On the Accurate Identification of Network Paths having a Common Bottleneck

Muhammad Murtaza Yousaf(student)
University of Innsbruck, Austria
murtaza.yousaf@uibk.ac.at

Michael Welzl(faculty)
University of Innsbruck, Austria
michael.welzl@uibk.ac.at

Bulent Yener(faculty)
Rensselaer Polytechnic Institute, NY, USA
yener@cs.rpi.edu

ABSTRACT

We present a new mechanism for detecting shared bottlenecks between end-to-end paths in a network. It only needs one-way delays from endpoints as an input and is based on the well known linear algebraic approach SVD (Singular Value Decomposition). Clusters of flows which share a bottleneck are extracted from SVD results by outlier detection. Simulations with varying topologies and different network conditions show the high accuracy of our technique.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*network monitoring*

General Terms

Measurement

1. INTRODUCTION

End-to-end flows in packet networks adversely influence each other when they traverse a single congested link ("shared bottleneck"). Given the large number of distributed applications on the Internet, where coordination between flows is quite feasible, we believe that the detection of shared bottlenecks is important. There are numerous possibilities for using such information — e.g., in network-aware Grid scheduling, efficient file transfer in overlays, replica management in P2P systems, and coordinated congestion management.

Despite the existence of some studies on shared bottleneck detection ([1], [2] and [3]), such endeavors are hardly undertaken, and the accuracy exhibited is limited. Our method, which only operates on end-to-end measurements (forward delays) and is easy to calculate, provides extraordinarily accurate results. It is based on SVD that has most of its applications in image processing and pattern recognition, but the nature of our problem makes it an ideal tool for us.

2. APPROACH

2.1 Singular Value Decomposition (SVD)

We briefly review the singular value decomposition [4] of matrices. Any $m \times n$ matrix A can be expressed as

$$A = \sum_{t=1}^{r} \sigma_t(A) u^{(t)} {v^{(t)}}^T \tag{1}$$

where $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_r(A) > 0$ are its singular values, $u^{(t)} \in \mathcal{R}^m, v^{(t)} \in \mathcal{R}^n, t = 1, \ldots, r$ are its left and right singular vectors respectively, and r is the rank of A. The $u^{(t)}$'s and the $v^{(t)}$'s are orthonormal sets of vectors; namely, ${u^{(i)}}^T u^{(j)}$ is one if $i = j$ and zero otherwise.

In matrix notation, SVD is defined as $A = U\Sigma V^T$ where U and V are orthogonal matrices of dimensions $m \times r$ and $n \times r$ respectively, containing the left and right singular vectors of A. $\Sigma = \mathbf{diag}(\sigma_1(A), \ldots, \sigma_r(A))$ is an $r \times r$ diagonal matrix containing the singular values of A.

2.2 Shared Bottleneck Detection

Most of the techniques proposed till now to detect shared bottlenecks are based on packet delays or packet loss. It has been shown that delay based techniques quickly converge towards accurate results [1]; hence we decided to rely on the same metric. We use one-way delays along a path to detect shared bottlenecks. Our mechanism consists of three stages which are discussed in the following sections.

2.2.1 Delay Measurements

First we measure the one-way delays for all the paths. We group samples on the basis of a flexible time windows and pick the average of a time window as a representative measurement. From these data we form a matrix D of forward delays of all flows where each entry d_{ij} is an average delay measurement of the i^{th} path for the j^{th} time window:

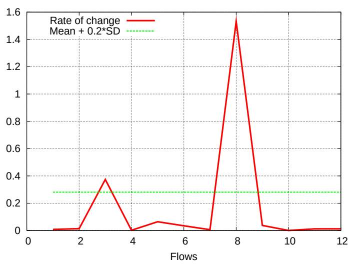
$$\mathbf{D} := (d_{i,j})_{i=1,\ldots,m;\ j=1,\ldots,n} \tag{2}$$

2.2.2 Getting Singular Values and Projection

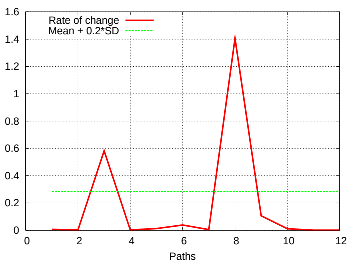
We apply SVD on the matrix $D_{m\times n}$ and get $U_{m\times r}$, $\Sigma_{r\times r}$, and $V_{n\times r}$ as discussed in section 2.1.

Here, the matrix $U_{m\times r}$ represents the association among the paths on the basis of similar delays caused by common bottlenecks and $\Sigma_{r\times r}$ represents the degree of this association. Usually, if the paths share a bottleneck then the diagonal matrix always shows a *good rank reduction*, i.e. we can approximate D by keeping only the topmost "principal component" and we need to project only on the first column of $\Sigma_{r\times r}$. We say that a matrix D has a "good" rank k approximation if the 2-norm and the Frobenius norm of $D - D_k$ is small; for a detailed treatment of SVD see [4]. In order to project $U_{m\times r}$ on the first column of $\Sigma_{r\times r}$, we multiply each row of $U_{m\times r}$ with only the first column of the $\Sigma_{r\times r}$. This results in a matrix $X_{m\times 1}$, as shown in equation 3, that has one entry corresponding to each path. These values are further used to group flows that share a bottleneck.

