Prophet: Fast Accurate Model-based Throughput Prediction for Reactive Flows in Data Center Networks

Kai Gao Jensen Zhang Yang Richard Yang Jun Bi

[1]*Tsinghua University* [2]*Tongji University* [3]*Yale University* [4]*TNLIST*

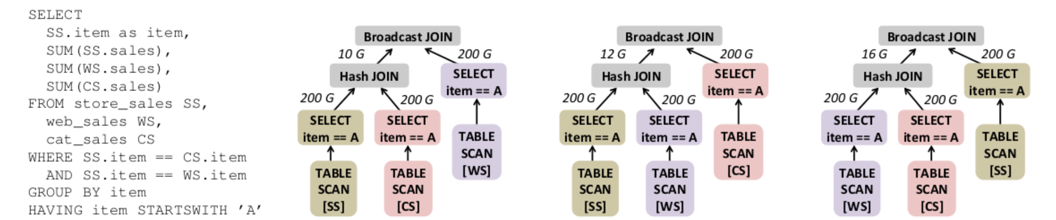
Table of Contents

Introduction

Throughput Prediction: Use Cases

Throughput prediction enables network awareness in applications.

Query Plan Selection:



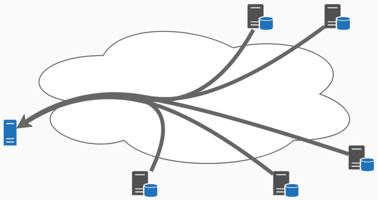
(a) A sample SQL Query (b) QEP-1 (c) QEP-2 (d) QEP-3

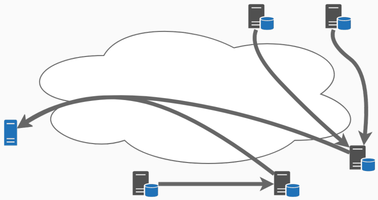
Raajay Viswanathan *et al.* , CLARINET, OSDI'2016

Throughput Prediction: Use Cases

Throughput prediction enables network awareness in applications.

Transfers in Overlay Networks:





Reactive Flows

Reactive flows (most importantly, TCP) are widely used.

Data center network: *can be more than 99%*

In this paper, we make two major contributions. First, we measure and analyze production traffic (>150TB of compressed data), collected over the course of a month from ~6000 servers (§2), ex-

The measurements reveal that 99.91% of traffic in our data center is TCP traffic. The traffic consists of query traffic (2KB to 20KB in size), delay sensitive short messages (100KB to 1MB), and throughput sensitive long flows (1MB to 100MB). The query

Alizadeh *et al.* , DCTCP, SIGCOMM’2010

Reactive Flows

Reactive flows (most importantly, TCP) are widely used.

Internet: *80% - 90%*

Table 1. Values of UDP/TCP Ratio.

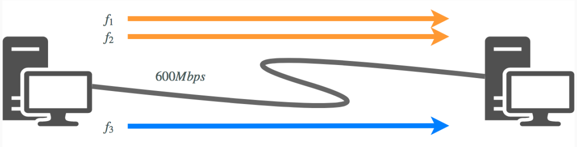
Trace	Sample	UDP/TCP Ratio			Total IP Traffic (pkts/bytes/flows)
		pkts	bytes	flows	
CAIDA-OC48	08-2002	0.11	0.03	0.11	(1371M/838GB/79M)
	01-2003	0.12	0.05	0.27	(463M/267GB/26M)
GigaSUNET	04-2006	0.06	0.02	1.06	(422M/294GB/9M)
	11-2006	0.08	0.03	1.45	
CAIDA-OC192	06-2008	0.14	0.05	1.43	(4427M/2279GB/197M)
	02-2009	0.19	0.07	2.34	(1922M/1410GB/110M)
OptoSUNET	01-2009	0.21	0.11	3.09	(1100M/657GB/41M)
	02-2009	0.20	0.11	2.63	

CAIDA, Internet traffic analysis, 2002-2009

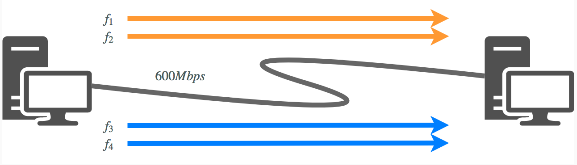
Throughput Prediction for Reactive Flows is Not Easy

Reactive flows interfere with each other.

