Veritas: Answering Causal Queries from Video Streaming Traces

Chandan Bothra*
Purdue University
Sanjay Rao
Purdue University
Jianfei Gao*
Purdue University
Bruno Ribeiro
Purdue University

ABSTRACT
In this paper, we consider the task of answering what-if questions in the context of adaptive bit rate (ABR) video streaming without access to randomized control trials (RCTs) (e.g., no A/B testing) — i.e., given recorded data of an existing deployed system, what would be the performance impact if we changed its design. Our work makes three contributions. First, we show the problem is challenging since data may only be available for a single ABR algorithm without RCTs, and since it is necessary to deal with the cascading effects that past ABR decisions have on future decisions. Next we present Veritas, the first framework that tackles causal reasoning for video streaming without requiring data collected through RCTs. Integral to Veritas is an easy-to-interpret domain-specific ML model that relates the latent stochastic process (intrinsic bandwidth that the video session can achieve) to actual observations (download times), while exploiting counterfactual queries via abduction using the observed TCP states (e.g., congestion window) for blocking the cascading dependencies. Third, we evaluate Veritas’s ability to accurately answer a wide range of what-if questions using emulation experiments, and data of real video sessions from Puffer. The results show that (i) Veritas accurately tackles a wider range of what-if questions (e.g., change of buffer size or video quality) that existing approaches cannot; (ii) Veritas without RCT training data achieves performance comparable or better than a recent parallel approach that requires RCT data; and (iii) in many scenarios Veritas achieves accuracy close to an ideal oracle.

CCS CONCEPTS
• Networks → Application layer protocols; Network measurement; Network performance modeling;

KEYWORDS
Video streaming; Causal Inference; Predictive models; Applying machine learning to networks.

1 INTRODUCTION
A central theme of data-driven networking is answering what-if questions — given data obtained from a real-world deployment of an existing deployed system, we want to infer what would have happened if we had used a different system design. For instance, given data collected from real video streaming sessions, a video publisher may wish to understand the performance if a different Adaptive Bitrate (ABR) algorithm were used (Figure 1), or if a new video quality (e.g., an 8K resolution) were added to the ABR selection, or an existing bit rate choice were removed (e.g., during the COVID crisis, many video publishers restricted the maximum bit rate [4]). Answering what-if questions of this nature is also known as causal reasoning. Causal inference considers the effect of events that did not occur while the data was being recorded [34], and has been explored in domains as diverse as economics [9] and epidemiology [38].

Shortcomings of traditional (associational) machine learning (ML). Several widely-used ML tools are inadequate for causal inference. Many approaches (e.g., neural networks and decision trees) merely capture correlations in collected data, limiting them to associations predictions, i.e., predictions that are related to associations between observations in a deployed system. Associations, however, are inadequate to answer causal questions. For instance, people carrying umbrellas on a sunny morning is a good predictor of rain in the afternoon. However, forbidding people to carry umbrellas in the morning does not prevent rain in the afternoon. Similarly, in video streaming, an ABR algorithm could choose lower bitrates when network conditions are poor, resulting in an association between lower video bitrates and rebuffering events. However, decreasing bitrate will not cause more rebuffering events — rather, the opposite is likely to happen.

The approach widely considered to be the gold standard for causal inference is Randomized Control Trials (RCTs). Both RCTs and other approaches such as Reinforcement Learning allow reasoning about a redesigned system but require active interventions that involve changing a system, and observing its performance among real users. While these approaches have several advantages, they must be conservatively deployed as they could be disruptive to the performance of real users, and may increase inequality of service.

*Both authors contributed equally to this paper and can be contacted at following: cbothra@purdue.edu, gao462@purdue.edu

ACM Reference Format:
Chandan Bothra*, Jianfei Gao*, Sanjay Rao, and Bruno Ribeiro. 2023. Veritas: Answering Causal Queries from Video Streaming Traces. In ACM SIGCOMM 2023 Conference (ACM SIGCOMM ’23), September 10–14, 2023, New York, NY, USA. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3603269.3604828
Veritas only relies on easy-to-interpret and low-complexity ML models, while only requiring pre-recorded data. The challenge that Veritas tackles is abduction [34, Section 4.2.4], which involves (i) inferring a set of likely values for latent variables consistent with the observations; and (ii) modeling the proposed changes to return the answer to a what-if query using the inferred latent variables. While abduction is challenging in general, the key insights of Veritas are (a) a careful selection of control variables (the TCP states at the start of each chunk download) that simplifies the causal task, and (b) a ML method to perform abduction that is principled, yet accessible and easy to interpret given it leverages domain insights. More specifically, as part of Veritas, we have designed a domain-specific ML model that relates the latent stochastic process (intrinsic bandwidth that the video session can achieve if TCP were in steady state throughout the session) to actual observations (actual throughput observed by chunk downloads), when also given a sequence of additional control variables in the form of the TCP states at the start of each chunk download. This control is needed since the actual observed throughput depends on the TCP state of the connection (e.g., whether slow-start is in progress), and the size of the downloaded object. The control allows us to “invert” the observed throughput variables in order to get the latent bandwidth variables.

To ensure we represent the statistical dependencies in the latent bandwidth time series during the inversion process, we develop an High-order Embedded Hidden Markov Model (HoEHMM), which embeds a domain-specific model for the emission process. A Bayesian posterior sampling of the HoEHMM allows us to capture the uncertainty inherent in the combination of our inversion, stochastic modeling, and the data. Once a sampled inverted bandwidth represents the answer to a what-if query. Rather than a single point estimate, Veritas provides a range of potential outcomes reflecting the inherent uncertainty in inferences that can be made from the data.

Evaluation. We evaluate Veritas with respect to its ability to answer a range of what-if causal queries including the impact of (i) changing the ABR algorithm; (ii) changing the buffer size; and (iii) changing the set of video qualities that the ABR algorithm could select. We compare the predictions from Veritas with: (a) a baseline approach that uses the logs directly without explicit causal adjustments; and (b) CausalSim [8], a parallel work, which also seeks to perform causal inference for video streaming. CausalSim has an RCT requirement in the training phase where each of N sessions is assigned to one of many ABR policies at random. We perform comparisons in: (a) controlled emulation settings where ground truth bandwidth information is available. Here, we also compare Veritas with an oracle that knows ground truth; and (b) “in-the-wild” settings using data of real video streaming session data obtained from Puffer [50]. Here, we also compare Veritas’ predictions with the true performance experienced by a distributionally similar group of users.

Results: We summarize our key results:
• For what-if questions pertaining to use of higher video qualities or a different buffer size, Veritas significantly outperforms CausalSim and the baseline while performing close to the oracle – e.g., when predicting the median quality across sessions if higher video qualities are used, Veritas achieves a prediction error of less than 0.022%,
while the baseline and CausalSim have a prediction error of 4.06% and 3.89% respectively. Note that the counterfactual capabilities of CausalSim are limited, since it is unable to accurately evaluate actions that were outside the scope of the initial RCT experiment as is the case with these what-if queries.

- For questions pertaining to the performance of a new ABR algorithm, our results show Veritas without RCT data (i.e., with data of only a single previously deployed algorithm) performs comparably to CausalSim with RCT data obtained from at least two other ABR algorithms. Further, Veritas outperforms both the baseline and CausalSim without RCT training data. For example, in a setting where both Veritas and CausalSim with RCT data correctly predict no rebuffering, CausalSim without RCT data incorrectly predicts rebuffering for about 18% of the sessions.
- Veritas without RCT data matches the performance of CausalSim with RCT data in answering what-if questions using real world video streaming obtained from Puffer, in some cases even performing better in estimating buffer occupancy and rebuffering ratio of the video sessions.

Overall, the results show new capabilities and the promise of Veritas in tackling a wide range of what-if queries while not requiring RCT training data. We have made the source code of Veritas publicly available1.

2 BACKGROUND AND MOTIVATION

In this section, we motivate the need for causal reasoning in context of video streaming, and why ML tools used for associational predictions, and approaches such as Reinforcement Learning and Randomized Control Trials fall short.

2.1 Causal vs. associational queries.

Causal vs. associational queries. Video streaming today typically involves splitting video into chunks, each encoded at multiple qualities. Clients pick qualities for each chunk using Adaptive Bit Rate (ABR) algorithms so as to balance between achieving high video quality, while avoiding rebuffering based on network conditions [7, 20, 21, 28, 41, 47, 51].

Consider data collected from a video streaming system, which includes the sizes and download times of chunks for each session. The following questions showcase the difference between associational and causal queries:

Q1. Given a set of observations of chunk sizes and download times of a video session, if the next download in the log is a chunk of size \( s \), what would be the download time?

