The Measurement Manifesto

George Varghese, Cristian Estan

Abstract—**Useful measurement data is badly needed to help monitor and control large networks. Current approaches to solving measurement problems often assume minimal support from routers and protocols (e.g., active measurement) or place the entire burden on the router to support heavyweight mechanisms (e.g., NetFlow). In this paper we argue that the research agenda in measurement must change to consider measurement solutions which enlist the cooperation of routers, protocols, and tools. We believe that the need is so urgent that the deployment issues associated with such holistic solutions can be finessed by cooperation between a few key ISPs and a few key router vendors. If this agenda is accepted, there is a rich vein of technical problems, hitherto considered only from an active measurement perspective, for which there can be new and effective orchestrated solutions. We illustrate this thesis using two examples. Our major example is that of measuring traffic matrices using class counters and class sampling (as opposed to the implementation cost of per-prefix counters, or the errors inherent in tomography). We also provide a smaller example of measuring route stability by modifying route computation. Beyond specific techniques, we hope the guidelines in this paper can provide a focus for discussion among researchers, router vendors, protocol designers, and network operators — the stakeholders in the measurement enterprise.**

I. Introduction

This paper deals with the problem of producing effective measurements to help run large networks more effectively. We focus on novel approaches to *passive measurement* [7]. Passive measurement helps determine the *causes* of network performance problems as opposed to *active measurement* [10], which provides insights into the *effects* of network problems on users. Once causes — such as links that are unstable or have excessive traffic – are identified, network operators can take action. Thus measurement is crucial not just to understand the network but to better engineer its behavior.

Control mechanisms that network operators currently use include adjusting OSPF link weights and BGP policy to spread load, setting up circuit-switched paths to avoid hot spots, and buying new equipment. This paper focuses only on network changes that address the measurement problem, making a network more *observable*. Making a network more *controllable*, by adding more tuning knobs, is an equally important problem we do not address.

Unlike the telephone network, where observability and controllability was built into the design, the simplicity of the Internet service model has made it difficult to observe [4]. In particular, there appears to be a great semantic distance between what users (e.g., ISPs) want to know, and what the network provides. In this tussle [3] between user needs and the data generated by the network, users respond by distorting [3] existing network features to obtain desired data.

For example, Traceroute uses the TTL field in an admittedly clever but distorted way. Tools like Sting [10] use TCP in ingenious ways to yield end-to-end measures. Even tools that make more conventional use of network features to populate traffic matrices (e.g, [7], [14]) bridge the semantic gap by correlating vast amounts of spatially separated data and inconsistent configuration information. This is clever, but is it engineering? For example, civil engineers do not primarily test the stability of bridges indirectly by driving cars of random weight across the bridge, but instead by directly building in measurement sensors.

The problem is complicated by the attitudes of the stakeholders [3] in the measurement enterprise. *Router vendors* avoid adding measurement features because it impacts forwarding performance. *ISPs* appear resigned to the use of indirect means. Finally, many *network researchers* may regard passive measurement[1] as a boring exercise that could be solved by slapping down a few counters in the right places.

And yet there are signs that change is possible. Cisco Express Forwarding offers per-prefix counters, a massive step up in utility (and implementation complexity) from SNMP counters. ISPs are putting pressure on router vendors to add features. Juniper's DCU solution [11] uses routing protocol assistance to reduce administrative complexity for accounting. Finally, researchers are making imaginative proposals for new network measurement primitives [4], [5].

Despite these winds of change, other than the Juniper DCU solution [11], no solution represents a concerted effort to put all the options (i.e., new router implementation features, protocol changes, new tools) together to orchestrate effective systems solutions to user needs. Systems solutions are those that exploit the fact that a system consists of a number of components, all of which can be modified to help users.

