Hindawi Publishing Corporation
Research Letters in Communications
Volume 2009, Article ID 939840, 5 pages
doi:10.1155/2009/939840

Research Letter

The Measurement Paradox in Valiant Network Design

Matthew Roughan

School of Mathematical Sciences, University of Adelaide, Adelaide 5005, Australia

Correspondence should be addressed to Matthew Roughan, matthew.roughan@adelaide.edu.au

Received 27 April 2009; Accepted 1 July 2009

Recommended by Adnan Kavak

Valiant network design was proposed, at least in part, to counter the difficulties in measuring network traffic matrices. However, in this paper we show that in a Valiant network design, the traffic matrix is in fact easy to measure, leading to a subtle paradox in the design strategy.

1. Introduction

In recent years the difficulties in measurement and prediction of Internet traffic matrices have prompted a number of routing and network design strategies broadly termed "oblivious" [1–3]. They are oblivious in the sense that they guarantee performance under *any possible* traffic matrix. This appealing property has a cost: extra capacity is needed to ensure that performance is maintained under all possible inputs, though several papers have shown reasonable bounds to this additional cost.

In this paper we examine Valiant network design (sometimes called load balancing) a strategy extended from switch design to the design of a whole network [2, 3]. The basic principle is to build a completely connected network—a clique—and use load balancing to share all traffic across all two hop paths. The remarkable property of this network is that with only twice the capacity of an optimal network, it can carry any allowable traffic matrix, without congestion!

The irony of Valiant network design is that it is predicated on the assumption that traffic matrices are hard to measure, and yet in this paper we show that such a design creates a network in which it is actually possible to measure the traffic matrix. However, this fact is of little use, because if we redesign the network based on this improved information, we then lose the ability to make ongoing measurements, leading to a paradoxical situation.

It is a classic case where "you cannot have your cake, and eat it too!" Where we have the capability to make good measurements (courtesy of Valiant design) we cannot make use of them, and where we do not have such a design, the measurements are much harder to obtain. As a result, we suggest an alternative, which takes advantage of the properties of Valiant network design in addition to the ability to measure traffic matrices.

We should note that there are other reasons for using Valiant network design, for instance resilience to network failures, or errors in traffic predictions, and these may outweigh the issue of difficulties in traffic matrix estimation. However, the problem of measuring traffic matrices has been found interesting in a number of contexts, and so here we examine the measurement aspect of a Valiant network.

2. Background

A Traffic Matrix (TM) describes the amount of traffic (the number of packets or more commonly bytes) transmitted from one point in a network to another during some time interval, and they are thus naturally represented by $T_t(i, j)$ which represents the traffic volume (in bytes or packets) from i to j during a time interval $[t, t + \Delta t)$. The locations i and j may be physical geographic locations making i and j spatial variables, or logical variables such as a group of IP addresses, but in this paper we will associate locations with PoPs (Points of Presence). Often, for convenience, TMs are written as column vectors by stacking the columns of the matrix. This allows us to write a series of such matrices into a new matrix X, whose columns each represents a single snapshot of a TM. In this paper we need only single snapshots, and so our notation will refer to TMs as column vectors $\mathbf{x}$.

TMs are the basic input into many network engineering problems. Of particular relevance here is the network design problem (the problem of determining where links will appear in the network, and what capacity they should have, along with the subsidiary problem of determining the routing of traffic in this network). However, TMs are not easy to measure directly due to problems with data collection, and the scale of data required [4].

On the other hand SNMP (the Simple Network Management Protocol) data is easy to collect and almost ubiquitous. However, SNMP data only provides link load measurements, not TM measurements [5]. The link measurements $\mathbf{y}$ are related to the TM, which is written as a column vector $\mathbf{x}$, by the linear relationship

$$\mathbf{y} = A\mathbf{x}, \qquad (1)$$

where A is called the routing matrix [6]. If A is invertible the solution to this system of equations is obvious, but in general, A is not even square. A network with N nodes has $N(N-1)$ traffic demands, so the length of $\mathbf{x}$ is $O(N^2)$, but in a typical network design the number of links and hence the length of $\mathbf{y}$ are $O(N)$. As N becomes large, the system of equations above becomes underconstrained. In most real networks, the problem is highly underconstrained. The resulting problem of inferring the TM from link measurements is a classic underconstrained, linear-inverse problem. There are a number of good techniques for solving such problems (see, for instance, [5, 7]), but the ill-posed nature of the problem means that there are likely to be some errors in the estimates.

In response to these difficulties, an alternative set of ideas have developed: oblivious routing [1] and Valiant network design [2, 3], which seek to design a network and its routing such that it will work well for any arbitrary traffic matrix. That is they try to design the network in the absence of standard input information. The cost is a loss of efficiency. The network must be overengineered by at least a factor of two in most cases.