Q2. Given a set of observations of chunk sizes and download times of a video session, if the designer had intervened in the session and had asked to next download a chunk of size \( s' \), what would be the download time?

Question Q1 pertains to passively observing the system at hand with its existing ABR algorithm and settings. These offline observations can be used to make predictions about the system under similar conditions. More broadly, an associational prediction seeks to predict outcomes of a system without interfering (intervening) with its operation. In contrast, many real-world networking tasks are like Q2, which require going beyond passively predicting the outcomes of an existing system. These tasks require causal inference, which predicts the outcome of an intervention, a change in the way the system operates. Specifically, Q2 pertains to the impact of an interventional change to the system design: how changing the ABR logic to download a chunk of different size impacts download time (and the session as a whole). More generally, the designer may wish to understand the implications on performance if some aspect of the system were changed (e.g., changing the set of video qualities the client could choose from, the buffer size, or the ABR algorithm).

We next define the two types of causal inference algorithms of interest in our work.

Definition 1 (Interventional inference for network tasks). Given (i) existing recorded sessions running an old method; (ii) and a new method; our task is to predict the performance of the new method on new sessions.

Definition 2 (Counterfactual inference for network tasks). Given (i) existing recorded sessions running an old method; (ii) and a new method; our goal is to predict the performance of the new method if it had been used in place of the old method in the same recorded sessions.

2.2 Confounders with video streaming.

Most ML methods work by learning associations in existing data, and, hence, are only appropriate for associational predictions. Unfortunately, the result of an associational prediction may be wildly inaccurate for a causal question owing to confounders. In the context of video streaming there are many confounders:

Intrinsic network bandwidth (INB) as an unobserved confounder. INB captures the bandwidth the network is intrinsically capable of, without considering dependence on size, and the transport protocol – i.e., what the transport protocol would intrinsically see if it were running in steady state. In order to explain INB as an unobserved confounder, we first present an illustrative example. We conduct controlled emulation experiments on 100 FCC throughput traces [2], split equally between poor [0-0.3 Mbps] and good [9-10 Mbps] network conditions, with the MPC algorithm [35] in an emulation testbed (details in §4.1). Figure 2(a) shows download times of chunks across all video sessions, with each boxplot corresponding to chunks within a particular size range.

Figure 2(a) shows a seemingly odd association, whereby increasing the chunk size may decrease the download time. This is a consequence of the adaptive bit rate (ABR) algorithm making the INB an unobserved confounder between chunk size and download times. When network conditions are poor (i.e., the INB is low), the ABR tends to select smaller chunk sizes. When network conditions are good (i.e., the INB is high), the ABR tends to select larger chunks. Hence, smaller chunks can have longer download times than larger chunks.

TCP state as an unobserved confounder. One may naively think that, while the association between download times and chunk sizes has the INB as a confounder, the association between observed throughput and chunk sizes should not have unobserved confounders. This is not the case. Figure 2(b) shows how observed throughput is dependent on the size of chunks owing to TCP behavior [11, 29, 50]. Figure 2(b) shows the distribution of throughput

1Available at https://github.com/Purdue-ISL/Veritas
for payloads in a given size range in controlled experiments using TCP where we emulated a constant network bandwidth of 18 Mbps. The experiments involved sending payloads of varying sizes (2KB to 4MB) in the same TCP session. The graph shows that for small sizes (less than the bandwidth delay product of the network), throughput is much smaller, while it is closer to the intrinsic network bandwidth for larger sizes. Thus, simply using the throughput observed in logs in a trace-driven simulation is inadequate as it may not accurately reflect the performance if a different size had been chosen.

2.3 Why not actively intervene on live users?

Rather than making predictions by passively observing a system, Randomized Control Trials (RCTs), A/B Testing, and Reinforcement Learning (RL) [44] can evaluate the impact of a design change by actively intervening (changing) the system, and observing the performance. While these approaches are valuable, they must be used judiciously as active intervention may lead to degraded performance to some viewers. In practice, A/B testing is typically used in a conservative fashion only after an initial offline analysis approach indicates the design change has sufficient potential.

RL may be viewed as a sequential RCT in that the agent dynamically learns the best decisions to take at each state of the system. A drawback of RCTs in general, and RL in particular, is that it only answers the question for pre-defined decisions. If our set of possible decisions changes, the RCT/RL algorithms must be run again. Further, both RCTs and RL cannot directly answer counterfactual queries, although their randomized (exploration) measurements may still be used by counterfactual estimators in some special cases (e.g., [10]). For instance, imagine seeing rare network conditions where a deployed algorithm performed poorly. RCTs and RL are generally not applicable in this scenario since the event is in the past, and any RCT to test a new intervention on the system can only be applied in future sessions (where the rare event may be difficult to reproduce).

2.4 Recent work and limitations.

A parallel work, CausalSim [8], supports causal queries but requires a training phase where each of $N$ sessions is assigned to one of many ABR policies completely at random (RCT traces). While a good advance, there are two crucial limitations of CausalSim.

First, the approach inherently requires training data using RCTs. To better understand this requirement, consider that we are given traces from the MPC algorithm [35], and we would like to understand the performance if we moved to the BBA algorithm [20]. Figure 3(a) presents the rebuffering ratios (ratio of stalls in a session to total duration of the session) seen with CausalSim in such a setting ($§4$ provides details of the evaluation methodology). Clearly, the rebuffering ratio predicted by CausalSim (No RCT) incorrectly has a sharp tail. In contrast, if two different algorithms (MPC and Bola [41]) were assigned to sessions using an RCT, and then CausalSim were trained using data collected from both algorithms, the accuracy in predicting rebuffering with BBA is far improved and closer to Ground Truth. Deploying ABR algorithms using RCT to collect traces can impact the performance of real-world users as discussed earlier and such data may not always be available.

Second, CausalSim can fail out-of-distribution, where new actions are available outside the scope of the initial RCT experiment (e.g., what if the ABR now allowed higher video qualities, or if we used a different buffer size). Consider CausalSim trained on data from two ABR algorithms: MPC and BBA, but each deployed with a small set of video qualities. Now, lets assume we are interested in evaluating performance if higher qualities were used with BBA. Figure 3(b) shows that in such scenario, CausalSim rebuffering ratio predictions are far from Ground Truth. This is because CausalSim was trained with data where video quality looks independent of rebuffering ratios, since for low qualities the bandwidth was sufficient to avoid rebuffering events. Unfortunately, this association is incorrect for higher qualities, where the bandwidth is now inadequate (as shown by the Ground Truth in Figure 3(b)). Veritas causal inference will not be impacted by this lack of association.

3 VERITAS: A CAUSAL INFERENCE FRAMEWORK FOR VIDEO STREAMING

This section presents Veritas, our framework for answering causal queries in video streaming. §3.1 presents the causal graph (DAG) in Figure 4, which models the variables involved in video streaming and their causal relationships. Using the DAG in Figure 4 we choose variables that block the cascading dependencies to propose an efficient (and theoretically sound) abduction procedure in §3.2.
A key factor that impacts the decisions made by a video streaming algorithm is the intrinsic network bandwidth (INB). Figure 4 shows a directed acyclic graph (DAG) describing the causal dependencies for video streaming. Figure 4 describes the evolution of INB as a discrete process over discrete time intervals $t \in \{1, \ldots, T\}$ (each of wall-clock time length of $\delta$), with the INB within an interval assumed constant. Time is assumed to be discrete to simplify our approach, since $\delta$ can be as fine-grained as necessary.

The session downloads a series of chunks $1 \ldots N$. Chunk $n \in \{1, \ldots, N\}$ starts downloading at time $s_n \in \{1, \ldots, T\}$ and finishes at time $e_n \in \{s_n, \ldots, T\}$. The variables that evolve over time are: (i) $C_t \in C$, the average INB in time interval $(t-1)\delta, t\delta]$; (ii) $B_t$, the amount of buffer in the video player at time $t \in \{1, \ldots, T\}$, and (iii) $W_t$, the TCP state at time $t$. The TCP state includes parameters such as the congestion window, RTT and min RTT.

The variables that evolve at each chunk request are: (i) the size $(S_n)$ of the $n$-th requested chunk and (ii) $D_n$, its download time, $n = 1, \ldots, N$. The throughput observed during the download ($Y_n$) can be calculated using $S_n$ and $D_n$.