Orange: Background flows   Blue: New flows (to be predicted)



$$x_1 = x_2 = 200\text{Mbps}$$
$$x_3 = 200\text{Mbps}$$

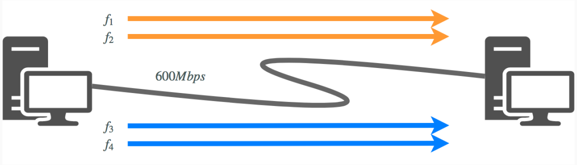


$$x_1 = x_2 = 150\text{Mbps}$$
$$x_3 = x_4 = 150\text{Mbps}$$

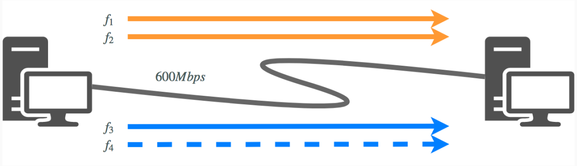
Throughput Prediction for Reactive Flows is Not Easy

There exist different congestion control algorithms.

Solid: TCP Vegas **Dashed**: TCP Reno



$$x_1 = x_2 = 150\text{Mbps}$$
$$x_3 = x_4 = 150\text{Mbps}$$

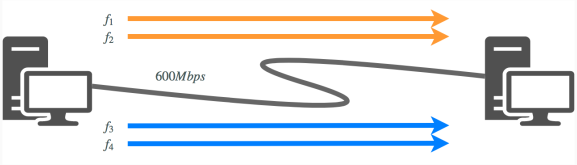


$$x_1 = x_2 = ?\text{Mbps}$$
$$x_3 = ?\text{Mbps}, x_4 = ?\text{Mbps}$$

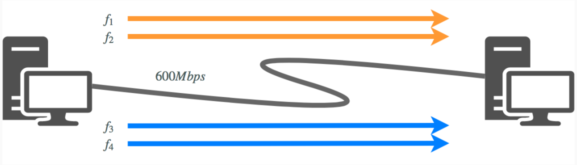
Throughput Prediction for Reactive Flows is Not Easy

Throughput can also be affected by source constraints.

Top: $x_4 \leq 180\text{Mbps}$ **Bottom**: $x_4 \leq 60\text{Mbps}$



$$x_1 = x_2 = 150\text{Mbps}$$
$$x_3 = x_4 = 150\text{Mbps}$$



$$x_1 = x_2 = 180\text{Mbps}$$
$$x_3 = 180\text{Mbps}, x_4 = 60\text{Mbps}$$

Summary

- Throughput prediction is useful
- Reactive flows (TCP) are widely used
- Throughput prediction for reactive flows is not easy
    - Reactive flows interfere with each other
    - Heterogeneous reactive mechanisms
    - The effects of source constraints

Basic Ideas

Motivation: Model-based Throughput Prediction

We want to answer this question (Q1):

> *Given a set of (TCP) flows, what is the expected throughput of each flow?*

Motivation: Model-based Throughput Prediction

We want to answer this question (Q1):

Given a set of (TCP) flows, what is the expected throughput of each flow?

Instead of answering it directly, we answer this question (Q2):

How does the network allocate bandwidth for TCP flows?

Motivation: Model-based Throughput Prediction

We want to answer this question (Q1):

Given a set of (TCP) flows, what is the expected throughput of each flow?

Instead of answering it directly, we answer this question (Q2):

How does the network allocate bandwidth for TCP flows?

There is an answer to this question! $\boxed{\textit{Network Utility Maximization}}$

$$\underset{A\boldsymbol{x} \leq \boldsymbol{c}}{\arg\max} \sum U_i(x_i)$$

(The allocation maximizes the total utility. . .)

(. . . subjet to link capacity constraints)

Breakdown

To solve NUM (Q2), we need to answer the following questions:

Q3 *How do we know the utility function of each flow?*

Srikant's utility function $U_i(x_i) = \rho_i \frac{x^{1-\alpha_i}}{\alpha_i - 1}$ (leading to **Q6** and **Q7**).

Q4 *How do we obtain the capacity constraint information?*

Global location mapping, assuming non-blocking switch.

Q5 *How do we handle source constraints?*

Explicit declaration in queries and lazy identification in samples.

Further Breakdown

To compute Srikant's utility function (**Q3**), we need

Q6 *How do we obtain α_i?*

Existing works by Oshio *et al.* .