In this paper we consider Valiant Network Design (VND), sometimes also called Valiant load balancing after its central idea. We will consider the simplest example of such design, for clarity (though the concepts presented here extend to the more complicated case). We have N PoPs which must be connected, but we do not know the TM. The only information we do possess is the total access capacity at each PoP. For simplicity, assume this capacity is C for all PoPs. The access capacity determines the maximum amount of traffic that can come in or depart from a PoP. Hence it limits the traffic matrix, because the row and column sums of this matrix cannot exceed C, so in the absence of additional information, our job is to design the network which minimizes our cost subject to the constraints

$$\sum_{i=1}^{N} T(i,j) \leq C, \qquad \sum_{j=1}^{N} T(i,j) \leq C. \qquad (2)$$

The basic principle of VND is that the network should be a clique (a completely connected network) and that traffic should be shared in even proportions across all two hop paths. Figure 1 illustrates the network design for a 6-node network and shows one of the N paths from p to q through node i.

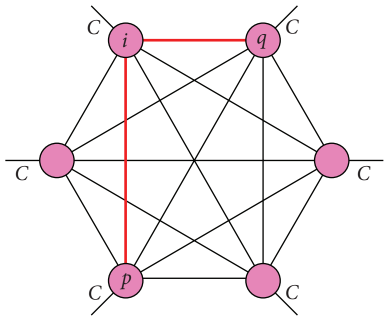


Figure 1: Valiant network design: each PoP has access capacity C, and traffic between p and q is shared over all two hop paths through each possible intermediate node i.

The key result of VND is that almost all traffic goes on two hop paths so in order to carry a maximal traffic matrix, the network requires approximately $2NC$ capacity, which when shared amongst the links results in a required link capacity of $2C/N$. (Note that traffic is evenly split across all N possible intermediate nodes, including the end points, i.e., we include paths $p-p-q$ and $p-q-q$ in the set of load-balanced paths.) Capacity estimates exist for the more complicated case with unequal access capacities, as well as extensions of VND to networks requiring resilience to failures [2, 3], but these are not germain to the question under consideration here, that is, how much information can we obtain about the TM of a VND?

2.1. Valiant Network Design Routing Matrix. The important thing to notice in the above is that VND needs a completely connected network. This may be implemented as a VPN on top of some other physical network, but even in this case, we can obtain link traffic measurements with ease using SNMP. Note that in a completely connected network there are $N(N-1)$ links and $N(N-1)$ elements in the TM, so the routing matrix is square. We may hope that in this case the routing matrix is invertible, and if this were the case, then we could solve the TM measurement problem by the simple expedient of taking

$$\mathbf{x} = A^{-1}\mathbf{y}. \qquad (3)$$

So we need to consider the routing matrix that results from VND. Formally, $A = \{A_{ir}\}$ is the matrix defined by

$$A_{ir} = \begin{cases} F_{ir}, & \text{if traffic for } r \text{ traverses link } i, \\ 0, & \text{otherwise,} \end{cases} \qquad (4)$$

where F_{ir} is the fraction of traffic from source/destination pair $r = (p,q)$ that traverses link i. A network with N nodes and L links will have a $L \times N(N-1)$ routing matrix (as the $i \rightarrow i$ TM elements are inconsequential here). In VND F_{ir}

can only take the values 0, $1/N$, or $2/N$. As the properties of A are not determined by the constant denominator N, we will instead look at the matrix $R = NA$, which has the values 0, 1, and 2.

We give a simple example for a 3-node network below in which both the origin-destination pairs (p, q) and the links (i, j) are ordered in the following order:

$$(1,2),(1,3),(2,1),(2,3),(3,1),(3,2). \tag{5}$$

To derive the matrix we separate it into two components: in terms of the traffic between origin/destination pair (p, q),

R_1 shows the routing of traffic on its first hop after entering the network at node p, and

R_2 shows the routing of traffic on its second hop before it reaches its destination q.

It is simple to derive R_1 as it specifies that traffic from node p will be split evenly over all links $p \to m$, so R_1 has a simple block diagonal structure:

$$R_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{6}$$

For instance, the second column of R_2 says that $1/N$ of the traffic from $1 \to 3$ goes along each of the links $1 \to 2$ and $1 \to 3$. R_2 is just the dual of R_1, that is, traffic arriving at a node follows the same pattern as traffic departing a node, so the matrix would have the same block diagonal structure if the links and origin/destination pairs were ordered by destination. Permuted to give the same ordering as above we get