Henceforth, for any random variable $X$ we define the sequences $X_{nsb} := (X_{n}, \ldots, X_{b})$ and $X_{sa_b} := (X_{sa}, \ldots, X_{ab})$. Moreover, let $\mathcal{S} := \cup_{n=1}^{N}\{s_n\}$ and $\mathcal{E} := \cup_{n=1}^{N}\{e_n\}$ be the set of random variables of showing the discrete times where a chunk starts and ends downloading, respectively. We assume that the variables in $W_{s_n, N}, B_{s_n, N}, S_{s_n, N}$. The observed variables in video streaming sessions (that is, all the information regarding them is either directly available, or can be calculated from the data). Note that TCP state information is easy to collect (e.g. using the tcp_info structure in Linux systems [6]). Further, although we could collect the information, we do not require the values $\{W_t\}_{t \in \{1, \ldots, T\}\setminus \mathcal{S}}, \{B_t\}_{t \in \{1, \ldots, T\}\setminus \mathcal{S}}$, and treat these variables as hidden.

Note that Figure 4 only illustrates the embedded process of \{\(C_t\)\}_{t \in \mathcal{S}\cup\mathcal{E}}, \{\(W_t\)\}_{t \in \mathcal{S}\cup\mathcal{E}}, \text{and} \{\(B_t\)\}_{t \in \mathcal{S}\cup\mathcal{E}} at the event times where a new chunk is requested or finishes downloading. It is important to note that the variables $C_{1:T}, W_{1:T}, B_{1:T}$ also evolve in the time between these chunk events, but for any time $t \in \{1, \ldots, T\}\setminus \mathcal{S}\cup\mathcal{E}$, that happens between chunk start and end times, the random variables $B_t$ depends only on $B_{t-1}$ (just the video being played) and $C_t$ depends only on $C_{t-1}$, but $W_t$ depends on both $W_{t-1}$ and $C_{t-1}$ if there is an active chunk download at time $t$ (and only on $W_{t-1}$ if there is no active download).

The $n$-th chunk size $S_n$ is influenced (through the ABR algorithm) by both the buffer state $B_{s_n}$ at the start of download of chunk $n$ and the last observed throughput $Y_{n-1}$. The chunk size value $S_n$ influences the download time $D_n$. Further, the TCP state $W_{s_n}$ (which includes the initial congestion window and RTT) along with $S_n$ and $C_{s_n}, \ldots, C_{e_n}$ also influence the download time $D_n$. $W_{s_n}$ itself potentially depends on the buffer at the end of the previous chunk $B_{e_{n-1}}$, as this determines the idle time between chunk downloads, that can impact TCP state for some implementations. Finally, as discussed above $S_n$ and $D_n$ together determine $Y_n$.

**Confounders:** The DAG in Figure 4 shows that $C_{1:T}$ are confounder variables between $S_{1:N}, D_{1:N}$, and $W_{s_n,N}$. Confounders are variables (often not available in the data) that cause spurious associations between multiple observed variables. Moreover, we make the simplifying assumption that $C_{1:T}$ are not influenced by any other variable in the model (that is, chunk downloads do not impact the INB). Veritas current assumes we are running a particular version of TCP (e.g., Cubic, or BBR) and currently does not address the impact of what-if questions where the TCP version itself changes.

### 3.2 Veritas abduction for causal queries

Since no other variables affect the confounder variables $C_{1:T}$ but $C_{1:T}$ directly or indirectly affect all other variables (i.e., all other variables are descendants of some variable in $C_{1:T}$), if we could infer $C_{1:T}$ we would be able to handle any counterfactual or interventional query needed. This procedure to infer a confounder ($C_{1:T}$) to respond to causal queries is known as abduction [34, Section 4.2.4]. Abduction involves \(i\) “inverting” the observed variables to get the hidden confounders; and \(ii\) then modeling the proposed changes (assuming the hidden confounder values are now known) to return the answer to the what-if query. Abduction approaches in ML typically rely on composable statistical models using high-level programming languages [12, 14, 30], and do not effectively deal
with the use of “if” statements and deterministic decision functions common in networking. Hence, we propose a custom abduction method tailored to our task.

The task: In our setting, abduction requires sampling the network INB given all the observations in a session:

$C_{1:T} \sim P(C_{1:T} = c_{1:T}) \{\text{All Observed Variables}\}, \quad (1)$

where $C \sim P(C = c[H = h])$ means random variable $C$ is sampled from its distribution conditioned on observing $H = h$. Once the confounding variables $C_{1:T}$ are sampled given the observables, abduction allows us to simulate the effect of the causal query over the sampled $C_{1:T}$ (now assumed known). We discuss how Veritas achieves this next.

Veritas’s High-order Embedded Hidden Markov Model (HoEHMM). Sampling the INB time series as described in Equation (1) is non-trivial. Hidden Markov Models (HMMs) [48] are commonly used to sample time series, but standard HMMs would require the emission $Y_n$ to be only depend on a single hidden variable (say, $C_{\eta_n}$). Unfortunately, $Y_n$ in the DAG of Figure 4 depends on a large set of variables.

Extending HMMs: Fundamentally, HMMs relies on the concept of $d$-separation to avoid cascading temporal dependencies. A sufficient condition for a set $U$ of random variables to $d$-separate a set $A$ and $B$ is that all undirected paths in the DAG between $A$ and $B$ include at least one variable from $U$, and no such paths have arrows collide “head-to-head” in the variables in $U$ [34, Definition 1.2.3]. One of the challenging aspects of Veritas is that $C_{\eta_n}$ does not $d$-separate $Y_n$ and $Y_{n-1}, \ldots, Y_1$ in the DAG of Figure 4, which then creates cascading dependencies between $Y_n$ and all other variables at time steps $Y_n, \ldots, Y_{n-1}, e_1, \ldots, s_1$. If we want to create a Markov model for Veritas, we must achieve $d$-separation. For this, Veritas’s Markov chain is defined over an extended variable-dimensional set of states.

Inspecting the DAG in Figure 4, we can see that $C_{\eta_n}, W_{\eta_n}, B_{\eta_n}$, and $S_{\eta_n}$, circled in red, block any undirected paths between $Y_n$ and $\{Y_{n-1}, \ldots, Y_1\}$. Moreover, $Y_n$ also depends on the sequence $C_{\eta_n:\eta_s}$, which has variable size.

We now define Veritas’s High-order Embedded Hidden Markov Model (HoEHMM), characterized by (i) a set of variable-dimensional high-order hidden states $\{(C_{(s_{\eta_n+1}, s_{\eta_n-1})}, B_{\eta_n}, S_{\eta_n}, W_{\eta_n})\}_{\eta_n=1}^{N}$; (ii) a matrix that captures the transition probabilities from one hidden state to another; (iii) a set of observations (which we will define later); (iv) a set of emission probabilities for $\{Y_1, \ldots, Y_n\}$, which capture the likelihood of a particular observation being generated from a given hidden state; and (v) an initial probability distribution over the states.

Creating a hidden Markov model of the evolution of $\{(C_{(s_{\eta_n+1}, s_{\eta_n-1})}, B_{\eta_n}, S_{\eta_n}, W_{\eta_n})\}_{\eta_n=1}^{N}$ would be unnecessarily complex. Thankfully, we can tame this complexity through partial observations. Specifically, we observe $\{(B_{\eta_n}, S_{\eta_n}, W_{\eta_n})\}_{\eta_n=1}^{N}$. Buffer size $B_{\eta_n}$ and chunk size $S_{\eta_n}$ are common observations in video streaming. Observing TCP state at the start of a chunk request $W_{\eta_n}$, while not as common, has precedent in the literature [50], and is key to tame temporal dependencies.

Veritas’s HoEHMM also departs from standard HMM models in other ways. First, HMMs traditionally use common parameterized probability distributions (e.g., multinomial, Gaussian) to model emission probabilities. Instead, Veritas embeds a domain-specific model for its emissions. The model captures how INB, chunk sizes, and TCP states gets translated into observed throughput.

Second, in traditional HMMs, each hidden state is associated with a single observation. However, in our context, observations are only associated with those hidden INB states where chunks are being downloaded. But the hidden INB itself still changes during the off periods (without chunk downloads) and no observations are available during these intervals. Further, it is possible that there are multiple chunks downloaded in the same time interval $(t - \delta, t + \delta)$, $t \in \{1, \ldots, T\}$. To handle this, Veritas’s HoEHMM allows each INB state to be associated with zero, one or more observations (corresponding to the number of chunks downloaded in the corresponding interval). Veritas’s HoEHMM is consistent with prior work [7, 43], which has modeled TCP throughput evolution as a Markov process, but Veritas addresses significant complexities associated with embedding a custom emission process, and $d$-separation. Further, our focus is on abduction for causal inference.