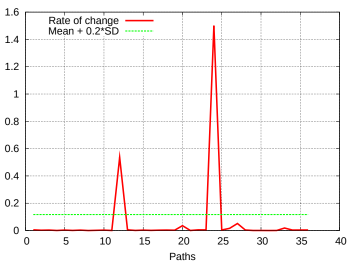
$$X_{m\times 1} = U_{m\times r} \times \Sigma_{r\times r}(:, 1) \tag{3}$$



(a) 12 Flows with with corresponding sources and destinations



(b) 12 Flows with random sources and random destinations



(c) 36 Flows with each sender sending 3 flows to random destinations in each branch

Figure 1: Results of experiments without common end points

2.2.3 Grouping Flows

The matrix $X_{m\times 1}$ contains one value for each path. The paths that share a bottleneck obtain similar values, which differ significantly from those paths that share a different bottleneck. If we sort all the entries of matrix then we can observe a grouping on the basis of stable slopes and sudden drops in the slope indicating the start of a new cluster. We use the *rate of change* (*roc*) as a measure to detect groups of paths that share a bottleneck:

$$roc = \frac{x_i - x_{i+1}}{x_{i+1}} \tag{4}$$

The detection of sudden drops in the slope becomes a problem of detecting outliers. We use the threshold τ defined as

$$\tau = \bar{z} + 0.2 \times s \tag{5}$$

where $\bar{z}$ is the mean *roc* and s is its standard deviation. The coefficient of 0.2 was selected after a detailed study of data obtained in simulations which showed that our method works very well with this value in a wide range of realistic network conditions and scenarios.

3. VALIDATION

Using the ns-2 network simulator, we tested our mechanism with various network conditions; here, we discuss only one scenario with multiple sources and destinations (it is a major assumption in related work to have a common end point). The topology is shown in figure 2. Its center represents an overprovisioned wide-area network backbone; three bottlenecks are placed at the access points of local networks, which resembles the most common real life case of Internet connectivity. Two of them are limited by their small physical capacity (1.5 Mbps and 3.0 Mbps — all other links have 10Mbps), whereas the third bottleneck is due to heavy cross traffic. All links were assigned random delays from 1ms to 20ms, the other parameters are given in table 1.

In our first experiment we generated 12 TCP flows, one from each source on the left to the corresponding destination on the right to compute delay measurements from all 12 paths. This is a basic step for showing that our mechanism does not need any synchronization between packet sending times and does not need packets of different flows to be adjacent, which was required in related work. We randomly start all the flows during the first second of the simulation. Our mechanism accurately detected three clusters of paths as shown in figure 1(a) which indicates that the first three flows, next five flows, and last four flows share a bottleneck.

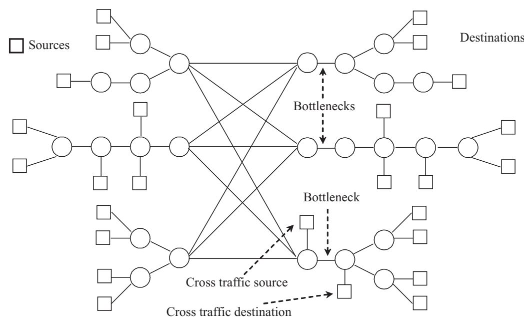


Figure 2: Topology without common end points

Table 1: Simulation parameters

Background traffic	256 Kbps Pareto flows (all sources to all destinations)
Reverse traffic	64 Kbps Pareto flows (all destinations to all sources)
Cross traffic	256 Kbps CBR flows (24)
Queue size	250 packets
Drop policy	DropTail

As a next step, we generated 12 TCP flows from random sources to random destinations. Figure 1(b) shows that we obtained accurate results. Finally, we generated three TCP flows from all sender to a random destination in each branch of the bottleneck to investigate 36 paths. This way, we generated foreground traffic from every sender to every bottleneck. Even with such a heavy load our mechanism was stable and reliable enough to detect clusters of paths having common bottlenecks, which is shown in figure 1(c).

4. REFERENCES

[1] Rubenstein, D., Kurose, J., Towsley, D.: Detecting shared congestion of flows via end-to-end measurement. IEEE/ACM ToN, 10(3):381-395 (June 2002).

[2] Kim, M.S., Kim, T., Shin, Y., Lam, S.S., Powers, E.J.: A wavelet-based approach to detect shared congestion. In ACM SIGCOMM 2004 (Aug. 2004).

[3] Younis, O., Fahmy, S.: Flowmate: Scalable on-line flow clustering. IEEE/ACM ToN, 13(2):288-301 (Apr. 2005).

[4] Golub, G., Loan, F.V.: Matrix Computations. Johns Hopkins University Press (1996).