$$R_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

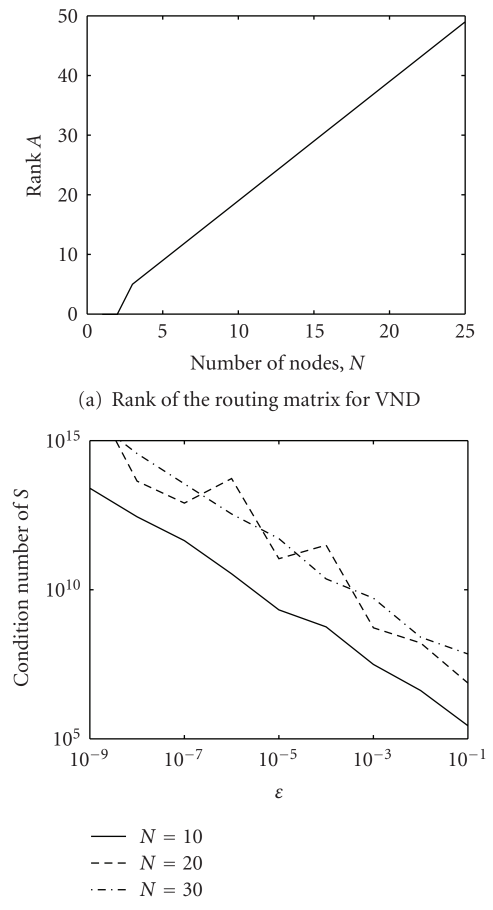
Then

$$R = R_1 + R_2. \tag{8}$$

Note that the "2" entries of R lie along the diagonal and that R is symmetric.

The question of interest is "is the matrix R invertible?" In this simple example the answer is a resounding no. In fact, all of the examples we tried (up to $N = 30$) resulted in singular matrices. For N nodes, the routing matrices were of size $N(N-1)\times N(N-1)$, but as shown in Figure 2(a), their rank was approximately $N/2$. The trend in rank suggests that the matrix will never be invertible for any N. So although A is square, it is not invertible. The underconstrained nature of the problem remains.

However, there is a fix.



(a) Rank of the routing matrix for VND

(b) Condition numbers of modified routing matrix A'

FIGURE 2: Functions of the routing matrix in VND.

3. Routing Jitter

As noted above, the routing matrix for the VND is not invertible. However, with a very small change, we can make it so. The change we introduce is to vary the traffic spread by a small amount that we will call *routing jitter*. Rather than spreading the traffic perfectly evenly we introduce a random vector $\mathbf{r}$ of length $N-2$ with sum zero, spread uniformly over the range $[-\epsilon/2, \epsilon/2]$. We keep the same amount of traffic on the direct (one hop) path between two nodes, but use $\mathbf{r}$ to modify the proportions of traffic on each of the $(N-2)$ 2-hop paths. The effect is to create a new matrix $S = R + E$, from which we can derive our new routing matrix $A' = S/N$. The key result is that, for $N > 4$, this new A' will be invertible with high probability, and the TM estimation problem now has a unique solution. Note that the possibility that A' is close to singular can be easily avoided by testing for this condition prior to its use, and applying a different jitter if the matrix is close to singular.

Note that the even load balancing in the simple VND is an artifact of the simple example we have considered with all nodes having equal capacity. In more realistic settings, VND load balancing is already uneven, so small additional changes to this routing, such as we perform above, are not a big problem, but they do have a cost. The total traffic on link (i, j) can be calculated by adding the traffic on this link arising from traffic with destination k, following path $i \to j \to k$ for some $k \neq i$ and traffic with destination j following path $m \to i \to j$ for some $m \neq j$. The traffic on link (i, j) is therefore given by

$$y_{i,j} = \frac{1}{N}\sum_{k\neq i} T(i,k)\left(1+\epsilon_{i,j,k}\right) + \frac{1}{N}\sum_{m\neq j} T(m,j)\left(1+\epsilon_{m,i,j}\right), \tag{9}$$

where $\epsilon_{i,j,k}$ is the extra traffic from i to k steered onto intermediate node j. Note that by construction we limit $|\epsilon_{i,j,k}| < \epsilon/2$, so that we can write

$$y_{i,j} \leq \frac{1+\epsilon/2}{N}\left[\sum_{k\neq i} T(i,k) + \sum_{m\neq j} T(m,j)\right] \leq \frac{(2+\epsilon)C}{N}, \tag{10}$$

using (2). The standard VND (without consideration for link/node failures) requires capacity $2C/N$, so the additional cost of our rerouting is (in the worst case) $\epsilon C/N$ capacity on each link. So clearly, we should aim to choose ϵ to be reasonably small.