Hidden state transitions of Veritas’s HoEHMM. In Figure 4, only $S_n, W_{\eta_n}$ and $C_{\eta_n:\eta_s}$ affect $Y_n$. Since $B_{\eta_n}, S_{\eta_n}$ and $W_{\eta_n}$ are observed for any $h, 1 \leq h \leq N$, we now only need to focus on the transition probabilities $P(C_{(s_{\eta_n+1}, s_{\eta_n-1})} = C_{\eta_n} | C_{\eta_n-1})$. Our model assumes a time-homogeneous first-order Markov process $P(C_{t+1} | C_t) = P(C_t | C_{t-1})$, $1 \leq t \leq T$, where $C_t$ denotes the average INB during time interval $(t - \delta, t + \delta)$ (see §3.1 for details). For instance, $e = 0.5$ implies that the hidden states are $C = \{0.0\text{Mbps}, 0.5\text{Mbps}, 1.0\text{Mbps}, \ldots\}$. Both hyperparameters $\delta$ and $e$ can be as small as desired if computationally feasible. The conditional distribution $P(C_t | C_{t-1})$ is given by the transition probability matrix

$$A_{t,j} = P(C_t = j | C_{t-1} = i); \quad 1 < t \leq T, \quad (2)$$

where the prior $P(C_1)$ is also a hyperparameter of our model (assumed uniform in our experiments).

Parameterized hidden state transition matrix. In our model, practitioners can define custom parameterized state transition models via Pytorch [5] differentiable parameters (henceforth denoted $\theta$). For instance, a fully flexible transition model can be defined as $A_{t,j} = \theta_{i,j}$. Our experiments consider the following parameterized transition matrix:

$$A_{t,j}(\theta) = \frac{(1 - \eta)}{Z_i} \exp \left( -\frac{1}{2 \theta^2} (j - i)^2 \right) + \eta, \quad (3)$$

where $\theta^2 > 0$ is the learnable variance of a zero-mean Gaussian distribution, $\eta \in [0, 1)$ is a smoothing hyperparameter, and $Z_i$ is normalization factor ensuring that $\sum_j A_{t,j} = 1$. In all experiments, we use a fixed smoothing $\eta = 0.05$.

Domain-specific emission model. The throughput $Y_n$ observed by video chunk $n$ with start time $s_{\eta_n}$ and end time $e_{\eta_n}$ is a function of INB $C_{\eta_n:\eta_s}$, chunk size $S_{\eta_n}$ and the starting TCP state $W_{\eta_n}$ which includes congestion window $W_{\eta_n}^{\text{cwnd}}$ and minimum RTT $W_{\eta_n}^{\text{minRTT}}$. We develop a simple TCP model (Algorithm 4 in the Appendix) denoted by $f$ to estimate $Y_n$. The model is based on the following insight: the observed throughput matches INB if chunk sizes are sufficiently large, and transmission is not limited by cwnd. However, throughput is lower if limited by size or cwnd.

In more detail, we first calculate the Bandwidth Delay Product (BDP) using the INB, $C_{\eta_n:\eta_s}$ and $W_{\eta_n}^{\text{minRTT}}$. If both $W_{\eta_n}^{\text{cwnd}}$ and $S_{\eta_n}$ are larger than the BDP, $Y_n$ is close to the intrinsic network bandwidth,

743
INB. However, if $W_{w_{n}}$ is large but $S_{n}$ is smaller than BDP, then our TCP connection is size-limited and $Y_{n}$ is close to $S_{n}/W_{\text{minRTT}}$. When throughput is limited by cwnd ($W_{cwnd}$ is smaller than BDP), we calculate the number of transmission rounds needed to transmit size $S_{n}$ and estimate $Y_{n}$ using the number of transmission rounds, $W_{cwnd}$ and $S_{n}$. We use a linear increase to model the growth of $W_{cwnd}$ across rounds. Although this simplifies TCP behavior, we preferred this approach since it is more generic across congestion control algorithms and because Veritas tolerates error in $f$ (as discussed below). Further, even this simple approach produces promising empirical results (§4). We have extensively experimented with alternate $f$ functions which indicated that while it is important to model the impact on throughput with low cwnd, the performance was less sensitive to how changes in cwnd within a chunk download were modeled. Finally, $f$ is a hyper-parameter for Veritas, and custom $f$ functions that capture details of specific TCP algorithms can be easily incorporated (although this may require logging more TCP state information (e.g., ssthresh, time since last loss event, etc.).

If $f$ were to perfectly model TCP behavior we could define the emission probability distribution as

$$P(Y_{n} = y | W_{w_{n}}, S_{n}, C_{w_{n}}, c_{n}) = 1(y = f(C_{w_{n}}, c_{n}, W_{w_{n}}, S_{n})),$$

where $1(\cdot)$ is the indicator function. Since our TCP model is imperfect, we add uncertainty in the form of Gaussian white noise with a learnable variance. In our experiments we use a higher variance for the first few chunks to model TCP slow start effects at the start of the session. Please refer to Equation (5) in the Appendix for further details.

**Evolution of the embedded INB.** We now discuss how to estimate hidden states $C_{1:T}$ using an embedded Markov chain. Our embedding is inspired by the embedding in Neal et al. [31] (see Figure 5). More precisely, for $t \in \{1, \ldots, T\}$, instead of modeling $P(C_{t} | C_{t-1})$, we model the transitions $P(C_{n} | C_{n-1})$, where $1 < n \leq N$. For chunks $n-1$ and $n$, we will define $P(C_{n} = j | C_{n-1} = i) = (A^{n}_{i,j})_{i,j}$, where $A = \min(\frac{\Delta_{n}}{\delta}, \hat{\Delta}_{n})$ and $A$ is defined in Equation (2).

**Learning Veritas’s HoEHMM.** Finally, the algorithm to learn the model parameters is a variation of the Baum-Welch algorithm [36] with gradient descent updates, specifically tailored to our model (see Algorithm 2 in the Appendix).

**Abduction of $C_{1:T}$.** Once the model parameters are learned for a given session, Veritas’s abduction is performed by sampling $C_{1:T}$ given those HoEHMM parameters. We sample $C_{1:T}$ according to the posterior in Equation (1) using traditional Bayesian sampling methods [15, 37, 39]. The sampling requires computing the joint probability $P(C_{n} = i, C_{n+1:1} = j | Y_{1:N}, W_{r_{n:N}}, S_{1:N})$, which we obtain from our variant of the Baum-Welch forward-backward algorithm (see Algorithm 1 in the Appendix) and Algorithm 3 in the Appendix presents the complete sampling algorithm for $C_{1:T}$. It is divided into two steps. First, we sample $C_{n+1:N}$. Then, we sample the intermediate values $C_{1}$, where $t \in \cup_{n=2}^{N}(\{s_{n-1} + 1, s_{n-1} - 1\}$ according to transition matrix $A$ based on chunk samples $C_{s_{n:N}}$. Note that it is possible to sample INB values beyond time $T$ if necessary.

### 3.3 How Veritas answers causal queries

Figure 6 shows how Veritas may be used to answer counterfactual and interventional queries. The system deployed in the wild (Setting A) produces logs which for each chunk, which includes (i) size; (ii) start time of download; (iii) end time of download; and (iv) TCP state including cwnd, minRTT, and RTT [6] at the start of each download.

Veritas performs the abduction step by sampling $K$ likely INB sequences $C_{1:T}$ (Equation (1)) as discussed earlier. The counterfactual query is a video session emulated with the new Setting B using the sampled INB traces $C_{1:T}$ (e.g., Setting B may correspond to a different algorithm, or buffer size). Veritas’s emulation provides $K$ outcomes for the counterfactual query rather than just a single one, capturing the uncertainty inherent in the abduction step given the observed data. While the above description pertains to counterfactual queries, Veritas can also be used for interventional queries as described in §4.3.
4 EVALUATION

We evaluate Veritas with respect to its ability to answer a range of what-if questions using two sets of experiments:

- **Emulation experiments**, which compares Veritas’s predictions to ground truth. These experiments allow evaluations with variety of what-if questions (change of video qualities, buffer size and ABR algorithm), and also allow evaluation of counterfactual questions (i.e., questions pertaining to the exact same trace if alternate setting were considered).

- **Real world experiments**, where we use Veritas to predict the result of what-if questions on logs of real world video sessions collected by Puffer [50]. Here, ground truth is unavailable for counterfactual queries, but the dataset allows evaluation of interventional queries (i.e., predictions on sessions with similar characteristics) though only a class of queries pertaining to a change of ABR algorithm may be validated (§4.3).