Our central thesis is that *sytems solutions can bridge the semantic gap in network data today with reasonable implementation cost.* Further, such solutions are deployable with achievable cooperation among the stakeholders. While incremental deployment has been the gold standard for Internet research in the academic community, it has not stopped major vendors from introducing concerted changes such as MPLS and DiffServ in recent years. If the customer need can be demonstrated to be sufficiently important (e.g., traffic engineering for MPLS, QoS for DiffServ) vendors can be persuaded. The need *is* important: failure to address the semantic gap will cause the situation to steadily worsen, with a continued proliferation of ad hoc tools and a lack of coherent data. All this has a large hidden cost in reduced productivity.

The rest of this paper is organized as follows. Section II describes guidelines that we use in searching for orchestrated measurement solutions. Section III reviews existing primitives and proposals in the light of these guidelines. Section IV describes a detailed attack on the problem of measuring traffic matrices via a solution that generalizes existing approaches ranging from tomography to per-prefix counters. Section V describes a smaller encounter with the problem of measuring route stability. Section VI summarizes the paper.

[1] Active measurement is another matter; finding clever ways to manipulate an unsuspecting network to yield its secrets (e.g., TBIT, Sting) continues to fascinate researchers.

II. Three Guidelines

The RISC revolution can be partly attributed to the following observations. First, architects learned from circuit designers that decoding complex instructions required large clock cycles. Second, architects found that many complex instructions were not used in user benchmarks. Third, architects learned to simulate complex instructions in software, and to move some aspects (e.g., pipeline scheduling) traditionally done in hardware to the compiler. These lessons can be abstracted into three straightforward guidelines:

Guideline P1, Understand real implementation costs: Understand costs and hence the space of feasible implementations.[2]

Guideline P2, Understand real user needs: Determine those aspects of current solutions that do not match user needs, and can thus be potentially simplified. Modify measurement primitives to reduce the semantic gap between user needs and network data.

Guideline P3, Leverage other aspects of the system: Recognize that a system consists of multiple components that can cooperate to form effective solutions.

While these guidelines appear trite, we will attempt to show that they have some teeth in the measurement context by examining existing proposals using the principles as a yardstick in Section III, and then using the principles to suggest new solutions to two *specific* problems in Section V and Section IV. Our descent from a 30,000 feet view of the measurement world down to ground level in Section IV and Section V will be somewhat rapid, and may disconcert the reader. Unfortunately, since measurement is ultimately about measuring specific things, it is difficult to appreciate the challenges without considering specific measurement problems. We step back for a broader view once again in Section VI.

III. Existing and Proposed Schemes

We review standard measurement primitives in Section III-A, consider new research proposals in Section III-B, and consider an imaginative (at least to our minds) and orchestrated solution for accounting from Juniper Networks.

A. SNMP and NetFlow

The following measurement primitives are standard. While useful, building tools based on them is akin to writing programs in assembly language: low-level, tedious, and error-prone.

SNMP Counters: Routers implement a large number of SNMP counters, but for measuring the traffic mix on a link the most relevant are packet and byte counters. SNMP counters are easy to implement (**P1**), and are useful (**P2**) for managers to determine congested links. Unfortunately, in terms of **P2**, SNMP does not go far enough; today's hardware can easily support more discriminating counters.

NetFlow: NetFlow allows managers to log flow records keyed on TCP/IP header fields. Because of the need to write these headers to slow DRAM, earlier implementations slow down routers considerably. These problems are partly addressed by Sampled NetFlow and aggregated NetFlow. The solutions recognize (**P2**) that users can get good statistics about traffic from samples, and that users often only want the sum of traffic for a given 5-tuple. Many implementations of Sampled NetFlow are still problematic, and the vast amounts of data generated can swamp managers and tools. Despite this, NetFlow is invaluable for its role in diagnosis though other solutions can meet other user needs more efficiently. [7] correlates NetFlow data at various routers to route information. This is error-prone and must deal with incomplete data. Note that the AT&T production network uses the tomogravity approach [14].

[2]While the fact that RISC stripped away features to make hardware simpler may have been technologically right twenty years ago, it is equally a mistake to underestimate the potential of modern hardware. Thus, increasingly complex instructions have been creeping back into even classic RISC machines like the MIPS.

