical deployment regions.

To capture such biases, we recommend using a training dataset that is obtained from different geographical locations. If we use this geographically diverse dataset to infer a causal structure, then if a variable that signifies the difference of datasets of two regions is missing in the dataset, then the variable representing the "region" will not be *separable* from the outcome variables. As a result, if we observe that there are variables in the DAG that are children of the "regional" variables, but there is no good explanation for this dependency based on understanding of the working of the system, then we must investigate what the missing variables may be. Unfortunately, there is no automated way of determining what variables may be missing.

Finally, it is difficult to compile a dataset that covers CDN system dynamics under *all* possible values of the variables in the system and from all the regions. As a result, if WISE learns the causal structure from a non-comprehensive dataset, then we cannot be certain that the causal structure captures all the variables and causal relationships. Such limitations can be detected by using data from one subset of geographical regions, applying the dataset and causal structure to predict response time for another region, and comparing that with the ground truth. We present an evaluation for completeness of causal structure and variables in Section VII-F. We also discuss a case where a missing variable in the dataset results in larger prediction error.

VII. Evaluating WISE on a Deployed CDN

We have implemented WISE with the Map-Reduce framework [7] using the Sawzall logs processing language [16] and Python Map-Reduce libraries. We chose this framework to best

exploit the parallelization and batching opportunities offered by the WISE design We describe the implementation further in our technical report [19].

This section describes our experience applying WISE to a large dataset obtained from Google's global CDN for Web-search service. We use WISE to predict two metrics:

- Network-level server response time (*nrt*) is the time duration between client's host sending a HTTP request with Web-search query and the client receiving the last byte of the response from the CDN for Web search.
- Browser-level response time (*brt*) is the time duration between the user clicking the "Search" button on the Web page and when the browser completes loading the results pages returned by the CDN for Web search.

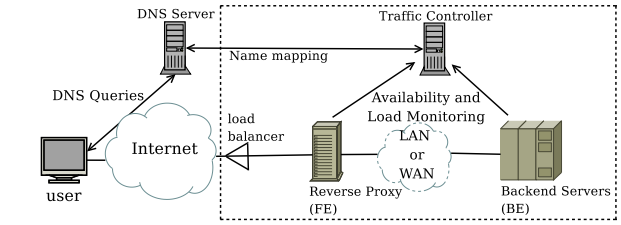
We start by briefly describing the CDN and the service architecture in Section VII-A. In Section VII-B we describe the dataset from the CDN. In Section VII-C, we describe the factors that make it challenging to estimate *nrt* and *brt* using this dataset. In Section VII-D the causal structure discovered from this dataset using WCD. In Sections VII-E—VII-G we evaluate WISE's ability to predict response-time distribution for the *what-if* scenarios.

A. Web-search Service Architecture

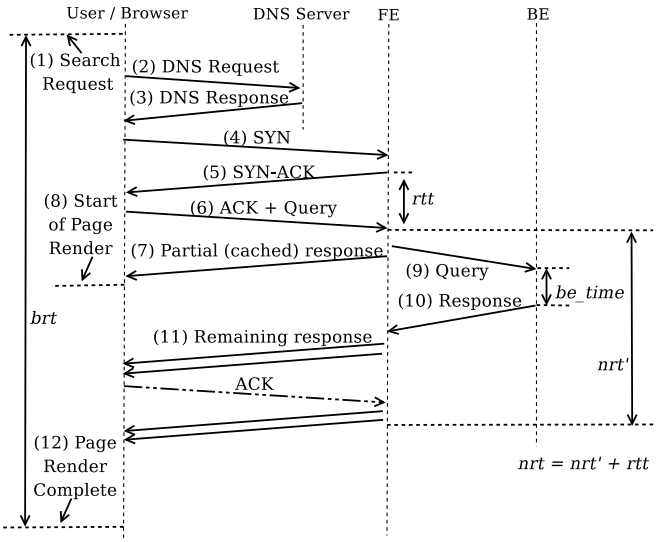
Figure 5(a) shows Google's Web-search service architecture. The service comprises a system of globally distributed HTTP reverse proxies, referred to as the front-end (FE), and a system of globally distributed clusters that house the Web servers and other core services, referred to as the back-end (BE). A DNS-based request redirection system redirects the user's queries to one of the FE in the CDN. The FE process forwards the queries to the BE servers, which generate dynamic content based on the query. The FE caches static portions of typical reply, and starts transmitting that part to the requesting user as it waits for reply from the BE. Once the BE replies, the dynamic content is also transmitted to the user. The FE servers may or may not be colocated in the same data center with the BE servers. If they are colocated, they can be considered to be on the same local area network and the round-trip latency between them is only a few milliseconds. Otherwise, the connectivity between the FE and the BE is typically on a well-provisioned connection on the public Internet. In this case the latency between the FE and BE can be several hundred milliseconds.