### 4.1 Evaluation with counterfactuals

**Schemes compared.** We compare Veritas’s ability to perform counterfactual inference against several approaches:

- **Ground Truth:** This refers to the metrics collected when emulating the intrinsic ground truth network bandwidth (INB), defined in §3.1. This serves as the ideal benchmark other approaches must achieve.

- **Baseline:** This scheme estimates INB using the observed throughput of each chunk over the duration of chunk downloads. During off periods, when no estimate is available, linear interpolation of the throughput observed by the previous and next chunks is used. This scheme is commonly used in most video streaming evaluations today [7, 28, 51] but does not account for unobserved confounders.

- **CausalSim:** As discussed in §2.4, CausalSim [8] answers causal queries but requires training data obtained using an RCT where sessions are assigned to one of $K$ ABR policies at random. We use the code provided by the authors [1] in our experiments.

**Evaluation setup.** We use an evaluation setup similar to Figure 6. A video session in Setting A is emulated using a ground truth network bandwidth (INB) trace. The resulting logs are provided to the different schemes. Both Veritas and Baseline produce traces inferring INB. A video session is emulated in Setting B with the traces inferred by these schemes as well as the original INB trace to obtain the metrics predicted by these schemes, and ground truth. Veritas samples multiple inferred traces (five by default) for each video session, each of which are emulated. CausalSim does not require emulation — instead, it directly predicts the metrics of interest using the video logs from Setting A as input to a Neural Network model.

**Training.** Table 1 summarises the what-if questions that we explore, and the training data used for each of Veritas and CausalSim. Note that CausalSim requires RCT training data collected from two different ABR algorithms to evaluate performance in the target algorithm in Setting B, while Veritas only requires data from the source algorithm in Setting A. We train the CausalSim model [8] for Ground Truth evaluation with the optimal hyperparameters shared by the authors [1]. Veritas uses the HoEHMM described in §3.2 and is trained with the video logs from the Setting A. We optimise the HoEHMM model using native gradient descent algorithm. In our experiments we discretise INB using $\epsilon \in \{0.05, 0.5, 1\}$ Mbps, and discretise the time steps with $\sigma \in \{1, 5\}$ seconds. For the construction of transition matrix of HoEHMM, we assume the maximum capacity to be 1.5x the maximum observed throughput observed in the session.

**Metrics.** For any what-if question, each scheme predicts standard video session metrics such as video quality (measured by SSIM [49]), rebuffering ratios. We also present the distribution of buffer occupancy, and chunk download times predicted by each scheme.

**Setup details.** We use the evaluation setup of Yan et al. [50] to run our emulation experiments with different ABR algorithms and system settings. We emulate FCC throughput traces [2] to play a 5 minute pre-recorded video clip with bitrate ranging from 0.1 Mbps to 4 Mbps using Mahimahi [32]. Our emulation experiments use TCP CUBIC [18], and disable TCP Slow Start Restart [33, 33] as typically done in production video servers (our Puffer data in §4.3 is based on sessions running BBR). We use a standard video provided with Puffer [50], whose average SSIM index for the lowest and highest quality are 10.36 dB and 18.58 dB respectively. The clients are launched inside a mahimahi shell with a 80 ms end to end delay and downlink bandwidth limited by FCC traces. We select FCC traces with network bandwidth varying between 1 Mbps to 5 Mbps, a range of bandwidth typically used for non-trivial bitrate adaptations [7, 28, 50].

![Figure 7: Comparing INB, Baseline and Veritas samples in a typical experiment.](image-url)

Table 1: What-if questions explored in emulation experiments. Veritas only needs training data from a single ABR algorithm, while CausalSim requires training data from multiple algorithms using an RCT.

| Question | CausalSim (RCT required) | Veritas (No RCT required) |
|----------|--------------------------|--------------------------|
| Change of ABR (MPC to BBA) | 50% MPC, 50% Bola1 | 100% BBA (15s) |
| Change of buffer (15s to 5s) | 50% BBA (15s), 50% MPC (15s) | 100% BBA (15s) |
| Change of qualities (Low to High) | 50% BBA (Low), 50% MPC (Low) | 100% BBA (Low) |
4.2 Results with counterfactuals

We first illustrate Veritas’s sampled INBs for a typical FCC trace. Then, compare Veritas’s ability to accurately answer counterfactual queries with the ability of existing methods.

**Illustrating Veritas.** Figure 7 illustrates Veritas in action in a typical experiment. The green curve (INB) refers to the intrinsic ground truth bandwidth which is emulated. The red curve shows the trace created using the Baseline approach. Clearly, Baseline is conservative in its estimation of INB, especially in periods where the ABR algorithm selects smaller chunk sizes (either lower qualities, or lower-sized chunks of higher quality given variable bit rate video). Thus, trace-driven emulations using Baseline to answer "what-if" questions will lead to incorrect results.

The blue curves in Figure 7 show five sample traces inferred by Veritas for the same INB trace. All these samples are closer to INB than Baseline and significantly less conservative. Veritas may exhibit more uncertainty in regions where a range of different INB values may result in the same throughput observations. This is the intended behavior, since our causal estimates must account for estimation uncertainty.

**Change of video qualities.** Consider a scenario where a video streaming application has been deployed with a given set of low video qualities and we want to know the counterfactual what would have happened if a set of higher video qualities were used instead?

Figures 8(a) and (b) present the cumulative densities (CDFs) of average SSIM and rebuffering ratios across sessions for all tested methods. Figure 8(c) presents the CDF of the buffer occupancy at the start of each chunk across all sessions. Since Veritas provides many samples per session, we plot a CDF of all samples.

Figure 8 shows Veritas performs better than all alternatives. First, Veritas is close to Ground Truth for all metrics. Second, Baseline predicts lower buffer occupancies, higher rebuffering ratios and lower SSIMs than Ground Truth. This makes sense since Baseline uses observed throughput which tends to be conservative. Third, CausalSim predicts much higher buffer occupancies, higher SSIM and lower rebuffering than INB. This is because the training data (low qualities) is from a regime where the bandwidth is sufficient to support the video qualities. However, the what-if queries that relate to higher video qualities pertain to a regime where the same bandwidth is now inadequate. This highlights the limitation of predicting counterfactual queries using associations in the training data (as CausalSim’s neural network does).

Note that CausalSim predicts buffer occupancies much higher than the client buffer (15s). Hence, we also consider an alternate policy, CausalSim-Bounded, where predicted buffer occupancy is the minimum between CausalSim predictions and the maximum buffer capacity. This buffer value is then used by the simulated ABR algorithm in CausalSim. Figure 8 shows that while this caps the predicted buffer to the maximum 15s, the resulting predictions are still not close to ground truth. Overall, the results not only show the effectiveness of Veritas but also the limitations of CausalSim when evaluating actions outside the scope of the initial RCT.

**Change of buffer.** Next, we consider a what-if query pertaining to a change in client buffer size. Given logs of an ABR algorithm...
deployed with a 15s buffer, the goal is to predict the performance if the buffer size were reduced to 5s to move closer to a live streaming setting. Figure 9 presents results. Again, Veritas performs close to Ground Truth in all metrics. Baseline is more conservative predicting lower buffer occupancies, lower SSIM and higher rebuffering ratios. In contrast, CausalSim is optimistic in these metrics. In particular, it predicts high buffer occupancies similar to the values observed in the training data (which is based on a 15s buffer). During inference, since the predicted buffer values are high, CausalSim predicts higher quality chunks are selected, and predicts low rebuffering.

We again consider a policy CausalSim-Bounded, which limits the predicted buffer size to maximum of 5s or the predicted buffer occupancy, whichever is lower. The ABR algorithm simulated by CausalSim in the what-if query takes this adjusted buffer prediction. CausalSim-Bounded predicts buffer occupancies of 5 sec for most chunks, still not close to ground truth. Interestingly however, it now underestimates SSIM, and overestimates rebuffering ratios. This is because in the training data, a 5s buffer is associated with poorer network conditions in the original training data. This leads CausalSim-Bounded to predict lower video qualities and higher rebuffering ratios.