B. Newer Proposals

The following proposals in the last three years have attempted to raise the level of abstraction of network data.

[4] uses a common hash function to synchronize sampling of a packet across all routers, and a second common hash function as a content digest. [6] attempts to finesse the need for NetFlow collection at a router by providing an algorithm to directly compute (at high speeds) the flows over a threshold. [5] proposes a sampling technique that can sieve the amount of NetFlow data sent to a manager while preserving any estimates of high flows.

Trajectory sampling still requires great complexity in a tool to gather and correlate labels from routers. On the other hand, [6] and [5] only provide local views of heavy-hitters on one link, and do not provide the network-wide view that ISPs need.

DCU: Juniper Network's Destination Class Accounting (DCU) solution [11] combines all three guidelines. DCU addresses the issue of an ISP wishing to collect traffic statistics on traffic sent by a customer in order to charge the customer differently depending on the type of traffic and the destination of the traffic. For example, assume that ISP Z wishes to bill Customer A at one rate for all traffic that exits via ISP X, and at a different rate for all traffic that exits via ISP Y. One way to do this would be for router $R1$ to keep a separate counter for each prefix that represents traffic sent to that prefix.

Instead, the Juniper DCU solution [11] has two components. First, each forwarding table entry has a 16 bit class ID. Each bit in the class ID represents one of 16 classes. Thus if a packet matches prefix P with associated class ID C, if C has bits set in bits 3, 6, and 9, the counters corresponding to all 3 set bits are incremented. Thus there are only 16 classes supported but a single packet can cause multiple class counters to be incremented. This is easily implementable in a forwarding ASIC (**P1**). The solution also supports 16 tariffs (**P2**) for traffic charging. More interestingly, to attack the problem of changing prefix routes (which would result in the tool having to constantly map each prefix into a different class), the DCU solution enlists the help of the routing protocol (**P3**).

The idea is that all prefixes advertised by ISP X are given a color (which can be controlled using a simple route policy filter), and prefixes advertised by ISP Y are given a different color. Thus when a router gets a route advertisement for prefix P with color c, it automatically assigns prefix P to class c. This small change in the routing protocol greatly reduces the work of the tool.

IV. Measuring the Traffic Matrix

If even half the attention to 'rocket science traffic modeling' were devoted to how to estimate a reasonable ingress-egress traffic matrix, network engineers, particularly of large clouds, would find their job substantially easier

— Dennis Ferguson, 1996 ISMA

For our second problem, consider a network such as those used by ISPs like Sprint and AT&T. The network can be modeled as a graph with links connecting router nodes. Some of the links are *external* as they go to routers belonging to other ISPs or customers. External links directed towards the ISP router are called input links; external links directed away from an ISP router are called output links.

The traffic matrix of a network enumerates amount of traffic that was sent (in some arbitrary period) between *every* pair of input and output links of the network. For example, the traffic matrix could tell managers of an ISP Z that 60 Mbits of traffic entered during the day from Customer A of which 20 Mbits exited on the peering link $E2$ to ISP X. Network operators find traffic matrices (over time scales ranging from hours to months) indispensable. They are used to optimize routing decisions (by changing OSPF weights), for knowing when to set up circuit switched paths (avoiding hot spots), for network diagnosis (understanding causes of congestion), and for provisioning (knowing which links to upgrade).

Unfortunately, existing legacy routers only provide a single aggregate counter (SNMP byte counter) of all traffic traversing a link, which aggregates traffic sent between all pairs of input and output links that traverse the link. Inferring the traffic matrix from such data is hard because there are $O(V^2)$ possible pairs in the matrix (where V is the number of external links) and many sparse networks may only have say $O(V)$ links (and hence $O(V)$ counters). Even after knowing how traffic is routed, one has $O(V)$ equations for $O(V^2)$ variables, which makes deterministic inference impossible. This has led to two very different solution approaches.