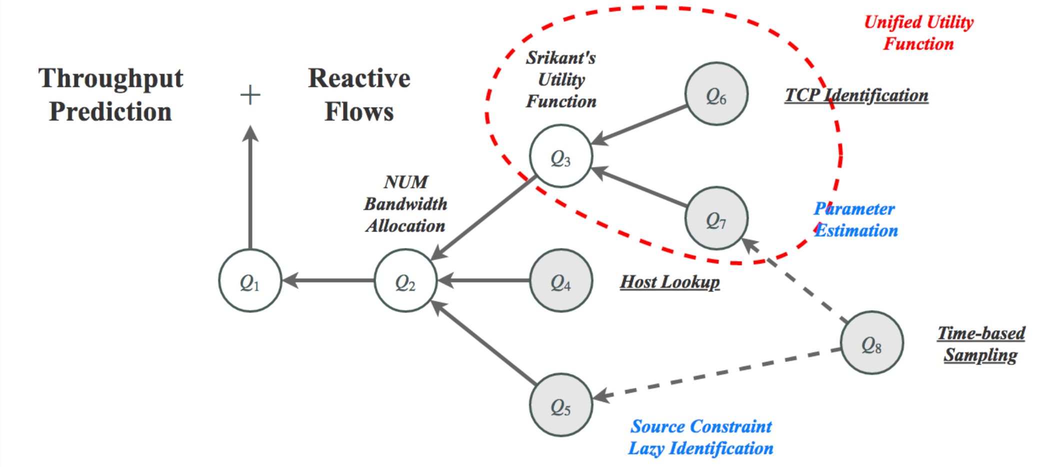
Q7 *How do we obtain ρ_i?*

Parameter estimation by analyzing the **samples**.

Q8 *How do we get the samples?*

Time-based traffic mirroring.

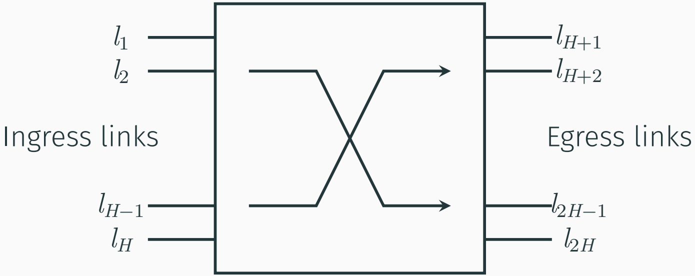
Overview


Throughput Prediction
+
Reactive Flows
Srikant's Utility Function
Unified Utility Function
TCP Identification
NUM Bandwidth Allocation
Parameter Estimation
Host Lookup
Time-based Sampling
Source Constraint Lazy Identification
Q_1
Q_2
Q_3
Q_4
Q_5
Q_6
Q_7
Q_8

Problem Formulation

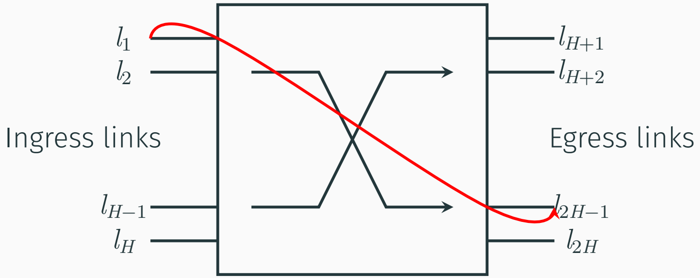
Network Model

We assume there are no bottlenecks in the core of the network (non-blocking switch assumption of DCN). The network is a bipartite graph. For a network with H hosts, there are $L = 2H$ links numbered as $l_1, \ldots, l_L$ and $\mathcal{L} = \{l_1, \ldots, l_L\}$.



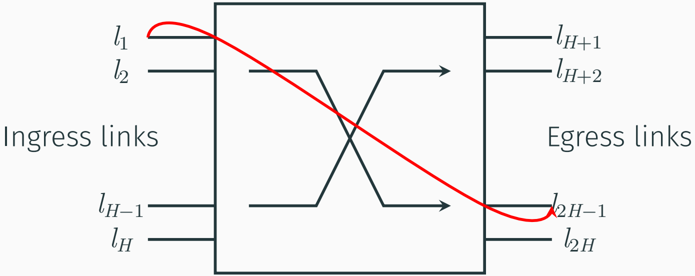
Flow Model

Assume there are N background flows numbered as $f_1, \ldots, f_N$ and M queried flows numbered as $q_1, \ldots, q_M$. Let $\mathcal{F} = \{f_1, \ldots, f_N\}$ and $\mathcal{Q} = \{q_1, \ldots, q_M\}$. Let $A = \{a_{ij}\}_{L \times N}$ and $B = \{b_{ij}\}_{L \times M}$ be the **routing matrix**.



Flow Model (Cont.)