Change of algorithm. Consider that the video streaming application has been deployed with a given ABR. We ask the counterfactual what would have happened if an alternate ABR algorithm were instead used. We study this question in context of moving from the MPC [35] algorithm to BBA [20] algorithm. Recall that in this setting, Veritas does not have access to RCT data, but CausalSim does. Figure 10 presents results which show despite this, both Veritas and CausalSim perform similarly. We also present CausalSim (No RCT), where the scheme like Veritas is trained without RCT data (to make the comparison fair). Without RCT, CausalSim incorrectly predicts rebuffering for about 13% of the sessions, and it erroneously under-predicts and over-predicts buffer occupancies in some cases as indicated by the tails. Finally, when the SSIM metric is considered, CausalSim underpredicts, and Veritas overpredicts, but the prediction errors are comparable. We also note that a more conservative estimate is possible with Veritas by taking the more conservative of its predictions across samples (we elaborate further in §4.3). Overall, the results show that CausalSim (No RCT) performs poorly, but Veritas matches CausalSim (RCT) without requiring RCT data.

4.3 Validations with real world data

We next validate our results with real Internet video session data collected by Puffer [50], a video streaming platform. In Puffer, each video session is assigned an ABR algorithm randomly chosen from a set of algorithms. Owing to the random assignment, the distribution of network characteristics across sessions assigned to different algorithms may be assumed similar. Puffer makes logs available for all video sessions which include information such as chunk sizes, chunk download times, and buffer size.

Given logs collected from an ABR algorithm A1 (source algorithm), consider a what-if query that asks what would be the performance if an algorithm B (target algorithm) were used instead. Since INB is unknown, we cannot validate the resulting prediction with ground truth. However, the predictions of this query may be compared to the performance observed by sessions assigned to algorithm B. Thus, the predictions may be validated with data from a different set of sessions, but with similar network characteristics. Note that we can only validate questions related to a change of ABR algorithm using this dataset as Puffer does not assign buffer sizes or qualities randomly to sessions.

Evaluation setup: We consider a day of data (Aug 24, 2020) and focus on three ABR algorithms deployed on this day: BBA [20], and two versions of BOLA 2 [41], which we refer to as Bola1 and Bola2. We consider all six combinations of source and target algorithms shown in Table 2, and compare the predictions of Veritas and CausalSim. CausalSim is trained on RCT data from multiple source ABR algorithms, while Veritas is only trained on data from a single source as shown. We train CausalSim for each target algorithm using that day’s data and tune their loss hyperparameter for each training dataset using author provided scripts.

Results. We present results for slow streams (defined by Puffer as streams with mean delivery rate less than 6 Mbps), which are more likely to involve non trivial bitrate adaptation [8, 50]. Figure 11 compares CausalSim and Veritas with the real world data of the target algorithm. Results from all six source target combinations are combined for brevity for each metric. We make several points. First, although Veritas does not use RCT, it matches and even slightly outperforms CausalSim which needs RCT training data. When predicting download time across chunks and average SSIM per session, both schemes are almost indistinguishable from each other, and from the performance of real world target sessions. When buffer occupancy is considered, Veritas slightly overestimates but CausalSim underestimates – the median buffer occupancy of real world sessions is 9.08s, while the median with Veritas is 9.51s, and with CausalSim is 7.98s. Finally, Veritas slightly underestimates rebuffering, while CausalSim overestimates –for instance, although 12% of real world sessions see rebuffering, CausalSim estimates 20% see rebuffering, while Veritas estimates 10% see rebuffering.

Recall, Veritas generates $K=5$ candidate samples for each input trace and our results so far consider all predictions for each session (which is akin to taking the median prediction). A practitioner may instead wish to obtain conservative estimates of SSIM and rebuffering when making a proposed change. We also consider a

| Source ABR | Target ABR | CausalSim (RCT) training | Veritas training |
|------------|------------|--------------------------|-----------------|
| Bola1      | BBA        | Bola1, Bola2              | Bola1           |
| Bola2      | BBA        | Bola1, Bola2              | Bola2           |
| BBA        | Bola1      | BBA, Bola2                | BBA             |
| Bola2      | BBA        | BBA, Bola2                | Bola1           |
| Bola1      | Bola2      | Bola1, BBA                | Bola1           |
| BBA        | Bola2      | Bola1, BBA                | BBA             |

Table 2: Experiments on Puffer dataset. For each source and target pair, Veritas is only trained on data from a single source ABR, while CausalSim is trained on RCT data from multiple source algorithms.

---

2Puffer deploys two versions of BOLA with different quality objectives [3].
scheme that we refer to as Veritas(Conservative), which for each session only considers the most conservative estimate of SSIM and rebuffering ratios across the five samples. Veritas (Conservative) performs very close to the real world sessions, estimates a 95th percentile rebuffering ratio to be 5.97%, which is close to real world value of 7.39%. In addition, Veritas (Conservative) is practically indistinguishable from Veritas and other schemes in SSIM, indicating SSIM predictions across samples are close.

Finally, we also compared the performance of Veritas and CausalSim across all the sessions for the day. Veritas (without RCT) and CausalSim (with RCT) perform similarly and are comparable to real world sessions across all the metrics — we defer these results to the Appendix (Figure 12).

5 DISCUSSION

Veritas focuses on INB as a confounding variable while using observations of RTT and other TCP state variables. Veritas models packet loss implicitly as loss rate is one of the factors that impact INB. An open question for the future is whether modeling loss rate more explicitly can improve the performance of Veritas.

Veritas assumes that the bitrate decisions made by a video client does not impact the INB experienced by the video session. This is reasonable when the bottleneck link has large number of concurrent sessions. However, when the bottleneck link has only a small number of concurrent sessions (e.g., on a link connecting a home network to the Internet), more explicitly modelling the effect of concurrent sessions in Veritas may be important. Veritas assumes chunks of a video session are served from a single CDN server, but the chunks can be served from different layers of the CDN owing to cache misses in the edge layer, especially for less popular content [17]. Investigating and addressing such additional confounders could further improve the performance of Veritas.

Another important future direction is exploring how to combine Veritas and RCTs, which have complementary benefits. Dealing with confounding variables might be easier with RCTs as they directly measure the impact of intervention on active users. However, doing so can lead to degraded performance and thus RCTs tends to be used sparingly. Techniques such as Veritas can be used for offline analysis to explore a wide range of design alternatives with the most promising design choices then tested using RCTs.

6 RELATED WORK

• **Biases in video streaming.** A preliminary workshop paper [42] inspired both CausalSim [8] and our work. However, [42] is restricted to a square wave bandwidth process, does not model the dependence of observed throughput on chunk size, or handle the uncertainty in inference. Finally, the use of matching in [42] requires bitrates to be occasionally chosen randomly. Another work [11] has observed that smaller chunk sizes may see poorer throughput than larger ones owing to TCP slow start effects. To handle this, [11] compares the total reward seen by algorithm B on a trace collected from an algorithm A by only considering those chunks where the new algorithm picks the same bitrate as the old algorithm. The approach does not tackle *what-if* questions, assumes a constant bandwidth process, and does not model the causal dependence of chunk size selection by the ABR algorithm on bandwidth. We tackle
the harder problem of inferring a latent and variable bandwidth process from observed throughput, deal with the uncertainty in such inference, and address a wide range of causal what-if queries.

• Inferring causal dependencies and what-if analysis. Several works [23, 27, 45] infer causal dependencies using correlations but do not consider latent confounders. Some work [19, 23, 24, 46] deals with observed confounders – e.g., Krishnan et al. [24] explored whether video stream quality (e.g., rebuffering ratios) causally impacts user engagement metrics while accounting for observed confounders such as user connection type (DSL vs. mobile) and location. These works only infer if a correlation is an indication of a causal relationship but do not answer what-if questions, and do not deal with latent confounders. Other works [22, 40, 45] consider what-if analyses for various applications, but do not address confounding variables. Recent work [25] considers causal questions while considering implicit feedback in the context of cloud systems, relying on randomized experiments (from RL exploration).

7 CONCLUSION

In this paper, we make three contributions. First, we have shown the viability of answering what-if counterfactual and interventional queries related to video streaming without access to RCT data (A/B testing) through causal inference. Next, we present Veritas, the first framework that tackles causal inference for video streaming using data from a single ABR algorithm (i.e., no need for A/B testing, RCT training data). Veritas uses a High-order Embedded Hidden Markov Model (HoEHMM) that relates the unobserved INB time series to the throughput observed by the application. A key insight behind Veritas is exploiting information about the TCP state at the start of each chunk download to simplify the causal inference model. Third, we show the effectiveness of Veritas in answering a wide range of counterfactual and interventional queries using emulation testbed experiments and real-world datasets. In a counterfactual query pertaining to increase in video quality, Veritas estimates the median quality of video sessions within 0.022% of the Ground Truth while Baseline and CausalSim incur an error of 4.06% and 3.89% respectively. On questions that involve a change of ABR algorithm, Veritas (without RCT training data) performs comparably to Ground Truth and CausalSim which has access to RCT data; CausalSim without RCT data incorrectly predicts rebuffering for more than 13% of the sessions. Validations with real-world datasets confirm the promise of Veritas.