Approach 1, Internet Tomography: This approach [9], [14] does statistical inference based on SNMP counters. At the heart of the technique is a model of the underlying traffic distribution (e.g., gravity) and some statistical (e.g., maximum likelihood) or optimization technique (e.g., quadratic programming [14]). Early approaches based on Gaussian distributions did poorly [9], but a new approach based on gravity models does much better, at least on the AT&T backbone [14]. The great advantage of tomography is that it works without retrofitting existing routers, and is cheap to implement in routers. Disadvantages of this method are errors (off by as much as 20% in [14]) and sensitivity to routing errors (a single link failure can throw an estimate off by 50%).

Approach 2, Per-prefix counters: Some newer routers offer per-prefix counters. By pooling together router per-prefix counters and with a knowledge of routes, a tool can reconstruct the traffic matrix. One advantage of this scheme is that it provides perfect traffic matrices. A second advantage is that it can be used for differential traffic charging based on destination address. The two disadvantages are the implementation complexity of maintaining per-prefix counters (and the lack thereof in legacy routers), and the large amount of data that needs to be collected and synthesized from each router to form traffic matrices.

Instead, we propose a scheme that dials between these two earlier approaches using per-class counters that aggregate multiple prefix counters into each per-class counter. Our scheme is related to but different from the DCU proposal [11].

Implementation Complexity: Routers have to work for 5-10 years. Factoring growth, such a router needs to support perhaps 1 million prefixes. Byte counters are 64 bits. To keep up with wire speeds at say 40 Gbps, counters must be in fast (1- 5nsec cycle time) SRAM. 64 Mbits of fast SRAM is expensive, and has other costs in terms of area and power. Shah et al. [12] show how to reduce SRAM width by storing only low order counter bits in SRAM but storing all 64 bits in a cheaper DRAM backing store. However, the implementation is still complex and requires sorting.

On the other hand, implementing a small number (e.g., 1000) counters in on-chip SRAM is trivial. A hidden cost even for a small number of counters is the cost of mapping a packet to a counter. One simple way to do this is to add to each next hop table entry a *class ID*. For example with 1000 counters, a 10 bit class ID has to be added to each forwarding entry. For say a million prefixes, this is 10 Mbits! However, many next-hop entries are large (20 bytes to store multiple adjacencies for load balancing), and some routers already add a 16-bit class ID. Better still is to finesse the need for a class ID by mapping to a class based on information (e.g., output port, see local matrix proposal in Section IV-B) already present in the forwarding entry.

The rest of this section is organized as follows. Section IV-A uses the principles to derive the general scheme, and Section IV-B describes five interesting special cases, three of which yield new schemes. Section IV-C proposes using a small counter space and yet eliminating tomography completely using sampling on the class counter space.

A. General Solution using Class Counters

We apply the guidelines of Section II to the traffic matrix problem. First, we have just seen (**P1**) that 100-1000 counters are almost as simple as 1 counter except for the potential storage per prefix for class mapping, and even that can be eliminated if the mapping is a trivial function of existing next-hop information.

Second, the major applications (**P2**) of prefix counters seem to be accounting and traffic matrices. In both cases, the solution seems unaligned with real needs. For example, prefix counters seem to be overkill for the traffic matrix problem because there may be millions of prefixes (120,000 today) but an ISP may have far fewer external links.

The RocketFuel [13] data indicates that the large ISPs had between 6000 and 12,000 external links in 2002. Similarly, an accounting application may have only have hundreds of different providers and much fewer tariff structures (Juniper, observing that cell phones offer at most 8 tariffs, allows 16). Thus the final tool (for traffic matrix calculation or accounting) would aggregate thousands of prefixes into equivalence classes anyway.

Why not have the router do this in the first place, reducing complexity for *both* the router and the management tool?

Mapping from prefixes to equivalence classes can cause problems when prefixes change. If a prefix advertised by ISP X begins to be advertised by ISP Y, who should tell each router of the new mapping? Rather than have the tool or managers do this, it is far more "administratively scalable" [11] to enlist help from the routing protocol. Thus the proposed solution has two aspects:

1. Class Counters: Each prefix is mapped to a small class ID of 8-14 bits (256 to 16,384 classes) using the forwarding table.