Network-level server response time (nrt) for a request is the time between the instance when the user issues the HTTP request and the instance when the last byte of the response is received by the users. We estimate *nrt* as the sum of the round-trip time estimate obtained from the TCP three-way handshake, and the time between the instance when the request is received at the FE and when the last byte of the response is sent by the FE to user. Key contributors to *nrt* are: (i) the transfer latency of the request from the user to the FE, (ii) the transfer latency of request to the BE and the transfer latency of sending the response from the BE to the FE, (iii) processing time at the BE, (iv) TCP transfer latency of the response from the FE to the client; and (v) any latency induced by loss and retransmission of TCP segments.

Browser-level service response time (brt) for a request is the time between the instance when the user presses the "Search" button on the Web page and when the browser finishes rendering the Web page with search results that the server sends. We estimate *brt* using a browser plug-in and javascript embedded in the Web page returned by the server. The following factors contribute latency to *brt*: (i) the DNS request from the user before it establishes a connection with the Web server; (ii) the network-level server response, which captures the latency in host contacting the servers, servers preparing a search response and sending the response to the client; and (iii) the time it takes the browser to render the results Web page and execute javascript embedded in the page.



(a) Google's Web-search Service Architecture



(b) Messages Events for Network-level and Browser-level Response Time

Fig. 5. Google's Web-search service architecture and message exchange for a search request on a *fresh* TCP connection.

B. Dataset

We use data from an existing network monitoring infrastructure in Google's network. FE and BE servers export reports with values for many performance-related variables. A browser plug-in tracks browser-level events and reports these statistics for users that opt to participate in monitoring. The browser plug-in and FE add unique identifiers to each request. This identifier allows us to collate the records in dataset obtained from FE, BE, and the browser plug-ins.

WISE applies the feature selection tests (ref. Section. V-A) on the variables in the dataset. Table I describes the variables that WISE found to be relevant to network and browser response-time variables.

Feature	Description
ts, tod	A timestamp of instance of arrival of the request at the FE. Hourly time-of-day (tod) is derived from the timestamp.
sP	Number of packets sent by the server to the client excluding retransmissions.
srP	Number of packets retransmitted by the server to the client, either due to loss, reordering, or timeouts at the server.
sB	Size in bytes of the encoded response sent by the FE server.
encoding	Encoding type used by FE server.
region	User's IP address, /16, /24 network prefixes, AS number, and geographical mapping to state and country, are collectively referred to as *region*.
fe, **be**	Identifiers for the FE data center at which the request was received and the BE data center that served the request.
rtt	Round-trip time between the user and FE estimated from the initial TCP three-way handshake.
bw	An estimate of access network bandwidth for the /24 IP address block for the user's IP address. It is estimated as the 90% TCP throughput for unthrottled responses greater than 4KB sent from FE to the IP addresses in /24 prefix.
febe_rtt	The network level round-trip time between the front end and the back-end clusters.
be_time	Time taken by BE to process the request forwarded by FE.
firstReq	Binary variable indicating whether the HTTP request is first on a TCP connection.
browser	Browser name and version obtained from HTTP User-Agent string.
nrt	Network-level response time for the request as seen by the FE (see Section VII-A). *nrt* is a target variable.
brt	Browser-level response time for the request as seen by the FE (see Section VII-A). *brt* is a target variable.

TABLE I
FEATURES IN THE DATASET FROM GOOGLE'S CDN.

C. Challenges in Estimating nrt and brt

Figure 5(b) shows the process by which a user's Web search query is serviced. The message exchange and events have many factors that affect *nrt* and *brt* in subtle ways, so it is hard to make accurate "back-of-the-envelope" calculations in the general case.

Challenges for Network Response Time (*nrt*). Several factors make it difficult to estimate *nrt*:

Asynchronous transfer of content to the user. Once the TCP handshake is complete, user's browser sends an HTTP request containing the query to the FE. While the FE waits on a reply from the BE, it sends some static content to the user; this content—essentially a "head start" on the transfer—is typically brief and constitutes only a couple of IP packets. Once the FE receives the response from the BE, it sends the response to the client and completes the request. A client may use the same TCP connection for subsequent HTTP requests.

Spliced TCP connections. FE processes maintain several TCP connections with the BE servers and reuse these connections for forwarding user requests to the BE. The FE also supports HTTP pipelining, allowing the user to have multiple pending HTTP requests on the same TCP connection.

Spurious retransmissions and timeouts. Because most Web requests are short TCP transfers, the duration of the connection is not sufficient to estimate a good value for the TCP retransmit timer, and many Web servers use default values for retransmits, or estimate the timeout value from the initial TCP handshake round-trip time. This causes spurious retransmits for users with slow access links and high serialization delays for MTU-sized packets.

Challenges for Browser Response Time (*brt*). Several factors make it difficult to estimate *brt*:

DNS lookup not visible to network or browser. The user's host needs to resolve the domain name for the search engine site before it can contact the servers. If the DNS reply is cached at the client, latency for DNS resolution is minimal; however, if the DNS reply is not cached or has expired, then DNS latency can be significant. Therefore, not all search requests include DNS resolution latency. Because DNS lookup is transparent to the browser, we have no data to indicate the latency for DNS for a particular request at the browser level. Further, because clients typically use local DNS resolvers, the DNS request is not seen by the DNS servers of the CDN either. As a result, our data has no direct indication of latency contributed by DNS.