The invertibility of A' for all but pathological cases of r should be obvious, but it is not the only issue. Numerical matrix inversion can be highly inaccurate if the condition number of the matrix (the ratio of the largest and smallest singular values) is too high. Figure 2(b) shows simulated condition numbers for A' for several values of N and a range of values of ϵ. We can see that the condition number increases as ϵ decreases. The smaller epsilon is, the closer to ill-conditioned the matrix becomes. However, we found that for moderately sized problems (say $N = 30$) that $\epsilon < 10^{-6}$ posed no problem (for Matlab's standard matrix inversion function), resulting in errors in the inverse on the order of 10^{-7}. As N increases, condition numbers appear to increase, so larger problems may be more difficult, but the magnitude of this effect is inconsequential compared to the following.

Real traffic consists of packets, and load balancing mechanisms can only divide traffic at this granularity. Also, in order to avoid reordering of packets in a flow, one often performs load balancing on a source/destination basis. This introduces additional granularity into the traffic flows, preventing perfect load balancing. Errors in the load balancing shares are, in effect, errors in A' the routing matrix. We need our value of ϵ to be larger than the typical values of these errors in order to be able to obtain meaningful traffic estimates, so we suggest a value of the order of 0.01–0.05, requiring an additional 1%–5% capacity, which will in addition easily result in reasonably conditioned routing matrices.

4. Discussion

The above shows that minor modification of VND's load balancing mechanism results in an identifiable TM estimation problem in the sense that the problem now has a unique solution, and in the absence of measurement errors, we can obtain the actual TM. This is ironic, considering that VND was at least in part predicated on the inability to measure this matrix.

However, we cannot just throw away the VND, because without it, we would no longer be able to make these measurements. So in the case that we have the measurements, we do not need them, and where we do need measurements, we cannot get them. This paradox is more annoying than intriguing.

In addition, VND also allows resilience to unexpected networks demands, either due to temporary surges or attacks, or due to long-term errors in traffic predictions. Surely there is some happy middle ground?

The obvious solution is to continue to use a Valiant-like network design, that is, one which uses load balancing over a clique. However, we can use the fact that we can measure the matrix to improve the design. Valiant design has a cost, roughly twice the capacity of an optimal network, which is needed in a VND. If we instead steered a percentage X of the traffic along the direct path between two nodes, then we could trade off between flexibility with respect to unexpected changes in traffic, against a reduced cost of the network design. The choice of X allows us to interpolate between the two extreme cases:

(i) $X = 1$: we get a direct routing, and given the input TM we can determine the minimum capacity network required.

(ii) $X = 2/N$: we get VND, with its resilience to unexpected traffic.

In either case, the TM is measurable.

The total capacity requirements for such a network consist of NC times the direct component plus $2NC$ times the VND component, noting that in the simple version of VND $X = 2/N$. So, the total capacity requirement is

$$P = NC + \left(\frac{1-X}{1-2/N}\right)NC, \tag{11}$$

for $X \in [2/N, 1]$. Of course, in reducing the capacity of the network, we lose some ability to deal with random variations in traffic matrices. The factor of 2 in capacity is the cost for being oblivious, so if we use the above methodology, we will no longer be able to carry any traffic matrix, but we will be able to carry the most likely traffic.

5. Conclusion

The conclusion of this paper is that there in an inherent paradox in the nature of Valiant network design. The choice to create a clique (as the underlying network structure) creates the possibility of making the traffic matrix problem identifiable. Hence, for a Valiant network design, we have (with a minor modification) enough information to measure the traffic matrix, and from this we could build some other design. Of course, if we actually change the network design (to a nonclique-based design), then we lose our measurement capability, but there is a possible alternative in choosing a design between the two possible extremes.

It should be noted that VND is also robust to prediction errors. Hence, VND can alleviate problems that may have occurred as the result of poor planning, not just because traffic matrices are hard to measure. VND can also be used to create networks that are highly resilient to node and link failures, and this is another reason we may wish to use this design methodology.

References

[1] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*, pp. 313–324, Karlsruhe, Germany, August 2003.

[2] Z.-S. Rui and N. McKeown, "Designing a predictable internet backbone," in *Proceedings of the 3rd Workshop on Hot Topics in Networks (HotNets-III '04)*, pp. 1–6, San Diego, Calif, USA, November 2004.

[3] Z.-S. Rui and N. McKeown, "Designing a predictable internet backbone with valiant load-balancing," in *Proceedings of the 13th International Workshop on Quality of Service (IWQoS '05)*, vol. 3552 of *Lecture Notes in Computer Science*, pp. 178–192, Passau, Germany, June 2005.

[4] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: existing techniques and new directions," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '02)*, pp. 161–174, Pittsburgh, Pa, USA, 2002.

[5] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*, pp. 206–217, San Diego, Calif, USA, June 2003.

[6] Y. Vardi, "Network tomography: estimating source-destination traffic intensities from link data," *Journal of the American Statistical Association*, vol. 91, no. 433, pp. 365–377, 1996.

[7] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An information-theoretic approach to traffic matrix estimation," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*, pp. 301–312, Karlsruhe, Germany, August 2003.