Let x_i be the throughput of f_i and y_i be the throughput of q_i. Let τ_i be the source constraint of f_i and π_i be the source constraint of q_i. Let $\boldsymbol{x} = \{x_1, \ldots, x_N\}^{\mathrm{T}}$, $\boldsymbol{y} = \{y_1, \ldots, y_M\}^{\mathrm{T}}$, $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_N\}^{\mathrm{T}}$ and $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_M\}^{\mathrm{T}}$



Throughput Prediction Problem

The throughput prediction of $\mathcal{Q}$ is the solution to this problem:

$$\arg\max_{\boldsymbol{y}} \left(\sum_{i=1}^{N} U_{f_i}(x_i) + \sum_{i=1}^{M} U_{q_i}(y_i) \right)$$

Subject to:

$$\begin{pmatrix} A\,B \\ I\,O \\ O\,I \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{\tau} \\ \boldsymbol{\pi} \end{pmatrix}$$

Parameter Estimation

We consider the simpler form (let $\mathcal{Q} = \emptyset$ and ignore source constraints $\boldsymbol{\tau}$), replace $U_{f_i}(x)$ with Srikant's utility function:

$$\arg\max_{\boldsymbol{x}} \sum_{i=1}^{N} \left(\rho_i \frac{x_i^{1-\alpha_i}}{1-\alpha_i} \right)$$

Subject to:

$$A\boldsymbol{x} \leq \boldsymbol{c}$$

The solution $\hat{\boldsymbol{x}}$ can be considered as a function $\hat{\boldsymbol{x}}(\mathcal{F}, \boldsymbol{\alpha}; \boldsymbol{\rho})$ because the utility functions are concave and the domain is convex, or simply $\hat{\boldsymbol{x}}(\boldsymbol{\rho})$ for given $\mathcal{F}$ and $\boldsymbol{\alpha}$.

Parameter Estimation (Cont.)

For a given set of flows $\mathcal{F}$, assume the α parameter in Srikant's utility function of each flow is known (denoted as $\boldsymbol{\alpha}$). Assume we have K samples $\tilde{\boldsymbol{x}}^{(\mathbf{1})}, \ldots, \tilde{\boldsymbol{x}}^{(\boldsymbol{K})}$, the estimated ρ parameter of all flows (denoted as $\hat{\boldsymbol{\rho}}$) is the solution of the following problem:

$$\hat{\boldsymbol{\rho}} = \arg\min_{\boldsymbol{\rho}} \frac{1}{2K} \sum_{k=1}^{K} \|\hat{\boldsymbol{x}}(\boldsymbol{\rho}) - \tilde{\boldsymbol{x}}^{(k)}\|^2 \qquad \text{(Minimize the error)}$$

Practical Issues

- **I1** To estimate throughput for all flows, we need a ρ_i for each possible (src, dst) pair. The number of **potential parameters** is large (H^2 for a network with H hosts)
- **I2** Solving the parameter estimation problem by **compute gradient by definition** can be slow.
- **I3** The estimation problem has not considered **source constraints** yet.
- **I4** It is impractical to **monitor all the flows**. How to sample the traffic and extract useful information.

Optimizations

Practical Issues and How to Solve Them

I1 To estimate throughput for all flows, we need a ρ_i for each possible (src, dst) pair. The number of potential parameters is large (H^2 for a network with H hosts)

Reduce #variables by dividing (src, dst) pairs into equivalent classes.

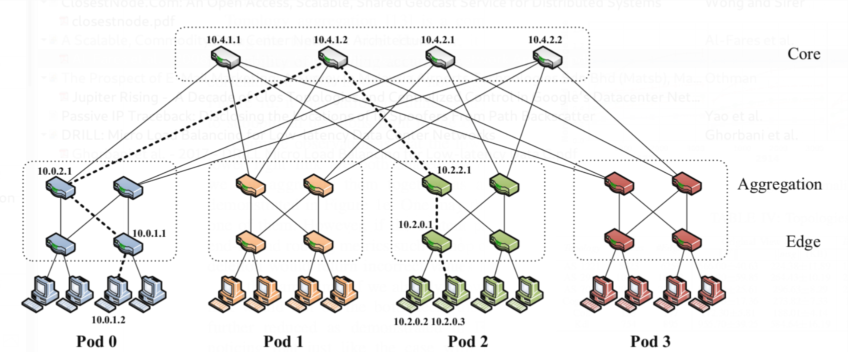
I2 Solving the parameter estimation problem by compute gradient by definition can be slow.

I3 The estimation problem has not considered source constraints (which might be unknown) yet.

I4 It is impractical to monitor all the flows. How to sample the traffic and extract useful information.

Reduce the Number of Variables

- The utility function depends only on the end-to-end cost.
- In DCN, the topology is highly structured and many (src, dst) pairs have similar end-to-end costs.