Browser may fetch additional resources. Web page rendering in the browser is only complete after the browser has rendered the HTML and other embedded content, such as images or scripts referred to in the HTML page. This content is often cacheable, but if it not cached at the host or has expired, then the browser may need to fetch this content before it can complete rendering. Unfortunately, to encourage caching, the URLs for this content are generic and cannot be collated with the main HTTP request for Web search. As a result, our data has no direct indication of latency contributed by fetching of embedded content.

Browser type and user host's computation capability. Web page rendering time varies with the type of browser. We can infer the browser type from the HTTP User-Agent string, but determining the relationship between rendering speed and browser type is not straightforward. Rendering time also depends on type of content (HTML, javascript, images), type of compression or encoding used in HTTP, the order of download of content, the computation capability of the host, and the load on the host at the time of search query. Not only is the host's computation capability and load not observed in the dataset, the relationship of these factors is not well-understood.

D. Causal Structure in the Dataset

To obtain the causal structure, we use a small sampled data subset collected in July 2009 from several data center locations. This dataset has roughly 50 million requests, from clients in more than 10,000 unique ASes.

We seed the WCD algorithm with the *region* and *ts* variables as the *no-cause* variables. Figure 6 shows the causal structure that WCD produces. Most of the causal relationships

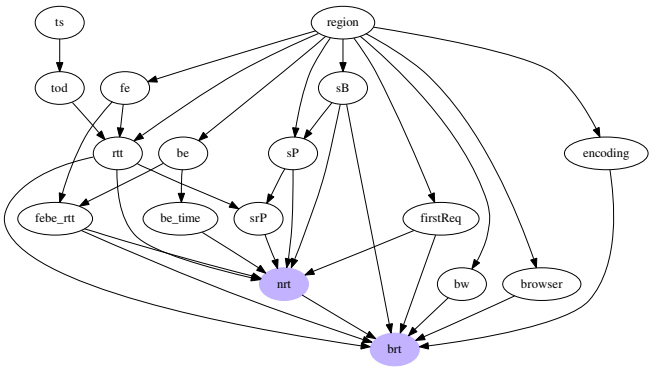


Fig. 6. Inferred causal structure in the dataset. $A \rightarrow B$ means A *causes* B. Target variables are shown in shaded nodes.

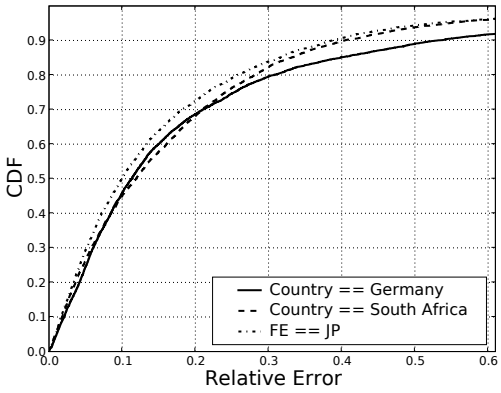


Fig. 8. Relative prediction error for the scenarios in Figure 7.

in Figure 6 are straightforward and make intuitive sense in the context of networking, but a few relationships are quite surprising. WCD detects a relationship between the $region$ and sB attribute (the size of the response from the server); we found that this relationship exists due to the differences in the sizes of search response pages in different languages and regions as well as the encoding and compression schemes used by the servers. Another unexpected relationship is between $region$ and sP attributes; We suspect that this relationship exists due to different MTU sizes in different parts of the world. Unfortunately, our dataset did not have load, utilization, or data center capacity variables that could have allowed us to model the be_time variable. All we observed was that the be_time distribution varied somewhat among the data centers. Overall, we find that WCD algorithm not only discovers relationships that are faithful to how networks operate but also discovers relationships that might escape trained network engineers.

Note that many variables, including the target variables (*nrt, brt*) are not direct children of the $region$, ts, fe, or be variables. This means that when conditioned on their respective parents, these variables are independent of the region, time, choice of FE and BE, and *we can use training data from past, different regions, and different FE and BE data centers to estimate the distributions for these features*. Further, while most of the variables in the dataset are correlated, the in-degree for each variable is smaller than the total number of variables. This reduces the number of dimensions that WISE must consider for estimating the value of the variables during scenario evaluation, allowing WISE to produce accurate estimates, more quickly and with less data.

E. Estimation of Response Time

Our primary metric for evaluation is *prediction accuracy*. There are two sources of error in response-time prediction: (i) error in response-time estimation function (Section V-E) and (ii) inaccurate input, or error in estimating a valid input distribution that is representative of the scenario (Section V-D). To isolate these errors, we first evaluate the estimation function accuracy alone in Section VII-F and later consider the overall accuracy for a complete scenario in Section VII-G.