2. Routing Support: As in the DCU proposal, for the traffic matrix, a similar idea can be used to color routes based on the matrix equivalence class (e.g., all prefixes arising from same external link or network in one class).

Some differences between Junipers DCU proposal [11] and ours are:

Scalability: Our scheme is more scalable because N classes require at most $\log_2 N$ class ID bits per prefix, and at most one increment. Incrementing multiple counters per packet, as in the DCU proposal, requires an N bit class ID, and a more complex implementation that is unlikely to scale to $N > 1000$.

Different Application: Given 10,000 external links, the traffic matrix application motivates the need for a larger number of classes.

Different routing support: We suggest coloring routes based on the topological information (see Section IV-B), not the tariff class.

Subtleties: This is not a complete proposal; there are tricky issues (e.g., OSPF load balancing) that need to be addressed, and can be. Two issues worth noting are as follows. First, some applications may wish to select traffic (or accounting) classes based on QoS. This can be easily done using the DiffServ bits (in addition to the class ID) to select a class. Second, many papers [7] point out that the real traffic matrix is a point-to-multipoint demand because traffic from say Customer 1 may have multiple egress points to the same Provider. This is easily handled by aggregating all these egress points into the same class.

Our proposal scales by allowing only one counter to be incremented per packet. But accountants may want tariff data, and operations may need the traffic matrix. A small generalization is to map a prefix to a superclass ID which can then be parsed to yield a small number of class IDs. For example, a 16 bit superclass ID could be partitioned into an 8 bit class ID for accounting, and an 8 bit class ID for traffic matrices. A received packet increments both. Note that we have separated the degree of parallelism from the number of counters; these two are confounded in the DCU proposal. Class counters, are of course, specified abstractly by specifying only that routers increment a counter for every class in a superclass.

B. Five Special Cases of Class Matrices

The mechanism above provides the traffic counts from each input link to each destination class. By aggregating input links (at either the router or the tool) also into classes, this method yields the class to class traffic matrix. Let N be the number of classes. We now consider five special cases (Figure 1 shows three cases in a different order from the text below) of class matrices. The first two are well known.

1, One class: If $N = 1$, we have a single SNMP counter, which requires tomography to find the traffic matrix.

2, Prefixes: If N is the number of prefixes we get per-prefix counters — see bottom of Figure 1. Note that the class mechanism is a strict generalization of the two earlier solutions (SNMP and per-prefix counters) which represent two extremes. The next three are more interesting.

3, External Links: We have already referred to this idea, shown in the middle of Figure 1. This provides the traffic to and from external links. Links could be aggregated by provider to handle point-to-multipoint data. RocketFuel data [13] indicates that an $N = 12,000$ sufficed in 2002 for the largest ISP, but this number is probably growing, especially with ISPs consolidation and mergers.

4, PoPs: A PoP is a physical location where an ISP houses a collection of routers. Many ISPs find the PoP-to-PoP traffic matrix to be very valuable [2] and aggregate the router-to-router matrix to find this. This can be done directly by classes by setting each PoP into a separate class. RocketFuel data [13] indicates a great reduction in the number of PoPs, with $N = 150$ sufficing for the largest ISP.

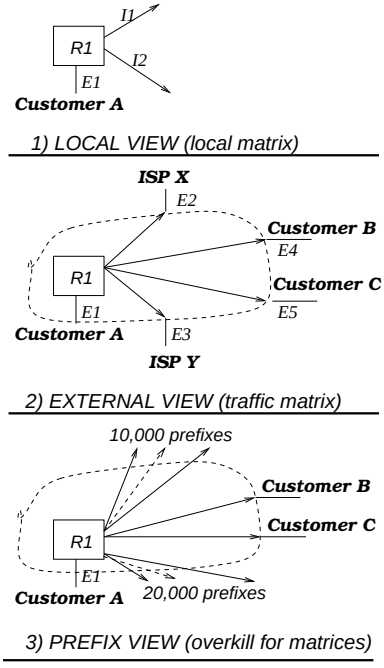

Fig. 1. Using the class counter to produce various views that range from the standard per-prefix view of traffic (bottom) to a local traffic matrix (top).