Al-Fares, SIGCOMM'2008

Reduce the Number of Variables

Basic idea: Divide (src, dst) pairs to different equivalent classes

In a 3-layered fat tree topology, each host has 3 equivalent classes:

- In the same rack
- In the same Pod
- In different Pods

$3H$ parameters instead of H^2

Problem Transformation

Let ϱ_i^j represent the ρ parameter for the i-th equivalent class of j-th host, let $\bar{\boldsymbol{\rho}} = \{\varrho_1^1, \varrho_2^1, \ldots, \varrho_3^H\}$. Let p_i **be the equivalent class index of f_i**, we have $\rho_i = \bar{\rho}_{p_i}$ and its matrix form

$$\boldsymbol{\rho} = \Lambda \bar{\boldsymbol{\rho}} \quad \text{where} \quad \Lambda = \begin{bmatrix} \underbrace{0 \cdots 0}_{p_1 - 1} \; 1 \; 0 & \cdots\cdots\cdots\cdots & \\ \vdots & \ddots \;\; \ddots \;\; \ddots & \\ \underbrace{0 \cdots\cdots\cdots 0}_{p_i - 1} \;\; 1 \;\; 0 & \ddots & \\ \vdots & \ddots \;\; \ddots \;\; \ddots & \\ \underbrace{0 \cdots\cdots 0}_{p_N - 1} \; 1 \; 0 \cdots\cdots & \ddots & \end{bmatrix}_{N \times 3H}$$

Parameter Estimation for Equivalent Classes

For a given set of flows $\mathcal{F}$, assume the α parameter in Srikant's utility function of each flow is known (denoted as $\boldsymbol{\alpha}$). Assume we have K samples $\tilde{\boldsymbol{x}}^{(\boldsymbol{1})}, \ldots, \tilde{\boldsymbol{x}}^{(\boldsymbol{K})}$, the estimated $\bar{\rho}$ parameter of all equivalent classes (denoted as $\hat{\bar{\boldsymbol{\rho}}}$) is the solution of the following problem:

$$\hat{\bar{\boldsymbol{\rho}}} = \arg\min_{\bar{\boldsymbol{\rho}}} \frac{1}{2K} \sum_{k=1}^{K} \|\hat{\boldsymbol{x}}(\Lambda\bar{\boldsymbol{\rho}}) - \tilde{\boldsymbol{x}}^{(k)}\|^2 \qquad \text{(Minimize the error)}$$

Practical Issues and How to Solve Them

- **I1** To estimate throughput for all flows, we need a ρ_i for each possible (src, dst) pair. The number of potential parameters is large (H^2 for a network with H hosts)
- **I2** Solving the parameter estimation problem by compute gradient by definition can be slow.
  Derive the gradient from KKT conditions.
- **I3** The estimation problem has not considered source constraints (which might be unknown) yet.
- **I4** It is impractical to monitor all the flows. How to sample the traffic and extract useful information.

Deriving the Gradient

We first consider the simplified NUM problem and according to Karush-Kuhn-Tucker condition (justified because the constraints are linear, i.e., the problem is LCQ):

$$\nabla_{\boldsymbol{x}} f + \hat{\boldsymbol{\lambda}}^{\mathrm{T}} \nabla_{\boldsymbol{x}} \boldsymbol{g} = 0 \quad \Rightarrow \quad \forall j, \quad \bar{\rho}_{p_j} \hat{x}_j^{-\alpha_j} = \sum_k a_{kj} \hat{\lambda}_k, \qquad \text{(Stationarity)}$$

$$\hat{\boldsymbol{\lambda}}^{\mathrm{T}} \boldsymbol{g}(\hat{\boldsymbol{x}}) = 0 \quad \Rightarrow \quad \forall k, \quad \hat{\lambda}_k \Big(\sum_j a_{kj} \hat{x}_j - c_k\Big) = 0.$$

(Complementary Slackness)

Deriving the Gradient (cont.)

We now consider the partial derivatives of (Stationarity) and (Complementary Slackness) and after some reorganization, we have

$$\begin{pmatrix} \operatorname{diag}(\bar{\rho}_{p_j}\alpha_j x_j^{-1-\alpha_j}) & A^{\mathrm{T}} \\ \operatorname{diag}(\lambda_k)A & \operatorname{diag}(\sum_j a_{kj}x_j - c_k) \end{pmatrix} \begin{pmatrix} \mathbf{J}_{\boldsymbol{x}}(\bar{\boldsymbol{\rho}}) \\ \mathbf{J}_{\boldsymbol{\lambda}}(\bar{\boldsymbol{\rho}}) \end{pmatrix} = \begin{pmatrix} \Gamma \\ O \end{pmatrix}$$

where

$$\Gamma_{j,i} = \begin{cases} x_j^{-\alpha_j} & \text{if } p_j = i, \\ 0 & \text{if } p_j \neq i. \end{cases}$$

To be able to compute the gradient $\mathbf{J}_{\boldsymbol{x}}(\bar{\boldsymbol{\rho}})$, the leftmost matrix must be invertible.

Deriving the Gradient (cont.)