F. Accuracy of Response-time Function

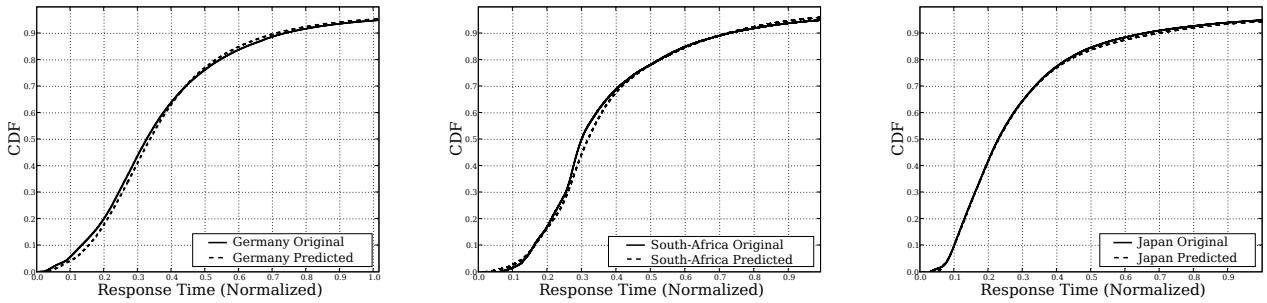
Accuracy of the response-time function estimator depends on following: (a) whether the WISE model captures all the variables, (b) whether the causal structure that WISE infers is accurate, and (c) whether there is sufficient training data available to make the prediction. In this section, we evaluate whether these factors are reasonably addressed by WISE by estimating the accuracy of predictions of (*nrt, brt*) for existing deployments.

To evaluate prediction for existing deployments we can try to evaluate a scenario: "What-if I make no changes to the network?" This scenario is easy to specify with WSL by not including any optional scenario update statements. For example, a scenario specification with the following line:
`USE WHERE country==deu`
would produce an input distribution for the response-time estimation function that is representative of users in Germany without any error and any inaccuracies that arise would be due to inaccurate input from scenario specification.

Recall from Sections V-D and V-E that WISE uses the parents of *nrt* and *brt* to find the nearest-neighbor samples from the training dataset and predict the response time. Region, FE, BE, or *ts* are not parents in the CBN in Figure 6. As a result, when WISE performs NN search, it does so without regard for the region, servers, or timestamp of the samples. Still, to ensure that the nearest-neighbor samples that WISE uses for prediction do not come from same data center or country for which we wish to evaluate a *what-if* scenario, we exclude the data from those data centers and countries from the training dataset for all of the evaluations that follow.

Network-level Response Time (*nrt*) Prediction. We start with demonstrating that WISE predicts entire response time distribution accurately. We then compare WISE predictions for *nrt* with simpler parametric models.

Predicting Distribution of nrt. We examine accuracy of *nrt* prediction for three scenarios. First two scenarios specify estimating the response-time distribution for the users in Germany, and South Africa, and the third scenario tries to estimate the response-time distribution for users that are served from FE in Japan. This FE data center primarily serves users in South and East Asia. We used the dataset from the third week of



(a) Germany (b) South Africa (c) FE in Japan

Fig. 7. Prediction accuracy: comparison of normalized response-time distributions for the scenarios in Section VII-F.

June 2007 as our training dataset and predicted the network response time distribution for these scenarios for the fourth week of June 2007.

Figures 7(a)-(c) show the results for the three scenarios, respectively. The ground-truth distribution for response time is based on the response-time values observed by the monitoring infrastructure for fourth week of June 2007, and is shown in solid line. The response time for the same period that WISE predicts is shown in a dotted line. The ground-truth and predicted distributions are identical.

WISE also predicts response time for individual requests accurately. Figure 8 shows the relative prediction error for individual requests in these experiments. The error is defined as $|rt - \widehat{rt}|/rt$, where rt is the ground-truth value and $\widehat{rt}$ is the value that WISE predicts. The median error lies between 8-11%.

Comparing WISE with Parametric Models. Because Google uses TCP to transfer the data to clients, It is reasonable to ask how well we could do using one of simpler parametric models used for estimating TCP-transfer latency? We have adapted work of Arlitt et al. [2] to account for additional latency that occurs due to round-trip between FE and BE (`febe_rtt`) and the back-end processing time (`be_time`). We refer to this model as AKM.

Table II presents a comparison of relative error in mean network response time estimates for ten regions using WISE and AKM. Table II-a presents errors for cases where the serving FE and BE are colocated in the same data center. Table II-b presents errors for cases where serving FE not in the same data center as the BE. These are also cases where access network is constrained, and there are significant packet losses and retransmissions. The relative error for mean response time for colocated cases is comparable between WISE and AKM. For non-colocated cases, error for AKM is between 3.5% and 9% more than WISE.

AKM uses a compensation multiplier called *'CompWeight'* to minimize the error for each trace. This factor minimizes the error on the mean at cost of higher error in the tails of the distribution. WISE, on the other hand, can predict the entire distribution accurately and also predicts the mean response time more accurately than AKM. Further WISE has advantage of generality. As we show in the next subsection, WISE can be easily extended to predict browser-level response time.