5, Router Ports: To dial down the number of classes still further, consider making every output port in a router a class Figure 1). This provides the *local traffic matrix* at a router — i.e., the traffic between every input port and output port on each router. Two implementation advantages that accrue are: first, from packets to classes is now trivial and there is no need for a class ID in each forwarding entry (output port is effectively the class ID); second, many routers have a small number of ports from 16 to at most 256 and the reduced class count implies a smaller number of counters.

The conjunction of local matrices at all routers *does not* provide the global traffic matrix. However, local matrices provide more equations that help constrain tomography schemes even more, and thus seems likely to produce more accurate solutions. Although one may expect a factor of P^2 gain in the number of equations where P is the number of ports (e.g., this is a factor of 256 for a 16 port router), the equations are not independent. In practice, some early experiments [8] show a factor of 2 increase in the number of equations.

Despite only a two-fold gain in the number of *equations*, for inference on the AT&T network, the accuracy improvement of knowing the local matrix everywhere is roughly the same as knowing the top 15 complete rows (e.g. by turning on NetFlow at 15 busiest backbone routers). With no noise, simulations show that the best-known tomographic inference scheme [14] has an error of 11%; the error reduces to 3% using the extra information in our local matrices idea. With a noise level of 5% in the SNMP link data, simulations show a relative error of 14% without local matrices, and only 5% with local matrices.

For a noise level of 10%, the relative error is 18% without local matrices and 7% with local matrices. This is perhaps surprising because the final error is less than the amount of input noise assumed! While this was on the AT&T network, local matrices should help tomography even more on a sparse network; if the topology is a star, note that the local matrix gives the complete traffic matrix! Overall, local matrices increase the accuracy of the best known tomography schemes ([9], [14]) by a factor of 2.5 to 4.

Local matrices can be generalized to compute the use the two-hop (or k-hop) local matrices. The marginal gain in terms of independent equations generated by $k > 1$ hop matrices is small, and thus the knee of the tradeoff curve seems to be at $k = 1$.

C. Sampling Classes as an Alternative to Tomography

We have seen that current data [13] indicates that an implementation must use around 12,000 classes for the external link matrix, 150 for the POP matrix, and even as few as 32 for the local matrix. We claimed earlier that 100-1000 counters is easy to implement. Assuming that on-chip SRAM sizes can scale faster than the growth rate of external links, we believe that even handling the external link matrix is feasible using our proposal, now and in the future.

However, some implementations may wish to have a smaller number of classes (say 32) in order to use on-chip SRAM for other purposes. An interesting question worth answering is whether one can infer larger matrices (e.g., external link matrix) using a number of classes that is strictly smaller than the number of rows of the desired matrix.

One approach, as in the local matrices idea above, is to compute some exact numbers and use them to constrain statistical inference in tomography. But a completely different approach is to realize that the final output of tomography is a set of *statistical* traffic pair estimates based on *deterministic* inputs. Why not consider turning this proposition on its head and calculate *deterministic* traffic-pair estimates based on *statistical inputs*?

In other words, even if we have only 32 class counters and we wish to watch say 12,000 external links in the AT&T network, why not sample (in time) the links being watched? Thus in any subinterval, 32 randomly selected output links are watched faithfully by the class counter. At the next subinterval, another 32 randomly selected output links are watched.

Assume a traffic estimate is required for K subintervals (e.g., for estimating the hourly traffic matrix, one may wish to use 3600 one second subintervals). Suppose traffic from Customer A to ISP Y was watched for k out of the K possible subintervals. Then a simple estimate for the traffic of Customer A to ISP Y can be found by scaling the sum of the traffic over the watched intervals by the factor K/k.