We let

$$\begin{pmatrix} M_1 \, M_2 \\ M_3 \, M_4 \end{pmatrix} = \begin{pmatrix} \operatorname{diag}(\bar{\rho}_{p_j} \alpha_j x_j^{-1-\alpha_j}) & A^{\mathrm{T}} \\ \operatorname{diag}(\lambda_k) A & \operatorname{diag}(\sum_j a_{kj} x_j - c_k) \end{pmatrix}.$$

M_1 is invertible, so we need to prove $\det(M_4 - M_3 M_1^{-1} M_2) \neq 0$.

$$(M_4 - M_3 M_1^{-1} M_2)_{ik} = \begin{cases} \sum_j a_{kj} \hat{x}_j - c_k - \hat{\lambda}_k \Phi_{kk}(\bar{\boldsymbol{\rho}}) & \text{if } i = k \\ -\hat{\lambda}_i \Phi_{ik}(\bar{\boldsymbol{\rho}}) & \text{otherwise} \end{cases}$$

where

$$\Phi_{ik}(\bar{\boldsymbol{\rho}}) = \Phi_{ki}(\bar{\boldsymbol{\rho}}) = \sum_j a_{ij} a_{kj} \bar{\rho}_{p_j}^{-1} \alpha_j^{-1} \hat{x}_j^{1+\alpha_j}.$$

Deriving the Gradient (cont.)

We can reorganize the matrix (by changing the order of constraints) by whether $\hat{\lambda}_k = 0$:

$$\det \begin{pmatrix} \mathrm{diag}(\sum_j a_{kj}\hat{x}_j - c_k) & O \\ M_1' & \mathrm{diag}(\hat{\lambda}_k) M_2' \end{pmatrix} = \prod_{\hat{\lambda}_k = 0} \left(\sum_j a_{kj}\hat{x}_j - c_k \right) \prod_{\hat{\lambda}_k \neq 0} \hat{\lambda}_k \det(M_2')$$

- If $\hat{\lambda}_k = 0$, usually $\sum_j a_{kj}\hat{x}_j - c_k \neq 0$. Otherwise, we can add a disruption to c_k to make sure $\sum_j a_{kj}\hat{x}_j - c_k \neq 0$ without affecting the solutions.
- $\det(M_2')$ is a polynomial of $\bar{\boldsymbol{\rho}}$, it will not always be zero.

With a small disruption, we ensure the matrix is full-rank. So we use numerical methods to get $\mathbf{J}_{\hat{\boldsymbol{x}}}(\bar{\boldsymbol{\rho}})$.

Deriving the Gradient (cont.)

With $\mathbf{J}_{\hat{x}}(\bar{\boldsymbol{\rho}})$, we can derive the gradient of the error function as:

$$\nabla E = \frac{1}{K}\sum_{k=1}^{K}(\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^{(\boldsymbol{k})})^{\mathrm{T}}\mathbf{J}_{\hat{x}}(\bar{\boldsymbol{\rho}})$$

In practice, we use spherical coordinates to avoid choosing step sizes. The final gradient is

$$\nabla_{\phi} E = \frac{1}{K}\sum_{k=1}^{K}(\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^{(\boldsymbol{k})})^{\mathrm{T}}\mathbf{J}_{\hat{x}}(\bar{\boldsymbol{\rho}})\mathbf{J}_{\bar{\boldsymbol{\rho}}}(\boldsymbol{\phi})^{\mathrm{T}}$$

Practical Issues and How to Solve Them

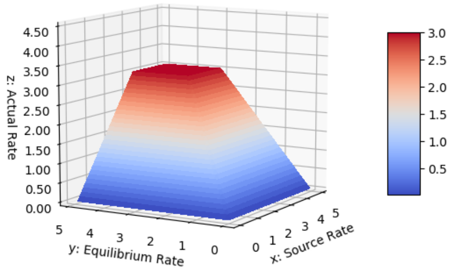
- **I1** To estimate throughput for all flows, we need a ρ_i for each possible (src, dst) pair. The number of potential parameters is large (H^2 for a network with H hosts)
- **I2** Solving the parameter estimation problem by compute gradient by definition can be slow.
- **I3** The estimation problem has not considered source constraints (which might be unknown) yet.
  Consider effective source constraints through lazy identification.
- **I4** It is impractical to monitor all the flows. How to sample the traffic and extract useful information.

Handling Unknown Source Constraints

Basic idea: Consider only the *effective* source constraints.

A sample might be constrained by an **effective** source constraint if:

- Sampled throughput is significantly smaller than the estimated throughput (of the same equivalent class).
- Sampled throughput is at a relatively fixed rate.



Practical Issues and How to Solve Them

I1 To estimate throughput for all flows, we need a ρ_i for each possible (src, dst) pair. The number of potential parameters is large (H^2 for a network with H hosts)

I2 Solving the parameter estimation problem by compute gradient by definition can be slow.