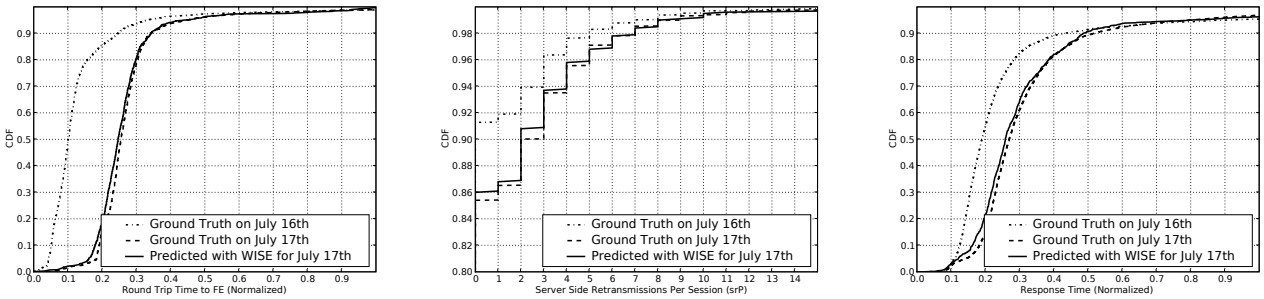
Browser-level Response Time (brt) Prediction. Browser-level response time is a high variance metric because of nature of Web page rendering process. For a typical region, both mean and standard deviation of *brt* is between 2 to 3 times more than mean and standard deviation for *nrt*. Further, as discussed in Section VII-C, our dataset does not contain variables that capture important contributing factors such as, DNS latency, how many embedded pieces of contents are required to render the page, whether the browser downloaded the embedded content or obtained these from cache, the computation capability of the user's host and the load on that host at the time of request. This makes estimating browser-level response time significantly more difficult.

(a) Co-located FE & BE			(b) Non-colocated FE & BE		
Region	WISE	AKM	Region	WISE	AKM
C1 Europe	2.0%	1.1%	C6 S. America	-2.2%	-11%
C2 Europe	0.3%	1.0%	C7 Australia	-0.9%	-10%
C3 E. Europe	0.3%	-6.7%	C8 S. America	0.2%	-8.9%
C4 Europe	0.0%	4.4%	C9 Asia	-3.0%	-7.7%
C5 N. America	1.4%	1.1%	C10 Asia Pacific	2.2%	5.9%

TABLE II
COMPARISON OF RELATIVE ERROR OF MEAN NETWORK-LEVEL RESPONSE TIME ESTIMATE BETWEEN WISE AND PARAMETRIC APPROACH.

We use WISE to estimate the browser level response time distribution for seven regions, served from a mix of colocated and non-colocated FE and BE locations. We use datasets from first two weeks in September 2009. Table III presents the relative error on key percentiles of the distribution of *brt* as well as mean *brt* for these regions. Prediction error on all key percentiles as well on mean for most regions is less than 10%. This is remarkable considering that *brt* is a very volatile metric and the dataset is missing indicators for a number of contributors of *brt* latency.

For two regions, D and G, the relative error for mean *brt* is greater than 10%. By examining javascript execution script in a limited scope experiment, we found that the reason for larger error is that the browser and computation capability combination in these countries is pretty unique, but because there is no feature in the dataset to indicate computation capability of hosts, the nearest-neighbor search algorithm in WISE ends up using samples that are similar in other dimensions but differing in computation capability, resulting in higher prediction error. We further ascertained this observation by using data from the same regions as training data. The results are shown at the bottom of Table III as D* and G*. When we use the local



(a) Client-FE RTT (b) Server Side Retransmits (c) Response Time Distribution

Fig. 9. Predicted distribution for response time and intermediary variables for the India data center drain scenario.

Region	Error on Percentiles			Error on Mean
	25^{th}	50^{th}	75^{th}	
B	12.0%	3.6%	4.1%	4.8%
C	1.7%	-3.9%	-10.1%	4.1%
D	0.4%	-1.6%	-4.2%	-10.9%
E	-1.2%	-3.6%	-5.2%	4.7%
F	0.3%	-1.5%	-2.5%	4.1%
G	9.2%	9.9%	10.7%	11.6%
I	4.0%	4.8%	5.8%	7.7%
D*	-3.1%	-2.6%	-3.4%	-7.8%
G*	1.4%	1.9%	4.4%	5.5%

TABLE III
RELATIVE ERROR OF MEAN BROWSER-LEVEL RESPONSE TIME ESTIMATE WITH WISE

datasets, prediction error drops significantly.

We are working to extend our datasets to include features that are indicative or computation capability of the hosts. Once these variables are available, we believe we will be able to improve prediction accuracy without using local datasets.

G. Evaluating a Live What-if Scenario

We evaluate how WISE predicts the response-time distribution for the affected set of customers during the scenario that we presented in Section II as a motivating example. In particular, we will focus on the effect of this event on customers of AS 9498, which is a large consumer ISP in India.