The random sampling can be implemented by the route processor by picking a class counter to replace, reading the value, and updating the affected prefixes. This is fairly time consuming and thus implementation constraints will require subintervals of at least seconds to avoid burdening route processors. While there is the usual tradeoff between the sampling rate and accuracy, the analysis for such class sampling seems different from that of standard sampling (e.g., [4]). This is because the samples are batched with a random interval between batches and not between individual samples.

Such batch sampling could cause problems with bursty distributions. For example, suppose a flow class sends traffic during the busiest hour and nothing else for the other hours. If the subintervals are in units of hours, then there is a strong chance that the sampling will miss this flow's traffic. However, this is not true if the subintervals are in units of seconds. Thus as with tomography there is some dependence on the input traffic model. However, the dependence is quite different and can (hopefully) be abstracted in terms of some simple summary property of the distribution such as the maximum (with high probability) burst length. One conjecture is that class sampling should do well if the sampling subinterval is much smaller than the maximum burst length, but this needs to be studied.

Further, the accuracy of class sampling can be improved using the sample-and-hold technique of [6]. This comes at the cost of only finding the large elements in the traffic matrix, but this appears to suffice [7] for the traffic matrix and accounting applications. The results in [6] indicate that the use of sample-and-hold can make N counters have the accuracy of using roughly N^2 counters and ordinary sampling. This should provide good accuracy for even small values of N. In general, comparing the accuracy and robustness (especially to assumptions about traffic distributions) of class sampling versus tomography seems an interesting open problem.

V. Measuring Route Stability

We turn to a quick second example of measuring route stability. For imagine that the route from Customer A to Customer B usually uses the direct path through link $I1$. However, if link $I1$ is flaky, assume that a backup path is chosen instead. If link $I1$ keeps coming up and down, traffic to Customer B (say an important web site) may be affected because of routing instability.

This phenomenon could be measured indirectly by observing link failures via say SNMP counts and then correlating with the topology and the protocol to determine that say the route to Customer B will be affected. However, there is a simpler and more direct way to measure route instability with help from the routing protocol or its implementation.

The idea is illustrated in the case of OSPF (by far the most common IGP used by ISPs). When new link state packets arrive at announcing the failed link, the tree moves around to include the backup path. Normally, a router high in the Dijsktra is blissfully unaware of the route instability if its own next hop for Customer B stays the same when the link bounces. However, a trivial, incrementally deployable, change to Dijsktra's algorithm can notice that a node in the tree has changed parents. All children of a node whose parent has changed are marked as having changed, by passing an "instability flag" down the tree.

The route processor can now maintain summary route stability statistics — such as the number of changes in the last day — on a per-prefix basis. If this is considered too expensive, a management station attached to router can compute this information if it receives all link state packets. Similar ideas can be applied to path vector protocols like BGP by passing an instability attribute with a route.

This approach can be simulated by having a PC pretend it is a router and trick an adjacent router into believing it is a router. However, such route stability measurements could be more useful if routers include summary link statistics (e.g., traffic utilization, link errors, etc.) in link state packets. Instead of using these statistics for dynamic routing, which has been considered difficult to do without endangering network stability, the management station could this information to potentially correlate routing problems to causes.

VI. Conclusions

We first state our *specific* conclusions based on the specific new techniques we examined in this paper. First, route stability can be measured precisely and even possibly explained by watching routing traffic, and passing link statistics in LSPs. Second, traffic matrices can be calculated more directly using class counters and routing support. Even using an easily feasible number of class counters, local traffic matrices can be used to improve the accuracy of tomogravity schemes [14], PoP matrices can possibly be directly calculated, and link to link matrices can be estimated by class sampling techniques.

But the main message of this paper is broader than the two specific examples we used to illustrate the ideas. The *general* conclusions are as follows. First, we believe that there is a large semantic gap between data provided easily by the network and the needs of users like ISPs. This leads to either the need to trick the network into providing data (e.g., [10]), or to collect, correlate, and synthesize vast amounts of spatially separated data (e.g., [7]). Second, we believe this semantic gap can be bridged at reasonable implementation cost by a systems approach.