I3 The estimation problem has not considered source constraints (which might be unknown) yet.

I4 It is impractical to monitor all the flows. How to sample the traffic and extract useful information.
Consider low-throughput flows as non-reactive flows.

Sampling

Basic idea: Identify individual *throughput-sensitive* flows and accumulate the *low-throughput* flows.

Short-lived and low-throughput flows **may not reach the equilibrium** and their throughput **cannot be measured very accurately**. But **many short-lived, low-throughput flows cannot be ignored.**

Solution

Extract the individual flows with **more than 5% of the total link capacity as samples** and compute the **accumulated traffic demand** from the rest. Let v_k be the traffic demand on the k-th link, the link capacity used in the prediction is computed as $c_k - v_k$.

Evaluation

Settings

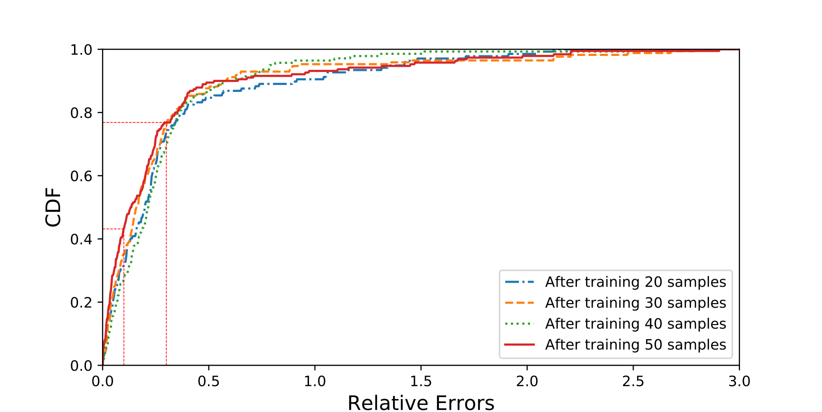
Topology: A Clos topology with $K = 4$, 16 hosts (32 links)

Flows: 100 initial flows, 60 samples each with 10-20 flows.

- We use NS2 to run the traffic and get the throughput for each sample. The parameter estimation and prediction are analyzed offline.
- Each sample is first used as a query to measure the prediction error and prediction time.
- We update the $\boldsymbol{\rho}$ every sample and measure the estimation time.

Prediction Accuracy (Vegas)

Metric: Relative errors: $|\frac{\tilde{x}-\hat{x}}{\tilde{x}}|$

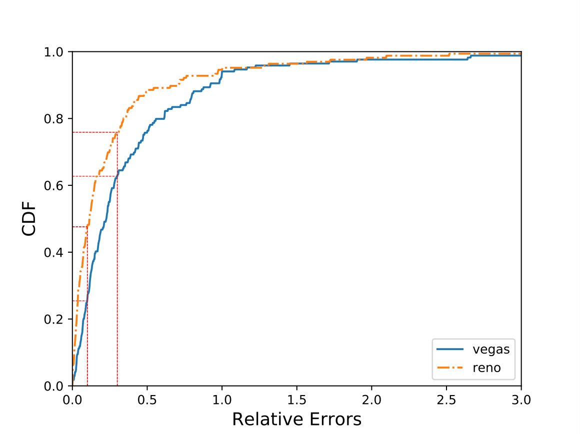


>75% flows have a smaller relative error than 30%.

Prediction Accuracy (Multiple TCP)

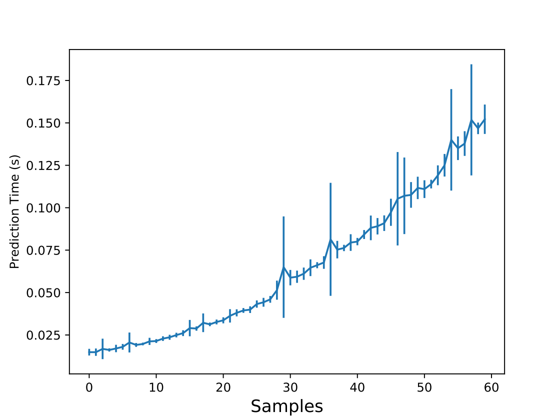
Metric: Relative errors: $|\frac{\tilde{x}-\hat{x}}{\tilde{x}}|$