To appreciate the complexity of this scenario, consider what happens on the ground during this reconfiguration. First, because the FE in Taiwan is colocated with the BE, $febe\_rtt$ reduces to a typical intra-data center round-trip latency of 3ms. Also we observed that the average latency to the FE for the customers of AS 9498 increased by about 135ms as they were served from FE in Taiwan (`tw`) instead of the FE in India (`im`).

If the training dataset already contains the rtt estimates for customers in AS 9498 to the fe in Taiwan then we can write the scenario in two lines as following:

```
USE WHERE as_num==9498 AND fe==im AND be==tw
INTERVENE SET fe=tw
```

WISE uses the CBN to automatically update the scenario distribution. Because, fe variable is changed, WISE updates the distribution of children of fe variable, which in this case include $febe\_rtt$ and rtt variables. This in-turn causes a change in children of rtt variable, and similarly, the change cascades down to the rt variable in the DAG. In the case when such rtt is not included in the training dataset, the value can be explicitly provided as following:

```
USE WHERE as_num==9498 AND fe==im AND be==tw
INTERVENE SET febe_rtt=3
INTERVENE SET rtt+=135
```

We evaluated the scenario using the former specification. Figure 9 shows ground truth of response-time distribution as well as distributions for intermediary variables (Figure 6) for users in AS 9498 between hours of 12 a.m. and 8 p.m. on July 16^{th} and the same hours on July 17^{th}, as well as the distribution estimated with WISE for July 17^{th} for these variables. We observe only slight under-estimation of the distributions with WISE—this underestimation primarily arises due to insufficient training data to evaluate the variable distribution for the peripheries of the input distribution; WISE was not able to predict the response time for roughly 2% of the requests in the input distribution. Overall, maximum cumulative distribution differences[1] for the distributions of the three variables were between 7-9%.

VIII. DISCUSSION

In this section, we discuss the limitations and extensions for WISE. We present the difficulties that arise for specifying hypothetical values for network round-trip times for evaluating server placement or peering scenarios, and our on-going work to address these. We also discuss the merits of using non-parametric estimation in WISE and how we can extend it to using parametric models. Additional aspects are presented in our technical report [19] but omitted here for lack of space.

WISE also makes stability assumptions, i.e., the causal dependencies remain unchanged under any values of intervention, and the underlying behavior of the system that determines the response times does not change. We believe that this assumption is reasonable as long as the fundamental protocols and methods that are used in the network do not change.

Values of Variables for Scenario Specification. To predict response-time distribution, WISE requires the designers to specify the *what-if* value of one or more variables. While this considerably facilitates scenario specification and evaluation,

[1]We could not use the relative error metric here because the requests in the input distribution prepared with WISE for *what-if* scenario cannot be pairwise matched with ones in the ground-truth distribution; maximum distribution difference is a common metric used in statistical tests, such as Kolmogorov-Smirnov Goodness-of-Fit Test.

it is still a big problem in some cases. Specifically, specifying a scenario with correct value of network round-trip time (RTT) distribution to evaluate a new peering, or data center deployment has proven to be difficult.

Predicting the RTT distribution for a change in the network is difficult for a many reasons: A new peering or transit relationship can affect BGP level routing for many AS. WISE in its current form, does not try to predict the AS whose traffic would be affected by the change. Further, even when the subset of traffic that would be affected is identified, the change in RTT is usually not a simple step-function: the changes in RTT value different at different percentiles of distributions, higher percentiles are usually affected less by a shorter network path than lower percentiles.

WSL allows the designers to specify full distribution of a RTT using the `SETDIST` directive, if the distribution is known. We are separately investigating on two aspects that would facilitated this process further. (1) We are working modeling of RTT distribution by separately modeling the propagation and queuing delay factors in RTT and their interaction. This separation will allow the designers to specify the changes in the propagation delay and WISE would automatically estimate the likely queuing delay component and estimate the RTT distribution more accurately. (2) By analyzing the Internet connectivity graph, we are trying to estimate the propagation delays between AS and target CDN node locations. This would allow the designers to simply specify a hypothetical location for CDN node and WISE will be able to determine the propagation delays, queuing delays and the RTT distribution automatically. This RTT distribution can then be used to estimate overall response-time for the service.

Parametric vs. Non-Parametric. WISE uses the assumption of functional dependency among variables to update the values for the variables during the statistical intervention evaluation process. In the present implementation, WISE only relies on non-parametric, nearest-neighbor search based technique for estimating this function. This makes it difficult to use WISE for cases where dataset may not have suitable nearest-neighbor samples. Nearest-neighbor based prediction is also data intensive. Fortunately, nothing in the WISE framework prevents using parametric functions. If the dependencies among some or all of the variables are parametric or deterministic, then we can improve WISE's utility. Such a situation can in some cases allow *extrapolation* to predict variable values outside of what has been observed in the training dataset.