Third, The systems approach is codified in this paper using three guidelines. Any such list is necessarily incomplete, but they can perhaps serve as a first cut. Fourth, we believe that besides the two examples we used in this paper (route stability and traffic matrix calculation), *there are many other network-wide metrics whose calculation can be reconsidered using these principles. Other examples include link bandwidth, end-to-end error rates, and routing update delays.* These three examples were taken from papers on these topics (all of which assumed the network to be a black box) in the 2001 Internet Measurement Workshop. A more careful survey could probably find a larger number of such problems.

Fifth, we believe that it is possible for the stakeholders to work together. We believe network researchers can be convinced that the field of measurement is not merely a boring study of counters, but can furnish rich and beautiful problems. We believe that ISPs are already discovering a business case for better data collection that can lead to running their networks more profitably. Finally, we believe that router vendors can make changes when pressed.

Amplifying the last point, it may be argued that protocol changes, as advocated by guideline **P3**, are hard to accomplish. However, this flies against the number of protocol changes made in the last few years (for MPLS traffic engineering, supporting VPNs, etc.). In general, it appears that as long as there is a strong business case for the two dominant router companies, protocol and implementation changes can and do happen.

To establish such a business case it would clearly help for ISPs to form a user consortium to specify measurement requirements. After all, router vendors already take the NEBS standards for reliability in telecommunications equipment very seriously. Thus we end, somewhat tongue-in-cheek and with apologies to Karl Marx, with a final slogan:

ISPs of the world unite — you have nothing to lose but the chains that imprison the data you need!

References

[1] S. Bhattacharya et al Network Measurement and Monitoring: A Sprint Perspective draft-bhattacharyya-monitoring-sprint-00.txt

[2] S. Bhattacharya et al. Pop-level and Access-link traffic dynamics in a Tier-1 pop. In *SIGCOMM Internet Measurement Workshop*, November 2001.

[3] D. Clark et al. Tussle in cyberspace: Defining tomorrow's internet. In *Proceedings SIGCOMM 2002*, September 2002.

[4] N. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *Proceedings ACM SIGCOMM*, August 2000.

[5] N. Duffield, C. Lund, and M. Thorup. Charging from sampled network usage. In *SIGCOMM Internet Measurement Workshop*, November 2001.

[6] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings SIGCOMM 2002*, September 2002.

[7] A. Feldmann et al. Deriving traffic demands for Operational IP networks: Methodology and experience. In *SIGCOMM 2000.*

[8] Y. Zhang et al An information theoretic approach to Estimation of Traffic Matrices from link traffic measurements. In *SIGCOMM 2003*,

[9] A. Medina et al. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings SIGCOMM 2002*, September 2002.

[10] S. Savage. Sting: A TCP-based network measurment tool. In *USENIX Symposium on Intenet Technologies and Systems*, 1999.

[11] C. Semeria and J. Gredler. Juniper networks solutions for network accounting. In *Juniper White Paper, 200010-001*, 2001.

[12] D. Shah et al. Maintaining statistics counters in router line cards. In *IEEE Micro*, Jan 2002.

[13] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies using RocketFuel. In *Proceedings SIGCOMM 2002*, September 2002.

[14] Y. Zhang et al. Fast accurate computation of large-scale IP matrices from link loads. In *ACM SIGMETRICS 2003*, May 2003.

Acknowledgements: Albert Greenberg and Jennifer Rexford (of ATT) and Christophe Diot (formerly of Sprint Labs) provided us with invaluable information about ISP operations. After reading this paper, Christophe Diot pointed out an unpublished Internet draft [1] that provides a valuable Sprint perspective on passive monitoring, and also states the need for router changes to support measurement. We are grateful to Yin Zhang for implementing our local matrix idea in simulations of the ATT tomogravity approach and providing us with quantative results.