This work does not raise any ethical issues.

8 ACKNOWLEDGEMENT

We thank Cloudlab [16] for providing us with the computing resources to run the experiments. We thank Francis Yan and Keith Winston for helpful discussions about the Puffer testbed. Finally, we thank our shepherd, Michael Schapira, and the anonymous reviewers for their feedback which greatly helped improve the paper. This work was funded in part by an Amazon Research Award, and by National Science Foundation (NSF) Awards IIS-1943364 and CNS-2212160.

REFERENCES

[1] CausalSim. GitHub. https://github.com/CausalSim/Unbiased-Trace-Driven-Simulation.
[2] Federal communications commission. 2016. raw data - measuring broadband america. (2016). https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016.
[3] Implementing BOLA-BASIC on puff. https://puffer.stanford.edu/bola/
[4] Netflix and YouTube agree to reduce bitrate during Coronavirus crisis. https://www.broadbandtvnews.com/2020/03/19/netflix-agrees-to-reduce-bitrate-during-coronavirus-crisis/.
[5] PyTorch. https://pytorch.org/.
[6] tcp - Linux manual page. https://man7.org/linux/man-pages/man7/tcp.7.html.
[7] Zohaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: auto-tuning video ABR algorithms to network conditions. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication - SIGCOMM ’18, pages 44–58, Budapest, Hungary, 2018. ACM Press.
[8] Abdullah Alomar, Pouya Hamadian, Arash Nasr-Esfahany, Anish Agarwal, Mohammad Alizadeh, and Devavrat Shah. Causalsim: Toward a causal data-driven simulator for network protocols. arXiv preprint arXiv:2201.06112, 2022.
[9] Jochen D. Angrist, Guido W Imbens, and Donald B Rubin. Identification of causal effects using instrumental variables. Journal of the American statistical Association, 91(434):444–455, 1996.
[10] Eliaas Bareinboim, Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. Advances in Neural Information Processing Systems, 28:1342–1350, 2015.
[11] Mihovil Bartolovic, Junchen Jiang, Sivaraman Balakrishnan, Vyas Sekar, and Bruno Sinopoli. Bases in Data-Driven Networking, and What to Do About Them. In Proceedings of the 16th ACM Workshop on Hot Topics in Networks - HotNets-XVI, pages 192–198, Palo Alto, CA, USA, 2017. ACM Press.
[12] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Nennja Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Storril, Paul Horstall, and Noah D Goodman. Pyro: Deep Universal Probabilistic Programming. Journal of Machine Learning Research, 20(28):1–6, 2019.
[13] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. TCP Congestion Control. RFC 5681, September 2009.
[14] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A Probabilistic Programming Language. Journal of Statistical Software, 76(1), 2017.
[15] Siddhartha Chib. Calculating posterior distributions and modal estimates in markov mixture models. Journal of Econometrics, 75(1):79–97, 1996.
[16] Dmitriy Dwylyakin, Robert Ricci, Alekander Mariç, Gary Wong, Jonathan Duerig, Eric Eide, Leigh Stoller, Mike Hilder, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cechet, Snigdhaswar Kar, and Prabodh Mishra. The design and operation of Cloudlab. In Proceedings of the USENIX Annual Technical Conference (ATC), pages 1–14, July 2019.
[17] Elbah Ghabashneh and Sanjay Rao. Exploring the interplay between cdn caching and video streaming performance. In 2020 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2020.
[18] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: A new tcp-friendly high-speed tcp variant. SIGOPS Oper. Syst. Rev., 45(2):64–74, jul 2008.
[19] Hadrien Hours, Ernst Biersack, and Patrick Loiseau. A Causal Approach to the Study of TCP Performance. ACM Transactions on Intelligent Systems and Technology, 7(2):1–25, December 2016.
[20] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM ’14, pages 187–198, New York, NY, USA, 2014. ACM.
[21] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, CoNEXT ’12, pages 97–108, New York, NY, USA, 2012. ACM.
[22] Yurong Jiang, Lenin Ravindranath Sivalingam, Suman Nath, and Ramesh Govindan. WebPerf: Evaluating What-if Scenarios for Cloud-hosted Web Applications. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication - SIGCOMM ’16, pages 258–251, Florianopolis, Brazil, 2016. ACM Press.
[23] Satoru Kobayashi, Katsuki Otomo, Kensuke Fukuda, and Hiroshi Esaki. Mining Causality of Network Events in Log Data. Journal of Machine Communications - SIGCOMM ’16, pages 258–251, Florianopolis, Brazil, 2016. ACM Press.
[24] S. Shunmugan Krishnam and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In Proceedings of the 2012 Internet Measurement Conference, IMC ’12, pages 211–224, Boston, Massachusetts, USA, November 2012. Association for Computing Machinery.
Veritas: Answering Causal Queries from Video Streaming Traces

[25] Mathias Lécuyer, Sang Hoon Kim, Mihir Nanavati, Junchen Jiang, Siddhartha Sen, Alekseandr Slivkins, and Amit Sharma. Sayer: Using Implicit Feedback to Optimize System Policies. ACM Symposium on Cloud Computing (SOCC), New York, NY, USA, 2021.

[26] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. ACM SIGCOMM Computer Communication Review, 42(4):359–370, 2012.

[27] Ajay Anil Mahimkar, Zhihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Qi Zhao. Towards automated performance diagnosis in a large IPTV network. ACM SIGCOMM Computer Communication Review, 39(4):231–242, 2009. Publishers: ACM New York, NY, USA.

[28] Hongti Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pages 197–210. ACM, 2017.

[29] Yun Seong Nam, Jianfei Gao, Chandan Bothra, Ehab Ghabashneh, Sanjay Rao, Bruno Ribeiro, Jihun Zhan, and Hui Zhang. Xatu: Richer neural network based prediction for video streaming. ACM SIGMETRICS, 2022.

[30] Siddharth Narayanaswamy, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank D. Wood, and Philip H. S. Torr. Learning disentangled representations with semi-supervised deep generative models. In NIPS, pages 5927–5937, 2017.

[31] Ravi Netravali, Aniruddh Sivaraman, Keith Winston, Somak Das, Ameesh Goyal, and Hari Balakrishnan. Mahimahi: A lightweight toolkit for reproducible web measurement. In Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM ‘14, page 129–130, New York, NY, USA, 2014. Association for Computing Machinery.

[32] Jitendra Padhye, Sally Floyd, and Mark J. Handley. TCP Congestion Window Validation. RFC 2861, June 2000.

[33] Judea Pearl. Causality: Cambridge university press, 2009.

[34] Yanyuan Qin, Ruofan Jin, Shuai Hao, Krishna R Pattipati, Feng Qian, Subhabrata Sen, Chaoqun Yue, and Bing Wang. A control theoretic approach to abr video streaming: A fresh look at pid-based rate adaptation. IEEE Transactions on Mobile Computing, 2019.

[35] Lawrence R Rabiner. A Tutorial on Hidden Markov Models and Selected Applications. In the Workshop on Network Meets AI & ML, NetAI ’20, page 42–47, New York, NY, USA, 2020. Association for Computing Machinery.

[36] Kenneth J. Rothman and Sander Greenland. Causation and Causal Inference in Epidemiology. American Journal of Public Health, 95 Suppl 1:S144–150, 2005.

[37] Steven L Scott. Bayesian analysis of a two-state markov modulated poisson process. Journal of Computational and Graphical Statistics, 8(3):662–670, 1999.

[38] Rahul Singh, Prashant Shenoy, Maitreya Natu, Vaishali Sadaphal, and Harrick Yin. Analytical modeling for what-if analysis in complex cloud computing applications. ACM SIGMETRICS Performance Evaluation Review, 40(4):53–62, April 2013.

[39] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitarasam. Bola: Near-optimal bitrate adaptation for online videos. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pages 1–9. IEEE, 2016.

[40] P. C. Sruthi, Sanjay Rao, and Bruno Ribeiro. Pitfalls of data-driven networking: from error visibility to structural similarity. IEEE Transactions on Mobile Computing, 13(4):600–612, 2004.

[41] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, Janines Hong, Keyi Zhang, Philip Levis, and Keith Winston. Learning in situ: a randomized experiment in video streaming. In 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), pages 495–511, 2020.