IX. Related Work

We are unaware of any technique that uses WISE's approach of answering *what-if* deployment questions, but WISE is similar to previous work on TCP throughput prediction and the application of Bayesian inference to networking problems.

A key component in response time for Web requests is TCP transfer latency. There has been significant work on TCP throughput and latency prediction using TCP modeling [2], [14], [6]. Due to inherent complexity of TCP these models make simplifying assumptions to keep the analysis tractable; these assumptions may produce inaccurate results. Recently, there has been effort to embrace the complexity and using past behavior to predict TCP throughput. He et al. [9] evaluate predictability using short-term history, and Mirza et al. [13] use machine-learning techniques to estimate TCP throughput. We also use machine-learning and statistical inference in our work, but techniques of [13] are not directly applicable because they rely on estimating path properties immediately before making a prediction. Further, they do not provide a framework for evaluating *what-if* scenarios. The parametric techniques, as we show in Section VII-E, unfortunately are not very accurate for predicting network-level response time. Further, TCP transfer is not sufficient for predicting browser-level response time.

WebProphet [12] uses dependencies between Web-page objects to predict browser-level response time. Although this technique is promising, the content of Web pages served from a CDN can vary widely, even for a single service; using this approach to predict response-time distributions would thus require analyzing each variant separately. Additionally, WebProphet requires collecting client-side measurements, which may not always be available to a CDN operator. As we show in Section VII-E WISE can accurately predict browser-level response time using only server-side logs.

Recent work has used Bayesian inference for fault and root-cause diagnosis. SCORE [11] uses spatial correlation and shared risk group techniques to find the best explanation for observed faults in the network. Shrink [10] extends this model to a probabilistic setting, because the dependencies among the nodes may not be deterministic due to incomplete information or noisy measurements. Sherlock [3] additionally finds causes for poor performance and also models fail-over and load-balancing dependencies. Rish et al. [17] combine dependency graphs with active probing for fault diagnosis. These projects focus on diagnosis but do not evaluate *what-if* scenarios.

X. Conclusion

Network designers must routinely answer questions about how changes to configuration and deployment will affect service response times. Without a rigorous method for evaluating these scenarios, the network designers must rely on ad hoc methods or resort to costly field deployments. This paper has presented WISE, a tool for evaluating *what-if* deployment scenarios for content distribution networks. To our knowledge, WISE is the first tool to automatically derive causal relationships from Web traces and apply statistical intervention to predict networked service performance. Our evaluation demonstrates that WISE is both fast and accurate: it can predict response-time distributions in "what if" scenarios to within an 11% error margin. WISE is easy to use; its scenario specification language makes it easy to specify complex configurations in just a few lines of code. WISE is also deployed: Googlehas used WISE to evaluate what-if scenarios in practice. WISE uses almost no domain knowledge to derive causal dependencies. Similar causal inference techniques might also help network designers understand the dependencies that affect more complex phenomena (e.g., misconfigurations or bugs). Another possible avenue for future work is to combine statistical techniques with domain knowledge to provide even more accurate predictions.

Acknowledgments

We would like to thank Andre Broido and Ben Helsley at Google, and the anonymous reviewers for ACM SIGCOMM

2008 for the feedback that helped improve this work. We would also like to thank Jeff Mogul for sharing source code for his prediction methods [2]. This work was supported in part by NSF grants CNS-0721559 and CNS-0721581, and NSF CAREER Award 0643974.

REFERENCES

[1] Akamai. http://www.akamai.com, 2010.
[2] M. Arlitt, B. Krishnamurthy, and J. Mogul. Predicting Short-transfer Latency from TCP arcana: a Trace-based Validation. In *Proc. ACM SIGCOMM Internet Measurement Conference*, New Orleans, LA, Oct. 2005.
[3] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.
[4] A. Barbir, B. Cain, R. Nair, and O. Spatscheck. *Known Content Network (CN) Request-Routing Mechanisms*. Internet Engineering Task Force, July 2003. RFC 3568.
[5] L. A. Barroso, J. Dean, and U. Holzle. Web Search for a Planet: The Google Cluster Architecture. Number 2, pages 22–28, 2003.
[6] N. Cardwell, S. Savage, and T. Anderson. Modeling tcp latency. In *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.
[7] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. 6th USENIX OSDI*, San Francisco, CA, Dec. 2004.
[8] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *Proc. First Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, Mar. 2004.
[9] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer tcp throughput. In *Proc. ACM SIGCOMM*, Philadelphia, PA, Aug. 2005.
[10] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: a tool for failure diagnosis in ip networks. In *ACM SIGCOMM workshop on Mining network data (MineNet'05)*, pages 173–178, New York, NY, USA, 2005. ACM.
[11] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Ip fault localization via risk modeling. In *Proc. 2nd USENIX NSDI*, Boston, MA, May 2005.
[12] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. Greenberg, and Y.-M. Wang. Webprophet: Automating performance prediction for web services. In *Proc. 7th USENIX NSDI*, San Jose, CA, Apr. 2010.
[13] M. Mirza, J. Sommers, P. Barford, and X. Zhu. A machine learning approach to tcp throughput prediction. In *Proc. ACM SIGMETRICS*, San Diego, CA, June 2007.
[14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM*, pages 303–323, Vancouver, British Columbia, Canada, Sept. 1998.
[15] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2000.
[16] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the data: Parallel analysis with sawzall. *Scientific Programming Journal: Special Issue on Worldwide Computig Programming Models and Infrastructure*, 13(4):227–298, Dec. 2005.
[17] I. Rish, M. Brodie, and S. Ma. Efficient fault diagnosis using probing. In *AAAI Spring Symposium on Information Refinement and Revision for Decision Making: Modeling for Diagnostics, Prognostics, and Prediction*, 2002.
[18] P. Sprites and C. Glymour. An algorithm for fast recovery of sparse causal graphs. In *Social Science Computer Review.*, volume 9, pages 62–72, 1991.
[19] M. B. Tariq, K. Bhandakar, V. Valancius, N. Feamster, and M. Ammar. Answering "what-if" deployment and configuration questions with wise: Techniques and deployment experience. Technical Report GT-CS-10-01, Georgia Tech School of Computer Science, Jan. 2010. http://www.cc.gatech.edu/~mtariq/pub/wise-tr.pdf.
[20] M. B. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering "What-if" Deployment and Configuration Questions with WISE. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.
[21] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference.* Springer, 2003.
[22] J. R. Wolberg. *Data Analysis Using The Method Of Least Squares: Extracting The Most Information From Experiments*. Springer, 2005.