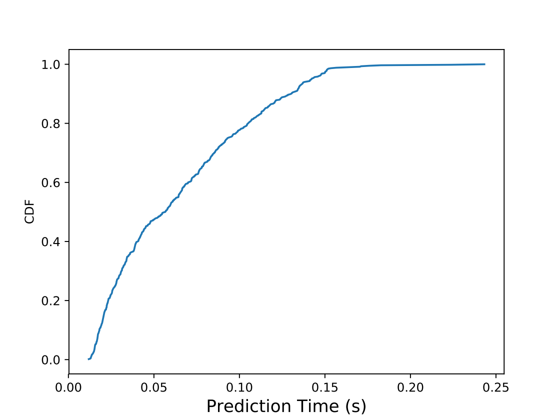
- Predictions become less accurate (60%/75% with relative error smaller than 30%)
- Prediction for TCP Reno is more accurate than Vegas



Prediction Time

Metric: The time used to conduct a prediction

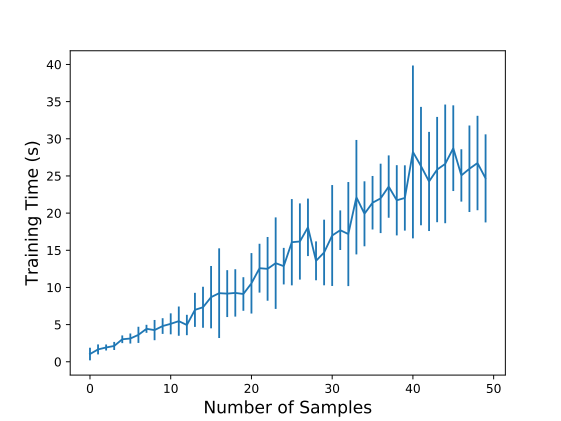




Less than 0.2s for up to 1056 flows (19 fg and 1037 bg).

Parameter Estimation Convergence

Metric: The time used to get an optimal parameter estimation.



Discussions

Prophet

A Model-Driven Throughput Prediction System

- Predict throughput by solving NUM problem
- Handle heterogeneity with unified utility functions (Srikant's utility function)
- Obtain traffic samples with advanced monitoring techniques (Per-flow monitoring)
- Estimate unknown parameters ρ using the gradient descent method

Limitations

Overall Framework

- Limited to special network topologies and special TCP variants.

Optimization

- Impact of multiple paths between two hosts is not analyzed.
- Convergence of the parameter estimation is not proved.
- Mistakes in the published paper (corrected in this presentation).

Evaluation

- Not enough real traffic analysis

Limitations

Overall Framework

- Limited to special network topologies and special TCP variants.

Optimization

- Impact of multiple paths between two hosts is not analyzed.
- Convergence of the parameter estimation is not proved.
- Mistakes in the published paper (corrected in this presentation).

Evaluation

- Not enough real traffic analysis

Limitations

Overall Framework

- Limited to special network topologies and special TCP variants.

Optimization

- Impact of multiple paths between two hosts is not analyzed.
- Convergence of the parameter estimation is not proved.
- Mistakes in the published paper (corrected in this presentation).

Evaluation

- Not enough real traffic analysis

Revisit the Assumptions

- DC Network
  What about an arbitrary network?
- Vegas and Reno
  What about other TCP congestion control algorithms?
- Random Sampling
  Are there better sampling methods?

Revisit the Assumptions

- DC Network
  What about an arbitrary network?
- Vegas and Reno
  What about other TCP congestion control algorithms?
- Random Sampling
  Are there better sampling methods?

Possible directions:

- In SDN, get routing matrix and propagation delay from the controller.
- Use function approximation to model the utility function for arbitrary TCP variants.
- Sketch-based sampling

Parameter Estimation: How to Get Unique ρ

A minimal example:

$$A = \begin{pmatrix} 100 \\ 110 \\ 101 \end{pmatrix}, \boldsymbol{c} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \boldsymbol{\alpha} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \tilde{\boldsymbol{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

There are actually multiple $\boldsymbol{\rho}$s which minimizes the estimation error as long as:

$$\rho_1 = \frac{1}{3}\left(2 + x_1 - x_2 - x_3\right)$$
$$\rho_2 + \rho_3 = \frac{1}{3}\left(1 - x_1 + x_2 + x_3\right)$$

Prophet

A Model-Driven Throughput Prediction System

- Predict throughput by solving NUM problem
- Handle heterogeneity with unified utility functions (Srikant's utility function)
- Obtain traffic samples with advanced monitoring techniques (Per-flow monitoring)
- Estimate unknown parameters ρ using the gradient descent method

Q & A

We'd like to thank the reviewers for their insightful feedback!

Questions and comments are highly appreciated!

Kai Gao gaok12@mails.tsinghua.edu.cn

Jensen Zhang jingxuan.n.zhang@gmail.com

Prophet

A Model-Driven Throughput Prediction System

- Predict throughput by solving NUM problem
- Handle heterogeneity with unified utility functions (Srikant's utility function)
- Obtain traffic samples with advanced monitoring techniques (Per-flow monitoring)
- Estimate unknown parameters ρ using the gradient descent method