[42] Xiaoqi Yin, Aphiishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM ’15, London, United Kingdom, 2015.
A APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A.1 Models and Algorithms

In this part, we will clarify some details of our models and present the pseudo code for all algorithms: our Baum-Welch variant Algorithms 1 and 2, and intrinsic network bandwidth (INB) sampling Algorithm 3, and network throughput estimator Algorithm 4 in our HoEHMM.

Why \( B_{s_{1:N}} \) need not be observed. In Figure 4’s DAG, start time \( s_{1:N} \) are not defined as random variables to simplify exposition. If we had defined \( s_{1:N} \) as an observed random variable, \( s_{1:N} \) could have been used in place of \( B_{s_{1:N}} \) to define the sufficient set of observed variables in our \( d \)-separation argument. Looking at the dependence between \( P(C_{s_n} = j|C_{s_{n-1}} = i) \) and \( \Delta_n \) makes it clear that observing \( s_{1:N} \) is also necessary for our Markov model. To conclude, then, we do not actually need to log \( B_{s_{1:N}} \) since \( s_{1:N} \) is necessary and sufficient and readily available in the trace.

Domain specific emission process. We use a simple model \( f \), which estimates throughput given INB, TCP state and size of related download chunk. The pseudo code is provided in Algorithm 4. Besides, we can not guarantee \( f \) being a perfect estimator for real-world throughputs, thus we also need to take uncertainty of \( f \) by Gaussian noise (Equation (4)) as shown below in Equation (5). The variance is higher in the initial stage of the session to model TCP slow start effects at the start of a session.

\[
P(Y_n|W_{s_n}, S_{s_n}) = \begin{cases} \frac{1}{\sigma_1^2} \exp\left(-\frac{1}{2\sigma_1^2} (f(C_{s_n}, W_{s_n}, S_{s_n}) - Y_n)^2\right) & s_n \leq T_{\text{stable}} \\ \frac{1}{\sigma_2^2} \exp\left(-\frac{1}{2\sigma_2^2} (f(C_{s_n}, W_{s_n}, S_{s_n}) - Y_n)^2\right) & s_n > T_{\text{stable}} \end{cases}
\]

(5)

Here, \( T_{\text{stable}} \) is the switching time between the initial phase that includes slow start and the steady state phase and \( Z_t \) is a normalization factor. The uncertainty of \( f \) will vary between two stages, and we take this into consideration with two variance \( \sigma_1^2 \) and \( \sigma_2^2 \) for each stage respectively.

Parameter Tuning. There are two groups of learnable parameters: One is transition matrix parameters \( \theta \), the other is estimator uncertainty parameters \( \sigma \). Since our HoEHMM can be treated as a variant of HMM-Gaussian model, we can utilize the Baum-Welch algorithm, which can provides the best estimation of transition matrix and emission variances (uncertainty) given training data [36]. Suppose the transition matrix and emission variances learned by Baum-Welch are \( A^\ast \) and \( (\sigma^\ast)^2 \), we can construct a Mean Squared Error (MSE) loss function

\[
I(\theta, \sigma) = \|A^\ast - A(\theta)\|^2 + \| (\sigma^\ast)^2 - \sigma^2 \|^2
\]

and use vanilla gradient descent algorithm to update learnable parameters \( \theta \) and \( \sigma \) in each Baum-Welch iteration. The pseudo code of a full update iteration is provided in Algorithm 2.

Algorithm Pseudo Code. As introduced in Section 3.2, our Baum-Welch variant is nearly the same as their origins, but replace the transition matrix from constant matrix \( A \) to \( A^h \) where \( \Delta_n \) as shown in Section 3.2 and Figure 5, and replace the emission process by our proposal as Equation (5). The pseudo code of variants for Baum-Welch forward-backward and update are provided in Algorithm 1 and Algorithm 2. Besides, we also provide IND sampling algorithm in Algorithm 3. In what follows we denote the pair distribution

\[
\Gamma_{i,j,n} = P(C_{s_n} = i|C_{s_{n+1}} = j|Y_{1:N}, W_{s_{1:N}}, S_{1:N})
\]

(6)

Input: State Space \( C \), Transition times \( T \), Initial distribution \( u_1 \), Transition matrix \( A \), Emission process \( F \) (Equation (5)), Throughputs \( Y_{1:N} \), TCP states \( W_{s_{1:N}} \), Chunk sizes \( S_{1:T} \), interval gaps \( \Delta \), capacity unit \( e \)

Output: Forward Distribution \( \alpha \), Backward Distribution \( \beta \), Conditional Distribution \( \gamma \), Conditional Joint Distribution \( \Gamma \)

/* Alias */
\[
\gamma_{s_{1:N}, n} = A_{s_{1:N}, n}^{\Delta_n} P(Y_{s_{1:N}, n}, S_{s_{1:N}}) \forall i, j \in C, 2 \leq n \leq N
\]

/* Forward */
\[
\alpha_{i,n} = u_1 Y_{i,n} P(Y_{1:n+1}, S_{1:n+1}) \forall i \in C
\]

for \( n = 2 \rightarrow N \) do

\[
\alpha_{i,n} = \sum_{j \in C} \alpha_{i-1,n} \gamma_{s_{1:N}, n} \forall i \in C
\]

end

/* Backward */
\[
\beta_{i,n} = f_{\text{for}} \gamma_{s_{1:N}, n+1} \forall i \in C
\]

for \( n = N - 1 \rightarrow 1 \) do

\[
\beta_{i,n} = \sum_{j \in C} \beta_{i,n+1} \gamma_{s_{1:N}, n+1} \forall i \in C
\]

end

/* Posterior */
\[
\Xi_{i,n} = \frac{\alpha_{i,n} \beta_{i,n}}{\sum_{j \in C} \alpha_{j,n} \beta_{j,n}}
\]

end

Algorithm 1: Forward-Backward Algorithm. It first computes forward distribution \( \alpha_{i,j,n} = P(C_{s_n} = i|Y_{1:n}, W_{s_{1:N}}, S_{1:N}) \); then computes backward distribution \( \beta_{i,j,n} = P(C_{s_n} = i|Y_{n+1:N}, W_{s_{n+1:N}}, S_{n+1:N}) \); and finally achieve conditional joint distribution \( \Gamma_{i,j,n} = P(C_{s_n} = i, C_{s_{n+1}} = j|Y_{1:N}, W_{s_{1:N}}, S_{1:N}) \) by combining \( \alpha \) and \( \beta \) for all \( i,j \) in INB state space from 1 to \( N - 1 \) chunks.
Algorithm 2: HoEHMM Update. It is a combination of Baum-Welch and gradient descent update. It first uses Baum-Welch algorithm to achieve expected transition matrix and uncertainty, then uses gradient of Mean Squared Error (MSE) loss to update transition and uncertainty parameters.

Input: State space $C$, Length $T$, Forward distribution $\alpha$ from Algorithm 1, Transition A, Pair distribution $\Gamma$

Output: A sampled capacity trace $C$

/* Chunk-level sampling. */

$C_{SN_i} \sim \text{Multinomial}(\alpha_n / \sum_{j \in C} \alpha_{n,j}) \cdot \epsilon$

for $n = N - 1$ to 1 do

$\xi_n, i = \Gamma_i C_{n+1, i}, n, i \in C$

$\pi_n, i = \xi_n, i / \sum_{j \in C} \xi_n, j, i \in C$

$C_{SN} \sim \text{Multinomial}(\pi_n) \cdot \epsilon$

end

/* Interval-level sampling. */

for $n = 1$ to $N - 1$ do

$p_n^t = P(Y_t | W_n, S_n, C_{SN})$

for $t = s_n + 1$ to $s_{n+1} - 1$ do

$\xi_n^t = A_{n, t-1} \cdot p_n^t \cdot A_{n+1, n+1}^{-1} \cdot i, e \in C$

$\pi_n^t = \xi_n^t / \sum_{j \in C} \xi_j^t, i \in C$

$C_t \sim \text{Multinomial}(\pi_n^t) \cdot \epsilon$

end

end

Algorithm 3: Capacity Sampler. It obtains the last state $N$ as the last state of Viterbi output, then forward samples each state $1 \leq n < N$ based on sampled state $n + 1$ and scores defined by Equation (6). After sampling capacities of chunks, it samples a second time to generate capacities of every interval.
Figure 12: Validation with all real video sessions obtained from Puffer during a day. Veritas without RCT data matches the performance of CausalSim with RCT data. The curves are indistinguishable for download time, buffer and SSIM. While Veritas underestimates and CausalSim overestimates the rebuffering ratio, Veritas (Conservative) is much closer to Ground Truth rebuffering ratio.