Mukarram Bin Tariq is a Ph.D. student in Computer Science in the College of Computing at Georgia Tech. He is a member of Network Operations and Internet Security Lab. His research interests include performance modeling for cloud computing applications and causal analysis for networked systems. He received his B.S. degree from National University of Science & Technology, Rawalpindi, Pakistan in 2001.

Kaushik Bhandankar is an engineer atGoogle Inc. He received his M.S. degree from College of Computing at Georgia Tech, in 2008 and B.S. degree from National Institute of Technology, India in 2005. His research interests include understanding interaction and interdependence between networking systems with the eventual goal of using this knowhow to decrease latency of these systems in practice.

Vytautas Valancius is a Ph.D. candidate at Georgia Institute of Technology, advised by professor Nick Feamster. His research interests include interdomain routing and network virtualization. Prior to his Ph.D. studies, Vytautas obtained an M.S. at KTH, Sweden and worked in the networking industry as a consultant for 5 years, earning CCIE certification (#14359).

Amgad Zeitoun Amgad Zeitoun is a Senior Engineer at Google Inc at the large scale data processing group. He received his Ph.D. in Computer Science and Engineering from University of Michigan in 2004. He has received his M.S. degree in Computer Science and B.S. degree in Computer Engineering from Cairo University, Egypt in 1997 and 1999, respectively. His research interests include large-scale distributed systems, network measurements and traffic analysis, and workflow management systems.

Nick Feamster is an assistant professor in the College of Computing at Georgia Tech. He received his Ph.D. in Computer science from MIT in 2005, and his S.B. and M.Eng. degrees in Electrical Engineering and Computer Science from MIT in 2000 and 2001, respectively. His research focuses on many aspects of computer networking and networked systems, including the design, measurement, and analysis of network routing protocols, network operations and security, and anonymous communication systems. In December 2008, he received the Presidential Early Career Award for Scientists and Engineers (PECASE) for his contributions to cybersecurity, notably spam filtering. His honors include a Sloan Research Fellowship, the NSF CAREER award, the IBM Faculty Fellowship, and award papers at SIGCOMM 2006 (network-level behavior of spammers), the NSDI 2005 conference (fault detection in router configuration), Usenix Security 2002 (circumventing web censorship using Infranet), and Usenix Security 2001 (web cookie analysis).

Mostafa Ammar received the S.B. and S.M. degrees from the Massachusetts Institute of Technology in 1978 and 1980, respectively and the Ph.D. in Electrical Engineering from the University of Waterloo, Ontario, Canada in 1985. For the years 1980-82 he worked at Bell-Northern Research (BNR), first as a Member of Technical Staff and then as Manager of Data Network Planning. Dr. Ammar's research interests are in the areas of computer network architectures and protocols, distributed computing systems, and performance evaluation. He is the co-author of the textbook "Fundamentals of Telecommunication Networks," published by John Wiley and Sons. He is also the co-guest editor of April 1997 issue of the IEEE Journal on Selected Areas in Communications on "Network Support for Multipoint Communication." He also was the Technical Program Co-Chair for the 1997 IEEE International Conference on Network Protocols, 2002 Networked Group Communication Workshop, 2006 Co-Next Conference, and 2007 ACM SIGMETRICS Conference. Dr. Ammar is the holder of a 1990-1991 Lilly Teaching Fellowship and received the 1993 Outstanding Faculty Research Award from the College of Computing. He served as the Editor-in-Chief of the IEEE/ACM Transactions on Networking (1999-2003) and served on the editorial board of Computer Networks (1992-1999).